



SharePoint® Server 2010

Enterprise Content Management

Todd Kitta, Brett Grego, Chris Caplinger, Russ Houberg

SHAREPOINT® SERVER 2010 ENTERPRISE CONTENT MANAGEMENT

| | |
|---|------------|
| INTRODUCTION..... | xxix |
| ► PART I INTRODUCTION TO ENTERPRISE CONTENT MANAGEMENT | |
| CHAPTER 1 What Is Enterprise Content Management? | 3 |
| CHAPTER 2 The SharePoint 2010 Platform..... | 17 |
| ► PART II PILLARS OF SHAREPOINT ECM | |
| CHAPTER 3 Document Management..... | 33 |
| CHAPTER 4 Workflow | 87 |
| CHAPTER 5 Collaboration | 133 |
| CHAPTER 6 Search | 173 |
| CHAPTER 7 Web Content Management | 215 |
| CHAPTER 8 Records Management..... | 243 |
| CHAPTER 9 Digital Asset Management..... | 275 |
| CHAPTER 10 Document Imaging..... | 293 |
| CHAPTER 11 Electronic Forms with InfoPath | 357 |
| CHAPTER 12 Scalable ECM Architecture..... | 381 |
| ► PART III SHAREPOINT ECM SUPPORT CONCEPTS | |
| CHAPTER 13 ECM File Formats..... | 421 |
| CHAPTER 14 The SharePoint ECM Ecosystem..... | 451 |
| CHAPTER 15 Guidance for Successful ECM Projects | 469 |
| INDEX..... | 495 |

SharePoint® Server 2010 Enterprise Content Management

SharePoint® Server 2010 Enterprise Content Management

Todd Kitta
Chris Caplinger
Brett Grego
Russ Houberg



WILEY

John Wiley & Sons, Inc.

SharePoint® Server 2010 Enterprise Content Management

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

Copyright © 2011 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-58465-1
ISBN: 978-1-118-16730-4 (ebk)
ISBN: 978-1-118-16731-1 (ebk)
ISBN: 978-1-118-16732-8 (ebk)

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 also publishes its books in a variety of electronic formats and by print-on-demand. Not all content that is available in standard print versions of this book may appear or be packaged in all book formats. If you have purchased a version of this book that did not include media that is referenced by or accompanies a standard print version, you may request this media by visiting <http://booksupport.wiley.com>. For more information about Wiley products, visit us at www.wiley.com.

Library of Congress Control Number: 2011928430

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.

For Shannon

— TODD KITTA

*I would like to dedicate my chapters to and thank
Linda, Kirsten and Chelsea for letting me give up
some quality family time to help author this book.*

— CHRIS CAPLINGER

*To Kim; you are my best friend and I love you more
than words could ever say.*

— BRETT GREGO

*This book is dedicated to my wife, Melanie, and my
two boys, Jared and Austin.*

— RUSS HOUBERG

ABOUT THE AUTHORS

TODD KITTA has a background that includes software architecture and development, project management, consulting, and technology advisement. He has been working with the .NET platform since the beta timeframe and has garnered a deep expertise on the Microsoft development platform as a whole. His expertise also spans across Microsoft SharePoint, Windows Azure, and the Microsoft Business Intelligence stack, as well as Microsoft's Connected Systems platform including BizTalk, Windows Workflow Foundation, and Windows Communication Foundation. Todd authored *Professional Windows Workflow Foundation*, which was published by Wrox. In addition, he commonly speaks at user group meetings and other special events in the Midwest and beyond.

CHRIS CAPLINGER is the CTO as well as one of the founders of KnowledgeLake, Inc. (www.knowledgelake.com), and a Microsoft Gold ISV specializing in Document Imaging. Chris is a member of executive and engineering teams at KnowledgeLake. Chris has been working in the document imaging, workflow, and ECM industries since 1996, working for systems integrators and as an independent contractor before helping build KnowledgeLake.

BRETT GREGO is the Director of Engineering at KnowledgeLake, Inc., where he is responsible for building a suite of enterprise content management (ECM) products for Microsoft SharePoint. He has more than 15 years of experience developing software and since the release of Microsoft SharePoint 2003 he has leveraged this platform to develop numerous successful products. His time at KnowledgeLake, Inc., began as a developer when he architected and implemented one of the first AJAX-based production document image viewers for Microsoft SharePoint back in 2005 and continues into the present as he manages teams of seasoned engineers to create some of the world's leading products in the SharePoint ECM market.

RUSS HOUBERG is a SharePoint Microsoft Certified Master (MCM) and has been a Senior Architect at KnowledgeLake for more than 6 years. Russ is responsible for designing the taxonomy and topology architecture for KnowledgeLake's document imaging customers who regularly require enterprise-class scalability. Russ has spent the last several years focused on pushing the boundaries of SharePoint scalability and has authored and co-authored several whitepapers, including two on behalf of Microsoft (the "SQL Server 2008 R2 Remote BLOB Storage" whitepaper and the "Using Microsoft Office SharePoint Server to implement a large-scale content storage scenario with rapid search availability" case study).

CREDITS

ACQUISITIONS EDITOR

Paul Reese

PROJECT EDITOR

Kelly Talbot

TECHNICAL EDITOR

Chris Geier

PRODUCTION EDITOR

Daniel Scribner

COPY EDITOR

Luann Rouff

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

PROOFREADER

Jen Larsen, Word One

INDEXER

Robert Swanson

COVER DESIGNER

LeAndra Young

COVER IMAGE

© iStock / kycstudio

ACKNOWLEDGMENTS

FIRST AND FOREMOST, all glory and honor to The Father, Son, and Holy Spirit. Thank you to my family for putting up with me while writing this book and for just being awesome. Thank you to my colleagues Chris, Brett, and Russ for collaborating on and rocking this book. Thank you to Kelly Talbot for doing a great job keeping this book on track. And thank you to Paul Reese and Chris Geier for your contributions to this book.

— TODD KITTA

I WOULD LIKE TO FIRST THANK Todd Kitta for leading and inspiring us all to put this book together. I'm hoping this is the first of many projects we can work on together. I would also like to congratulate Russ for completing his SharePoint 2010 MCM while we put the finishing touches on this book. To Todd, Russ, and Brett, thank you for finding the time to put together your chapters while working for the fast-paced and growing organization of KnowledgeLake. It's been great working with all of you and as much as I'm happy about being finished I will miss the camaraderie of doing this together. Now, let's go grab a beer.

— CHRIS CAPLINGER

I WOULD LIKE TO THANK my fiancée Kim for being so supportive as I wrote this book and reminding me how cool it is to be author. I would like to thank KnowledgeLake, Inc., for giving me the opportunity to leverage my talents to prosper in my career and for providing the best place in the world to work. I want to thank all of the people who work for me on the engineering team at KnowledgeLake. Without them, our products would not be where they are today. I would like to thank my mom and dad for instilling in me a strong work ethic and drive to succeed. I owe all of my success to them. I would also like to thank the editors for ensuring that everything in this book is clear and understandable and Wiley Publishing for giving us the opportunity to work on this project. And finally, I would like to thank my boss Chris Caplinger. You have been a great mentor and I have learned a lot and will continue to learn a lot from you.

— BRETT GREGO

FIRST AND FOREMOST, I'd like to thank my Father in heaven for blessing me with the skills and abilities that I have to work with SharePoint. Without Him, my career would not be possible. A close second, I'd like to thank Melanie, Jared, and Austin, who all sacrificed while I took the time to write my chapters, particularly over the holidays. I would also like to thank Darlene and Jim who back in the early 1980s let me tinker with what was, at that time, a cutting-edge new IBM personal computer. It was the birthplace of my desire to work with computers for a living. I also want to thank Dan, Gregg, Ron, and Bob and the rest of the folks at KnowledgeLake for creating and maintaining a culture of taking care of the people who take care of the customers. Finally, this book would not have been possible without Brett, Chris, Todd, Kelly, and the content and technical editors who I worked with on this project. It was a pleasure.

— RUSS HOUBERG

CONTENTS

INTRODUCTION

xxix

PART I: INTRODUCTION TO ENTERPRISE CONTENT MANAGEMENT

| | |
|--|-----------|
| CHAPTER 1: WHAT IS ENTERPRISE CONTENT MANAGEMENT? | 3 |
| Introduction to ECM | 4 |
| A Historical Perspective | 4 |
| Document Imaging | 4 |
| Electronic Documents | 6 |
| COLD/Enterprise Report Management | 6 |
| Business Process Management/Workflow | 6 |
| ECM Components | 7 |
| Capture | 7 |
| Paper | 7 |
| Office Documents | 8 |
| E-mail | 8 |
| Reports | 9 |
| Electronic Forms | 9 |
| Other Sources | 9 |
| Store and Preserve | 9 |
| Software | 10 |
| Hardware and Media Technologies | 10 |
| Cloud | 11 |
| Management Components | 12 |
| Document Management | 12 |
| Web Content Management | 12 |
| Business Process Management and Workflow | 12 |
| Records Management | 12 |
| Collaboration | 13 |
| Delivery | 13 |
| Search | 13 |
| Viewing | 14 |
| Transformation | 14 |
| Security | 16 |
| Summary | 16 |

| | |
|---|-----------|
| CHAPTER 2: THE SHAREPOINT 2010 PLATFORM | 17 |
| A Brief History of SharePoint | 18 |
| SharePoint 2010 | 18 |
| Capability Categories | 19 |
| Sites | 19 |
| Composites | 19 |
| Insights | 20 |
| Communities | 20 |
| Content | 21 |
| Search | 21 |
| SharePoint Concepts | 21 |
| Architecture | 23 |
| Development Concepts | 26 |
| ECM in SharePoint 2010 | 27 |
| Managed Metadata | 27 |
| Ratings | 28 |
| The Content Type Hub | 28 |
| Search | 28 |
| Workflow | 28 |
| Document Sets | 28 |
| Document IDs | 29 |
| Content Organizer | 29 |
| Records Management | 29 |
| Digital Asset Management | 29 |
| Web Content Management | 29 |
| Summary | 30 |
| PART II: PILLARS OF SHAREPOINT ECM | |
| CHAPTER 3: DOCUMENT MANAGEMENT | 33 |
| What Is Document Management? | 34 |
| Microsoft SharePoint As a Document Management System | 35 |
| Document Taxonomy | 35 |
| Document Libraries | 36 |
| The Document Library Programming Model | 36 |
| Columns | 39 |
| The Column Programming Model | 41 |
| Content Types | 44 |
| The Content Type Programming Model | 45 |
| Managed Metadata | 47 |

| | |
|--|-----------|
| Administering Managed Metadata | 48 |
| Creating a Global Term Set | 49 |
| Using a Term Set in a Column | 50 |
| The Managed Metadata Programming Model | 50 |
| The Managed Metadata Service | 55 |
| Content Type Syndication | 56 |
| The Content Type Syndication Programming Model | 57 |
| Management of Managed Metadata Service Applications | 58 |
| Location-Based Metadata Defaults | 60 |
| Configuring Location-Based Metadata Defaults | 60 |
| The Location-Based Metadata Defaults Programming Model | 61 |
| Metadata Navigation | 62 |
| Configuring Metadata Navigation | 63 |
| Using Metadata Navigation | 63 |
| The Managed Metadata Navigation Programming Model | 65 |
| The Document ID Service | 66 |
| The Document ID Programming Model | 68 |
| Document Sets | 69 |
| Implementing Document Sets | 69 |
| Creating Custom Document Sets | 70 |
| Using Document Sets | 70 |
| The Document Set Programming Model | 71 |
| Document Control | 73 |
| Security | 73 |
| Managing Users and Groups | 76 |
| The Security Programming Model | 76 |
| Check-In/Check-Out | 79 |
| How to Check Out a Document | 79 |
| Programmatically Checking Out a Document | 79 |
| Versioning | 81 |
| How to Configure Versioning | 81 |
| Version History | 81 |
| Programmatically Interacting with Version History | 82 |
| The Audit Trail | 83 |
| The Content Organizer | 84 |
| Summary | 85 |
| CHAPTER 4: WORKFLOW | 87 |
| Workflow and ECM | 87 |
| Windows Workflow Foundation | 88 |

| | |
|---|------------|
| WF Concepts | 88 |
| Activities | 88 |
| Workflow Modes | 90 |
| Persistence | 90 |
| The Role of Workflow in SharePoint | 91 |
| Workflow Scopes | 92 |
| Item | 92 |
| Site | 93 |
| Workflow Phases | 93 |
| Association | 93 |
| Initiation | 93 |
| Execution | 93 |
| Authoring and Workflow Types | 94 |
| Out-of-the-Box Workflows | 94 |
| The Approval Workflow | 95 |
| Declarative Workflows | 99 |
| Visio | 99 |
| SharePoint Designer Workflows | 105 |
| Visual Studio Workflows | 114 |
| Improvements | 115 |
| Creating a Workflow in Visual Studio: An Exercise | 116 |
| InfoPath | 125 |
| Out-of-the-Box Workflows | 125 |
| SharePoint Designer Workflows | 125 |
| Visual Studio | 126 |
| Pluggable Workflow Services | 126 |
| Why You Need Workflow Services | 126 |
| Authoring Custom Workflow Services | 127 |
| Workflow Event Receivers | 130 |
| Summary | 131 |
| CHAPTER 5: COLLABORATION | 133 |
| ECM and Collaboration | 134 |
| SharePoint Is Collaboration | 134 |
| Social Tagging | 134 |
| Tags | 135 |
| How to Create Tags | 135 |
| Tag Cloud | 137 |
| Notes | 137 |
| How to Create Notes | 138 |

| | |
|--|------------|
| Ratings | 139 |
| Enabling Ratings for a Document Library or List | 140 |
| How to Rate an Item | 140 |
| Bookmarklets | 141 |
| Registering the Tags and Notes Bookmarklet | 142 |
| Creating Tags and Notes Using Bookmarklets | 143 |
| Privacy and Security Concerns | 143 |
| Tagging Programming Model | 144 |
| Working with Tags Programmatically | 145 |
| Working with Notes Programmatically | 147 |
| Working with Ratings Programmatically | 149 |
| My Sites | 151 |
| My Profile | 151 |
| My Content | 153 |
| My Newsfeed | 153 |
| My Sites Architecture | 154 |
| Configuring My Sites | 154 |
| Configuring My Site Settings in the User Profile Service Application | 156 |
| Enabling the Activity Feed Timer Job | 157 |
| User Profiles | 157 |
| User Profile Policies | 158 |
| User Profile Programming Model | 159 |
| Working with a User Profile Programmatically | 160 |
| User Profile Service Application | 165 |
| People | 166 |
| Organizations | 166 |
| My Site Settings | 167 |
| Synchronization | 168 |
| Enterprise Wikis | 168 |
| Blogs | 169 |
| Microsoft Office Integration | 170 |
| SharePoint Workspace | 170 |
| Outlook Integration | 171 |
| Summary | 172 |
| CHAPTER 6: SEARCH | 173 |
| Introduction | 173 |
| Retrieval: The Key to User Adoption | 174 |
| The Corpus Profile | 176 |
| What Types of Documents Will Be Crawled? | 176 |

| | |
|--|------------|
| Is an IFilter Available for Full-text Crawling All Document Types? | 176 |
| How Many of Each Document Type Will Be Crawled? | 177 |
| What Is the Average File Size By Document Type? | 177 |
| How Often Are Existing Documents Changed? | 177 |
| How Much New Content Will Be Added During a Specific Period of Time? | 178 |
| Impact of the Corpus Profile | 178 |
| Search Solutions | 178 |
| SharePoint Server 2010 Enterprise Search | 180 |
| Topology Components | 180 |
| Configuration Components | 184 |
| The Search Center | 189 |
| Calling the Search API | 203 |
| FAST Search for SharePoint 2010 | 203 |
| Functional Overview | 203 |
| Index and Query Processing Path | 205 |
| Search Architectures for SharePoint ECM | 206 |
| Sample Architectures | 207 |
| 3-Million-Item Corpus | 208 |
| 10-Million-Item Corpus | 208 |
| 40-Million-Item Corpus | 208 |
| 100-Million-Item Corpus | 210 |
| 500 Million Documents | 211 |
| The Impact of Virtualization | 211 |
| Tuning Search Performance | 211 |
| Health Monitoring | 212 |
| Performance Monitoring | 212 |
| Improving Crawl Performance | 213 |
| Improving Query Performance | 213 |
| Summary | 214 |
| CHAPTER 7: WEB CONTENT MANAGEMENT | 215 |
| WCM Overview | 215 |
| Improvements in 2010 | 216 |
| Authoring | 216 |
| AJAX | 216 |
| Accessibility | 216 |
| Markup Standards | 217 |
| Content Query Web Part | 217 |
| Cross-browser Support | 217 |

| | |
|--|------------|
| Rich Media | 217 |
| Metadata | 217 |
| Spectrum of WCM in 2010 | 218 |
| The SharePoint Server Publishing Infrastructure | 218 |
| Templates | 218 |
| Features | 219 |
| Security | 221 |
| Approve Permission Level | 221 |
| Manage Hierarchy Permission Level | 222 |
| Restricted Read Permission Level | 222 |
| Groups | 222 |
| Content Types | 223 |
| “Content” Content Types | 223 |
| Infrastructural Content Types | 224 |
| Site Content | 225 |
| The Anatomy of a Page | 226 |
| Master Pages | 226 |
| Page Layouts | 227 |
| An Exercise with Taxonomy and Layouts | 227 |
| Metadata | 232 |
| Content Query Web Part | 233 |
| Web Part Options | 233 |
| Query Options | 233 |
| Presentation | 234 |
| The Content Authoring Process | 235 |
| Authoring Web Content | 235 |
| Using the Content Organizer | 238 |
| Content Deployment | 238 |
| Workflow | 239 |
| Enterprise Wikis | 239 |
| Other Major Considerations | 240 |
| Branding | 240 |
| Navigation and Search | 240 |
| Targeting Global Users | 241 |
| Reporting and Analytics | 241 |
| Summary | 242 |
| CHAPTER 8: RECORDS MANAGEMENT | 243 |
| What Is Records Management? | 244 |
| Why Records Management Is Important | 244 |

| | |
|---|------------|
| Microsoft SharePoint as a Records Management System | 245 |
| Records Management Planning | 245 |
| Identifying Roles | 245 |
| Analyzing Content | 246 |
| Developing a File Plan | 247 |
| Designing a Solution | 247 |
| Compliance and SharePoint | 248 |
| Managing Records | 249 |
| Recordization | 250 |
| In-Place Records Management | 250 |
| Records Center | 253 |
| Content Organizer | 255 |
| Workflow in Recordization | 258 |
| Programming Model for Recordization | 259 |
| Information Management Policy | 263 |
| Configuring Information Management Policy | 263 |
| Exporting and Importing Policy Settings | 266 |
| Programming Model for Information Management Policy | 267 |
| Retention | 267 |
| Creating Retention Schedules | 267 |
| Programmatically Creating Retention Schedules | 268 |
| Auditing | 270 |
| Configuring Auditing | 270 |
| Reporting | 271 |
| Audit Reports | 271 |
| File Plan Report | 272 |
| eDiscovery | 272 |
| Summary | 273 |
| CHAPTER 9: DIGITAL ASSET MANAGEMENT | 275 |
| <hr/> | |
| SharePoint Server 2010 Digital Asset Management Components | 276 |
| The Asset Library | 276 |
| Digital Asset Columns | 276 |
| Digital Asset Content Types | 277 |
| Media and Image Web Parts | 278 |
| Media Web Part and Field Control | 278 |
| Picture Library Slideshow Web Part | 280 |
| Image Viewer Web Part and Field Control | 280 |
| Content Query Web Part | 280 |
| Digital Asset Management Solution Scenarios | 280 |

| | |
|--|------------|
| Marketing and Brand Management | 281 |
| Media Development Project | 282 |
| Online Training Center | 283 |
| Audio or Video Podcasting | 284 |
| Media Resource Library | 284 |
| Taxonomy Considerations | 284 |
| Storage Considerations | 285 |
| Managing Content Database Size | 285 |
| Remote BLOB Storage | 286 |
| Maximum Upload Size | 286 |
| Performance Optimization | 287 |
| BLOB Caching | 287 |
| Bit Rate Throttling | 289 |
| Summary | 292 |
| CHAPTER 10: DOCUMENT IMAGING | 293 |
| <hr/> | |
| What Is Document Imaging? | 294 |
| SharePoint as a Document Imaging Platform | 295 |
| Setting Up the Scenario | 295 |
| Solution Data Flow Diagram | 295 |
| Model-View-ViewModel Primer | 296 |
| Creating a Simple WPF Capture Application | 298 |
| Architecture and Design | 299 |
| Implementation | 299 |
| Building the MVVM Infrastructure | 299 |
| Building the Target Dialog | 300 |
| Building the Main Window | 301 |
| Deployment | 322 |
| Creating a Simple Silverlight Viewer Web Part | 322 |
| Architecture and Design | 323 |
| Implementation | 323 |
| Building the Image Loader | 323 |
| Building the Imaging Web Service | 324 |
| Building the Imaging HTTP Handler | 325 |
| Building the Viewer Web Part | 325 |
| Making the Application Accessible from JavaScript | 326 |
| Deployment | 327 |
| Deploying the Imaging Services | 327 |
| Deploying the Viewer Application as a Web Part | 327 |

| | |
|---|------------|
| Setting Up the SharePoint Infrastructure | 338 |
| Configuring SharePoint Search | 338 |
| Creating the SharePoint Content Type | 339 |
| Creating the SharePoint Document Library | 340 |
| Creating the SharePoint Web Part Page | 340 |
| Setting Up the SharePoint Web Part Page | 341 |
| Customizing the Advanced Search Box Web Part | 341 |
| Customizing the Search Core Results Web Part | 343 |
| The Solution from End to End | 355 |
| Summary | 355 |

CHAPTER 11: ELECTRONIC FORMS WITH INFOPATH **357**

| | |
|--|------------|
| Electronic Forms Overview | 357 |
| Is It a Form or an Application? | 358 |
| InfoPath Overview | 358 |
| What's New in 2010 | 360 |
| More InfoPath Fundamentals | 360 |
| Forms Services | 360 |
| Deploying Forms | 361 |
| Templates and Form Data | 362 |
| Rules | 364 |
| External Data | 368 |
| Custom Code | 370 |
| Publishing | 371 |
| Determining a Forms Strategy | 372 |
| Creating a Custom Form: An Exercise | 374 |
| Form Data and Layout | 374 |
| Form Rules | 376 |
| Form Submission | 376 |
| Publishing the Form | 379 |
| Summary | 380 |

CHAPTER 12: SCALABLE ECM ARCHITECTURE **381**

| | |
|---|------------|
| Storage Architecture, the Key to Performance | 381 |
| Performance Pitfalls | 382 |
| Too Few Disks in the Array | 382 |
| Shared SAN vs. DAS vs. NAS | 383 |
| Content Storage Size Factors | 384 |
| Database Storage and Capacity Planning | 385 |
| SQL Server Supporting Concepts | 386 |

| | |
|--|------------|
| TempDB | 390 |
| Log Files | 392 |
| Crawl Databases | 393 |
| Content Databases | 395 |
| Property Databases | 396 |
| Service Application Databases | 397 |
| Management Databases | 400 |
| Prioritizing Disk I/O | 400 |
| Index Partition Storage | 401 |
| Storage Tuning and Optimization | 401 |
| Storage Performance Monitoring | 401 |
| Database Data File Management | 402 |
| Remote BLOB Storage | 403 |
| When to Implement an RBS Solution | 405 |
| RBS Provider Options | 406 |
| Backup and Restore Considerations | 407 |
| SQL Server Licensing Considerations | 407 |
| SharePoint 2010 Scalable Topology Design | 408 |
| Knowing the Users, the Corpus, and the Processes | 408 |
| Farm Size Definitions | 409 |
| The Case for Additional Web Servers | 412 |
| The Case for Additional Application Servers | 412 |
| The Case for Additional SQL Servers | 412 |
| Scalable Taxonomy Factors | 413 |
| Content Organization and Scalable Taxonomy | 414 |
| An Exercise in Scalable Taxonomy Design | 415 |
| Content Database Size Supported Limits | 416 |
| Performance and Resource Throttling | 417 |
| Summary | 418 |

PART III: SHAREPOINT ECM SUPPORT CONCEPTS

CHAPTER 13: ECM FILE FORMATS 421

| | |
|--|------------|
| It's Alive — Your Document, That Is | 422 |
| Microsoft Office Formats | 422 |
| Microsoft Office Binary | 422 |
| Office Open XML | 423 |
| Viewing and Editing Microsoft Office | |
| Formats with Office Web Apps | 425 |
| Word Automation Services | 428 |
| Open Document Format | 437 |

| | |
|---|------------|
| Archive Formats | 437 |
| TIFF | 438 |
| OCR and iFilters | 438 |
| Markup | 442 |
| Development | 442 |
| PDF | 442 |
| OCR and iFilters | 442 |
| Markup | 443 |
| Development | 443 |
| Viewing and Editing | 443 |
| Living Document Conversion | 444 |
| PDF/A | 444 |
| Standardization | 445 |
| OCR and iFilters | 445 |
| Creating, Viewing, and Editing | 446 |
| XPS (Open XML Paper Specification) | 446 |
| OCR and iFilters | 447 |
| Markup | 449 |
| Development | 449 |
| Summary | 450 |
| CHAPTER 14: THE SHAREPOINT ECM ECOSYSTEM | 451 |
| <hr/> | |
| The Microsoft Partner Ecosystem | 451 |
| Becoming a Partner | 452 |
| ISV/Software Competency | 452 |
| The SharePoint Ecosystem | 453 |
| Technical Community | 453 |
| ISV Solutions | 454 |
| ABBYY | 455 |
| AvePoint | 458 |
| GimmelSoft | 460 |
| KnowledgeLake | 462 |
| Nintex | 465 |
| Summary | 467 |
| CHAPTER 15: GUIDANCE FOR SUCCESSFUL ECM PROJECTS | 469 |
| <hr/> | |
| Migrating to SharePoint 2010 | 470 |
| Identifying Content for Migration | 470 |
| Extracting Content from the Source System | 470 |
| File Shares | 470 |

| | |
|---|------------|
| Internally Developed Document Management Solutions | 471 |
| Other Legacy Document Management Solutions | 472 |
| Preparing Content for Importing | 472 |
| Setting the Content Type | 472 |
| Metadata Merge | 472 |
| Controlling the Import Process | 473 |
| General Metadata Cleanup | 473 |
| Importing Content into SharePoint | 473 |
| Protocols for Importing Content | 474 |
| Web Services | 474 |
| SharePoint Server Object Model | 477 |
| FrontPage Remote Procedure Calls (FPRPC) | 479 |
| Protocols for Updating SharePoint Content | 480 |
| SharePoint Server Object Model | 480 |
| SharePoint Client Object Model | 483 |
| Mapping Legacy ECM Features to SharePoint Solutions | 486 |
| Document Versions | 487 |
| Metadata-based Security | 487 |
| Document Protection and Rights Management | 488 |
| Annotations and Redaction | 488 |
| Search | 488 |
| Scanning and Indexing | 488 |
| Records Retention and Disposition | 489 |
| Workflow | 489 |
| Avoiding the Pitfalls | 489 |
| Capacity Planning | 489 |
| Illegal Characters in Metadata | 489 |
| Missing Required Data | 490 |
| Content Database Transaction Log Growth | 490 |
| Managing Upload File Size Restrictions | 490 |
| Upgrading a SharePoint ECM Farm to SharePoint 2010 | 491 |
| Know Your Farm | 491 |
| SharePoint Portal Server 2003 and WSS v2.0 | 491 |
| Microsoft Office SharePoint Server 2007 and WSS v3.0 | 492 |
| Imaging or Archive-Only Farm with No Customization | 492 |
| Collaboration Farm with Customizations | 492 |
| Collaboration Farm with Large Imaging or Archive Site Collections | 492 |
| Summary | 493 |
| INDEX | 495 |

INTRODUCTION

IN 2003 MICROSOFT RELEASED Windows SharePoint Services 2.0 and SharePoint Portal Server 2003, making their first true move into enterprise content management (ECM). It may not be a stretch to say they also created an entirely new technology space: collaborative document management. Seven years later, Microsoft launched the fourth version of SharePoint Services, now known as SharePoint Foundation (MSF), and the new server product is now called Microsoft SharePoint Server (MSS). These releases included new and refined features specifically targeted at the document management needs of an organization.

The rapid adoption of these ECM features inspired this book, which covers many major topics of this sophisticated platform. Most of the content comes from experience either building products on SharePoint 2010 or implementing them on customer sites. We hope that this cumulative experience will help others who are trying to create and meet the challenges of their own document management processes.

WHO THIS BOOK IS FOR

This book is for anyone who is currently using or planning to use the ECM features in SharePoint 2010. Regardless of the role you are providing within your SharePoint deployment, the chapters contain information that will enable you to understand these features and how they can help you make better decisions. We recommend starting with the first two chapters to get a basic understanding of ECM, the specific ECM topics covered in the book, and an overview of the SharePoint 2010 platform.

If you are a systems architect responsible for ECM features, you should find each chapter in the book helpful but you may want to skip ahead and read Chapter 6, “Search,” and Chapter 12, “Scalable ECM Architecture.” If you are a developer, Chapters 3, “Document Management,” 4, “Workflow,” and 8, “Records Management,” provide various coding examples. If you are a project manager, you should at minimum skim all the middle chapters on ECM features, and pay specific attention to Chapter 15, “Guidance for Successful ECM Projects.” Even if you are a business decision maker and won’t be getting your hands dirty with the design or implementation, reading enough of each chapter to understand what is possible will help you make better choices regarding your SharePoint ECM deployment.

WHAT THIS BOOK COVERS

Most of the material in this book applies specifically to SharePoint 2010. Some of the ECM concepts, topics, and features, however, may also exist in previous releases.

As you'll discover in Chapter 1, "What Is Enterprise Content Management?", ECM includes not only concepts and strategies, but also the tools necessary to facilitate them. Because SharePoint 2010 encompasses a broad range of ECM topics, the book contains both big-picture explanations of essential concepts and also detailed information and hands-on exercises that demonstrate how to enable these tools. Beyond ECM, you'll also learn all about SharePoint 2010's web content management (WCM) features.

This book addresses the gamut of ECM and WCM features. It covers basics like search and collaboration, and it examines workflow, scalability, compliance, master pages, layouts, and managing documents, records, web content, and other digital assets. It also delves into InfoPath, electronic forms, and document imaging.

HOW THIS BOOK IS STRUCTURED

This book is organized in three parts. Part I, "Introduction to Enterprise Content Management," provides an overview of ECM, which includes a history of both ECM and SharePoint, as well as an overview of the SharePoint ECM feature set. The chapters in this part of the book present both basic background information and a context that will be useful as you read the chapters in second part of the book.

Part II, "Pillars of SharePoint ECM," describes in detail the ECM features of SharePoint 2010. This part of the book explores document management, workflow, collaboration, search, web content management, records management, digital asset management, document imaging, electronic forms with InfoPath, and scalable ECM architecture. You can use these chapters as a reference when you later deploy your own ECM technologies with SharePoint 2010.

Part III, "SharePoint ECM Support Concepts," covers ECM document formats, explores the Microsoft — and, more specifically, SharePoint — ecosystem, and offers guidance for implementing successful ECM projects.

WHAT YOU NEED TO USE THIS BOOK

Although you could read this entire book without actually having a SharePoint 2010 installation, it would be best to have an installation of Microsoft SharePoint Server (MSS) that you can hack around on without having to worry about bringing down an administrator's production system! In order to get the most out of the features covered in this book, we recommend using the enterprise version of Microsoft SharePoint Server. Although MSS is preferred, many of the concepts and examples can be applied to Microsoft SharePoint Foundation and SharePoint Online.

If you plan on coding along with the examples provided in this book, we also recommend you have a copy of Visual Studio 2010. You'll want to install Visual Studio on the same Windows 2008

Server as SharePoint, or you can install both SharePoint and Visual Studio on Windows 7 x64 (also Windows Vista x64 Service Pack1).

Lastly, we recommend that you virtualize your installation if possible in order to easily roll back changes, as you are sure to make some mistakes along your SharePoint ECM journey.

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.



Boxes with a warning icon like this one hold important, not-to-be-forgotten information that is directly relevant to the surrounding text.



The pencil icon indicates notes, tips, hints, tricks, or asides to the current discussion.

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 filenames, 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 website is highlighted by the following icon:



Available for
download on
Wrox.com

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 978-0-470-58465-1.

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.

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, such as a spelling mistake or a 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 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 e-mail 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 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 e-mail 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.

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 e-mailed 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.

PART I

Introduction to Enterprise Content Management

- ▶ **CHAPTER 1:** What Is Enterprise Content Management?
- ▶ **CHAPTER 2:** The SharePoint 2010 Platform

1

What Is Enterprise Content Management?

WHAT'S IN THIS CHAPTER?

- Defining ECM as used by this book
- Gaining a historical perspective of ECM
- Defining the components of an ECM system

Considering that this is a book both by and for architects and developers, devoting an entire chapter to talking about the enterprise content management (ECM) industry and trying to define it, rather than just jumping into the bits and bytes that you probably bought the book for, might seem strange. However, by introducing ECM as part of an industry, instead of describing how the SharePoint world perceives it, we hope to provide a perspective that wouldn't otherwise be possible if you make your living inside the SharePoint ecosystem.

ECM, within or outside of the SharePoint world, seems to be a much-abused abbreviation used to describe a variety of different technologies. Of course, people often adopt new or existing terms, applying their own twist to the original meaning, and this is certainly the case with ECM. The difficult part is determining which meaning is actually correct. Sometimes even the words representing the initials are changed. For example, in the halls of our own company, sometimes “electronic” is used instead of “enterprise.” In other cases, ECM is confused with specific technologies that are part of it, such as DMS (Document Management System), IMS (Image Management System) or WCM (Web Content Management).

Clearly, ECM means a lot of different things to a variety of people. There is no doubt that some readers of this book will think something is missing from the definition, while other readers will find something included that does not fall into their own definition. That being said, this chapter introduces ECM not necessarily from a SharePoint perspective, but from a historical perspective; then it provides an overview of the components of an ECM system.

You can skip this information, but we believe it is important to clarify the problems we are trying to solve, rather than just write code based on our own assumptions.

INTRODUCTION TO ECM

The “content” aspect of *enterprise content management* can refer to all kinds of sources, including electronic documents, scanned images, e-mail, and web pages.

This book uses the definition of ECM from the Association for Information and Image Management (AIIM) International, which can be found on their website at www.aiim.org:

Enterprise Content Management (ECM) is the strategies, methods, and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists.

As this definition states, ECM is not really a noun. That is, it's not something as simple as an e-mail system or a device like a scanner, but rather an entire industry for capturing and managing just about any type of content. The key to the definition is that this content is related to organizational processes, which discounts information that is simply created but never used.

Moreover, ECM is meaningless without the tools that accompany it. You might say that the tools that solve your content problem also define it. This idea is explored in the next section, and hopefully clarified by a short history of a few of the technologies involved.

A HISTORICAL PERSPECTIVE

Although the term ECM is relatively new, many of the components that make up an ECM system started appearing in the 1970s. The world of information systems was vastly different 30–40 years ago. The Internet as we know it did not exist, the cost to store data was astronomical compared to today, server processing power was a mere fraction of what it is today, and desktop computers didn't even exist.

The history of ECM can be traced back to several technologies that formed that first stored and managed electronic content: document imaging, electronic document management, computer output to laser disc (COLD), and of course workflow, which formed the business processes.

Document Imaging

As evidenced by the first systems to take the management and processing of documents seriously, paper was one of the first drivers. These systems were often referred to as *electronic document management* or *document imaging* systems. By scanning paper and storing it as electronic documents, organizations found a quick return on investment in several ways:

- It reduced the square footage needed to store paper.
- It resulted in faster execution of paper-based processes by electronic routing.

- It eliminated the time it took to reproduce lost documents.
- It reduced overhead because paper documents could be retrieved electronically.

In addition to a reduction in manpower, there were other benefits to storing paper electronically — namely, security and risk benefits, which preceded regulations such as Health Insurance Portability and Accountability Act (HIPAA) and Sarbanes-Oxley by more than a decade. Some of these included the following:

- Password protection of documents
- Enterprise security restraint brought about by secure networks
- Management of records needed for legal holds
- Management of the document life cycle, such as retention periods
- Audit information about the document life cycle and requests about the document

The first document imaging systems for commercial consumption became available in the early 1980s, and they quickly started to replace the previous technology for removing paper from organizations, which was microfiche. Billions of documents were stored on microfiche, but indexes and location data were often stored in databases. Conversions of these systems to document imaging are surely still being handled today.

The ability to scan existing paper documents in order to create electronic documents, as discussed in the next section, led to the vision of a “paperless office,” a commonly used phrase by the end of the century. Of course, this lofty and often pursued goal of a paperless office has yet to materialize, and paper is still the original driver behind many business processes. As shown in Figure 1-1, focusing on paper is a good starting point to quickly begin realizing the benefits of an ECM system.

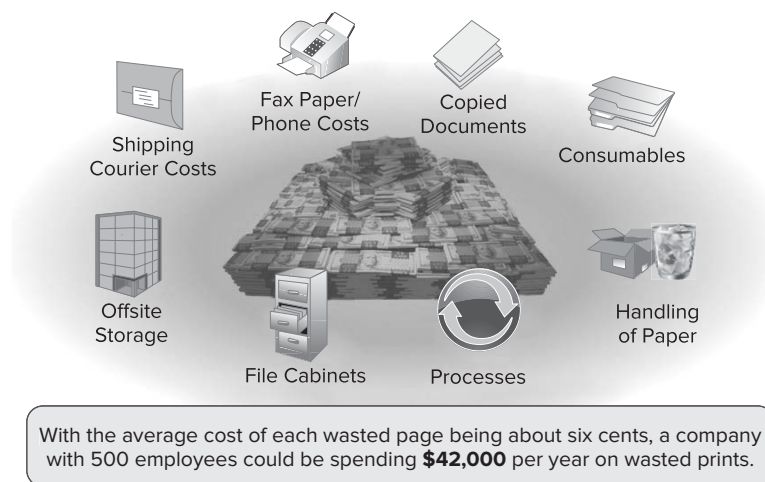


FIGURE 1-1

Electronic Documents

The invention of computer-based word processors (in the 1970s) created the need for a way to store and quickly retrieve these documents. Electronic documents share similarities with document imaging systems, yet they are unique in that they are typically dynamic; that is, they often require ongoing modification, whereas scanning paper was typically performed for archiving purposes.

The first electronic documents were created through word processing software, driven in the late 1970s by WordStar and Word Perfect. Although the former has been abandoned, WordPerfect still exists today and is part of an office suite from Corel.

Soon after personal computers and electronic word processors hit the market, electronic spreadsheets became available, beginning with VisiCalc, followed by Lotus 1-2-3 and eventually Microsoft Excel. Spreadsheet documents are now as commonplace as word processing documents.

Today, electronic documents exist in countless types and formats, ranging from simple ASCII text files to complex binary structures.

COLD/Enterprise Report Management

The widespread use of computers, beginning with the large mainframes, resulted in an unprecedented use of paper. Early computers all over the world started producing reports, typically on what is known as *green bar paper*. As the need for information from both mainframe and mini computers grew, so did the need for computer-generated reports. Necessary at first because structured methods for viewing data electronically did not exist, this excessive use of paper continued to plague organizations into the 1990s and even into this millennium.

Out of this problem grew a solution coined *computer output to laser disc (COLD)*. Instead of generating paper, these reports could be handled in a type of electronic content management system, typically storing the ASCII data and rendering it onto monitors. These systems enabled not only search and retrieval of the reports, but the addition of annotations, and of course printing of the documents when using a monitor is not adequate.

The term COLD was eventually replaced by *enterprise report management* when magnetic storage replaced the early optical storage systems.

Business Process Management/Workflow

Storing content electronically was a great step forward, but moving content to a digital medium quickly put the content in front of the right user at the right time.

The first *business process management (BPM)* systems, launched in the mid 1980s and called *workflow systems*, were created by the same companies that brought document imaging systems to market. These early systems were far less complex than the BPM systems used today, however, as they primarily enabled content to be put into queues to be processed by the same workers that processed the paper.

It was almost 10 years later when the first graphical components became available for creating complex workflow maps. This was the beginning of BPM as we know it today, which enables organizations to create, store, and modify business processes.

ECM COMPONENTS

In order to understand ECM, it is necessary to understand the common components that comprise such as system. The following sections provide a brief overview of these components, which, like the definition for ECM provided earlier, have been defined by AIIM.

Capture

Capture is the process of gathering the data, regardless of the source, including classification, indexing (sometimes called tagging), and rendition. These tasks are required before storage is possible, in order to understand what type of content is being managed, which keywords will be used to search for the content, and to ensure that the content is in a form that can be easily retrieved and viewed later.

Paper

Paper is still the primary driver of ECM. The reason is simple: Because of the volume of paper that most companies need to handle, efficiently managing that paper can provide the greatest return on investment. Figure 1-2 gives you some idea of the cost of paper in an enterprise.

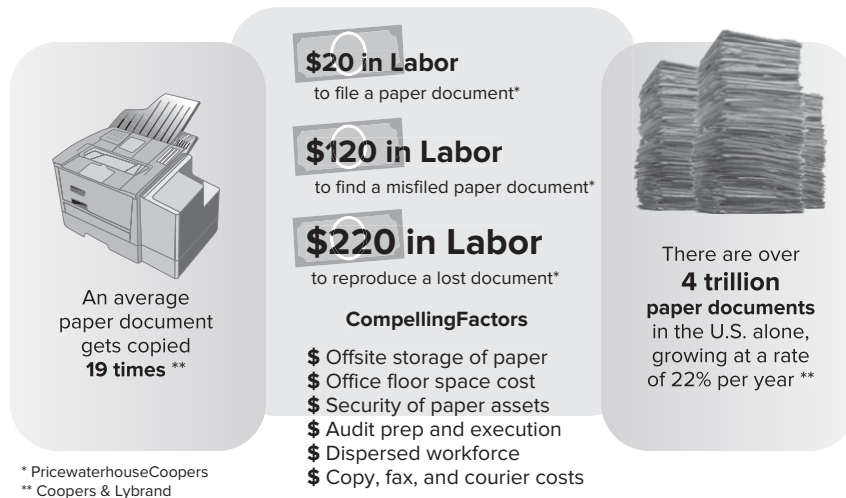


FIGURE 1-2

Paper capture is primarily done by using document scanners specifically built for the purpose. These scanners can capture both small and large batches of paper. Paper documents are typically divided into three categories: structured, semi-structured, and unstructured.

Structured documents typically represent forms such as tax documents, applications, or other preexisting forms. Because they are always the same, these documents are usually the easiest to automatically extract information from. Technology such as Zonal OCR can be easily applied to structured forms because the key data always exists in the same place.

Semi-structured documents are similar to structured documents, but they are different enough that structured zones can no longer be used to extract data. A common type of semi-structured document is the invoice. Most invoices are similar enough to be recognized as such, but each company designs its invoice with enough nuances to distinguish it from others, so these documents require either manual manipulation of the data or more intelligent automation.

Unstructured content represents the majority of the information in the average corporate enterprise. Almost all human correspondence is a good example of unstructured content. Although this content ends up being stored as the same content type in each company, on a per-page basis they have very little in common. Manual indexing is typically required for this type of data.

Developing applications to handle all the different types of paper that may come into an enterprise is a daunting task. Fortunately, toolkits are available to drive most document scanners on the market today, and these are normally compliant with either (or both) the TWAIN and ISIS driver standards. Although the physical process of scanning a piece of paper is simple, building a good process for either manually entering information or automatically extracting it is difficult, and probably not cost effective for custom applications.

Office Documents

Because this book is about ECM and (Microsoft) SharePoint, it focuses on the most common type of electronic documents that need to be managed: (Microsoft) Office documents. These documents are created using word processing software, spreadsheet software, presentation software, and so on. These documents are typically pre-classified on creation, as they frequently start from a template; therefore, extracting data for searching is often overlooked. Although each word from the document can be added to search indexers, it makes more sense to use specific keywords to identify the document. Sometimes pre-identified form fields are used, but often the most important data is keyed by hand before sending the document to storage.

E-mail

Capturing e-mails into an ECM system is becoming an increasingly common scenario. E-mail is often used to drive a business process, as it is becoming an acceptable form of correspondence in most organizations. Although some data can be indexed automatically, such as the sender, receiver, and subject, as shown in Figure 1-3, it can be difficult to extract the useful information contained in the body of the message, and manual intervention is usually required.

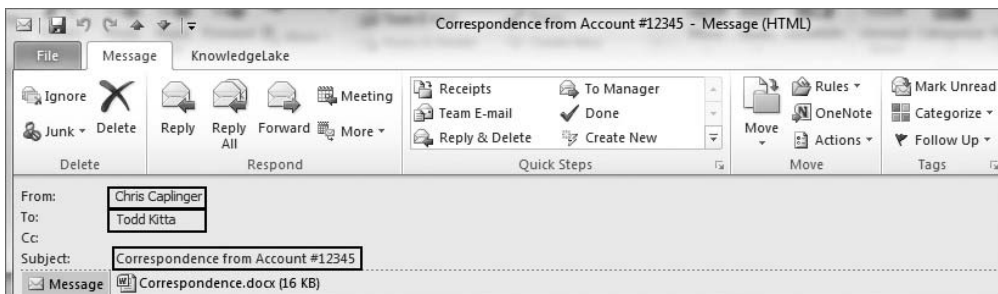


FIGURE 1-3

E-mail attachments are often more important than the e-mail message itself. Although Microsoft Outlook and other e-mail clients are improving ECM integration, extracting the attachment and exporting it to an ECM system usually requires specialized software in order to properly tag the content with searchable data.

Reports

As mentioned earlier, technology originally known as computer output to laser disc (COLD) and later as enterprise report management enables computer reports to be parsed into electronic files. Classifying and indexing these documents is typically automatic because they are in a form that is very structured.

Like the handling of paper, building a system for handling enterprise reports is most likely not cost effective when you compare the needed functionality versus the difficulty of obtaining it. Consider being able to read EDI or other electronic streams and extract the necessary data from them. Also, although data storage may not be an issue, you must consider how it will be displayed to users in a readable format.

Electronic Forms

It was initially believed that the goal of the paperless office would be achieved with the help of electronic forms. After all, the form templates provided in many applications, such as InfoPath, enable users to fill out preexisting fields and submit them directly to a content management system. With the type of content already known and the data being put into electronic form as it was gathered, it stood to reason that the paper forms could gradually disappear. However, human habits die hard. It may take another generation of computer users, who have been raised from birth with computers and who use them for everything from social networking to bill-paying, to realize the truly paperless office.

Other Sources

Although the most common types of capture have been identified here, there are many other possible data sources and data types. Multimedia, XML, and EDI are other well-known data formats that can arrive from many different sources. Indeed, just about any type of data can be consumed in an ECM system.

Store and Preserve

The store and preserve components of an ECM system are very similar; storage traditionally pertains to the temporary location of content, whereas preservation refers to long-term storage. In the past these were separated because online storage was costly. Content was usually stored in the temporary location only during its active life cycle, when it was frequently accessed as part of the business process. Once the active life cycle was complete, content would move to long-term storage, known as *offline storage* or *nearline storage*, which was much less expensive. The term “offline” reflects the fact that the content wasn’t accessible without human intervention; the term “nearline” typically refers to optical discs that were brought online automatically, such as in the case of a jukebox. The following sections describe both the software and hardware components of storage and preservation of content.

Software

The software required to store and preserve data varies widely according to the needs of an organization. Sometimes the software required is part of the operating system, and other times there are specific services used to store, deliver, and allow transactions on the content itself.

Repositories

Repositories refer to the actual software that defines the taxonomy of a storage system. In the case of SharePoint, the repository refers to the site, libraries, and content types that define the taxonomy of the system.

The term *library services* is also used to describe not just the repository but the services built around repositories.

Databases

Databases have been the standard for storing transactional data for decades; however, they were typically not a popular place for content storage until SharePoint began to use SQL Server for storage. As discussed later in this book, the rapidly growing number of SharePoint databases created for storage led third-party vendors to find new ways to move content into other storage mechanisms.

Hierarchical Storage Management

Hierarchical storage management (HSM) has dwindled in recent years due to the lower cost of online storage. The purpose of an HSM is to automatically transfer content between online and offline storage. These systems have enough intelligence to bring content offline when it was no longer actively accessed and back online when requested.

Often, the storage of content during its process life cycle and its permanent archival is the same; they are sometimes separated when the cost of having a large amount of documents for instant retrieval is too high.

File Systems

File systems are the most common location for content storage, but they are not the most common location in ECM systems. In fact, they are probably the worst possible place your content could be stored, especially if your system lacks a good backup strategy.

Hardware and Media Technologies

Software would not be very valuable if content were limited to computer memory. Very simplistic to advanced hardware and media technologies are used in combination with software to provide online, nearline, or offline storage locations for content.

Magnetic/RAID

At one time very expensive, the cost per gigabyte of magnetic storage has plummeted over the past 10 years. Storing content online has historically used magnetic disk; but now that the cost is so much lower, many systems continue to use magnetic storage for long-term preservation as well.

Redundant array of independent disks (RAID) allows several disks to be used together to add redundancy, which enables systems using magnetic technologies be more fault tolerant.

Optical

In the earliest days of ECM, specifically document imaging systems, large optical disks provided the least expensive long-term storage method for data. Although magnetic storage has become much less expensive, smaller optical media, such as DVDs, still offer a valid and often used method for permanent storage of content.

The obvious disadvantage of optical storage is that keeping it online is next to impossible; therefore, it is necessary to use nearline storage methods, using jukebox-type devices that bring the disks online as needed.

SAN/NAS

With the reduced cost of magnetic storage, both the *storage area network (SAN)* and *network area storage (NAS)* have become very popular in the world of ECM. While both can work for file-level and block-level data, a SAN is often optimized for the block-level access required by software such as databases. A SAN is typically several disk arrays that connect through fiber channels and that appear as local disks to a server. They all allow the fastest possible read times in order to meet the needs of relational database systems. NAS devices are detached storage arrays on less expensive hardware than a SAN, which makes them more cost effective in scenarios such as document archival. A NAS also works well with file-level access, making it a great solution for server-based file systems.

Cloud

- The cloud is the likely future location for content storage and quite possibly the entire ECM landscape. Cloud technologies combine both software and hardware technologies to provide centralized storage over the Internet (and sometimes intranets). One advantage of cloud technologies is that cloud data centers purchase hardware in bulk, drastically reducing the costs to customers.
- Cloud data centers are environmentally conservative, offering greatly reduced energy consumption.
- Software and hardware are optimized to work together.
- Commercial cloud providers offer a level of fault tolerance not otherwise possible for the same cost.

Of course, there are disadvantages too. Some are technical, such as the latency for users who need rapid access to large files. Others are more business related, such as security issues and loss of data control.

Cloud storage is rapidly evolving. In terms of total services provided, the current leaders seem to be Amazon S3 and Microsoft Azure. By the time this book is published, however, many other competitors will emerge to offer storage and management capabilities in the cloud.

Management Components

All the components described in the following subsections have their own chapters in this book, so here you are just briefly introduced to each of these management components of ECM.

Document Management

Document management can be a complete solution itself for many organizations, or it may represent a component of a larger platform. Chapter 4 focuses on the complete solution; but from a component standpoint, document management is an underlying technology for document imaging, records management, and web content management.

As a component, the most important functionality provided is the taxonomy. In terms of SharePoint, this means libraries, content types, columns, and even perhaps higher-level items like site collections and sites. When users refer to the taxonomy, they are typically referring to the assembly of these features within their organization.

Other functionality provided by document management systems are security, check in/out, versioning, and of course document retrieval. Document management systems can contain many more features, but these are the major components related specifically to documents.

Web Content Management

Although many users argue that web content management (WCM) is the primary driver of ECM, this component came along much later than the others mentioned earlier, with perhaps the exception of collaboration. WCM is an underutilized technology that can greatly reduce the costs of managing websites by introducing document management, publishing, and even workflow to the process.

WCM, unlike other components such as document management and BPM, is a technology used for the specific purpose of managing website authoring.

Business Process Management and Workflow

Business process management (BPM) and workflow enable electronic content to be delivered to the right people at the right time — and much more inexpensively than managing the content through manual processes.

These two terms are often used interchangeably but one is really part of the other. Workflow, in the context of ECM, refers to the process of automating a business process, while BPM is the process of not just automating but redefining, improving, and managing the business process. Workflow is part of BPM. The important thing to know is that BPM, or just workflow, can not only accelerate the return on investment of an ECM system, but also be the primary reason to adopt an ECM system.

Records Management

Records management, covered in detail in Chapter 8, is similar to document management and document imaging but has some key functional requirements that differentiate it. Although records

management often refers to both paper and digital records, this book focuses on records management from a digital perspective.

Records management is defined by ISO 15489-1:2001 as “the field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use, and disposition of records, including processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records.”

Records, in the context of records management, can then be understood as any evidence and information about business activities and transactions. The best source for more information on records management is ARMA International. Their public website is www.arma.org.

Collaboration

Collaboration is a relatively new management component of ECM, and it probably wouldn't be even considered today if not for SharePoint. The ability to create online content and have a group of users act on that content has been around since forums first started appearing on Internet newsgroups. These newsgroups thrived throughout the late 1980s and 1990s; and although they still exist, they have given way to website forums, blogs, and wikis. Collaborative information is the ultimate unstructured content.

Social Media

Social media, such as Twitter, YouTube, and LinkedIn, are not typically considered a part of ECM, but these popular sites have vast amounts of content, and it is managed.

Although organizations largely ignored social media in its early years, they are starting to understand the value of mining the information it contains, and they are looking for the best ways to manage it.

Document Collaboration

Document collaboration enables a set of users, typically authorized, to save documents to repositories, where they can be shared and manipulated. These documents are often a crucial part of the business process.

Although previous document management systems allowed some collaboration, SharePoint initially focused on collaboration, rather than ECM, until version 2 (SharePoint Portal Server and Windows SharePoint Services 2.0).

Delivery

The final component of ECM is delivery. Without a way to deliver the content, there is no reason to capture and store it in first place, and management would be impossible.

Search

The capability to search content is a key step to delivering information to users. How do users even know it exists unless it is easily identified?

In the earliest days of ECM, table indices on databases were used to find content with a structured taxonomy. Using this taxonomy to define the exact data needed made databases a great technology for searching content. This technology is still heavily used in ECM systems. Even SharePoint makes it possible with its new Metadata navigation capabilities on list column data as well as being able to make this column data available to the SharePoint and FAST search crawlers.

In the past decade or so, the industry has been switching to a more unstructured approach to searching. Whereas the first search systems for ECM were simply based on database tables and indices, now content is crawled by complex engines that can store seemingly unlimited amounts of data. These crawlers interrogate text and media on websites and in documents, as well as inside databases. Modern search technologies need to be able to aggregate this information for users while still being able to perform simple structured searching based on taxonomy.

Viewing

Of course, the viewing of content is a critical output point for an ECM system. Delivery of content is said to be completed when it is available to a viewer. The delivery method used varies according to the type of content and the purpose for which it is intended.

Document Readers

Document readers are used to display content, generally text, to users in an easily readable manner. Adobe Acrobat Reader is probably the most well-known example of a document reader. Although the term “reader” is being used, these applications often allow for much more than simply reading content and may also be the applications that were used to create the content to begin with. Applications such as Microsoft Word and Microsoft Excel are the best-known examples.

Web Browsers

Web browsers are rapidly becoming the most common delivery method for content, as much of the world’s unstructured information is now delivered within websites. Even most document viewers today have plug-ins for the most common browsers, or advanced viewers that are delivered directly within the browsing experience, as shown in Figure 1-4.

Others

Just as ECM systems can manage countless content formats, there are numerous ways to view that data. Although web browsers are becoming a standard mechanism for viewing all types of data, some types of content require a specialized viewer.

Transformation

The content within an ECM system is not useful unless it exists in a form that can be easily consumed. Often, part of the capture process includes transformation technologies that enable content to be stored in specific (or multiple) formats for easy consumption. Table 1-1 describes several common format types into which content may be transformed.

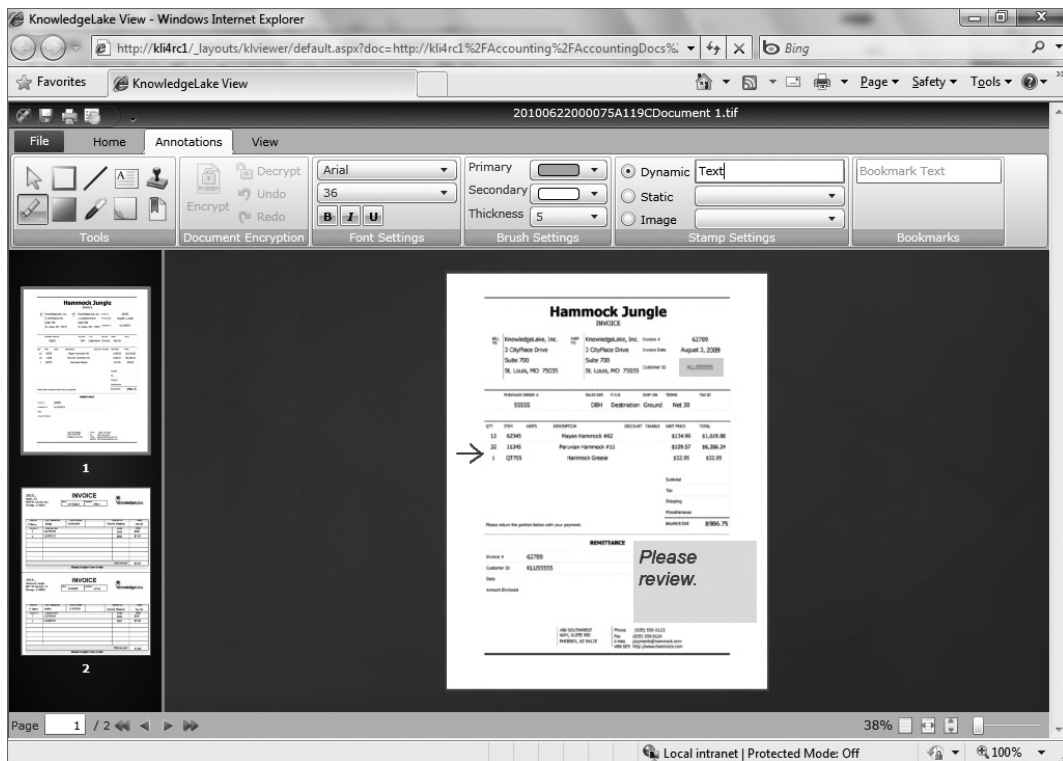


FIGURE 1-4

TABLE 1-1: Common Transformation Formats

| FORMAT | DESCRIPTION |
|----------------------------------|---|
| PDF (Portable Document Format) | Possibly the most common format for archived documents, PDF is a cross-platform format for storing almost any type of document. PDF documents typically display documents exactly as they will print. |
| XPS (XML Paper Specification) | Very similar to PDF but developed by Microsoft before given to the community. |
| XML (Extensible Markup Language) | An open format for describing just about any type of data. |
| HTML (Hypertext Markup Language) | The output format of most websites worldwide. |
| TIFF (Tagged Image File Format) | An image-only format often used when the original document's fidelity is no longer needed. |
| COLD/Enterprise Reports | Typically only report data is captured. However, when displayed, the data must be transformed to look like the original paper report that would have been generated. |

Security

Security ultimately applies to all the components of an ECM system. Good security systems enable system administrators to exercise complete control over how content should be secured. Administrators should be able to apply security in various ways: to each piece of content individually, automatically applied by capture systems, or controlled by other aspects of the system. Table 1-2 describes the three main security levels.

TABLE 1-2: Security Levels

| LEVEL | DESCRIPTION |
|----------|--|
| Parent | Security applied at the parent level forces content stored in the specific library or other repository item to take the same security level as the parent. Prior to Microsoft SharePoint Server 2007 (Windows SharePoint Service 3.0), this was the only option available. |
| Document | Document-level security allows specific security to be applied to a piece of content regardless of where it is located. |
| Type | Type-level security refers to security associated with a specific type of content. This set of permissions associated with an object is also known as an access control list (ACL). This type of security does not exist in SharePoint today. |

Digital/Electronic Signatures

Digital (or electronic) signatures guarantee the authenticity of the documents with which they are associated. Technologies that handle these signatures prevent a piece of content from being modified without the signature being compromised. Once the content is modified in any way, the signatures become invalid until reassigned.

Information/Digital Rights Management

Documents often need to be secured separately from the ECM security model itself. Because these documents can be transported outside the confines of the system, they need another layer of security to prevent unauthorized access to either part or all of their contents.

Within the industry, this technology is typically known as digital rights management; however, Microsoft has proprietary technology called Information Rights Management (IRM), designed specifically for content created by Microsoft software.

Digital rights management can control all aspects of a document, from preventing access to the document itself to how the document may be used. Examples include not allowing a document to be printed, saved, or even viewed.

SUMMARY

Without using a lot of technical jargon, this chapter provided a quick look at the ECM industry. Many developers understand ECM only as it applies to the SharePoint ecosystem, so the information provided here fills in some of the blanks by describing how the rest of world views ECM. Highlights of the chapter included a working definition of ECM from AIIM International, followed by a historical perspective, and finally the common components that make up an ECM system.

2

The SharePoint 2010 Platform

WHAT'S IN THIS CHAPTER?

- ▶ An overview of the SharePoint 2010 platform and the problems it addresses
- ▶ Core SharePoint concepts that are crucial for understanding deeper ECM concepts in this book
- ▶ An overview of the technical architecture of SharePoint 2010
- ▶ An overview of the SharePoint platform's development capabilities
- ▶ An introduction to the SharePoint 2010 ECM features covered in this book

The goal of this chapter is twofold. First, it contains information that provides a broad overview of SharePoint's purpose in general. This important context will provide you with the groundwork needed to understand the ECM-specific features of SharePoint and the remainder of the book. Second, it lays out a map for the rest of the book with regard to ECM functionality in SharePoint 2010. Therefore, use this chapter as a launchpad for determining where you would like to go next in this book.

This chapter is laid out as follows. First, some context around the history of SharePoint is provided. Next, the high-level capabilities of SharePoint 2010 are discussed. These are the items that make SharePoint SharePoint. The next sections discuss core SharePoint concepts (e.g., site collections, sites, etc.) as well as some items from a more technical perspective (Central Administration, features and solutions, etc.). Finally, the remaining sections provide the conduit to the other chapters in this book by covering the pertinent ECM features in SharePoint 2010.

A BRIEF HISTORY OF SHAREPOINT

Although the information in this section is not required for understanding the SharePoint platform, a short look at its history can be very useful for understanding why SharePoint 2010 looks and behaves the way it does from a capabilities perspective. If nothing else, it reinforces the adage that a good way to understand where you are is to understand where you came from.

First, a quick review of previous SharePoint versions is helpful:

- SharePoint Team Services/Microsoft SharePoint Portal Server 2001
- Microsoft SharePoint Portal Server 2003
- Windows SharePoint Services 3.0/Microsoft Office SharePoint Server 2007
- Microsoft SharePoint Foundation 2010/Microsoft SharePoint Server 2010

The first thing you'll notice is that SharePoint has gone through quite a few names and edition changes over the years! It took a few releases for Microsoft to learn enough about the challenges they needed to address as well as to flesh out the technologies used to build each release. The transition from 2001 to 2003 was largely technological; and the 2003-era product started to get some attention, whereas the 2001-era offerings were largely unnoticed by the masses.

SharePoint 2007 ushered in the modern era of SharePoint as we know it today, as it was again (mostly) rewritten in more modern technologies (which are touched on later in this chapter). For example, SharePoint 2007 was the first version to include a workflow engine and serious web content management capabilities. SharePoint 2007 also marked a turning point in terms of popularity. User adoption went through the roof; SharePoint became the fastest-selling product in Microsoft's history, hitting \$1 billion in revenue — a huge feat! SharePoint 2010 continues to build on the 2007-era product, offering huge improvements in features, and architectural changes with things like service applications and new search capabilities.

Many parts of SharePoint today owe their existence to products that used to be standalone. Following is a partial list of items that eventually worked their way under the SharePoint umbrella:

- Microsoft Content Management Server, now under the WCM features of SharePoint
- Microsoft Groove, replaced by SharePoint Workspace 2010
- Business Scorecard Manager, features of ProClarity, and the standalone PerformancePoint Analytics product, now under the PerformancePoint Services umbrella

SHAREPOINT 2010

The following sections provide the context necessary to dovetail into the subsequent chapters, which cover SharePoint from an ECM perspective. Some of these sections provide what might be considered basic information to those who have been working with SharePoint for several years. However, this material is a great introduction to SharePoint concepts, and can serve as a useful review for those who already have some SharePoint exposure.

First, we will review the core capabilities that define SharePoint and why it exists. Next, some foundational terms and concepts are defined and reviewed, such as how SharePoint is structured and how different SharePoint entities fit together. After that we will get a little bit more technical and review SharePoint from two perspectives: the architectural and the developmental.

Capability Categories

The SharePoint marketing team has done a good job of summarizing SharePoint’s offerings by dividing its capabilities into six categories. Although these divisions are marketing-oriented, they are useful for understanding, at a high level, what functional areas SharePoint provides. Figure 2-1 shows what is popularly referred to as the “SharePoint wheel.” Each of the topics contained in the wheel are discussed in the following sections.

Sites

This category is probably one of the more “squishy” of the six. It refers to the hierarchical objects in SharePoint that represent sites. These sites aren’t just for internal collaboration; they can exist for any number of scenarios, including the following:

- Business applications with workflow, forms, etc.
- Partner extranet for business collaboration
- Corporate intranet with news stories, policies, etc.
- Public-facing Internet site

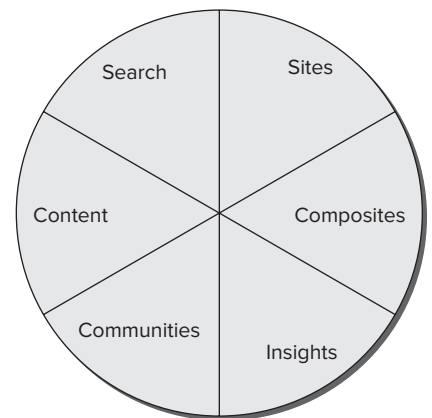


FIGURE 2-1

The sites concept also covers other various SharePoint features:

- Web Parts
- The fluent UI (also known as the *Ribbon*)
- Site management (e.g., creation, permissions, etc.)
- Publishing features
- My Sites
- Multilingual capabilities
- Mobile capabilities

As you can see, this area is somewhat of a catch-all for items that don’t necessarily fall into the other five areas described next.

Composites

New to SharePoint 2010, the concept of “composites” largely describes the capability that enables nondevelopers to build customized solutions in SharePoint 2010, without any code. SharePoint

provides several building block services and features that enable users to create DIY solutions, from the simple to the complex. Some of the services included in this bucket include the following:

- Access Services
- Business Connectivity Services
- Excel Services
- Visio Services
- InfoPath Forms Services

In addition to these items, the SharePoint platform enables configuration and customization from within the browser as well as within SharePoint Designer. For example, a nondeveloper can create custom workflow in SharePoint Designer that processes an InfoPath form through a business process. This is very powerful stuff!

Insights

The “insights” slice of the wheel describes features and capabilities in SharePoint 2010 that enable business intelligence capabilities. For the most part, the insights bucket refers to PerformancePoint Services, which is Microsoft’s web-based dashboarding tool. PerformancePoint can display data from various sources such as SQL Server Analysis Services cubes, databases, SharePoint lists, Excel Services, and more.

While PerformancePoint is the highlight of this feature area, it also offers some other supporting features, some of which have already been mentioned in the “Composites” section above. Excel Services and Visio Services both provide rich data visualization, and therefore can be used to support business intelligence initiatives. In addition, SharePoint ships with some charting Web Parts and key performance indicator (KPI) lists. These features are extremely powerful, especially when you consider that they do not necessarily require the intervention of IT in order to use them. However, when pursuing any serious business intelligence effort, governance is of the utmost importance; therefore, data sources and display mechanisms should not be configured by just anyone.

Communities

From wikis to workflows, SharePoint 2010 “communities” are all about helping people work together more effectively. Therefore, the functions of SharePoint that fall into this bucket are somewhat wide. For example, SharePoint team sites provide a great spot for teams of people to work together on various types of tasks, such as developing a document, developing a project plan, discussing how to solve a certain problem, and so on. As such, features like tagging, security, lists and libraries, and wikis fall into this category.

In addition, some of the 2010 desktop tools facilitate this collaboration. For example, Microsoft Word provides simultaneous multi-author editing capabilities, an extremely powerful tool that demonstrates the power of modern web-based applications. SharePoint Workspace and the Communicator tools also facilitate user interaction and are therefore included in this category.

Content

With the “content” section, we are really getting close to home with regard to ECM. This area of the SharePoint wheel describes all the ECM features of SharePoint 2010 and is what this book is all about. Its document management features make it easy to work with content type policies, taxonomies, workflow, and features like check-in/check-out and content approval. In addition, features of SharePoint 2010 that fall into the records management arena (see Chapter 8 for more information on records management) are also included in the content capacities group. Chapter 3 contains a wealth of information on content and document management.

Search

Search is one of the most powerful and advanced features in SharePoint 2010, so it’s not an afterthought by any means. Obviously, search is front and center in SharePoint; content isn’t of much use to users of the system if they can’t find the documents or items they are looking for.

SharePoint search comes in a few different flavors. First, there are the native search capabilities, which are improvements upon the search offerings in SharePoint 2007. This search provides powerful keyword searching, akin to what you are used to with search engines such as Bing or Google, as well as other refinement capabilities. Second, People Search is a big part of the SharePoint 2010 search offering. Finally, Microsoft’s FAST search engine provides even more capabilities than the core features, including thumbnails and document previews, much higher scale with regard to documents, extremely fast query response, more refinement capabilities, and more.

SharePoint Concepts

Now that we’ve reviewed the high-level attributes of the SharePoint platform, it’s time to dig a little deeper into some of the core concepts that support these feature areas. Figure 2-2, which shows a representation of the SharePoint object hierarchy, provides a good point from which to launch a discussion.

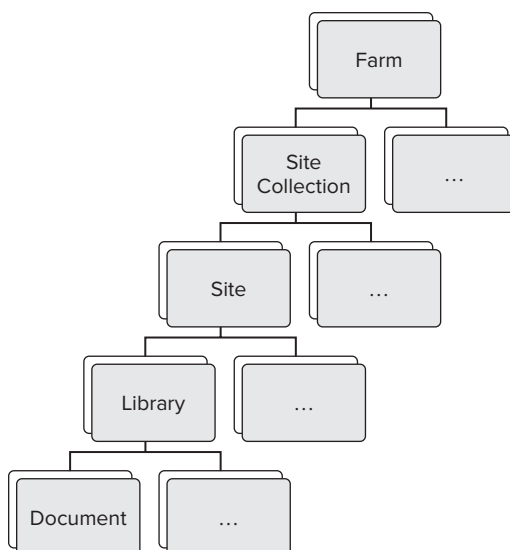


FIGURE 2-2

This diagram is largely conceptual because there are other entities (e.g., web applications) that are not represented here. However, the immediate goal of this section is to review the logical concepts before moving into some more of the technical pieces in the following sections.

Starting at the top of Figure 2-2 is a farm node. A SharePoint farm is the highest-level container for a logical SharePoint implementation. Granted, SharePoint implementations can and do contain multiple farms; but for all intents and purposes, individual farms are distinct animals. A farm contains all the elements necessary to make a SharePoint installation functional. This includes all the technical assets like servers and configuration. In addition, the farm acts as a logical container for all the other entities listed in Figure 2-2.

The next item in the tree is a site collection, which is self-explanatory: a site collection serves as a container for a collection of sites. Site collections always have a root site, meaning a site collection cannot exist without at least one top-level site. Site collections also provide a level of configuration and separation from other site collections. You can think of a site collection as a “walled garden” for configuration and content. Aside from some higher-level services at the farm scope, things can generally not be shared across site collections. This is a good thing as long as you understand the implications of how site collections and sites work when designing a SharePoint implementation.

SharePoint sites, of course, exist within a defined site collection. Sites are organized hierarchically, with the root site of the site collection always being the top-most parent. While some configuration is set at the site level (e.g., security, navigation, etc.), sharing configuration and content across sites in the same site collection is much easier and therefore desirable when compared to sharing across site collections.

Sites are also where all the real work happens; end users don't really ever do anything that isn't in the context of a site. Sites are where lists and libraries are defined, which contain all the lovely content that users enjoy so much. In addition, sites often house useful SharePoint forms and applications that drive business processes. It is useful to make a simple but very important distinction between lists and libraries. You can think of a list as a container for items. Items are simply distinct “rows” that contain metadata. For example, you could have a list containing items representing events. Events include data such as start date, end date, description, and so on. Lists really aren't any more complicated than that. Libraries are very similar except they contain documents. Documents are like list items in that they have metadata and they are distinct from one another; however, documents also have a file associated with them. Commonly, this is something like a Word document or a TIFF file.

So how is content represented in SharePoint? As already mentioned, items and documents have self-describing metadata associated with them. This metadata is made up of fields of given types. For example, out-of-the-box fields can be things like single lines of text, multiple lines of text, numbers, dates, people, options, and the list goes on. These fields can be grouped into a logic parent entity called a *content type*. The aptly named content type defines metadata and behavior. It defines metadata by simply containing references to columns. It defines behavior by providing the capability to run workflows when content of a given type is created, modified, and so on.

Content types are a crucial piece of the SharePoint ECM infrastructure. While they seem fairly straightforward conceptually, and they are, do not discount the importance of planning for content types in a SharePoint implementation. This becomes evident when you consider that content types can inherit from one another. This means that you can define a content type that specifies some

generic, high-level fields (e.g., Contract), and then define other content types that provide more specific fields (e.g., Employment Contract, Vendor Contract, etc.).

All the entities discussed in this section have important attributes and configurable information associated with them. For example, all these hierarchy levels have certain aspects of configurable security. In addition, some features can only be turned on or off at the site collection or site level. The remainder of the book touches on these topics, so this chapter does not provide an exhaustive list; it's just important to understand these high-level concepts at this point.

Architecture

Now that we've covered SharePoint capabilities and concepts from a high level, we can examine some of the more technical aspects of the platform. First, take a look at some of the technologies that SharePoint is built upon:

- Windows Server OS
- IIS
- SQL Server
- .NET 3.5
- ASP.NET web forms

Although the list isn't exhaustive, you can see that SharePoint is actually no different than most other applications you may have worked on or with previously. SharePoint is basically a very large .NET application that uses SQL for data storage! This statement is not meant to downplay the power or complexity of SharePoint but to highlight that the platform need not be viewed as a monolithic piece of technology that cannot be broken down and understood, even from a technical perspective.

From an architectural perspective, let's start with the farm concept introduced in the previous section. Server farms are still the top-level design concept even when looking at SharePoint from the technical angle. At the very least, a SharePoint farm requires one server. However, except for the smallest solutions, this implementation is not practical for a production farm. A more common scenario, albeit still small, is keeping all the web front ends (WFEs) and application logic on one server and keeping SQL on its own dedicated server. The point here is that SharePoint is infinitely scalable and configurable.

Another important entity that is scalable across servers (including backend databases) is the *service application*. Service applications are a generic infrastructure on which developers can build shareable application support systems. In fact, Microsoft ships SharePoint 2010 with many different service applications that support some of its core functionality, including the following (this list is not all-inclusive):

- Search
- User profiles
- Managed metadata

- Excel services
- Visio services
- Business data connectivity (BCS)

When you consider what each of these applications does, it's clear that the functionality provided by each would be very useful across an entire SharePoint farm or even multiple farms. The SharePoint 2010 architecture enables you to scale these service applications with each other or even independently depending on the system's requirements. Furthermore, an entire SharePoint farm can be dedicated to just hosting service applications! Then other SharePoint farms can connect to that application farm and consume whatever functionality is required. The opportunities to scale in SharePoint 2010 are virtually endless, and greatly improved from even SharePoint 2007.

Two other important architectural concepts to be aware of are *web applications* and their associated *zones*. Web applications and zones correspond directly to IIS websites. Each web application has a different domain that provides an additional layer of security. In addition, a service application can be assigned to different web applications. Zones are created by extending web applications, and they exist mainly for security purposes. Zones can have different authentication mechanisms (e.g., Windows authentication vs. forms-based) and can be accessed from different parts of the network (e.g., local intranet vs. extranet). Note that it is within web applications that site collections exist.

A special SharePoint component called Central Administration is used to manage the entire farm. This website can be hosted on its own web front end and has its own database called the *configuration database*. Figure 2-3 shows the main page of SharePoint 2010 Central Administration.

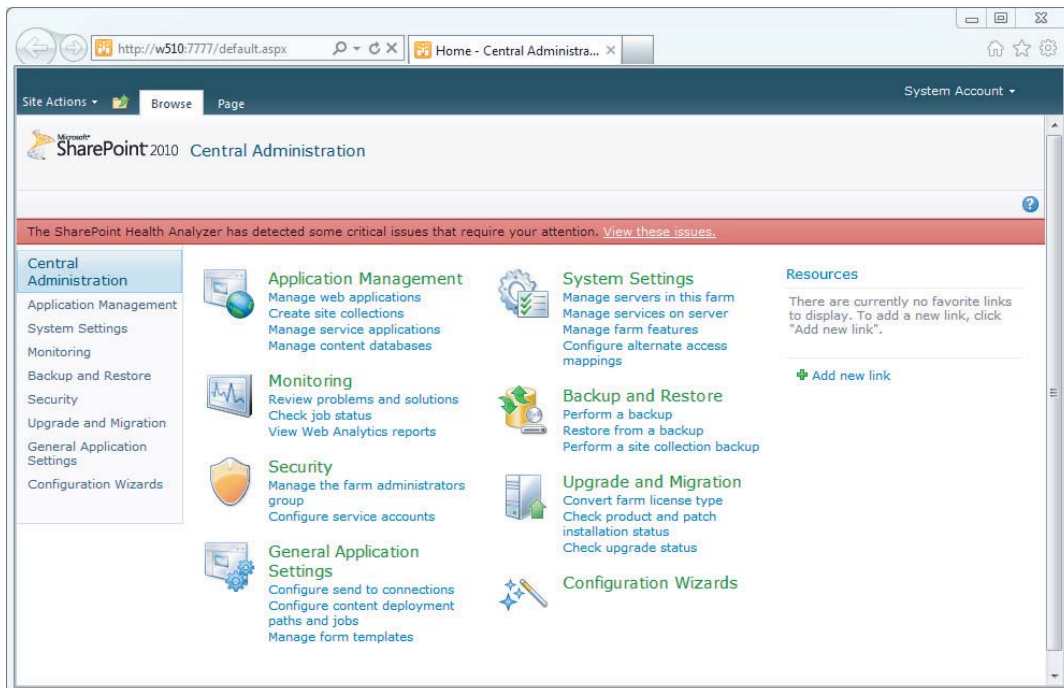


FIGURE 2-3

It is within Central Administration where the previously mentioned service applications as well as other farm-level items are configured. Web applications, zones, and site collections are created and managed here as well.

Another very useful SharePoint feature that is managed in Central Administration is the SharePoint timer job. Timer jobs are, not surprisingly, discrete jobs that run on a schedule. These jobs perform all kinds of important tasks that keep SharePoint running and enable it to support critical business processes. For example, there are timer jobs to send content alerts, process workflows, and monitor the health of your SharePoint implementation. Figure 2-4 shows the configuration screen for the workflow timer job in Central Administration.

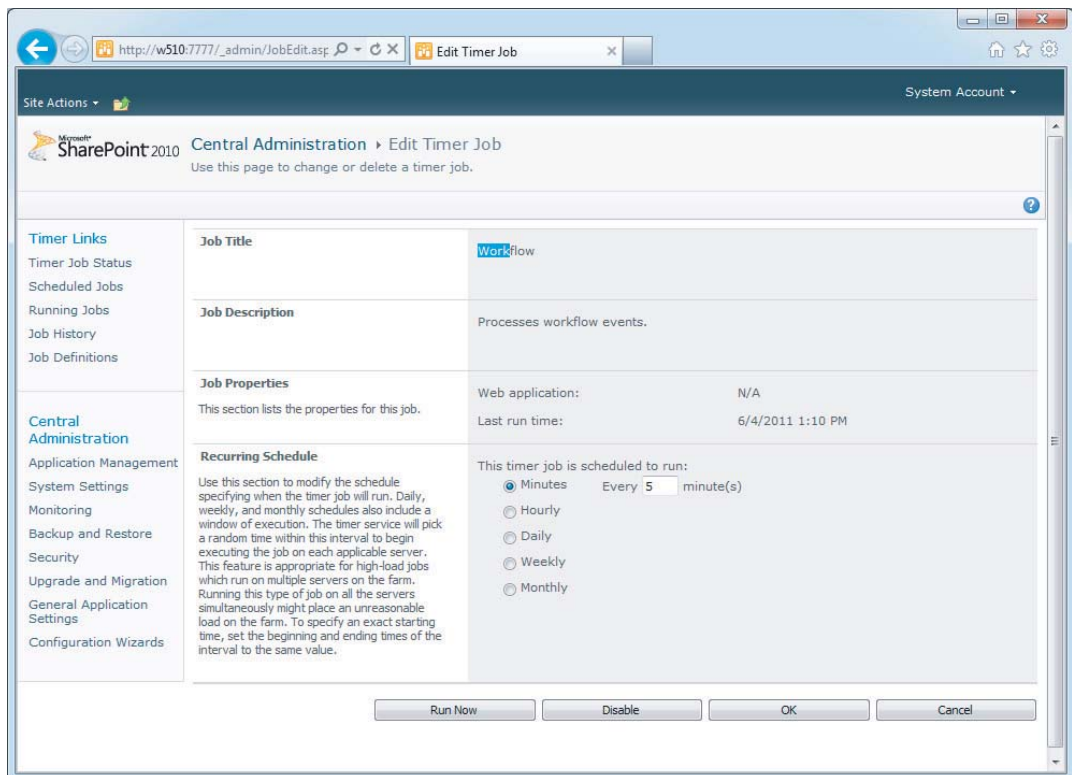


FIGURE 2-4

Finally, a major pillar of SharePoint 2010 is SQL Server. After all, an application isn't very useful if you don't have a place to put all the user-generated content. SQL Server stores important configuration information, which is set in Central Administration (in the configuration database), as well as the databases that house user content that was added to the farm's sites and libraries (known as *content databases*). Note that all the data for one site collection must reside in one SQL Server content database. However, multiple site collections can exist in one content database. This is an important consideration when architecting any SharePoint implementation; planning is crucial! In addition, SharePoint service applications can have their own databases. For example, the managed metadata application stores its information in a SQL database.

While this section is by no means a 500-level course in SharePoint architecture, it has described the most common technical aspects of SharePoint at a high level. Other chapters of this book, such as Chapter 12, “Scalable ECM Architecture,” cover these and other technical aspects in much greater depth.

Development Concepts

Hopefully, it’s apparent by now that SharePoint is not just an application per se, but a broad platform in much the same way that Windows is a platform. Any platform worth its salt is both extensible and well supported by a vibrant ecosystem of developers and extenders. SharePoint definitely meets both of these qualifications. Microsoft’s SharePoint team obviously put a lot of thought into how enterprises, systems integrators, and independent software vendors might want to extend the core SharePoint platform to do more than what it does out of the box; name any SharePoint feature or function and there is likely some sort of extension or add-on created for it.

One of the most common ways to extend SharePoint is to use custom code. Because SharePoint is written on the .NET Framework 3.5, that is the natural and obvious choice for developing a custom solution. Depending on what you are trying to accomplish, a custom solution might consist of UI components like ASP.NET web forms or Silverlight, workflows, timer jobs, Web Parts, and so on. Commonly, a custom solution consists of more than just a standalone assembly, UI element, or the like. Deploying and managing these items can be difficult without the mechanisms that SharePoint provides for these purposes.

SharePoint solutions (in the specific sense; they are actually called “solutions”) provide the unit of deployment and management for custom solutions (in the broad sense). Once packaged by the developer, SharePoint solutions end up as a single file with a `.wsp` extension. These files are actually CAB files. You can see for yourself by renaming one to `.cab` and double-clicking it.

SharePoint solutions contain farm-level items such as application pages, assemblies, and site definitions. In addition, solutions can and often do contain another deployment concept called a *feature*. Features are pieces of functionality that have to be activated at varying scopes of the farm (i.e., farm, web application, site collection, or site).

As mentioned earlier, the SharePoint solution infrastructure can deploy and manage technology assets developed on the .NET Framework. However, other assets, which are defined declaratively via a language called CAML, can be deployed as well. CAML is simply an XML-based schema that SharePoint understands. For example, a SharePoint developer can define new fields, content types, lists, and more just by defining some XML in a feature. Defining these assets by hand can be tedious and error-prone.

Luckily, SharePoint 2010 provides a huge leap forward in terms of development tools. Not surprisingly, Visual Studio is the tool of choice for custom solutions. To aid in these development efforts, the SharePoint team has provided several project and item templates, which can greatly speed up development. Figure 2-5 shows the New Project dialog for SharePoint in Visual Studio 2010.

The deployment and debugging capabilities provided in Visual Studio 2010 greatly enhance developer productivity. This fact is demonstrated by the F5 debugging capabilities, which enable developers to iteratively develop and test code very quickly. On the surface, this may not sound like a big deal, but when you consider all the tasks that must be performed behind the scenes when deploying a SharePoint solution, you can appreciate this capability.

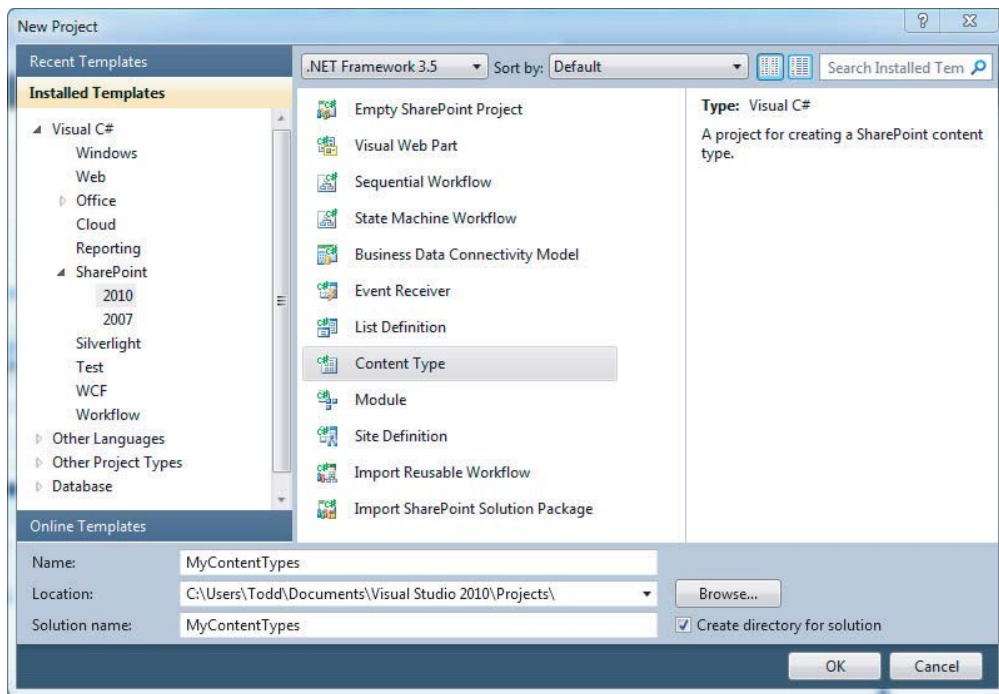


FIGURE 2-5

To deploy a SharePoint solution, it must first be registered with the farm (installed). Then it must be deployed (turned on) to the appropriate web applications if applicable. Next, any features contained within the solution need to be activated at the appropriate levels. Finally, any items in your solution, such as a workflow, must be associated with a list or library for testing, and you must navigate to some page to test your code. Visual Studio does all of this for you in a matter of seconds! This process was not fun in SharePoint 2007, to say the least.

ECM IN SHAREPOINT 2010

The following sections provide a brief overview of the main ECM features introduced in SharePoint 2010. You can use these sections as a guide to the remainder of the book, as each feature points to the chapter that covers that topic most extensively. However, you can also certainly read this book sequentially; whatever fits your needs the best. Keep in mind that the topics covered here are highlights of the new ECM-related features, not an all-inclusive list.

Managed Metadata

In the context of SharePoint 2010, *managed metadata* broadly refers to the capability to define, manage, and utilize a central set of items, which may be hierarchical. These items are logically grouped together, which enables them to be attached to content types for tagging and later retrieval. Because these items are centrally managed, governance and sharing can be readily handled, which wasn't really possible in SharePoint 2007 without heavy customization or development.

These items are referred to as *terms*, and they are grouped in *term sets*. You can configure term sets in Central Administration via a service application or in individual site collections. Because term sets can be managed in a service application, this means that they can be shared, even across multiple farms. In addition to managed terms, SharePoint 2010 supports what are called *enterprise keywords*, which users can create to globally and organically tag content from terms in a flat term set. For more information about managed metadata, see Chapter 3.

Ratings

If you have used Netflix or Amazon before, you probably realize the value of user-generated recommendations. When other people can designate whether or not an entity was valuable to them, and especially when this can be done by large groups of people, it makes it easier for popular (and presumably good) content to bubble up to the top. SharePoint 2010 enables users to rate content with a familiar five-point system (1 being the worst, 5 being the best). See Chapter 3 for more information on ratings.

The Content Type Hub

The content type hub will be a boon for anyone who has done large-scale content management projects in SharePoint 2007. Essentially, this feature enables you to manage content types on a designated central hub site, and then the content type definitions can be deployed to specified subscriber sites. This takes content governance to the next level. See Chapter 3 for more information on this feature.

Search

SharePoint search has plenty new to offer in 2010. From new features in the core search offering, which was available in 2007, to the availability of FAST technology, search is an extremely powerful feature set in SharePoint 2010. New features include capabilities like phonetic searching, search suggestions, Boolean query syntax, wildcard searching, and a “did you mean”-type feature when viewing results. See Chapter 6 for an in-depth overview of search.

Workflow

Workflow is a crucial aspect of ECM, as it can facilitate the entry of content into a system or even facilitate the content’s expiration. While workflow is not a brand-new feature in SharePoint 2010 (it was introduced in SharePoint 2007), it is greatly enhanced.

SharePoint workflow in 2010 has improvements such as workflow runtime services, site workflows, reusable SharePoint Designer workflows, workflow event receivers, and a massively improved development experience in Visual Studio. For more information on workflow, see Chapter 4.

Document Sets

Document sets are pretty much what you would expect based on the name. Essentially, document sets are special folders that group related documents together as an atomic unit. Examples might be a new employee packet, project documents, or a set of standard contracts. A document set content

type defines all the children documents that can be included, and which workflows can be run against the document set itself. For more information on document sets, see Chapter 3.

Document IDs

In previous versions of SharePoint, the only thing that pointed to a document was a link. If a document were moved somewhere else in the farm, then that old e-mail you saved with a link to that document is obsolete. The document ID service in SharePoint 2010 addresses this issue by assigning a unique identifier that becomes attached to a document upon entry into the system. For more information on the Document ID service, see Chapter 3, “Document Management.”

Content Organizer

The Content Organizer is a great new feature that aids in document management by automatically routing incoming content to the appropriate location. This can free users from having to memorize or think about taxonomy when adding content to SharePoint. It also ensures that rules are followed with regard to how many documents can be added to a single folder.

The Content Organizer uses rules based on metadata, as well as a special library called the Drop Off Library. If appropriate, all users can be redirected to this library when adding content to a site, and then the Content Organizer’s rules can take over and route the content to the correct destination. For more information on the Content Organizer, see Chapter 8.

Records Management

Records management is a whole discipline in itself. A record is essentially a piece of content, often a document, that represents an official event or transaction that occurred and that an organization is obligated to hold on to for a specified amount of time. Examples include contracts, e-mail, medical records, and so on. Some of the features that enable records management are the aforementioned Content Organizer, holds, and content type policies. Chapter 8 contains more detailed information on records management.

Digital Asset Management

Like records management, digital asset management (DAM) is more of a standalone concept or discipline than it is a feature. SharePoint 2010 provides the capability to manage digital assets such as images, videos, audio, and so on by providing features like the asset library, which enables life-cycle management of individual pieces of rich media. For more information on digital asset management, see Chapter 9.

Web Content Management

Web content management (WCM) is another entire area of SharePoint that is supported by various features. Some of the new WCM features in SharePoint 2010 are the greatly enhanced rich text editor, a vastly improved page creation and configuration experience, support for rich media, better browser support, support for managed metadata, and more. See Chapter 7 for more information on this very large topic.

SUMMARY

SharePoint 2010 is a multifaceted and extremely powerful platform that is capable of supporting the most demanding of ECM requirements. SharePoint's capabilities also range outside the ECM world as well, and include sites, composites, insights, communities, content, and search.

This chapter covered some core SharePoint concepts that are crucial for understanding much of the high-level functionality in the platform. These include items like the farm, the site collection, sites, and lists and libraries. In addition, it is important to grasp some of the architectural components, such as web applications, Central Administration, content databases, and the like.

SharePoint 2010 has both great new features and improvements to existing features that support ECM implementations. This includes items like document sets, the Document ID Service, the Content Organizer, the content type hub, and management metadata. Major improvements have been made in areas like web content management, records management, and workflow.

PART II

Pillars of SharePoint ECM

- ▶ **CHAPTER 3:** Document Management
- ▶ **CHAPTER 4:** Workflow
- ▶ **CHAPTER 5:** Collaboration
- ▶ **CHAPTER 6:** Search
- ▶ **CHAPTER 7:** Web Content Management
- ▶ **CHAPTER 8:** Records Management
- ▶ **CHAPTER 9:** Digital Asset Management
- ▶ **CHAPTER 10:** Document Imaging
- ▶ **CHAPTER 11:** Electronic Forms with InfoPath
- ▶ **CHAPTER 12:** Scalable ECM Architecture

3

Document Management

WHAT'S IN THIS CHAPTER?

- ▶ Understanding document management and how Microsoft SharePoint meets the needs of an enterprise-class document management system
- ▶ Exploring Microsoft SharePoint's extensive document management capabilities
- ▶ Administering and using Microsoft SharePoint's document management capabilities
- ▶ Leveraging Microsoft SharePoint APIs to programmatically interact with its numerous document management features

Organizations both large and small alike struggle to manage their document resources effectively. These organizations are often plagued by vast amounts of paper documents stored in file cabinets, desk drawers, and various other physical locations. These paper documents can consume large amounts of storage space, which may result in costly expenses. It is not uncommon for important documents to be misfiled, and it can be difficult and time consuming to find them even if they are filed properly. Unfortunately, this problem isn't limited to paper documents; many of the same issues occur with an organization's electronic documents as well — due to disorganized file shares and local user copies that lack centralized management, consistent structure, and organizational oversight. Managing documents in this way — whether they are in paper form or electronic form — is costly, inefficient, and ineffective.

Imagine a typical company employee who creates business documents on a daily basis. Many times, the employee will create a document and save a local copy to his or her local hard disk drive or network file share. This document is often e-mailed to other employees as a part of a manual business process, during which it may be modified and resent several times. These different versions of documents are difficult to keep track of because there is no clear history

or single working copy of the document. When a given document is needed in the future, it can be very difficult to retrieve it because there is no systematic means of easily locating it. This could be because no one knows which copy is the working copy or because the file is simply located in an unknown location on a network file share. The solution to effectively managing these documents is to take advantage of an electronic document management system.

WHAT IS DOCUMENT MANAGEMENT?

Document management is a foundational pillar of enterprise content management (ECM). It encompasses the storage, organization, classification, and control of electronic documents within a computing platform referred to as a *document management system*. The purpose of a document management system is to actively manage a document's life cycle for effective use within business processes.

Classifying electronic documents within the system involves associating pieces of descriptive data, called *metadata*, with the document to aid in document retrieval. While classifying documents is critical to ensuring that they can be recalled easily, organizing documents into various system-defined locations is equally important in order to efficiently manage both their security and their visibility, as well as optimize their use in business processes. The common term used to describe the organization and classification of documents within the system is *document taxonomy*.

Storing documents within a document management system enables companies to effectively manage their valued document resources. Whether documents are stored as paper in physical file cabinets or electronically on a file server, document management eliminates the inadequacies associated with these scenarios. It enables a business to avoid hidden silos of information, reduces the amount of time required to retrieve important information, promotes secure access to sensitive data, enables documents to be easily located, provides a foundation for application integration, and enables business processes to be automated.

A typical set of document management features found in most document management systems is listed in Table 3-1.

TABLE 3-1: Key Document Management Features

| FEATURE | DESCRIPTION |
|------------------------|--|
| Metadata | Descriptive data that can be associated with documents to aid in identification and retrieval |
| Storage | The organization of documents within a system such that they appear to be in different locations |
| Check-In/ Check-Out | Enables users to actively block other users from editing a document concurrently |
| Version History | Enables previous versions of a document to be tracked and reverted |
| Audit Trail | Tracks the usage of a document over its lifetime within the system |
| Security | Enables users to see or manipulate (e.g., edit, delete, etc.) particular documents based on assigned permissions |

MICROSOFT SHAREPOINT AS A DOCUMENT MANAGEMENT SYSTEM

Microsoft SharePoint Foundation (MSF) provides all the key features of an enterprise-level document management system within a web-based portal, and it can be easily expanded on-the-fly to meet critical document management requirements dictated by the business. SharePoint can be administered through simple web-based screens that are accessible based on user permissions. This provides an easy-to-manage environment that can be expanded without the need for custom development or IT involvement. However, the real power of SharePoint as a document management platform is that nearly everything that can be accomplished via application screens can also be accomplished by leveraging the extensive application programming interface (API) that SharePoint provides for programmatic interaction with the SharePoint Server. This API is referred to as the *SharePoint object model*.

The SharePoint object model provides various access points, enabling developers to build applications and components that can interact with SharePoint in various development scenarios. SharePoint provides a server-side object model for working with SharePoint on the server where it is actually running. SharePoint also provides three equivalent client-side object models for programmatic interaction with SharePoint from a machine that is remote from the SharePoint Server. The three client-side APIs are subsets of the server-side API but are designed to run on different development platforms. The client-side object model is provided as a managed API, a Microsoft Silverlight API, and an ECMAScript API. SharePoint also provides a set of traditional web services that can be used for remote programmability. Examples of using these various APIs will be provided throughout this chapter.

Microsoft SharePoint Server (MSS) is an advanced version of SharePoint that builds and expands on Microsoft SharePoint Foundation features and programming interfaces to provide a number of advanced document management capabilities that may either be required or significantly enhance the document management needs of an enterprise. MSS is provided in two levels — Standard, which provides a number of extended features, and Enterprise, which provides all the features of Standard and many other advanced features, which are described in this chapter.

DOCUMENT TAXONOMY

Document taxonomy refers to the hierarchical structuring of document storage locations and the metadata associated with these documents. Designing an effective document taxonomy is critical to organizing and using the documents within a document management system. A well-structured taxonomy simplifies the application of security, optimizes document retrieval, and enables manual navigation throughout the system to be intuitive. Ultimately, a document taxonomy facilitates the management of electronic content throughout its life cycle.

After the document taxonomy is defined, it can be used to populate storage locations with the appropriate electronic documents and provide the associated properties with values, a process referred to as *document classification*.

SharePoint provides a simple, dynamically expandable model for creating a document taxonomy through the use of its common platform-building components. These basic components, shown in Table 3-2, act as building blocks for structuring the document management system.

TABLE 3-2: Document Taxonomy Building Blocks

| COMPONENT | DESCRIPTION |
|-----------------|--|
| Web Application | The physical website hosted in IIS that hosts site collections. |
| Site Collection | A logical, hierarchical structure of sites consisting of a single top-level site and any sites below it. Site collections are hosted within a web application. |
| Site | An independent location within a site collection. Each site can have a group of sub-sites. The top-most site in the site collection is called the root site. |
| Lists | A location in a site that acts as a container for data. |
| Columns | A column describes the data that is stored in the list. Much like a table in a relational database, a list consists of a set of typed columns that store data. |
| Content Type | A content type provides a reusable set of columns and settings that can be added to a list and then associated within individual items in the list. |
| List Item | A list item is a single entry in a list. Each list item has a set of properties and values that represent the columns of a list. |

Document Libraries

File storage locations in SharePoint are referred to as *document libraries*. A document library is a special type of SharePoint list in which each item in the list is associated with a file. Because of this association, each list item in a document library is referred to as a document. Another type of list item that can be stored in a document library is a *folder*. Folders enable a document library to be structured into a hierarchical set of sublocations for organizing documents. From a file-storage perspective, document libraries are very similar to network file shares in that their main purpose is to store and organize files, but they also provide the features of a typical SharePoint list as well as key document management–related features.

As a central component of the document taxonomy, the columns within a document library are often referred to as a document’s *metadata*. This data is useful to describe the document represented by the list item, and it is central to document management in SharePoint.

Figure 3-1 provides an example of a document library within SharePoint.

The Document Library Programming Model

SharePoint provides a programming model for the interaction and management of document libraries from both the server-side and client-side object models. The code for the server-side object model resides in the `Microsoft.SharePoint` namespace, while the client-side code resides in the `Microsoft.SharePoint.Client` namespace. If you are using ECMAScript, the script belongs to the `SP` namespaces. SharePoint also exposes the `Lists` web service for remote interaction as well.

The screenshot shows a SharePoint document library with a list of 36 documents. The columns are Name, Modified, and Modified By. The documents are named 'Document 1' through 'Document 36'. The 'Modified' column shows dates from 10/28/2010 11:52 AM to 10/28/2010 11:53 AM. The 'Modified By' column shows the user 'BGWS2008R2X64\Administrator' for all documents.

| Name | Modified | Modified By |
|--------------------------------|---------------------|-----------------------------|
| 2010102800002419D92Document 1 | 10/28/2010 11:52 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 10 | 10/28/2010 11:52 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 11 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 12 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 13 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 14 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 15 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 16 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 17 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 18 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 19 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 2 | 10/28/2010 11:52 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 20 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 21 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 22 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 23 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 24 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 25 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 26 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 27 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 28 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 29 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 3 | 10/28/2010 11:52 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 30 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 31 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 32 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 33 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 34 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 35 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |
| 2010102800002419D92Document 36 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator |

FIGURE 3-1

When using the SharePoint object models to develop for SharePoint, it is important to remember that the client-side object model is a subset of the full server-side object model. Developing using these frameworks is simplified because the classes follow similar naming conventions except that the client-side classes do not have the SP prefix in their class names. Table 3-3 outlines the main classes that make up these object models.

TABLE 3-3: Main Classes in the Object Model for Document Libraries

| CLASS | DESCRIPTION |
|-------------------|--|
| SPSite | Represents a site collection. |
| SPWeb | Represents an individual site. |
| SPList | Represents a single list within a site. All lists within SharePoint inherit from the SPList class. |
| SPListCollection | Stores a collection of SPList objects. |
| SPDocumentLibrary | Represents a single document library within a site. The base class for this class is SPList because a document library is a type of SharePoint list. |
| SPListItem | Represents a single item within a list or document library. |

The following two listings demonstrate some basic programmatic interaction with document libraries. Listing 3-1 demonstrates the creation of a document library using the server-side object model. Listing 3-2 demonstrates uploading a document to an existing document library using the client-side object model.



LISTING 3-1: Creating a Document Library Using the SharePoint Server Object Model

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                //Create new Document Library
                Guid libraryId = web.Lists.Add("LibraryTitle",
                    "LibraryDescription",
                    SPListTemplateType.DocumentLibrary);

                //Access Document Library using List ID
                SPDocumentLibrary library =
                    web.Lists[libraryId] as SPDocumentLibrary;
            }
        }
    }
}
```

Listing 3-1 begins by initializing a `SPSite` class to get access to the SharePoint site collection indicated by the specified URL. Using this class, a reference to the corresponding `SPWeb` is obtained by calling the `Open` method. Once a reference to the `Web` is obtained, the `Add` method of the `Lists` property is called, which results in a new document library being created at the referenced `Web` with the name and description specified.



LISTING 3-2: Uploading a Document to a Document Library Using the SharePoint Client Object Model

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Client;
```

```

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ClientContext context = new ClientContext("http://10.1.0.169"))
            {
                string fileName = "sample.txt";

                //Uploading file to Document Library using byte[]
                byte[] fileBytes = System.IO.File.ReadAllBytes(fileName);

                List list = context.Web.Lists.GetByTitle("LibraryTitle");

                File file = list.RootFolder.Files.Add(new FileCreationInformation()
                {
                    Content = fileBytes,
                    Url = fileName
                });

                context.Load(file);
                context.ExecuteQuery();
            }
        }
    }
}

```

Listing 3-2 passes the URL for the site that will be interacted with to the `ClientContext` object. The `ClientContext` object is the main entry point of the client-side object model. After instantiating this object, a file is read into the program as a byte array, and is then passed to the `File` object obtained by calling the `Add` method on the `FileCollection` of the root folder of the list specified. `ExecuteQuery` is then called to tell the `ClientContext` object to carry out the specified actions.

Columns

Within SharePoint, columns are used to define and store the data that will be included in a list. Throughout SharePoint, columns are referred to in multiple ways. When viewing an individual item, columns are referred to as properties, whereas when interacting with an item programmatically, columns are referred to as fields. Another term used to describe this data is metadata. Essentially, metadata is data *about* data. In terms of SharePoint, columns represent additional data that describes the individual list item, which itself is data. Most of this metadata is predefined by an administrator and later populated with data when list items are created by users. However, some system-level metadata is automatically collected, such as who created or modified the item, and when.

Columns can be created at the site level or the list level. Columns created at the site level are called site columns and do not belong to a specific list, but rather act as column templates that can be later assigned to lists. Site columns are stored in the gallery list called the *site column gallery*. Columns can also be created and added directly to a list, or they can be added to a content type, which is discussed in the section on content types later in this chapter. These columns do not exist as templates,

so they cannot be reused across the site. They exist only in the location where they are defined. Figure 3-2 shows the New Site Column dialog.

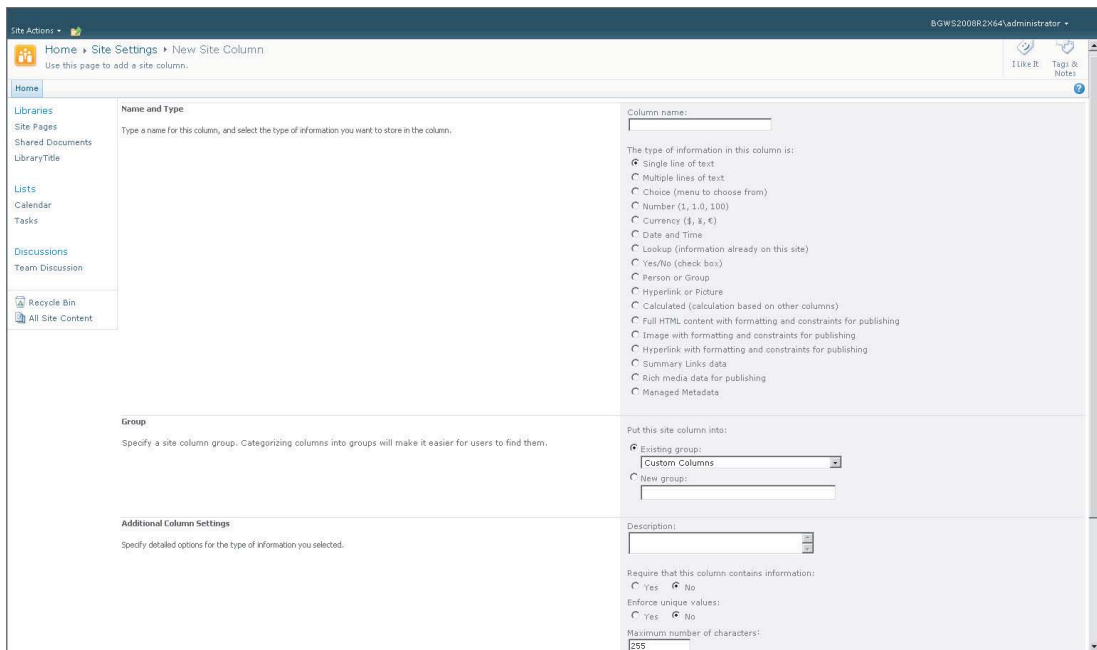


FIGURE 3-2

SharePoint provides the capability to create many different types of columns for storing data. Each column type provides a number of properties for setting various constraints used to enforce data integrity. Table 3-4 outlines the available column types.

TABLE 3-4: SharePoint Column Types

| COLUMN TYPE | DESCRIPTION |
|------------------------|--|
| Single Line of Text | Allows users to enter a textual value up to 255 characters in length. |
| Multiple Lines of Text | Allows users to enter textual values that can contain multiple lines and conditional formatting. |
| Choice | Allows users to select from a set of predetermined values using a drop-down list, radio buttons, or check boxes. |
| Number | Allows users to enter a numeric value or percentage. |
| Currency | Allows users to enter a numeric value formatted as a specific type of currency. |
| Data and Time | Allows users to enter a value that is either a date or both a date and a time. |

| COLUMN TYPE | DESCRIPTION |
|----------------------|--|
| Lookup | Allows users to select from a list of values that are derived from another list in the same site. |
| Yes/No | Allows users to enter a value that is either yes or no. |
| Person or Group | Allows users to select users or groups as a value. |
| Hyperlink or Picture | Allows users to provide a hyperlink to represent a link to another page or an image. |
| Calculated | Used to display a value that is calculated using a subset of Microsoft Excel formulas and other columns in the same list item. |
| External Data | Allows users to select from a list of values that are derived from a source that is external to SharePoint, such as a database or a web service. |
| Managed Metadata | Allows users to select from a list of hierarchical sets of values called <i>term sets</i> . Managed metadata is discussed in detail later in this chapter. |

The Column Programming Model

The server-side and client-side object models provide a programming model for the interaction and management of columns within SharePoint. Table 3-5 outlines the main classes used when manipulating columns programmatically.

TABLE 3-5: Main Classes in the Object Model for Columns

| CLASS | DESCRIPTION |
|--------------------|---|
| SPField | Base class that represents a field on a list or Web. All field classes inherit from this class. |
| SPFieldCollection | Stores a collection of SPField instances. |
| SPFieldBoolean | Represents a Yes/No column. |
| SPFieldCalculated | Represents a Calculated column. |
| SPFieldChoice | Represents a Choice column. |
| SPFieldCurrency | Represents a Currency column. |
| SPFieldDateTime | Represents a Date and Time column. |
| SPFieldLookup | Represents a Lookup column. |
| SPFieldLookupValue | Represents a single Lookup column value. |

continues

TABLE 3-5 (continued)

| CLASS | DESCRIPTION |
|-------------------------|--|
| SPFieldMultiChoice | Represents a Choice column that accepts multiple values. |
| SPFieldMultiChoiceValue | Represents a single multi-choice value. |
| SPFieldMultiLineText | Represents a Multiple Lines of Text column. |
| SPFieldNumber | Represents a Number column. |
| SPFieldText | Represents a Single Line of Text column. |
| SPFieldUrl | Represents a Hyperlink or Picture column. |
| SPFieldUser | Represents a Person or Group column. |
| SPFieldUserValue | Represents a single Person or Group column value. |

The following two listings demonstrate two simple examples of using this programming model. Listing 3-3 shows how to create a site column and add it to a document library using the server-side object model. Listing 3-4 demonstrates using the client-side object model to update a column value for an existing list item.



Available for
download on
Wrox.com

LISTING 3-3: Creating a Site Column and Assigning It to a Document Library Using the Server-Side Object Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                //Create a Single Line of Text Site Column that requires a value
                string internalName = web.Fields
                .Add("ColumnTitle", SPFieldType.Text, true);

                //Get Site Column from the web
                SPField siteColumn = web.Fields
                .GetFieldByInternalName(internalName);
                SPList list = web.Lists["LibraryTitle"];
            }
        }
    }
}
```

```

        //Add Site Column to List
        list.Fields.Add(siteColumn);
    }
}
}
}

```

The preceding code first gets a reference to the necessary `SPWeb` object. A new field is then created by calling the `Add` method of the `Fields` property, which returned the internal name of the field. The field's internal name is a static name; it does not change even if the field's display name is changed. Using the internal name, a reference to the corresponding `SPField` object representing the newly created site column was retrieved and subsequently added to the specified list as a library column.



Available for
download on
Wrox.com

LISTING 3-4: Updating a Column Value for an Existing List Item Using the Client-Side Object Model

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint.Client;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ClientContext context = new ClientContext("http://10.1.0.169"))
            {
                Web web = context.Web;

                //Get existing item from list
                List list = web.Lists.GetByTitle("LibraryTitle");
                ListItem item = list.GetItemById(7);

                //Set column value
                item["ColumnName"] = "Value";
                item.Update();

                //Execute actions
                context.ExecuteQuery();
            }
        }
    }
}

```

The preceding code updated a column value for a `ListItem` using the client-side object model. After getting a reference to the `Web` object for the URL specified, a reference to the desired `ListItem` was retrieved from the specified list. Using the `ListItem` object, the indicated column was then updated with a new value.

Content Types

Content types are objects in SharePoint that represent a reusable collection of columns and settings that can be associated with content within lists. By encapsulating these settings into a reusable object, administrators can easily manage a collection of columns and settings as a single entity in a centralized manner. For instance, when creating a document taxonomy, it is often necessary to create a consistent set of columns and settings to represent a particular type of content. For example, suppose a taxonomy is being created to store accounts payable documents. An administrator can create SharePoint content types that represent invoices and purchase orders. Each of these content types will consist of all the columns and settings that these types of documents need to possess. Once the content types are created, the administrator can then assign them to a document library as a single entity; and when content is added, users can select the appropriate content type for the uploaded content and then populate the associated columns with the necessary data as indicated by the content type's configured settings.

Just as columns can be created within the site column gallery, content types can also be defined in their own site-level gallery, called the Site Content Types gallery, as seen in Figure 3-3. Once content types are defined, they can easily be assigned to list instances so that they can be used when populating the metadata of new or existing content. An important thing to understand when assigning site content types to lists is that the list content type is actually a *local copy* of the site content type. Changes made to the site content type are not pushed down to the list content types unless the administrator instructs SharePoint to do so. However, any changes made directly to a list content type cannot be propagated to other content types in the inheritance hierarchy.

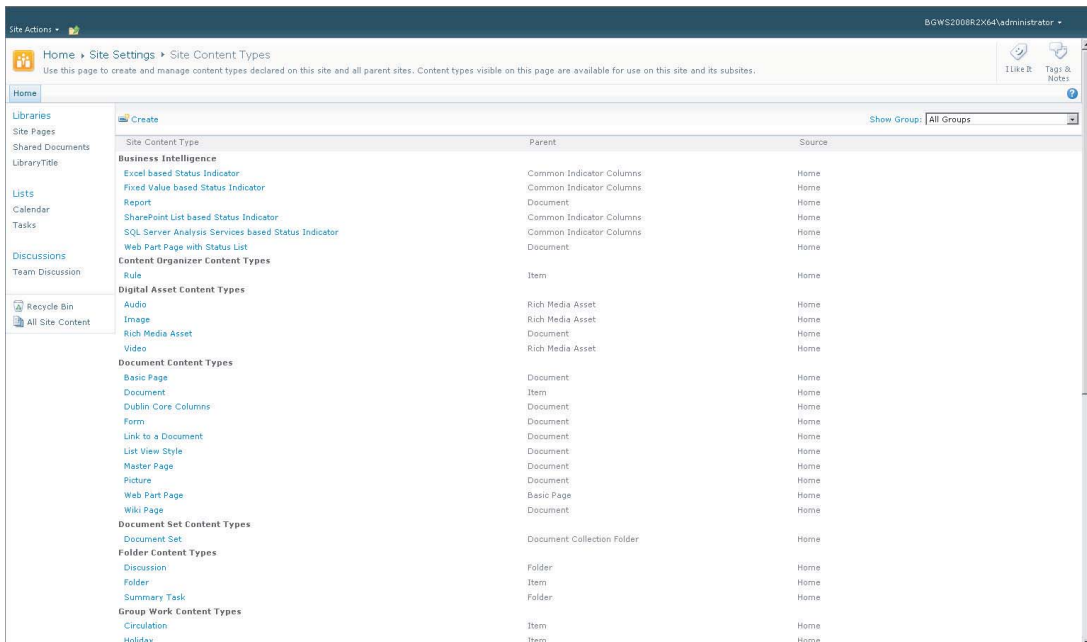


FIGURE 3-3

Out of the box, SharePoint provides a base hierarchical set of content types that represent different types of default content for built-in list types. Of all the built-in content types, only a few are relevant when defining a document taxonomy. Table 3-6 outlines these key content types.

TABLE 3-6: Base Content Types for Document Taxonomy

| NAME | DESCRIPTION |
|----------|--|
| System | The base content type from which all other content types are inherited |
| Item | The base content type for all list items |
| Document | The base content type for all list items in a document library that represent a document |
| Folder | The base content type for all list items in a document library that represent a folder |

The Content Type Programming Model

SharePoint provides a programming model for the interaction and management of content types within both the server-side and the client-side object models. Table 3-7 outlines the main classes used when manipulating content types programmatically.

TABLE 3-7: Main Class in the Object Model for Content Types

| CLASS | DESCRIPTION |
|-------------------------|---|
| SPContentType | Represents a content type within a list or Web |
| SPContentTypeCollection | Represents a collection of SPContentTypes |
| SPBuiltInContentTypeId | Enumeration that represents IDs for built-in content types |
| SPFieldLink | Represents a link to an existing SPField for use in a SPContentType |

The following two listings demonstrate two simple examples of using this programming model. Listing 3-5 shows how to create a site content type and add it to a document library as a list content type using the server-side object model. Listing 3-6 demonstrates using the client-side object model to assign a content type to an existing document in a document library and populate the associated fields with metadata.



Available for
download on
Wrox.com

LISTING 3-5: Creating a Site Content Type and Assigning It to a Document Library Using the Server-Side Object Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

continues

LISTING 3-5 (continued)

```

using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                //Get "Document" Content Type to be used as parent
                SPContentType documentContentType =
web.AvailableContentTypes
[SPBuiltInContentTypeId.Document];

                //Create new Content Type and add it to the web
                SPContentType customContentType =
                    new SPContentType(documentContentType,
web.ContentTypes,
                    "ContentTypeName");
                web.ContentTypes.Add(customContentType);

                //Add an existing Site Column
                SPField siteColumn = web.AvailableFields["ColumnTitle"];
                customContentType.FieldLinks.Add(new SPFieldLink(siteColumn));
                customContentType.Update();

                //Assign Content Type to existing library
                SPList list = web.Lists["LibraryTitle"];
                list.ContentTypes.Add(customContentType);
            }
        }
    }
}

```

The preceding code begins by gaining a reference to the `SPWeb` for the URL specified. A new `SPContentType` is then created and added to the `ContentTypes` collection of the `SPWeb`. Next, the `SPWeb` object is used to access the `AvailableFields` property, which contains a list of all the site columns that are valid for the Web. Using this property, a reference to the indicated field was returned and subsequently added to the newly created `SPContentType`. Finally, the content type was added to the specified list.



Available for
download on
Wrox.com

LISTING 3-6: Assigning a Content Type to an Existing Document and Populating Fields with Values Using the Client-Side Object Model

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint.Client;

```

```

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ClientContext context = new ClientContext("http://10.1.0.169"))
            {
                Web web = context.Web;

                //Get existing item from list
                List list = web.Lists.GetByTitle("LibraryTitle");
                context.Load(list, x => x.ContentTypes
                .IncludeWithDefaultProperties());

                //Get item from list
                ListItem item = list.GetItemById(12);
                context.ExecuteQuery();

                //Assign Content Type and Values
                foreach (ContentType contentType in list.ContentTypes)
                {
                    if (contentType.Name == "ContentTypeName")
                    {
                        item["ContentTypeId"] = contentType.Id;
                        break;
                    }
                }
                item["ColumnTitle"] = "Value";
                item.Update();

                //Execute actions
                context.ExecuteQuery();
            }
        }
    }
}

```

This listing showed the retrieval of a specific `ListItem`. Once a reference to the `ListItem` was retrieved, a `ContentType` with the indicated name was set for the item; finally, one of the column's values was changed.

Managed Metadata

Managed metadata is a column type provided by SharePoint Server that can be used in lists. It enables values to be provided from a centrally managed collection of hierarchically structured options (refer to Table 3-4 earlier in this chapter).

Within SharePoint, a hierarchical collection of managed metadata is referred to as a *term set*. Each item within this term set is referred to as a *term*. SharePoint refers to these term sets as managed metadata because they are managed from a central location independently of the columns themselves. A term set can be created as a local term set, which is defined within the context of a site collection, or it can be created as a global term set, in which case it can be used across site collections.

There is also a built-in term set called *enterprise keywords*, which consists of a single, nonhierarchical set of terms called the *keyword set*. The keyword set is essentially an ad-hoc list of terms that is built over time by user-entered values at the time of data entry. All other term sets have values that can be created dynamically at the time of data entry; this is a configurable option within the term set settings. All term set data is stored by SharePoint in a database referred to as the *term store*.

There are numerous benefits to using managed metadata. Managed metadata promotes uniformity of data by having a centrally managed list that provides consistent data throughout SharePoint. Ultimately, consistent data improves searching for information by ensuring that common terms are used accordingly. Managed metadata also enables term sets to be easily secured so that only users with the necessary permissions can manage the terms of the set. This also helps to control term consistency.

Figure 3-4 shows an example of a term set that represents major league baseball franchises.

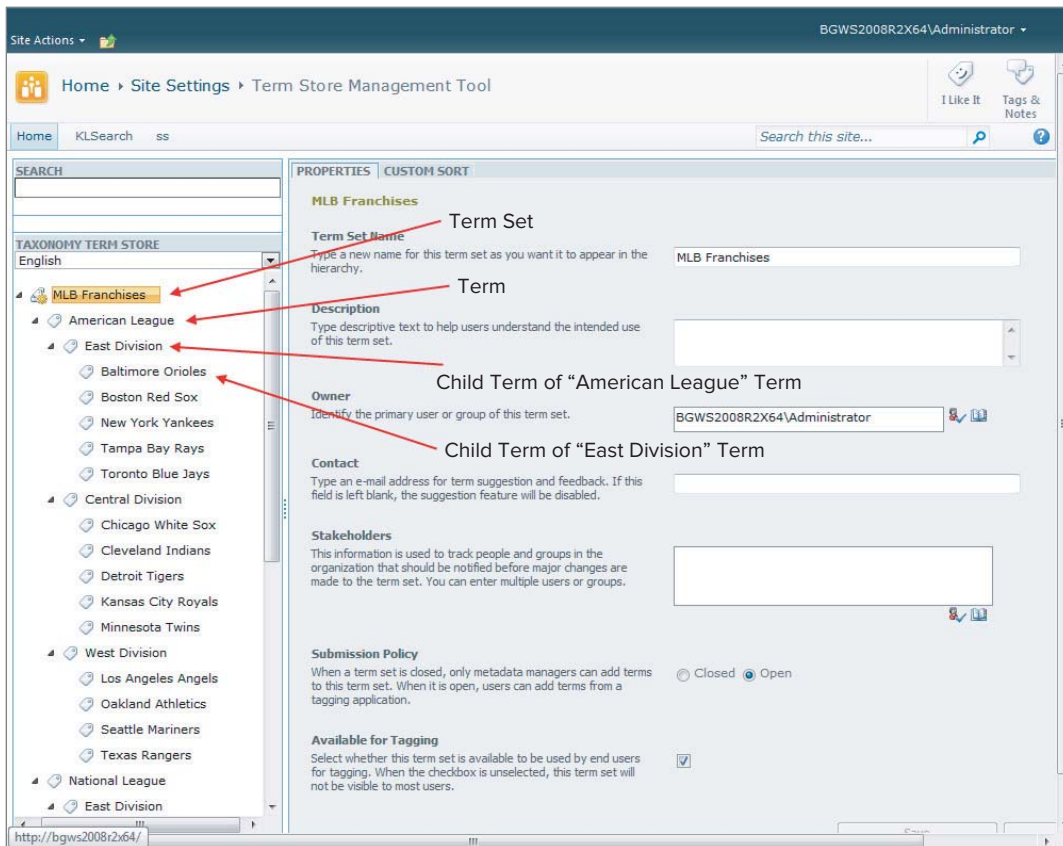


FIGURE 3-4

Administering Managed Metadata

In order to manage a term set, SharePoint provides a set of administrative pages referred to as the Term Store Management Tool. You can access this tool from the Managed Metadata Service

Application located in SharePoint 2010 Central Administration or from the Site Settings of any SharePoint site. The Managed Metadata Service Application is discussed in detail later in this chapter.

Creating a Global Term Set

You can create or import term sets using a CSV file. Manually creating a term set using the Term Set Management Tool as seen in Figure 3-5 is accomplished through the following steps:

1. Right-click a term set group and choose Create New Term Set.
2. Enter a name for the term set. Once a name is provided, the term set is created.
3. Provide a description for the term set to help users understand the term set's usage.
4. Provide the owner of the term set. By default, the user creating the term set is the owner.
5. Provide a contact e-mail address, which is used to receive feedback regarding the term set.
6. In the Stakeholders section, assign the person who should be notified of changes made to the term set.
7. Set the desired submission policy. A closed submission policy means that terms can be added only using the Term Store Management Tool. An open policy allows users to provide new values to the set at the time of data entry.
8. Select when the term set is available for tagging. This option allows the term set to appear when using SharePoint tagging.
9. Click Save.

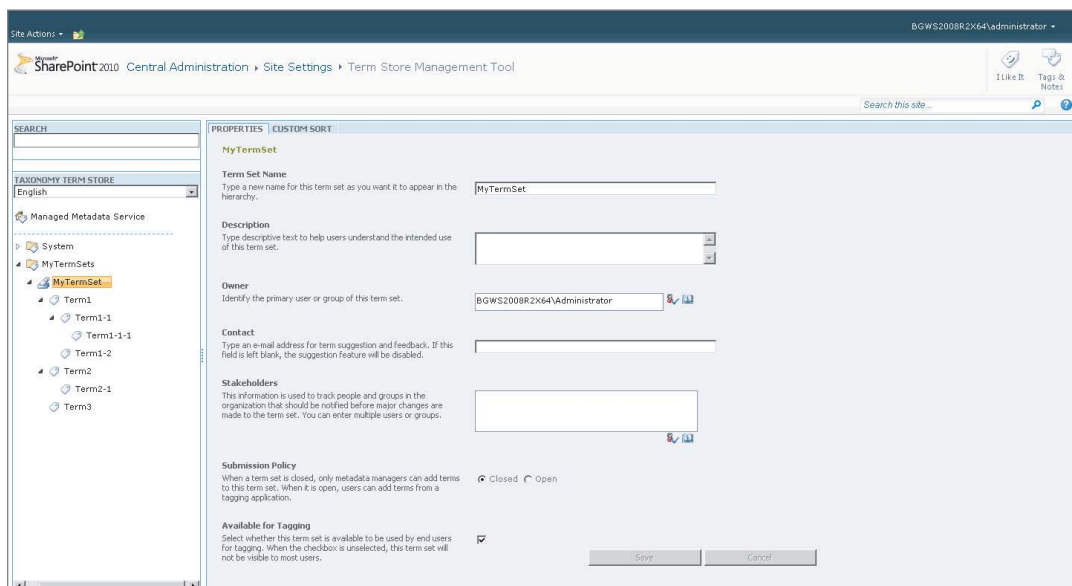


FIGURE 3-5

Using a Term Set in a Column

After a term set is created, it can then be used as the data source for a Managed Metadata column by selecting the appropriate term set at the time of column creation, as seen in Figure 3-6.

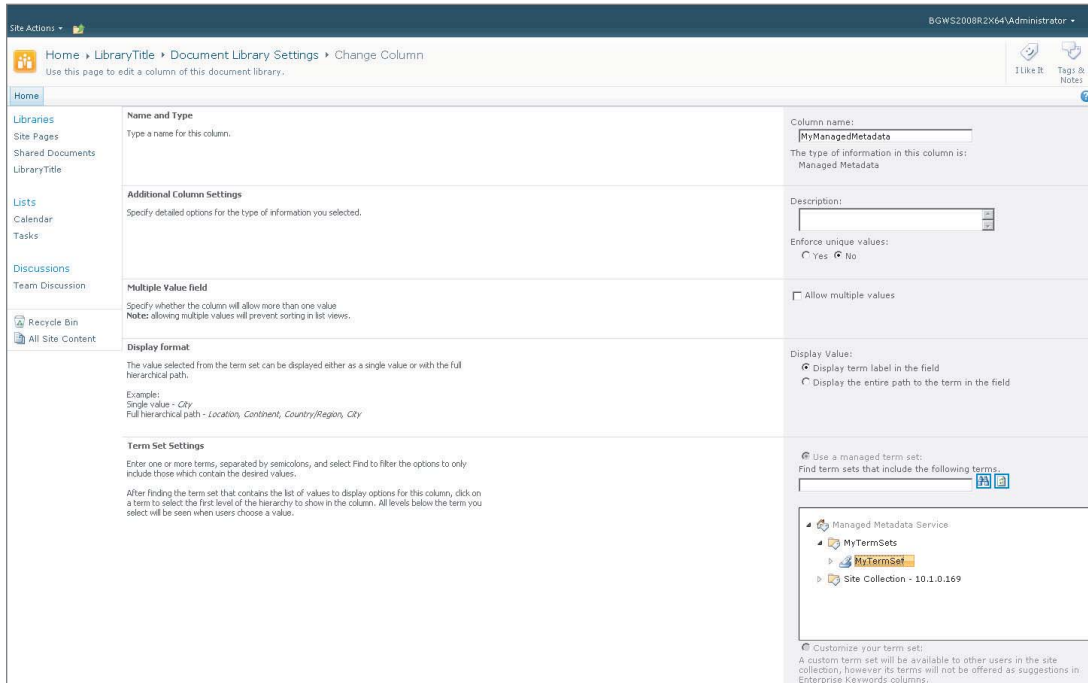


FIGURE 3-6

When users enter data for this column, they are presented with a list of suggestions based on the terms in the term set as they type. This is shown in Figure 3-7.

Users can also click the icon to the right of the managed metadata control, which opens the term selection dialog. This dialog displays the term set as a tree view, which enables users to view the term set in a hierarchical manner. If open submissions are allowed, new terms can be added to the term set from this dialog. An example of this dialog is shown in Figure 3-8.

The Managed Metadata Programming Model

SharePoint provides an extensive programming model for interacting with the term store. This programming model is referred to as the Metadata and Taxonomy Programming Model within SharePoint documentation. Taxonomy in this sense shouldn't be confused with the document taxonomy discussed earlier. The term "taxonomy" is used in this sense to describe the hierarchical nature of the term sets themselves.

The bulk of this programming model is provided through a collection of SharePoint namespaces, listed in Table 3-8. Note that these namespaces consist of server-side classes that must be executed on the SharePoint server, with the exception of the web service that SharePoint provides for client-side interaction. The main classes used in these namespaces are listed in Table 3-9. Listing 3-7

demonstrates the usage of this programming model to create a term set, while Listing 3-8 demonstrates reading existing term set data.

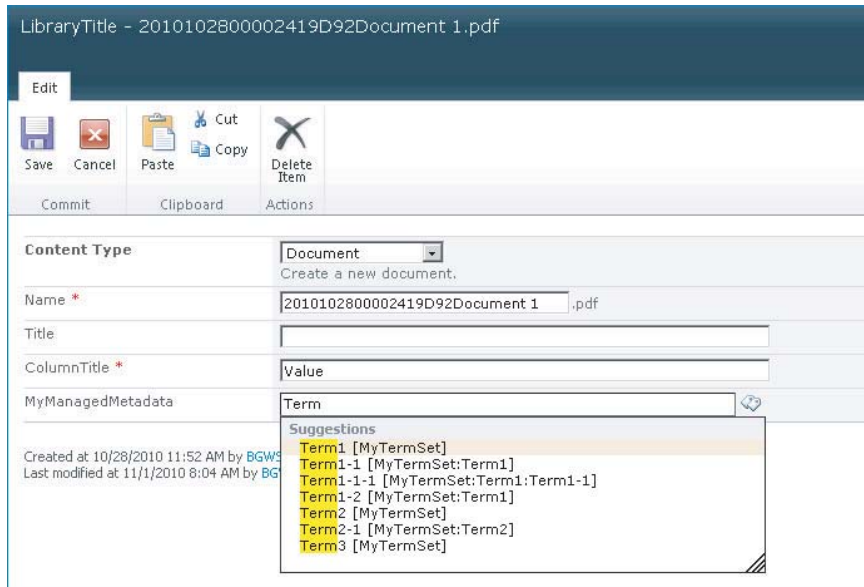


FIGURE 3-7

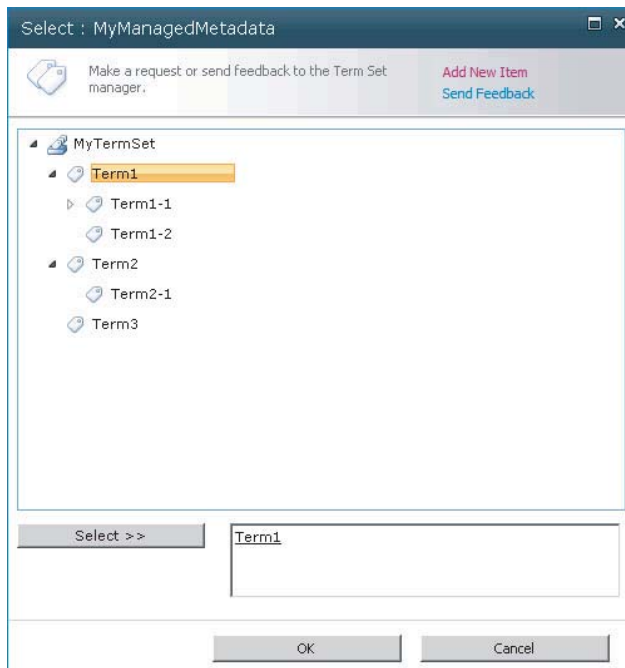


FIGURE 3-8

TABLE 3-8: Metadata and Taxonomy Namespaces

| NAMESPACE | DESCRIPTION |
|--|--|
| <code>Microsoft.SharePoint.Taxonomy</code> | Provides the core classes for managing term set data |
| <code>Microsoft.SharePoint.Taxonomy.ContentTypeSync</code> | Provides the core classes for ContentType synchronization among site collections |
| <code>Microsoft.SharePoint.Taxonomy.Generic</code> | Provides classes for storing collections of Managed Metadata items |
| <code>Microsoft.SharePoint.Taxonomy.WebServices</code> | Provides the classes used for the Taxonomy Web Service used for remote programmability |

TABLE 3-9: Main Classes for the Managed Metadata Object Model

| CLASS | DESCRIPTION |
|--|--|
| <code>ChangedGroup</code> | Represents a Group object change |
| <code>ChangedItem</code> | Represents a TermStore object change |
| <code>ChangedItemCollection</code> | Represents a collection of ChangedItem objects |
| <code>ChangedItemType</code> | Represents the item type for the changed object |
| <code>ChangedOperationType</code> | Represents the operation type that caused the change |
| <code>ChangedTerm</code> | Represents the Term object that changed |
| <code>ChangedTermSet</code> | Represents the TermSet that changed |
| <code>ChangedTermStore</code> | Represents TermStore that changed |
| <code>Group</code> | Represents a container of TermStore objects |
| <code>GroupCollection</code> | Represents a collection of Group objects |
| <code>HiddenListFullSyncJobDefinition</code> | Represents the timer job responsible for keeping TaxonomyField objects current |
| <code>ImportManager</code> | Provides the capability to import TermSet instances into a Group |
| <code>Label</code> | Represents a language-specific name of a Term object |
| <code>LabelCollection</code> | Stores a collection of Label objects |
| <code>StringMatchOption</code> | Indicates which string matching to use for string comparison |

| CLASS | DESCRIPTION |
|------------------------------|--|
| TaxonomyField | Represents a TaxonomyField object. The TaxonomyField inherits from the SPFieldLookup class. |
| TaxonomyFieldControl | Control used to edit a TaxonomyField object |
| TaxonomyFieldEditor | CodeBehind class used for the new Managed Metadata column page |
| TaxonomyFieldValue | Represents the value of a TaxonomyField object |
| TaxonomyFieldValueCollection | Represents the multi-value object for a TaxonomyField object |
| TaxonomyItem | Represents the base class TaxonomyItem class for an item in the term store |
| TaxonomyRights | Identifies the bitmask that represents the taxonomy permissions |
| TaxonomySession | Represents the TermStore objects for a SPSite object. |
| TaxonomyWebTaggingControl | A generic web control for selecting terms |
| Term | Represents a term or a keyword in a managed metadata hierarchy |
| TermCollection | Stores a collection of Term objects |
| TermSet | Represents a hierarchical or flat set of Term objects |
| TermSetCollection | Stores a collection of TermSet objects |
| TermSetItem | Provides an abstraction of the TaxonomyItem object that is a parent of Term objects |
| TermStore | Represents the storage of Group objects, TermSet objects, and Term objects |
| TermStoreCollection | Stores a collection of TermStore objects |
| TreeControl | Constructs a JSON representation of the data source and initializes an instance of the client tree control |



LISTING 3-7: Creating a Term Set

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
```

continues

LISTING 3-7 *(continued)*

```

using Microsoft.SharePoint.Taxonomy;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            {
                //Get Term Store
                TaxonomySession session = new TaxonomySession(site);
                TermStore termStore = session.TermStores[0];

                //Create a Term Set
                Group group = termStore.CreateGroup("MyTermGroup");
                TermSet termSet = group.CreateTermSet("MyTermSet");

                //Add Terms to the TermSet
                Term childTerm = null;

                Term parentTerm = termSet.CreateTerm("ParentTerm", 1033);
                childTerm = parentTerm.CreateTerm("ChildTerm1", 1033);
                childTerm = parentTerm.CreateTerm("ChildTerm2", 1033);

                //Commit changes
                termStore.CommitAll();
            }
        }
    }
}

```

Listing 3-7 begins by obtaining a reference to a `TermStore` object for the `SPSite` specified by the provided URL. Upon getting this reference, the `TermStore` was used to create a new `Group`. Using the newly created `Group`, a `TermSet` was created along with a small hierarchy of `Terms`. Finally, the `CommitAll` method was called on the `TermStore` in order to trigger the changes to be saved to the underlying database.

**LISTING 3-8: Reading Term Set Data**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Taxonomy;

namespace Chapter3Code
{

```

```

class Program
{
    static void Main(string[] args)
    {
        using (SPSite site = new SPSite("http://10.1.0.169"))
        {
            TaxonomySession session = new TaxonomySession(site);
            TermStore termStore = session.TermStores[0];

            foreach (Group group in termStore.Groups)
            {
                Console.WriteLine(group.Name + " -->");

                foreach (TermSet termSet in group.TermSets)
                {
                    Console.WriteLine(termSet.Name.PadLeft(10));

                    foreach (Term term in termSet.Terms)
                    {
                        PrintTermHierarchy(term, 0);
                    }
                }
            }

            Console.ReadLine();
        }

        private static void PrintTermHierarchy(Term term, int level)
        {
            Console.WriteLine(term.Name.PadLeft(level + 20));

            foreach (Term childTerm in term.Terms)
            {
                PrintTermHierarchy(childTerm, level + 30);
            }
        }
    }
}

```

Listing 3-8 shows the reading of existing `TermSet` data by getting a reference to a `TermStore` object. By iterating through each `Group` stored in the `TermStore` and each `TermSet` stored in each `Group`, the `Terms` in the `TermSet` are then recursively printed to the console.

The Managed Metadata Service

The *managed metadata service application* is an application provided by SharePoint Server that enables you to publish managed metadata from the term store to subscribing site collections throughout the SharePoint farm. Optionally, administrators can configure the managed metadata service to publish content types as well using a concept known as *content type syndication*.

When managed metadata is enabled in SharePoint, a connection to the managed metadata service application is automatically created. Each application identifies the database that will act as the term

store for managed metadata. When terms are created, deleted, or updated, the managed metadata service reflects these changes in the associated term store database. The subscribing web applications use the connection to retrieve the term store data.

It is possible to create multiple managed metadata service applications, but each instance must specify its own term store. These term stores can be shared from multiple site collections.

Content Type Syndication

The act of sharing a site collection's content types with other site collections in the farm is referred to as *content type syndication*. Content type syndication enables an administrator to specify a single site collection per managed metadata service to act a *content type hub*. If a site collection is specified as a content type hub, the managed metadata service will use this site collection's content type gallery as the source for publishing the content types contained in this gallery to the other site collections that have a connection to the managed metadata service application.

Content type syndication enables an organization to have a consistent taxonomy by allowing the same content types to be used within multiple site collections. This eliminates or reduces the need to define content types within each individual site collection and greatly simplifies the management of content types.

Configuring a Site Collection to Be a Content Type Hub

In order to configure a site collection to act as a content type hub, you must enable the content type syndication hub feature. To do this, perform the following steps, as seen in Figure 3-9:

1. Navigate to Site Settings for the root site in the site collection.
2. Within the Site Settings window, under Site Collection Administration, click Site Collection Features.
3. Next to Content Type Syndication Hub, click Activate.

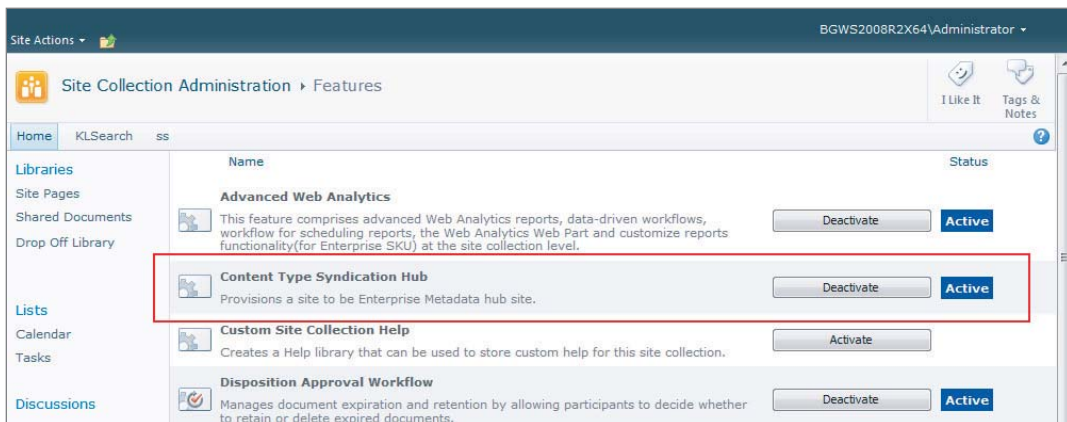


FIGURE 3-9

Determining Content Type Hubs for a Site Collection

To determine the content type hubs being used to publish content types to a specific web application, perform the following steps, as seen in Figure 3-10:

1. Navigate to Site Settings for the root site in the site collection.
2. Within the Site Settings window, under Site Collection Administration, click Content Type Publishing.
3. The service applications publishing content types to this site collection will be listed, along with a link to the content type gallery for the designated hub.

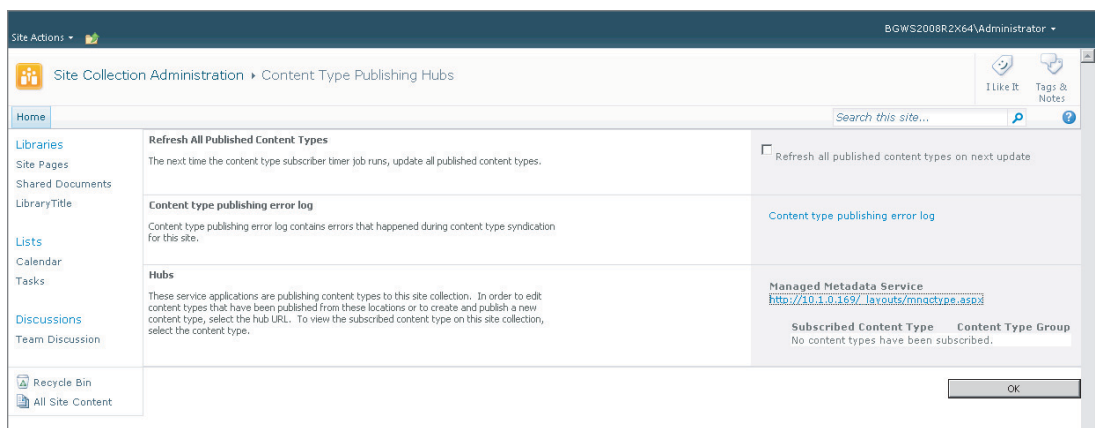


FIGURE 3-10

The Content Type Syndication Programming Model

SharePoint Server provides a server-side class named `ContentTypePublisher` for customizing the publishing and unpublishing of content types within a content type hub. Listing 3-9 demonstrates usage of this class.



LISTING 3-9: Programmatically Publishing a Content Type

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Taxonomy.ContentTypeSync;

namespace Chapter3Code
{
```

continues

LISTING 3-9 *(continued)*

```

class Program
{
    static void Main(string[] args)
    {
        using (SPSite hubSite = new SPSite("http://10.1.0.169"))
        {
            SPWeb rootWeb = hubSite.RootWeb;

            //Get "Document" Content Type to be used as parent
            SPContentType documentContentType = rootWeb
                .AvailableContentTypes[SPBuiltInContentTypeId.Document];

            //Create new Content Type and add it to the web
            SPContentType customContentType =
                new SPContentType(documentContentType,
                    rootWeb.ContentTypes,
                    "ContentTypeName");
            rootWeb.ContentTypes.Add(customContentType);

            //Determine if SPSite is a Content Type Hub
            if (ContentTypePublisher.IsContentTypeSharingEnabled(hubSite))
            {
                ContentTypePublisher publisher
                    = new ContentTypePublisher(hubSite);

                //If Content Type is not published, then publish
                if (!publisher.IsPublished(customContentType))
                    publisher.Publish(customContentType);

                //You can also, unpublish a Content Type
                if (publisher.IsPublished(customContentType))
                    publisher.Unpublish(customContentType);
            }
        }
    }
}

```

Listing 3-9 demonstrates using the `ContentTypePublisher` class to publish and unpublish a `SPContentType`. After obtaining a reference to the `SPWeb` indicated by the provided URL, a new `SPContentType` was created and added to the web's content types collection. Upon doing this, the `ContentTypePublisher` class was then used to determine whether content type sharing is enabled for the Web; if so, the `ContentTypePublisher` object is used to publish and unpublish the content type by calling the respective methods.

Management of Managed Metadata Service Applications

The following sections outline the process for creating, updating, and deleting managed metadata service applications. In order to perform these actions, you must be a member of the local Administrators group on the computer running SharePoint Central Administration, and a member of the Farm Administrators group in SharePoint.

Creating a Managed Metadata Service Application

To create a managed metadata service application, follow these steps:

1. Within SharePoint Central Administration, under Application Management, select Manage Service Applications.
2. On the Ribbon, click New and then select Managed Metadata Service.
3. On the Managed Metadata Service creation page, provide a name to identify the new service instance.
4. Provide the name of the database server and database name that will host the term store (optionally, provide the failover database server).
5. Select the desired method of authentication.
6. Select the appropriate application pool or choose to create a new application pool.
7. If the service instance will also be used for content type syndication, provide the URL that points to the site collection that will be designated as the content type hub.
8. In order to have a connection to this service instance automatically created for new web applications, select the option to add the service application to the farm's default list.
9. Click OK.

Updating a Managed Metadata Service Application

You can update a managed metadata service application by following these steps:

1. Within SharePoint Central Administration, under Application Management, select Manage Service Applications.
2. Select the appropriate service application from the application list.
3. On the Ribbon, select Properties.
4. On the Managed Metadata Service creation page, update the desired configuration properties.
5. Click OK.

Deleting a Managed Metadata Service Application

Follow these steps to delete a managed metadata service application:

1. Within SharePoint Central Administration, under Application Management, select Manage Service Applications.
2. Select the appropriate service application from the application list.
3. On the Ribbon, click Delete.
4. Click OK.

Location-Based Metadata Defaults

Location-based metadata defaults is a feature provided by Microsoft SharePoint Server that enables administrators to manage the default values of SharePoint properties based on the uploaded document's location within a document library. When documents are uploaded to SharePoint, precedence rules are evaluated by an `ItemAdded` list event handler that is automatically attached during configuration of the first location-based metadata default for a document library. The rules for evaluating the default value for a property are as follows:

1. If a property contains a value that is the same as the column default, the location-based metadata default is applied.
2. If a property does not contain a value, the location-based metadata default is application.
3. If a property contains a value that is different from the column default, the value is retained.

Location-based metadata defaults function in a hierarchical fashion. When defaults are configured for a parent folder, all subfolders inherit the same default values unless overridden by an individual subfolder. Removing all location-based metadata defaults will cause SharePoint to remove the `ItemAdded` event handler from the document library.

Configuring Location-Based Metadata Defaults

To configure location-based metadata defaults, follow these steps:

1. Within List Settings, under General Settings, click Column Default Value Settings.
2. Select a location from the list of folders in the left-hand pane, as shown in Figure 3-11.
3. Click the column in the right-hand pane for which you would like to configure a default value.
4. Select Use this Default Value, as shown in Figure 3-12.
5. Provide a value to use as the default.
6. Click OK.



FIGURE 3-11

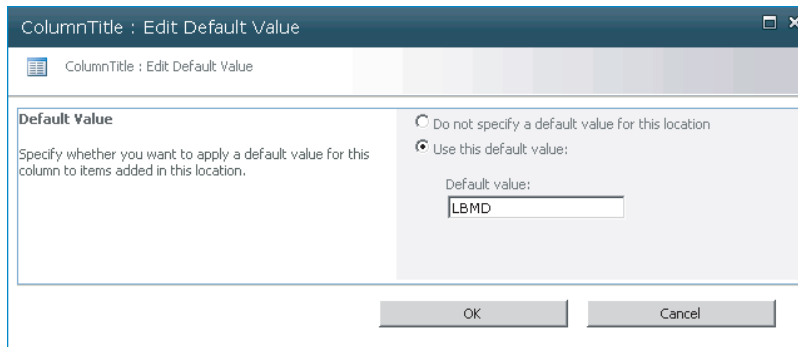


FIGURE 3-12

The Location-Based Metadata Defaults Programming Model

SharePoint Server provides a server-side class named `MetadataDefaults` for programmatically configuring location-based metadata default values for a document library. Listing 3-10 demonstrates how to use this class to set, get, and remove a field's default value.



Available for
download on
Wrox.com

LISTING 3-10: Configuring Location-Based Metadata Defaults

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.DocumentManagement;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                SPList list = web.Lists["LibraryTitle"];
                MetadataDefaults defaults = new MetadataDefaults(list);

                //Set Default Value
                defaults.SetFieldDefault(list.RootFolder,
```

continues

remember that SharePoint lists support a maximum of 20 indices. When the index limit is reached, automatic creation of indices is not possible.

TABLE 3-10: Metadata Navigation Field Types

| FIELD TYPE | NAVIGATION HIERARCHY | KEY FILTER | SINGLE INDEX | COMPOUND INDEX |
|---------------------------------|----------------------|------------|--------------|----------------|
| Content Type | X | X | X | X |
| Choice (single value) | X | X | X | |
| Managed Metadata (single value) | X | X | X | X |
| Managed Metadata (multi-value) | X | X | X | |
| Person or Group (single value) | | X | X | X |
| Person or Group (multi-value) | | X | | |
| Date and Time | | X | X | X |
| Number | | X | X | X |

Configuring Metadata Navigation

In order to use metadata navigation for a document library, an administrator must configure the necessary options. The following procedure outlines the steps necessary to configure metadata navigation:

1. Within List Settings, under General Settings, click Metadata Navigation Settings. You will be taken to the Metadata Navigation Settings page as shown in Figure 3-13.
2. Select the desired fields to act as hierarchical filters from the list of supported fields.
3. Select the desired fields to act as key filters from the list of supported fields.
4. Select whether to enable automatic creation of indices on the list.
5. Click OK.

Using Metadata Navigation

When metadata navigation is enabled, SharePoint creates a navigation panel below the Quick Launch bar on the Document Library page. This pane enables users to select a navigation hierarchy filter and any combination of key filters in order to filter the library view for the desired documents. This navigation pane also provides the capability to clear any selected filters. An example of the navigation pane is shown in Figure 3-14.

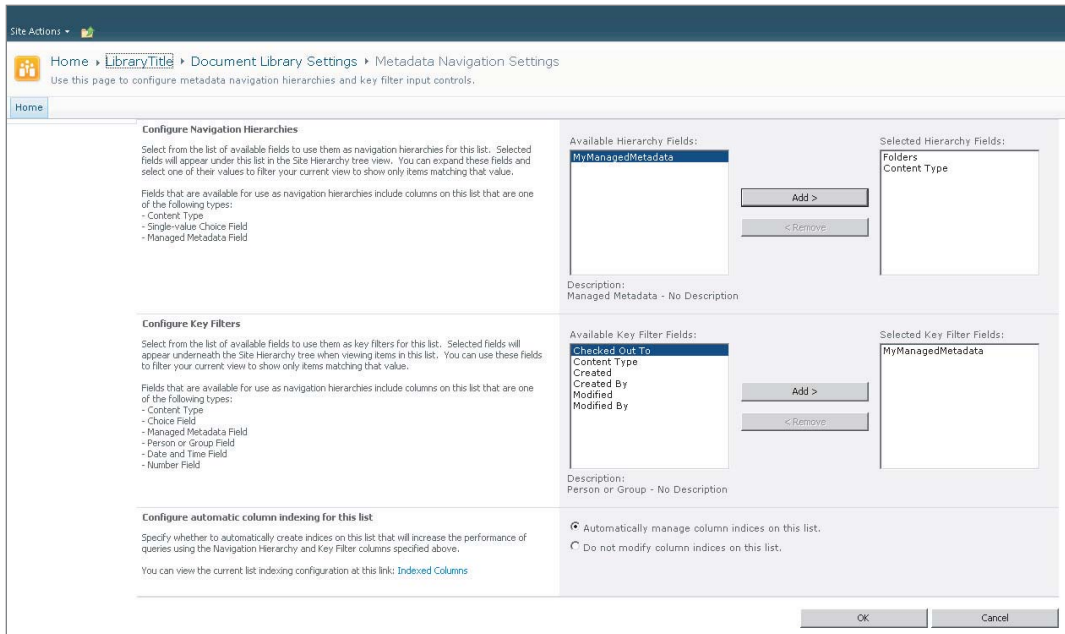


FIGURE 3-13

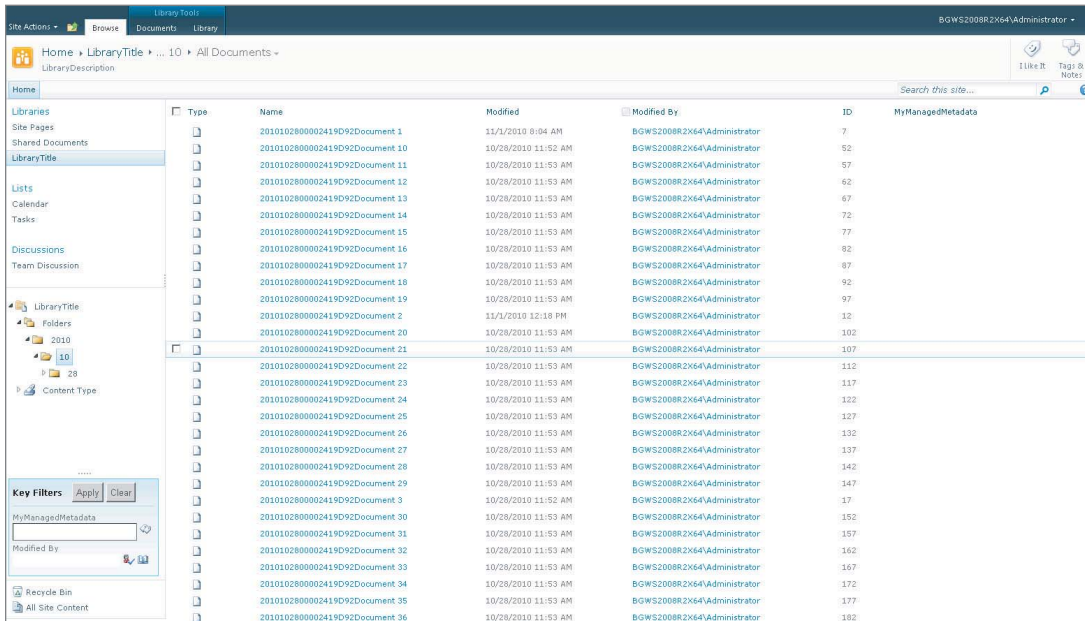


FIGURE 3-14

The Managed Metadata Navigation Programming Model

Programmatically using metadata navigation is made possible through a collection of classes contained in the `Microsoft.Office.DocumentManagement.MetadataNavigation` namespace. Table 3-11 lists the main classes used in this programming model. Listing 3-11 provides an example of how to use this programming model to create navigation filters.

TABLE 3-11: Classes Used for Metadata Navigation

| CLASS | DESCRIPTION |
|--|---|
| <code>MetadataNavigationContext</code> | Responsible for tracking the state of metadata navigation controls and the List View Web Part during an HTTP request |
| <code>MetadataNavigationHierarchy</code> | Represents a hierarchy of <code>ManagedNavigationItem</code> objects |
| <code>MetadataNavigationItem</code> | Represents an <code>SPIItem</code> object associated with the <code>MetadataNavigationHierarchy</code> object |
| <code>MetadataNavigationKeyFilter</code> | Represents a key filter used to perform filtering |
| <code>MetadataNavigationSettings</code> | Responsible for configuration of settings on a <code>ManagedNavigationItem</code> of an <code>SPList</code> to control the display of filters on list views |



LISTING 3-11: Programmatically Adding Metadata Navigation Filters to a List

Available for
download on
Wrox.com

```
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.DocumentManagement.MetadataNavigation;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                SPList list = web.Lists["LibraryTitle"];

                //Get settings from list
                MetadataNavigationSettings settings = MetadataNavigationSettings
                .GetMetadataNavigationSettings(list);
            }
        }
    }
}
```

continues

LISTING 3-11 *(continued)*

```

        //Add folder hierarchy filter to settings if not already configured
        MetadataNavigationHierarchy folderHierarchy =
            MetadataNavigationHierarchy.CreateFolderHierarchy();

        if (settings
            .FindConfiguredHierarchy(folderHierarchy.FieldId) == null)
            settings.AddConfiguredHierarchy(folderHierarchy);

        //Add field key filter to settings if not already configured
        SPField field = list.Fields["Modified"];

        if (settings.FindConfiguredKeyFilter(field.Id) == null)
            settings
            .AddConfiguredKeyFilter(new MetadataNavigationKeyFilter(field));

        //Set settings configuration on the list
        MetadataNavigationSettings.SetMetadataNavigationSettings(list,
            settings,
            true);
    }
}
}
}

```

This code starts out by obtaining a reference to the `SPWeb` indicated by the provided URL. The `SPWeb` object is then used to reference a specific `List`. Using this `List` object, a reference to the `MetadataNavigationSettings` object was retrieved. `MetadataNavigationSettings` were then used to create both a navigation hierarchy filter and a key filter. Finally, the `SetMetadataNavigationSetting` method was called, which triggers the actual changes to take place.

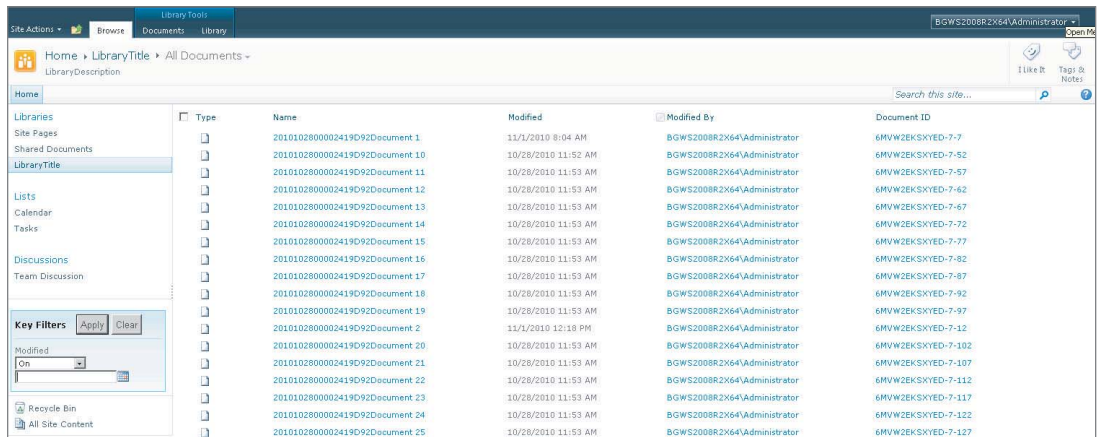
The Document ID Service

The document ID service is a SharePoint Server feature that provides the capability to automatically assign unique identifiers to a document in order to retrieve the document independent of its current location.

When the document ID service is enabled within the Site Collection Features administration page, SharePoint automatically adds three columns to the Document and Document Set content types. The first column that is added is named Document ID. This column is used to store a unique identifier for the document. The second column that is added is named Static URL. This column is used to store the URL that can be used to access the document regardless of its storage location. The third column is named PersistID. This column is used by SharePoint to determine if the existing document ID will be retained or reassigned when a document is copied to a new location.

In order for the document ID service to work, SharePoint assigns an `ItemAdded` event handler to all document libraries. This is used to assign a unique identifier to a new document when it is uploaded. Any documents that do not already have an identifier are assigned one through a SharePoint timer job called the Document ID assignment job that is registered when the document ID service is enabled. When an identifier is assigned to a document, SharePoint sets the `PreserveID` field to true to indicate that an identifier is assigned and that this identifier should be retained if the file is copied to a new location.

Figure 3-15 shows a view of a document library in which the Document ID column is displayed.

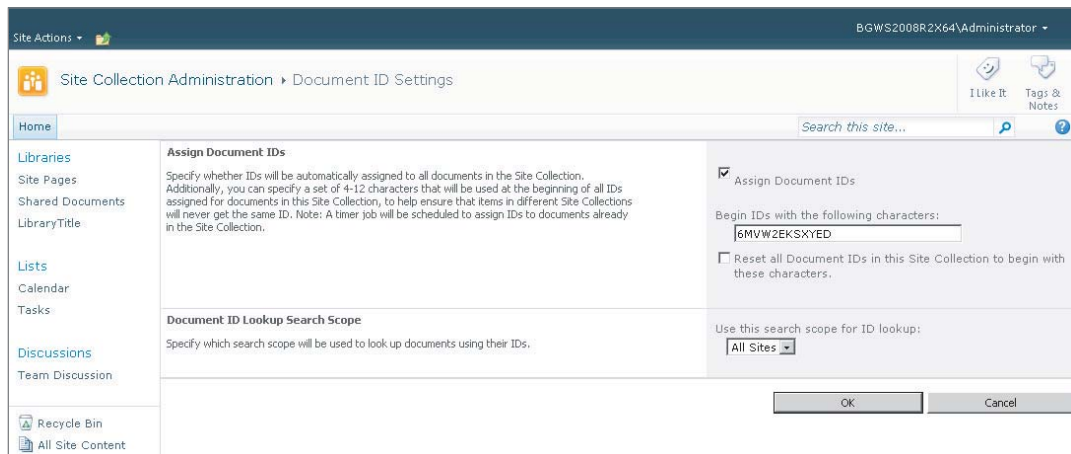


| Type | Name | Modified | Modified By | Document ID |
|--------------------------|--------------------------------|---------------------|-----------------------------|--------------------|
| <input type="checkbox"/> | 2010102800002419D92Document 1 | 11/1/2010 8:04 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-7 |
| <input type="checkbox"/> | 2010102800002419D92Document 10 | 10/28/2010 11:52 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-52 |
| <input type="checkbox"/> | 2010102800002419D92Document 11 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-57 |
| <input type="checkbox"/> | 2010102800002419D92Document 12 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-62 |
| <input type="checkbox"/> | 2010102800002419D92Document 13 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-67 |
| <input type="checkbox"/> | 2010102800002419D92Document 14 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-72 |
| <input type="checkbox"/> | 2010102800002419D92Document 15 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-77 |
| <input type="checkbox"/> | 2010102800002419D92Document 16 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-82 |
| <input type="checkbox"/> | 2010102800002419D92Document 17 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-87 |
| <input type="checkbox"/> | 2010102800002419D92Document 18 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-92 |
| <input type="checkbox"/> | 2010102800002419D92Document 19 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-97 |
| <input type="checkbox"/> | 2010102800002419D92Document 2 | 11/1/2010 12:18 PM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-12 |
| <input type="checkbox"/> | 2010102800002419D92Document 20 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-102 |
| <input type="checkbox"/> | 2010102800002419D92Document 21 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-107 |
| <input type="checkbox"/> | 2010102800002419D92Document 22 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-112 |
| <input type="checkbox"/> | 2010102800002419D92Document 23 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-117 |
| <input type="checkbox"/> | 2010102800002419D92Document 24 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-122 |
| <input type="checkbox"/> | 2010102800002419D92Document 25 | 10/28/2010 11:53 AM | BGWS2008R2X64\Administrator | 6MVW2EKSKYED-7-127 |

FIGURE 3-15

The default format for a document identifier is *[Prefix]-[List ID]-[List Item ID]*. Administrators can customize this format by providing a value for the prefix portion of the document identifier. The following procedure outlines how this is accomplished:

1. From the Site Settings window of the root site in a site collection, under Site Collection Administration, click Document ID Settings. This will take you to the Document ID Settings page, as shown in Figure 3-16.
2. Check whether to enable the assignment of document IDs.
3. Provide the desired prefix value to be used.
4. Check whether to reset all identifiers already assigned with a new value conforming to the new format.
5. Click OK.



Site Collection Administration > Document ID Settings

Assign Document IDs

Specify whether IDs will be automatically assigned to all documents in the Site Collection. Additionally, you can specify a set of 4-12 characters that will be used at the beginning of all IDs assigned for documents in this Site Collection, to help ensure that items in different Site Collections will never get the same ID. Note: A timer job will be scheduled to assign IDs to documents already in the Site Collection.

Assign Document IDs

Begin IDs with the following characters:

Reset all Document IDs in this Site Collection to begin with these characters.

Document ID Lookup Search Scope

Specify which search scope will be used to look up documents using their IDs.

Use this search scope for ID lookup:

OK Cancel

FIGURE 3-16

The Document ID Programming Model

If the built-in capability for customizing document identifiers is not sufficient, it is possible to create a custom provider, which is used to override the default provider used by SharePoint to assign an identifier. This allows organizations to fully customize the way a document identifier appears. For instance, an organization may choose to create an identifier that is based on particular metadata associated with the document. This enables the identifier to convey some contextual information with regard to the document itself. Listing 3-12 demonstrates how to create a custom document ID provider by implementing the abstract `DocumentIdProvider` class located in the `Microsoft.Office.DocumentManagement` namespace. Registering this custom provider can be accomplished with a feature receiver using the `SetDefaultProvider` or `SetProvider` method of the `DocumentId` class found in the same namespace. A feature receiver is a module that can be implemented and deployed along side a feature to execute custom code at the time of feature installation, activation, deactivation, or removal.



LISTING 3-12: Creating a Custom Document ID Provider

Available for
download on
Wrox.com

```
using System;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.DocumentManagement;

namespace Chapter3Code
{
    public class CustomDocumentIDProvider : DocumentIdProvider
    {
        public override bool DoCustomSearchBeforeDefaultSearch
        {
            get
            {
                //Return true to use call the GetDocumentUrlsById method
                //prior to using SharePoint's default search method
                throw new NotImplementedException();
            }
        }

        public override string GenerateDocumentId(SPListItem listItem)
        {
            //Return a custom Document ID based of field values of the list item
            throw new NotImplementedException();
        }

        public override string[] GetDocumentUrlsById(SPSite site,
                                                    string documentId)
        {
            //Perform custom search logic for finding documents
            //based on provided Document ID
            throw new NotImplementedException();
        }
    }
}
```



```

    }

    public override string GetSampleDocumentIdText(SPSSite site)
    {
        //Return a sample Document ID that is used by SharePoint
        //to guide users in searching for documents by Document ID
        throw new NotImplementedException();
    }
}
}
}

```

Document Sets

SharePoint Server provides a feature that allows multiple documents to be managed as a single entity. It accomplishes this through what is referred to as a *document set*. Much like folders, document sets are items containing documents that can be added to a document library, but they also behave like a document in terms of the functionality they provide. A typical usage for a document set is to manage a group of documents that represent a single workable product. For example, imagine a loan packet for a bank loan. This packet may consist of multiple individual documents. Using document sets, rather than managing each individual document separately, it is possible to manage the entire packet as a single unit.

Implementing Document Sets

Document sets are implemented as their own content type, inheriting from the folder content type within SharePoint. Therefore, they possess all the properties and behaviors of standard folders but add extensive logic and behavior to support a number of work product management features. Using document sets, you can track their version history, run workflows associated with the document set, limit the content types allowed within the document set, and set shared columns, which are used to synchronize values among all documents in the document set. When a document set is selected within a document library, the user is taken to a special page called the Document Set Welcome Page. This page provides a view of the documents contained within the set, as well as a display of the columns designated as Welcome Page columns. Figure 3-17 shows an example of a Document Set Welcome Page.

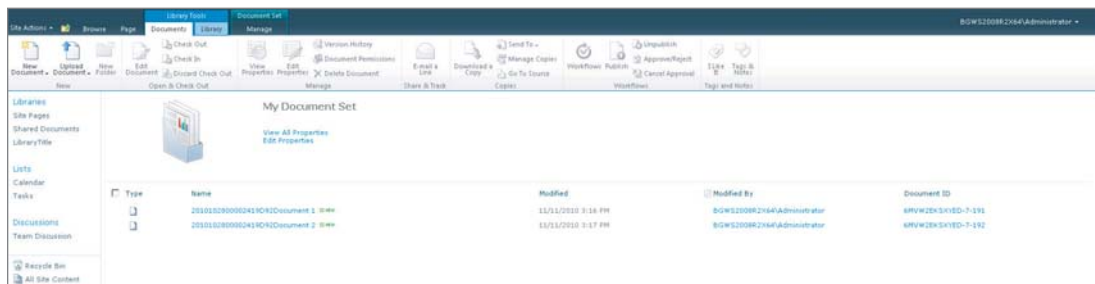


FIGURE 3-17

Creating Custom Document Sets

It is possible to create a custom document set by inheriting from the Document Set content type. Once you have created this content type, you can add new columns to it and configure all the appropriate document set settings. This content type can then be added to individual libraries in the same way as any other site content type.

Using Document Sets

In order to use a document set, the Document Set feature must be enabled within Site Collection Features. After the feature is enabled, the Document Set content type is added to the site's content type gallery. This content type can then be used to create custom document sets by creating content types that inherit from the Document Set content type. Any of these content types can then be added to individual document libraries.

Creating a Custom Document Set

The following procedure demonstrates how to create a custom document set content type:

1. From the Site Content Type Gallery, click Create.
2. Provide a name and description for the content type.
3. Select the Document Set content type to inherit from.
4. Click OK.

Configuring a Document Set

The following procedure demonstrates how to configure a custom Document Set content type:

1. From the Content Type settings page, add or create any desired site columns.
2. Click Document Set Settings. You will be taken to the Document Set Settings page, as shown in Figure 3-18.
3. Choose the allowed content types from the list provided.
4. Choose the default content that will be created when a new instance is provisioned.
5. Select the shared columns.
6. Choose the columns to be displayed on the Welcome Page.
7. Select whether to update the Welcome Page about inheriting content types.
8. Select whether to update all inheriting content types with changes made to these settings.
9. Click OK.

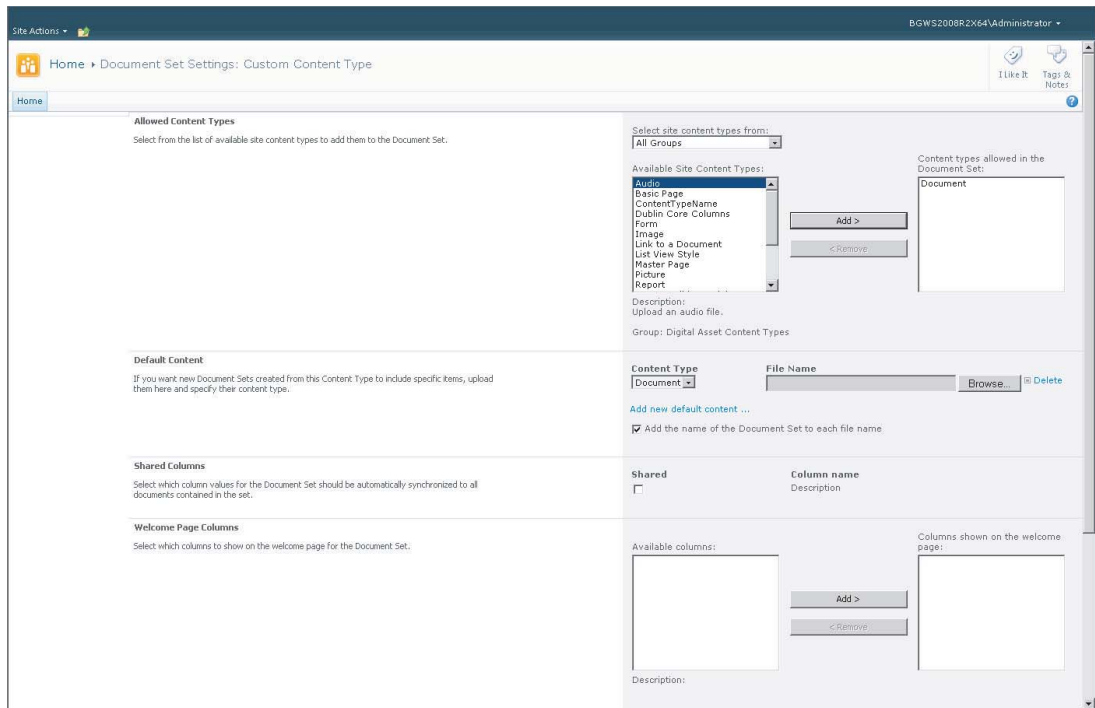


FIGURE 3-18

Creating an Instance of a Document Set

The following procedure demonstrates how to create a document set instance using a custom Document Set content type.

1. After adding the Document Set content type to the document library, select it from the New button in the Ribbon.
2. Provide a name, a description, and any other property values.
3. Click OK.

The Document Set Programming Model

SharePoint Server provides a server-side programming model for managing document sets within the `Microsoft.Office.DocumentManagement.DocumentSets` namespace. Table 3-12 lists the main classes provided in the object model. Listing 3-13 demonstrates how to use this object in order to create a document set instance.

TABLE 3-12: Document Set Object Model

| CLASS | DESCRIPTION |
|------------------------------|--|
| AllowedContentTypeCollection | Represents the list of content types that are allowed within a DocumentSet object |
| DefaultDocument | Represents a document that will be automatically created for the associated DocumentSet content type |
| DefaultDocumentCollection | Stores a collection of DefaultDocument objects |
| DocumentSet | Represents a Document Set |
| DocumentSetTemplate | Represents the template on which a DocumentSet object is based |
| DocumentSetVersion | Represents the major or minor version of the DocumentSet object |
| DocumentSetVersionCollection | Stores a collection of DocumentSetVersion objects |
| SharedFieldCollection | Specifies the fields for the associated SPContentType object of the current DocumentSet |
| WelcomePageFieldCollection | Specifies the fields to be displayed on the DocumentSet Welcome Page |

**LISTING 3-13:** Creating an Instance of a Document Set

Available for
download on
Wrox.com

```
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.DocumentManagement.DocumentSets;
using System.Collections;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                SPList list = web.Lists["LibraryTitle"];

                SPContentTypeId contentTypeId =
                    list.ContentTypes["Document Set"].Id;

                Hashtable properties = new Hashtable();
                properties.Add("Description", "My Description");

                DocumentSet documentSet = DocumentSet
```

```

        .Create(list.RootFolder, "DocumentSetName", contentTypeId, properties);
            }
        }
    }
}

```

The preceding code shows the creation of an instance document set. The listing starts by first using a reference to a `SPWeb` object to gain a reference to a specific list within the Web. The `List` object's `RootFolder` property was then passed as a parameter to the static `Create` method of the `DocumentSet` class along with the name and `SPContentTypeId`. The result is a new document set with the indicated name and content type in the folder of the list specified.

DOCUMENT CONTROL

In a document management system, the term *document control* refers to the system's ability to secure, govern, and record the usage of documents throughout their life cycle. Document control functions are critical to system effectiveness because they ultimately are responsible for how documents are used and by whom.

SharePoint provides a number of document control features that enable administrators to effectively manage document security, versioning, visibility, auditing, and organization.

Security

SharePoint provides a role-based system for managing security within SharePoint. Assigning users to roles effectively authorizes access to the various locations and items within the system.

In much the same way that security within the Microsoft Windows platform is managed, SharePoint provides the concept of groups and users. A user represents a single person within the system, while groups represent a collection of users as a single unit. SharePoint also provides the concept of permission level. A permission level represents a group of unique rights, or permissions, that can be granted to any group or user for a particular Web, list, or item within SharePoint (see Table 3-13). By default, SharePoint provisions the six permission levels listed in Table 3-14, although it is possible to create custom permission levels as well.

TABLE 3-13: Grantable Permissions

| PERMISSION | DESCRIPTION |
|--------------------|---|
| Managed Lists | Create and delete lists, add or remove columns in a list, and add or remove public views of a list |
| Override Check-Out | Discard or check in a document that is checked out to another user |
| Add Items | Add items to lists and add documents to document libraries |
| Edit Items | Edit items in lists, edit documents in document libraries, and customize Web Part pages in document libraries |

continues

TABLE 3-13 (continued)

| PERMISSION | DESCRIPTION |
|--------------------------------|---|
| Delete Items | Delete items from a list, and documents from a document library |
| View Items | View items in lists, and documents in document libraries |
| Approve Items | Approve a minor version of a list item or document |
| Open Items | View the source of documents with server-side file handlers |
| View Versions | View past versions of a list item or document |
| Delete Versions | Delete past versions of a list item or document |
| Create Alerts | Create alerts |
| View Application Pages | View forms, views, and application pages; enumerate lists |
| Manage Permissions | Create and change permission levels on the website and assign permissions to users and groups |
| View Web Analytics Data | View reports on website usage |
| Create Subsites | Create subsites such as team sites, Meeting Workspace sites, and Document Workspace sites |
| Manage Web Site | Grants the ability to perform all administrative tasks for the website, as well as manage content |
| Add and Customize Pages | Add, change, or delete HTML pages or Web Part pages, and edit the website using a Microsoft SharePoint Foundation compatible editor |
| Apply Themes and Borders | Apply a theme or borders to the entire website |
| Apply Style Sheets | Apply a style sheet (.CSS file) to the website |
| Create Groups | Create a group of users that can be used anywhere within the site collection |
| Browse Directories | Enumerate files and folders in a website using SharePoint Designer and WebDAV interfaces |
| Use Self-Service Site Creation | Create a website using Self-Service Site Creation |
| View Pages | View pages in a website |
| Enumerate Permissions | Enumerate permissions on the website, list, folder, document, or list item |
| Browse User Information | View information about users of the website |

| PERMISSION | DESCRIPTION |
|---------------------------------|--|
| Manage Alerts | Manage alerts for all users of the website |
| Use Remote Interfaces | Use SOAP, WebDAV, the Client Object Model, or SharePoint Designer interfaces to access the website |
| Use Client Integration Features | Use features that launch client applications; without this permission, users must work on documents locally and upload their changes |
| Open | Allows users to open a website, list, or folder in order to access items inside that container |
| Edit Personal User Information | Allows a user to change his or her own user information, such as adding a picture |
| Manage Personal Views | Create, change, and delete personal views of lists |
| Add / Remove Personal Web Parts | Add or remove personal Web Parts on a Web Part page |
| Update Personal Web Parts | Update Web Parts to display personalized information |

TABLE 3-14: Default Permission Levels

| PERMISSION LEVEL | DESCRIPTION |
|------------------|--|
| Full Control | Has full control of the site and can perform all site operations |
| Design | Can view, add, update, delete, approve, and customize content |
| Contribute | Can view, add, update, and delete list items and documents |
| Read | Can view pages and list items and download documents |
| Limited Access | Can view specific lists, document libraries, list items, folders, or documents when given permissions |
| View Only | Can view pages, list items, and documents; document types with server-side file handlers can be viewed in the browser but not downloaded |

When a new Web is created, SharePoint automatically creates the four default groups, each of which is assigned one of the out-of-the-box permission levels. These groups and their associated permission level are listed in Table 3-15. The permissions granted to these groups are inherited by all objects within the Web unless overridden at an individual child level. For each sub-Web that is created, it is possible to inherit these groups from the parent Web or have SharePoint automatically provision new congruent groups at the time of creation.

TABLE 3-15: Default Groups

| GROUP | PERMISSION LEVEL |
|---------------|------------------|
| Site Members | Contribute |
| Site Owners | Full Control |
| Site Visitors | Read |
| Viewers | View Only |

Managing Users and Groups

Managing users and groups can be accomplished at multiple levels. It is possible to add users or groups to a Web, list, or specific list Item within SharePoint. Each user or group added can even possess its own permission level. The following procedure outlines the process for adding users to an existing group in SharePoint:

1. From Site Settings, under Users and Permissions, click People and Groups.
2. Select a group from the Quick Launch menu.
3. Click the New button.
4. Enter users and groups in the control provided or select them from the global address book.
5. Click OK.

The Security Programming Model

SharePoint provides both a server-side and client-side object model for working with users and groups. Table 3-16 lists the main classes that make up the authorization object model for controlling security within SharePoint.

TABLE 3-16: Main Classes for the Authorization Object Model

| CLASS | DESCRIPTION |
|--------------------------------|---|
| <code>SPUser</code> | Represents a user in Microsoft SharePoint Foundation |
| <code>SPGroup</code> | Represents a group in Microsoft SharePoint Foundation |
| <code>SPGroupCollection</code> | Represents a collection of <code>SPGroup</code> objects |
| <code>SPUserCollection</code> | Represents a collection of <code>SPUser</code> objects |
| <code>SPRoleDefinition</code> | Defines a single role definition, including a name, description, management properties, and a set of rights |

| CLASS | DESCRIPTION |
|-----------------------------------|---|
| SPRoleAssignment | Defines the role assignments for a user or group on the current object |
| SPRoleDefinitionCollection | Represents a collection of <i>SPRoleDefinition</i> objects |
| SPRoleAssignmentCollection | Represents a collection of <i>SPRoleAssignment</i> objects |
| SPRoleDefinitionBindingCollection | Defines the role definitions that are bound to a role assignment object |

Listings 3-14 and 3-15 demonstrate simple examples of leveraging both the server-side and client-side object models, respectively, for interacting with SharePoint security.



LISTING 3-14: Creating a New Group and Adding Users Using the Server-Side Object Model

Available for
download on
Wrox.com

```
using System.Linq;
using System.Text;
using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                string groupName = "GroupName";

                //Create new group
                web.SiteGroups
                .Add(groupName, web.CurrentUser, web.CurrentUser, "GroupDescription");
                web.Update();

                //Retrieve group
                SPGroup group = web.SiteGroups[groupName];

                //Assign permission level to group
                SPRoleDefinition contributePermissionLevel =
                    web.RoleDefinitions["Contribute"];
                SPRoleAssignment assignment =
                    new SPRoleAssignment(group);
                assignment.RoleDefinitionBindings.Add(contributePermissionLevel);
                web.RoleAssignments.Add(assignment);
                web.Update();
            }
        }
    }
}
```

continues

LISTING 13-4 (continued)

```

                //Add user to group
                group.Users
            .Add("bgws2008r2x64\User1",
                "user1@somewhere.com",
                "User Name",
                "User Notes");
                group.Update();
            }
        }
    }
}

```

The preceding code created a new user group within SharePoint. After obtaining a reference to the indicated `SPWeb`, the `Add` method was called on the `SiteGroups` property of the `SPWeb`. The `SiteGroups` property is a reference to the `SPGroupCollection` that contains all of the valid groups for the current Web. Calling the `Add` method resulted in a new group being added to the collection. After adding the new group, the group was retrieved and the permissions given to the group by assigning the desired `SPRoleDefinition` to the role assignments for the Web. Once this was completed, the indicated user was added to the group through the `Users` property of the `Group` object.

**LISTING 3-15: Reading a Group and Its Users Using the Client-Side Object Model**

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Client;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ClientContext context = new ClientContext("http://10.1.0.169"))
            {
                Web web = context.Web;

                context.Load(web.SiteGroups,
                    x => x.Where(y =>
                        y.Title == "GroupName"));
                context.ExecuteQuery();

                Group group = web.SiteGroups[0];
            }
        }
    }
}

```

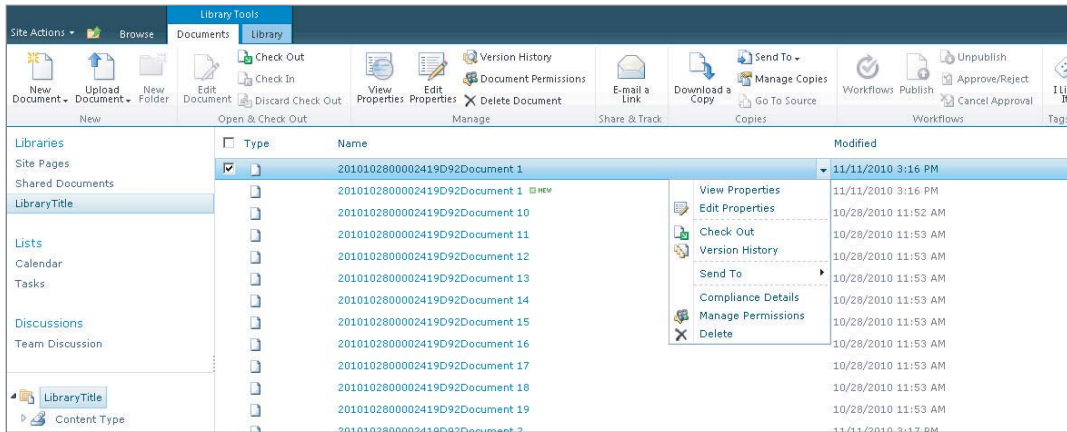



FIGURE 3-19



LISTING 3-16: Programmatically Controlling Document Check-Out

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                SPList library = web.Lists["LibraryTitle"];
                SPLListItem item = library.Items[0];

                //Check-out file
                item.File.CheckOut();

                //Make changes

                //Check-in to make changes available
                item.File.CheckIn("Check-In Comment");

                //Check-Out file
                item.File.CheckOut();

                //Make changes
            }
        }
    }
}
```

```

        //Discard changes
        item.File.UndoCheckOut();
    }
}
}
}

```

This listing begins by getting a reference to a specific document library in a Web. After obtaining this reference, the first document in the library is returned as a `SPListItem`. Because this `SPListItem` is an item in a document library, its `File` property represents a reference to the `SPFile` associated with the item. Using this `SPFile`, the file can then be checked out or checked in, or check-out can be undone.

Versioning

Documents within SharePoint also support the concept of *versioning*. Versioning tracks the changes made to a document by storing the previous version of the document each time it is edited. If a document is checked out, a new version will not be created until the document is checked back in. Along with each version of a document stored, SharePoint stores who edited the document, when it was edited, and a number used to identify the version. Administrators have the option to configure the version numbering scheme, the number of versions to retain, and whether content approval is required for users to be able to see a newly created version. Versioning is an important feature because it enables previous versions of a document to be restored as the current version should the need arise.

How to Configure Versioning

The following procedure outlines the steps necessary to configure versioning for a document library:

1. From Library Settings, under General Settings, click Versioning Settings. This will take you to the Versioning Settings page, as shown in Figure 3-20.
2. Select whether content approval is required (this enables draft item security settings).
3. Select a version number scheme.
4. If content approval is required, select draft item security.
5. Select whether to require a document to be checked out in order to edit it.
6. Click OK.

Version History

Once version history has been enabled, it is possible for anyone with sufficient permissions to view the version history of a document. From the Version History page, shown in Figure 3-21, it is also possible to restore the document to any previous version displayed in the history.

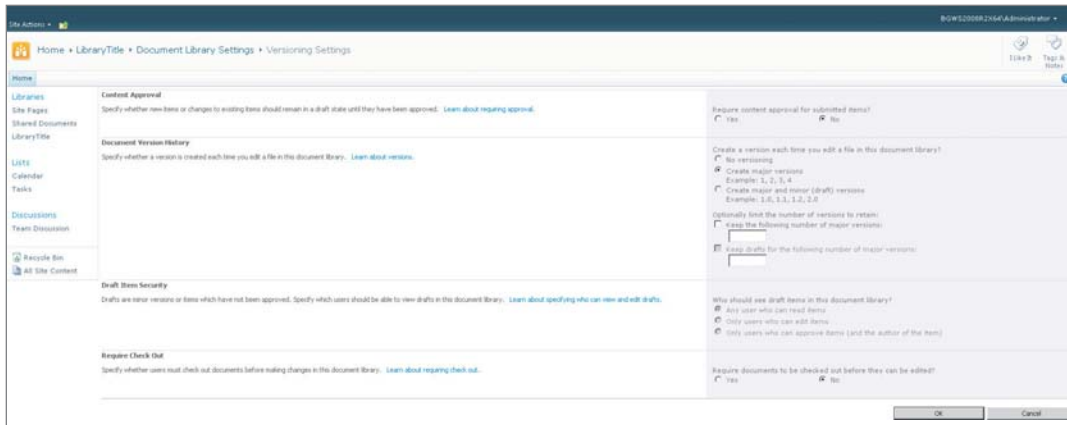


FIGURE 3-20

The screenshot shows the 'Version History' dialog box. At the top, there is a link for 'Delete All Versions'. Below it is a table with the following data:

| No. ↓ | Modified | Modified By | Size | Comments |
|-------|--------------------|-----------------------------|--------|----------|
| 3.0 | 11/12/2010 3:51 PM | BGWS2008R2X64\Administrator | 6.7 KB | |
| 2.0 | 11/12/2010 3:50 PM | BGWS2008R2X64\Administrator | 6.7 KB | |
| 1.0 | 11/11/2010 3:16 PM | BGWS2008R2X64\Administrator | 6.7 KB | |

At the bottom of the table, there are labels 'ColumnName' and 'Value'.

FIGURE 3-21

Programmatically Interacting with Version History

Using the client-side or server-side object model for document libraries, it is possible to interact with a document's version history. Table 3-3 earlier in this chapter lists the classes that make up the document library object model. Listing 3-17 demonstrates usage of the server-side object model to enumerate through the versions of a document as well as restore a previous version.



LISTING 3-17: Programmatically Interacting with Version History

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;

namespace Chapter3Code
{
    class Program
```

```

    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://10.1.0.169"))
            using (SPWeb web = site.OpenWeb())
            {
                SPList library = web.Lists["LibraryTitle"];
                SPLListItem item = library.Items[0];
                SPFile file = item.File;

                foreach(SPFileVersion version in file.Versions)
                {
                    Console.WriteLine(version.VersionLabel);
                }

                file.Versions.RestoreByLabel("1.0");

                Console.ReadLine();
            }
        }
    }
}

```

This listing demonstrated working with versions of a file by getting a reference to a `SPFile` object. Once a reference to the `SPFile` was obtained, the `Versions` property, which represents a `SPFileVersionCollection`, is iterated; and each version label is output to the console. After iterating over the collection, restoring a file version was demonstrated by calling `RestoreByLabel` and passing the label of the version, which is restored as the current version.

The Audit Trail

An important feature of any document management system is its ability to track usage information for the system, as well as its content. SharePoint Server provides an audit log for exactly this reason. Administrators can configure the information that is tracked by SharePoint through the Audit Log Settings page located within Site Settings. The settings page allows the following information to be configured for auditing purposes:

- Opening or downloading documents, viewing items in lists, or viewing item properties
- Editing items
- Checking out or checking in items
- Moving or copying items to another location in the site
- Deleting or restoring items
- Editing content types and columns
- Searching site content
- Editing users and permissions

After auditing is configured for a site collection, it is possible to view audit reports through the Audit Log Report page located within Site Settings. This page exposes a number of reports that can be scheduled to be generated and output to a designated Document Library as a Microsoft Excel file. Figure 3-22 shows the Audit Log Report page.

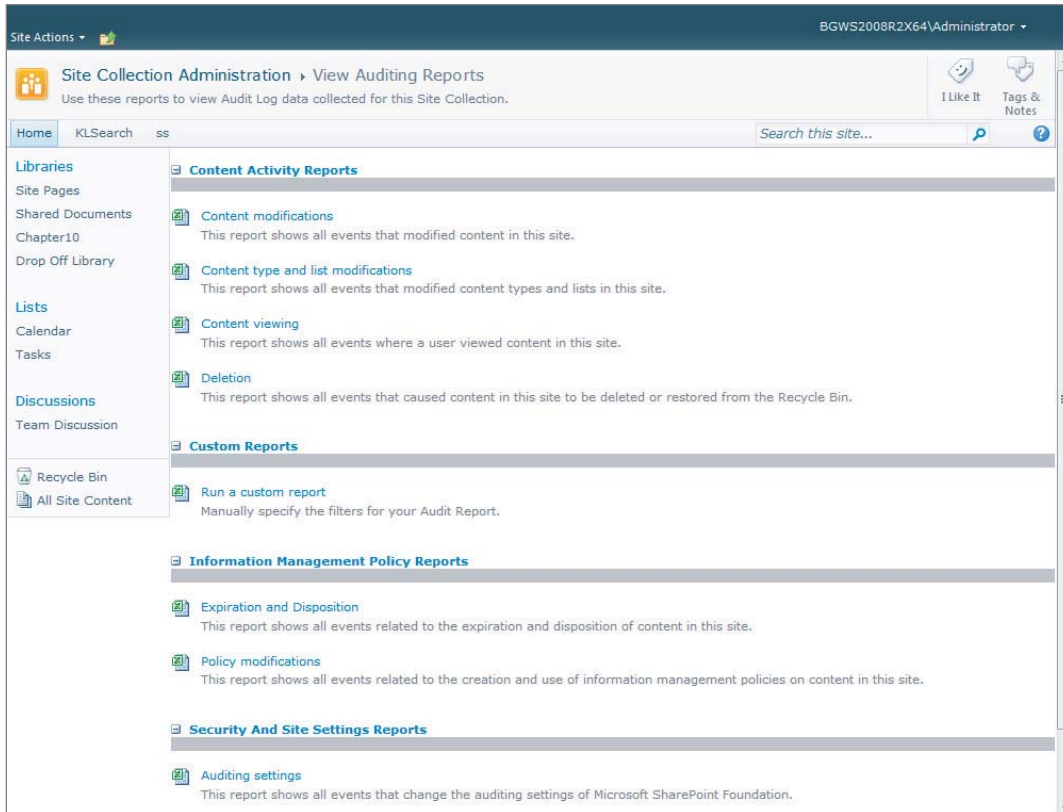


FIGURE 3-22

The Content Organizer

SharePoint Server provides a feature called the Content Organizer, which provides the capability to route documents to specific locations within a SharePoint site collection based on a set of predefined routing rules. When the Content Organizer is enabled within the Site Features page, a list for storing routing rules is created, as well as a special document library called the Drop Off Library.

When a user uploads documents to the Drop Off Library, the rules located in the routing rules list are evaluated in order to determine where to route the submitted documents. This enables documents to be sent to specific locations without the user needing to understand the end location of the document. It also ensures that documents meet a certain set of criteria prior to being sent to production locations.

Another feature enabled by the Content Organizer is its ability to auto-folder documents based on some threshold configured by an administrator. This can be useful for optimizing document library views. While the Content Organizer is useful from a document management perspective, it is also used extensively for records management, which is discussed in detail in Chapter 8.

SUMMARY

As a key pillar of enterprise content management, document management enables organizations to effectively and efficiently manage their document resources over the course of their lifetime. Ultimately, document management can be divided into two separate functional categories: document taxonomy and document control. Document taxonomy refers to the ability to organize and categorize documents within the system, whereas document control refers to the system's ability to sufficiently secure documents, as well as track and govern their usage within the system.

Microsoft SharePoint provides the facilities necessary to create an effective document taxonomy to hierarchically structure documents and their metadata in a way that simplifies organizing, controlling, and auditing their usage within the system. Using SharePoint, organizations can efficiently leverage their valued resources, which enables them to act in a more timely and knowledgeable manner, and automate document-centric business processes.

Overall, Microsoft SharePoint provides a full-featured document management system — one that offers an extensible, flexible platform with robust programmability support, which enables administrators, developers, and users to leverage the system to meet multiple business needs.

4

Workflow

WHAT'S IN THIS CHAPTER?

- Understanding workflow in the context of enterprise content management
- Understanding workflow capabilities in SharePoint 2010
- Choosing between different workflow options when implementing a solution
- Creating workflow solutions using the various development options in SharePoint 2010, including SharePoint Designer and Visual Studio

WORKFLOW AND ECM

Any ECM system worth its salt has some kind of workflow capability. Workflow in the context of ECM is commonly used in the following capacities:

- Content approval
- Content development
- Content disposition
- Custom business processes

The preceding list is certainly not exhaustive, but it is broad enough to describe many scenarios in which workflow is used in an ECM system. Content needs to be created, approved, and then have its life cycle managed in a way such that business rules can automatically be applied, without a human having to think about it or handle it.

Common requirements for workflow in ECM systems include parallel processing, integration with client applications such as Microsoft Office, process tracking and reporting, rule-based

routing, task management (such that reminders and task delegation can be easily handled), as well as obvious requirements such as creating workflows that can be easily managed and maintained.

WINDOWS WORKFLOW FOUNDATION

To fully understand workflow in SharePoint 2010 from an architectural perspective, it is important to understand the underpinnings of the SharePoint workflow engine. Whereas the rest of SharePoint is written with other Microsoft technologies at the core — namely, .NET and SQL Server — the workflow engine is based upon Windows Workflow Foundation (WF). Windows Workflow Foundation was introduced in .NET 3.0 in the fall of 2006. It was enhanced in .NET 3.5 and completely rewritten in .NET 4.0, which was released in the spring of 2010.

Although it sounds like a pretty big deal that WF was scrapped and rewritten from the ground up in .NET (it is), it is not relevant to a SharePoint 2010 developer, as SharePoint 2010, like its 2007 predecessor, utilizes the .NET 3.5 version of WF. While one could debate about why WF4 was a complete rewrite or why WF4 (or .NET 4, for that matter) was not implemented in SharePoint 2010, such a discussion would not be necessarily useful for this book. One potential upside of utilizing .NET 3.5 WF in SharePoint 2010 is that any skills a developer acquired or any tools that were written for SharePoint 2007 are still very relevant in 2010. One could assume that SharePoint vNext may very well utilize the .NET 4-style of WF, but that would be complete conjecture at this point in time. Note that any assertions made about how workflow currently functions in SharePoint refer to .NET 3.5-style workflow, not the newer .NET 4-style workflow.

WF Concepts

There are several concepts which, while they are core to WF and not necessarily specific to SharePoint, carry over into SharePoint workflow and are completely necessary to understand conceptually and technically. These topics are discussed in the next few sections. While this list is not exhaustive (comprehensive books are available on the topic of workflow), this material covers the main topics that are important for understanding workflow.

Activities

Put simply, activities (the moniker used in WF) are the building blocks that are used to compose processes known as workflows. By themselves they do nothing useful; it is when individual activities are configured and used in concert with other activities that real work is done.

Activities provide both very basic, but essential, functionality such as control flow as well as more specific functionality, such as executing SQL statements. Everything in a WF workflow, including the workflow itself, is an activity, and was developed with a very specific function in mind.

Again, there are much more comprehensive books on both workflow in general and SharePoint workflow in particular, so bear in mind that the following list represents, at a high level, the types of activities that are found in WF 3.5:

- `IfElseActivity` — This activity emulates a programming language's if, if else, and else statements. Figure 4-1 shows a very simple sequential workflow featuring an `IfElseActivity` with three branches. The first branch is followed if an invoice amount is greater than \$1,000, the second branch is followed if an invoice amount is greater than \$500 but less than \$1,000, and

the last branch is followed if neither of the first two conditions are met. If this logic were implemented in C#, it would look something like the following code snippet:

```
if(invoiceAmt > 1000)
{
    // code here...
}
else if (invoiceAmt > 500)
{
    // code here...
}
else
{
    // code here...
}
```

- **DelayActivity** — When this activity is executed within a workflow, it causes the execution to cease for a predetermined amount of time before resuming with the subsequent activity's execution. Logically, it is similar to a `Thread.Sleep` call in C#, but the technical implementation is much different, based on how persistence is implemented in WF. Persistence is a very important workflow topic and is discussed later in this chapter.
- **ParallelActivity** — This is another activity that utilizes the concept of branches. However, unlike `IfElseActivity`, which only executes one of its child branches based on some condition, `ParallelActivity` will execute each of its child branches in a parallel manner.

This can be useful when modeling a scenario in which two or more people or groups of people are working on something at the same time. Rather than force the actions to happen one after another, potentially lengthening the timeline of the flow unnecessarily, `ParallelActivity` can be used to shorten the time in which tasks are completed.

Note that the next activity after an instance of `ParallelActivity` will not be executed until each branch of the `ParallelActivity` has been completed.

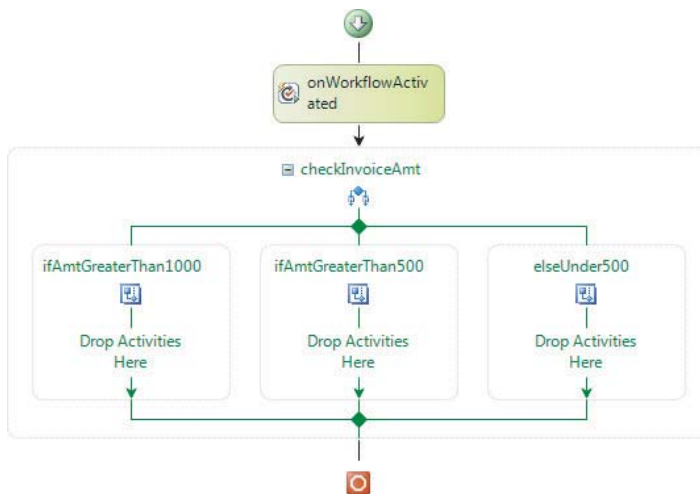


FIGURE 4-1

Workflow Modes

Workflow is generally thought of as moving from point A to point B while making decisions and completing some work along the way. While this general description is not inaccurate, there are different methods of getting from point A to point B. The most common type of workflow, and the one that most people automatically think of when the term is used, is the *sequential workflow*. In a sequential workflow, work moves in order from top to bottom (or left to right), and is finished once it has reached the last activity in the process definition. This type of workflow is actually not that different than old-style procedural code.

Another type of workflow, which like the sequential workflow type is supported by Windows Workflow Foundation and SharePoint, is called a *state machine*. If you have studied workflow concepts in any capacity before, you are likely already familiar with the concept of state machine workflows. These types of processes also take you from point A to point B, but they may take detours in the middle. A state machine is so-called because it is a collection of different “states,” which do not necessarily happen in a fixed order every time.

A state machine workflow has a beginning state and an end state and it moves from state to state based on rules specified in the workflow definition. In other words, a state machine workflow does not arbitrarily move from one state to another. Rather, the workflow definition determines which states can transition to other states, based on logic defined by the workflow author.

Consider the example of an invoice processing workflow. Some valid states might be as follows:

- PendingApproval
- PendingPayment
- ReviewDispute
- InvoiceOverdue
- InvoicePaid

Depending on real-world conditions, certain states may never need to be entered. For example, if a customer pays his or her invoice in full and on time, the `InvoiceOverdue` state will never be transitioned to. In addition, it is only logical that the `InvoiceOverdue` state can only be transitioned to from the `PendingPayment` state. Clearly, it wouldn’t make sense to move from `InvoiceOverdue` from `InvoicePaid`.

Although both of these workflow types are supported by SharePoint, not all of the workflow tools in SharePoint 2010 offer both types of workflow. For example, you can create a state machine workflow only by using Visual Studio. SharePoint Designer, which is discussed later in this chapter, only allows a workflow author to create sequential workflows.

Persistence

A distinct aspect of workflows, particularly those which interact with humans, is that they are long-running. Consider that long-running means anything that takes more than a couple seconds — minutes or even hours, days, and beyond. Generally, computer programs that do not require this type of extended interaction run in local memory, and relevant process data only survives a single execution

session of that program. A workflow that is coordinating the work of many people needs to be able to last for as long as necessary, as dictated by the business process.

For example, consider a vacation request workflow. A user may submit a request to take vacation sometime in the distant future. This request first requires the approval of the employee's immediate manager. If the manager is currently on vacation, then it may be at least another week or so until the request can be given the attention it needs. In addition, consider that the workflow needs to run until the vacation is actually taken by the employee. If the requested time away from work isn't until eight months in the future, then allowing the process to remain in memory on the server would be a recipe for disaster — not only from a scalability perspective (imagine if there were thousands of these requests active at any given time), but also from a durability perspective. If the server were to be rebooted or lost power, then all of the active requests would be lost forever.

To address these issues of scalability and durability, the workflow platform in SharePoint utilizes a mechanism called *persistence*. The state of each workflow instance — that is, the data that makes a specific workflow instance run — is taken from memory and placed into a persistent data store. Specifically, this information about a workflow instance is stored in a SharePoint content database in SQL Server.

The great thing about persistence in SharePoint workflow is that all of this plumbing happens for you automatically. As a developer or architect, you do not have to concern yourself with how or when to write this data to SQL Server. In addition, the workflow engine takes care of how and when to retrieve this data in order to resume sleeping instances.

So when is workflow state written to the database? For the most part, this persistence action occurs when the workflow has no immediate work to do. For example, imagine if a workflow assigned a task to a specific user and the next action to take is to process the response from that user. Given that a user might take a significant amount of time to respond to a task, this is a perfect time to persist the workflow state to the database. Once the user responds to the task, the workflow engine knows, through a process called *correlation*, which workflow instance the task is associated with, and that the state needs to be called up from the database so that the process can continue execution.

THE ROLE OF WORKFLOW IN SHAREPOINT

One of the greatest things about SharePoint from a platform perspective is its flexibility and the options available to solution architects and developers. This can also be one of SharePoint's greatest weaknesses in the hands of someone who does not understand all these options and how to determine when one potential solution should be chosen over another. This is no less true when looking at implementing a workflow solution in SharePoint.

The SharePoint 2010 platform provides numerous approaches to the development of workflow solutions. Depending on the capabilities that are needed, the available skillsets for a given project, or the policies of a given organization, one workflow solution might be more appropriate than another. Table 4-1 provides a brief introduction to the available workflow options in SharePoint 2010. The options are roughly ordered from least complex and capable to most. It can also be assumed each increase in complexity and capability also requires an increased skillset for development and ongoing support. Regardless of which technology is used, whether we're talking about

SharePoint or another non-Microsoft technology, understanding the business process is crucial to the success of any workflow project. Unfortunately, this simple and seemingly obvious point is often overlooked.

TABLE 4-1: Workflow Options in SharePoint 2010

| WORKFLOW APPROACH | SKILL LEVEL | DETAILS |
|---------------------|--|---|
| Out-of-the-Box | Low; only configuration is necessary. | Several out-of-the-box workflows are included, which represent canonical yet fairly straightforward business processes. Start here if possible. |
| Visio | Moderate; need to be comfortable designing business processes in Visio and familiar with SharePoint workflow capabilities. | Allows a visual way to orchestrate business processes and allows further refinement in SharePoint Designer or Visual Studio. |
| SharePoint Designer | Moderate; slightly more powerful than Visio and unfamiliar UI if the user is coming from a Visio background. | Uses a text-oriented approach to orchestrating a business process. Includes a good number of workflow actions out-of-the-box as well. |
| Visual Studio | Advanced; this is for .NET developers. | Visual Studio allows the ultimate flexibility, as custom code is possible, which is not the case for the other solutions described. |

The information in this table is only introductory in nature, and each of the workflow options, including associated pros and cons, pitfalls, and other information, is discussed in depth in this chapter. By the time you have finished reading this chapter, you should have a good understanding of what SharePoint 2010 offers from a workflow perspective; and you should be relatively comfortable determining when one approach might be better than another in a given circumstance.

Workflow Scopes

In SharePoint 2010, workflows can be associated with and executed at two different scopes: the item level or the site level. Both of these scopes are discussed in the following sections.

Item

Workflow in SharePoint was originally invented to process a specific item or document in a list or library. For example, consider a workflow that exists in order to solicit feedback on every new marketing document added to a specific document library. The process itself is the same for each document, but each document should be processed by a unique instance of the workflow. This is the perfect scenario in which an item-scoped workflow should be used.

Site

New in SharePoint 2010 is the site workflow. The site workflow fills a gap that existed in previous versions of SharePoint whereby it made sense to implement a process using a workflow, but the process didn't necessarily have an item or document with which it should be associated. A common approach to get around this limitation in the past was to create a "dummy" item that existed only to have a workflow running against it. The dummy item was itself inconsequential and filled no need other than serving as a "hangar" for the workflow.

Workflow Phases

The next three short sections define some important terms with regard to the various phases of a workflow's life cycle in SharePoint. Each of these phases is associated with the period of time after which a workflow has been developed and deployed to a SharePoint farm.

Association

The association phase describes the application of a workflow definition to a list, document library, content type, or site. At the time a workflow is associated to one of these entities, some basic information is required from the end user, such as a name for the workflow association, the task and history list to be used, and start options. These data elements are discussed in more detail in the section "Out-of-the-Box Workflows."

In addition to the standard data, custom information can be collected from the person who is performing the workflow association. This custom data can then be used by any running workflow instances to perform work.

Initiation

Workflow initiation describes when an already configured workflow association is starting and in the process of creating a new workflow instance. Workflow initiation can happen in a number of ways depending on how the workflow start options were configured during association. Workflows can be initiated in the following ways:

- Manual initiation if the user has the appropriate rights
- Automatic initiation if the workflow item was just created or changed
- Programmatic initiation

Just as with association, custom data can be collected from a user and then used inside the running workflow instance. However, initiation data is specific to one workflow instance, whereas association data is available to all instances of a particular association.

Execution

The last broad phase is fairly self-explanatory. During the execution of a workflow, the workflow is either actively processing input from users in the form of tasks or other events or sleeping and waiting for the next burst of execution based on external input.

A running workflow instance can also be manually terminated by a user who has appropriate rights. Users can also view information about running workflows, such as when an instance was started,

the last time it was interacted with, or the list of tasks and history items associated with a specific workflow instance.

AUTHORING AND WORKFLOW TYPES

This section discusses the spectrum of workflows and workflow development options in SharePoint 2010. Because there are varying levels of flexibility required for certain solutions and perhaps only certain skills that can be applied to a project, SharePoint 2010 provides an array of options with regards to implementing workflow.

First, SharePoint ships with several out-of-the-box workflow definitions that can simply be configured and used. Second, by way of SharePoint Designer, declarative workflows provide an “entry-level” workflow tool when the flexibility of code is not required. Finally, when the big guns need to be brought out, Visual Studio offers the most flexible, yet most complex, tooling for workflow development in SharePoint 2010.

Out-of-the-Box Workflows

SharePoint 2010 includes several workflow definitions out the box that can be used as-is with minimal configuration to meet specific business needs. These out-of-the-box workflows are described in Table 4-2.

TABLE 4-2: Out-of-the-Box Workflows in SharePoint 2010

| WORKFLOW | DESCRIPTION |
|----------------------|--|
| Approval | The Approval workflow routes a document to an approver or set of approvers for review and approval. Approvers can reject the document or reassign the approval task to someone else. |
| Collect Feedback | The Collect Feedback workflow solicits and compiles feedback regarding a document. The resulting feedback is provided to the document owner once the workflow is completed. |
| Collect Signatures | The Collect Signatures workflow is used to acquire one or more digital signatures in an Office document. This workflow must be started from within the Office document that requires the digital signatures. |
| Disposition Approval | The Disposition Approval workflow manages expiration and retention of a document by asking workflow task owners whether a document should be kept or deleted. |
| Publishing Approval | The Publishing Approval workflow, available in SharePoint publishing sites, routes a page for approval. Like the standard Approval workflow, this workflow allows the task owner to approve or reject a page or reassign the task. |
| Three-state | The Three-state workflow is used to manage list items against a process that has three states, or phases. A common scenario is issue tracking, which might have phases such as New, Active, and Closed. |

The preceding table should provide a thorough overview of what the Microsoft-provided templates attempt to accomplish. These six workflow templates provide a great deal of power and flexibility, especially considering no code and only configuration is required to use them. To get an idea of how to utilize and interact with out-of-the-box workflows, or really any SharePoint workflows, the following section reviews the Approval workflow. The Approval workflow is focused upon here because it is one of the most flexible of the out-of-the-box workflows and arguably the most commonly used.

The Approval Workflow

The Approval workflow is an out-of-the-box workflow that routes a document to one or more approvers. The approver reviews the document and then approves, rejects, or reassigns the document to someone else. The following example illustrates how to configure this workflow by associating it to a library, and provides steps for interacting with a running workflow instance.

Navigate to any SharePoint document library you choose. The Ribbon interface will expose the necessary command needed to associate a workflow definition to this library. To find this command, click the Library Ribbon tab in the Library Tools tab grouping. You can find the Workflow Settings command on the far right in the Settings group (see Figure 4-2).

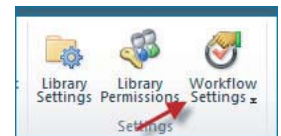


FIGURE 4-2

Click this button to display the Workflow Settings page, which shows any workflows currently associated with this library, and enables the association of new workflows. Next you are going to associate the Approval workflow to this library. To be more specific, you will associate the workflow to this library and the Document content type only. To specify the content type in this library to which the association will occur, select Document from the dropdown list labeled “These workflows are configured to run on items of this type.” Next, click the “Add workflow” link.

The subsequent page can be referred to as the workflow association page. It is here that vital information about the link between a workflow definition and a list, library, or content type is specified. This page is essentially the same for each and every workflow that could ever be associated. Table 4-3 describes the information that needs configuration.

TABLE 4-3: Workflow Association Information

| DATA ELEMENT | DESCRIPTION |
|--------------|--|
| Workflow | This allows the selection of a workflow definition. Workflows are available in a site collection whose associated SharePoint feature has been activated. |
| Name | A unique name for the workflow association. A column of this name will be added to the list/library and will display the workflow’s status for each item. |
| Task List | The task list that will be used to store workflow tasks for this association. An existing list can be chosen or a new one can be created from this user interface. |

continues

TABLE 4-3 (continued)

| DATA ELEMENT | DESCRIPTION |
|---------------|---|
| History List | The history list that will be used to store history items for this association. An existing list can be chosen or a new one can be created from this user interface. |
| Start Options | The start options specify how new workflow instances are started. Options are manual start, start on new item creation, start on item changed, and, for publishing sites, start when a major version of an item is published. |

Enter the following values on this page:

- Workflow: **Approval - SharePoint 2010**
- Name: **My Approval Workflow**
- Task List: new
- History List: new
- Start Options: manual

Click the Next button (because the Approval workflow has further options that need configuration) to display the Approval workflow settings, as shown in Figure 4-3.

The screenshot shows the 'Approval' workflow configuration page in SharePoint. It features several sections:

- Approvers:** Includes an 'Assign To' field with a user selection icon and an 'Order' dropdown menu set to 'One at a time (serial)'. Below this is a checkbox for 'Add a new stage' and a text box for entering names separated by semicolons.
- Expand Groups:** A checked checkbox with the text 'For each group entered, assign a task to every member of that group.'
- Request:** A large text area for a message, with a note below it: 'This message will be sent to the people assigned tasks.'
- Due Date for All Tasks:** A date picker field with the label 'The date by which all tasks are due.'
- Duration Per Task:** A text input field with the label 'The amount of time until a task is due. Choose the units by using the Duration Units.'
- Duration Units:** A dropdown menu currently set to 'Day(s)', with the label 'Define the units of time used by the Duration Per Task.'
- CC:** A text input field with a user selection icon and the label 'Notify these people when the workflow starts and ends without assigning tasks to them.'
- End on First Rejection:** An unchecked checkbox with the label 'Automatically reject the document if it is rejected by any participant.'
- End on Document Change:** An unchecked checkbox with the label 'Automatically reject the document if it is changed before the workflow is completed.'
- Enable Content Approval:** An unchecked checkbox with the label 'Update the approval status after the workflow is completed (use this workflow to control content approval).'

At the bottom of the page are two buttons: 'Save' and 'Cancel'.

FIGURE 4-3

By reviewing this dialog, you can see that this workflow is fairly flexible, especially when you consider that it only requires configuration in the UI, and no coding. Note the following highlights:

- The capability to assign tasks to people and/or groups serially or in parallel.
- The capability to create task stages. This functionality is new in the SharePoint 2010 version of this workflow.
- The capability to control task duration and due dates.
- The capability to CC people on each task so that others can be kept in the loop regarding what is happening in a workflow instance.
- The capability to control workflow termination. It is possible to terminate the workflow when the first person rejects the item and/or when the document changes.
- The capability to update the approval status of the item if it is under content control (generally in publishing sites).

For the purposes of this exercise, fill out this form with the values listed in Table 4-4.

TABLE 4-4: Values for Approval Workflow

| FORM FIELD | ENTER THIS INFORMATION |
|-------------------------|--|
| Approvers | To get an idea of how the Approval workflow functions, add a new stage in addition to the default one. In the first stage, enter a couple of accounts and select serial ordering; for the second stage, enter a couple of accounts and select parallel ordering. |
| Expand Groups | This option is only applicable if a group is entered in the preceding Approvers section. If you entered a group, go ahead and select this box to ensure that each user is assigned an individual task. |
| Request | Enter whatever text you would like here — something like Please review this document and indicate your approval status. |
| Due Date for All Tasks | Leave this textbox blank because you will provide a number of days to complete each task, rather than enter a specific date. |
| Duration Per Task | Enter 1 . |
| Duration Units | Select “Day(s)” from the dropdown box. |
| CC | Leave this blank for the purposes of this exercise. |
| End on First Rejection | Check this checkbox. |
| End on Document Change | Check this checkbox. |
| Enable Content Approval | Leave this checkbox blank. If you want to test this option in a library where content approval has been configured, please feel free to do so. Content approval can be turned on from the Version Settings screen for a document library. |

- Basic workflow information such as the date and time the workflow was started and last run, the initiator, the associated document, and status
- A Visio visualization of the workflow
- Actions such as the capability to terminate the workflow, cancel all tasks, add or update approvers, and so on
- A list of the associated tasks
- The workflow history associated with this workflow instance

At this point in the exercise, proceed by completing tasks that will automatically be assigned by the workflow to the users you configured. Feel free to try different paths by doing some variation of the following:

- Approve and reject the document at various points of the workflow.
- Change the content of the document at various points of the workflow.
- Reassign a task.
- Request a change in the document.
- Use the workflow status screen to cancel all approval tasks.
- Use the workflow status screen to add an approver.

As demonstrated here, the Approval workflow is flexible and full-featured, and it will meet the needs of many basic content approval workflows. However, Microsoft didn't gain its prominent position in the marketplace by offering cookie-cutter solutions, and they clearly recognize that their out-of-the-box workflows can't meet everyone's needs all of the time. Given that some or many customers have more complex requirements than those that can be addressed by the Approval workflow, many other approaches to workflow are available in SharePoint, as discussed in the following sections. This includes the capability, new in SharePoint 2010, to take an out-of-the-box workflow and create a new workflow from it as a baseline in SharePoint Designer.

Declarative Workflows

After out-of-the-box workflows, the next most complex and capable workflows type is the declarative workflow. This type of workflow is so-called because it does not use code to define a business process. Rather, user-friendly designers are used to define the process steps; and the resulting definition is eventually stored as XML, rather than a compiled assembly. The two types of declarative workflows, Visio and SharePoint Designer, are discussed in the following sections.

Visio

Visio has long been the tool of choice of business analysts for documenting the steps of a process. However, Visio's artifacts are generally just that: documentation. Wouldn't it be great if all that hard work done in the visual designer wasn't just informational but actually executed and guided important business processes from one step to the next? Well, Visio and SharePoint 2010 enable this to become a reality. . . to a point.

Before getting into the specifics of Visio, it is important to understand the general context of declarative workflows and how Visio workflows relate to SharePoint Designer workflows. First and foremost, Visio workflows cannot go straight to SharePoint for execution. Rather, Visio is used as a “getting started” tool by business analysts or anyone else who wants to create a human-readable version of a workflow. This workflow definition must then take another step before moving to SharePoint, and this step is into SharePoint Designer. For some initial context around this discussion, Figure 4-5 shows the various ways in which workflows can be authored and moved into SharePoint.

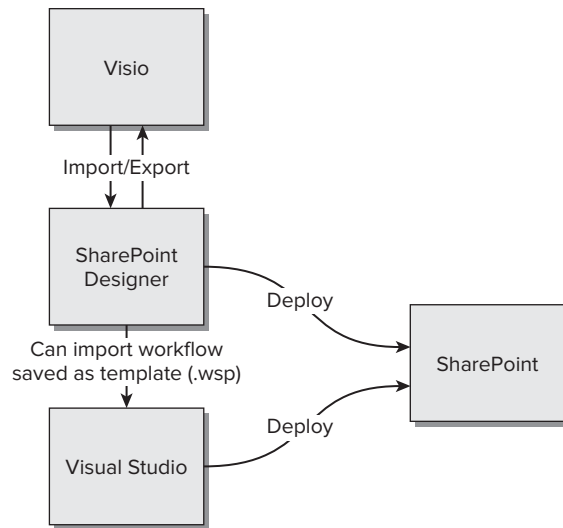


FIGURE 4-5

This figure shows a few different things. First, a custom workflow can begin its life in any of the three tools: Visio, SharePoint Designer, or Visual Studio. Second, a workflow can be transferred between various tools for further editing or refinement. For instance, a workflow that began its development life cycle in Visio can be moved along to SharePoint Designer for further refinement, and even further along into Visual Studio, where it is completed. Note that workflows edited in Visual Studio, whether that is where they started or not, cannot go back to SharePoint Designer or Visio. In contrast, workflows that have been edited in SharePoint Designer can be exported and edited in Visio. Finally, only workflows that have come from SharePoint Designer or Visual Studio can be deployed to SharePoint for execution. As mentioned earlier, Visio workflows require the extra step of being exported to at least SharePoint Designer before being used in SharePoint. With these relationships in mind, you are now ready to take a deeper dive into Visio workflows.

Visio Development Experience

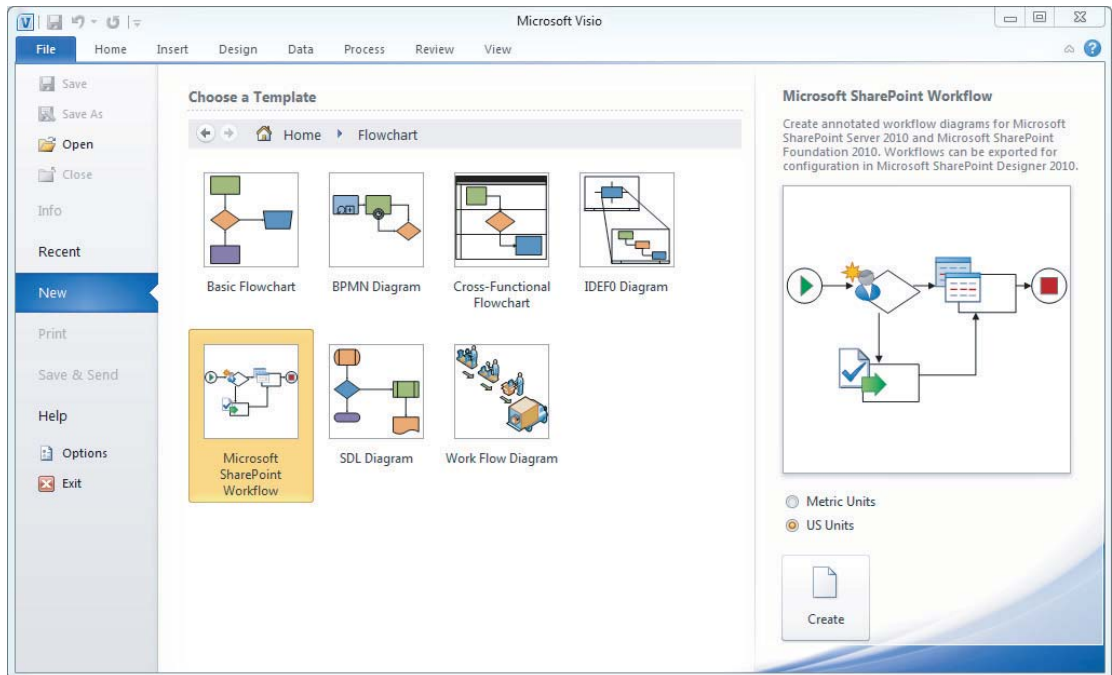


To develop SharePoint workflows in Visio, you need the Premium Edition of Visio 2010.

To create a new workflow in Visio 2010, navigate to the Office Backstage view and select New. From here, select Flowchart in the Template Categories section. In the Choose a Template dialog that appears, shown in Figure 4-6, double-click Microsoft SharePoint Workflow to create a new workflow.

The new workflow will contain several things in addition to the normal Visio options specific to SharePoint workflow development. First, three SharePoint workflow shape panels are available:

- SharePoint Workflow Actions
- SharePoint Workflow Conditions
- SharePoint Workflow Terminators


FIGURE 4-6

In addition to the shape panels, there is a Ribbon tab called Process. This tab contains several commands that are specific to SharePoint workflow development. This includes tools for the following:

- Managing subprocesses (these simply enable the workflow author to divide complex workflows into multiple pages in Visio)
- Validating the current workflow
- Importing/exporting the workflow to and from SharePoint Designer

To get started, simply drag and drop actions from the shapes window onto the design canvas. There are three categories of actions in Visio workflows, as described in the following sections.

Terminators

Each workflow that is built in Visio must start and end with a terminator. Therefore, it shouldn't be too surprising to learn that there are only two terminator actions: Start and Terminate (see Figure 4-7). If you do not start and end your workflow with the appropriate terminator actions, the workflow will not validate and you will not be able to export it.


FIGURE 4-7

Conditions

A workflow wouldn't be of much use without being able to make decisions. Visio enables you to use several different actions in order to control the flow of your process. The condition actions are shown in Figure 4-8.

Most of the condition actions are obvious, such as "Title contains keyword," "File type," or "Modified in date span." Less obvious is the "Compare data source" action. This condition is useful when none of the other actions are a good fit. Although Visio authors can't configure what the decision is supposed to evaluate, they can provide a useful label for the action. This label is not only useful for anyone else viewing the workflow definition, it is also displayed in SharePoint Designer after it has been imported into that tool.

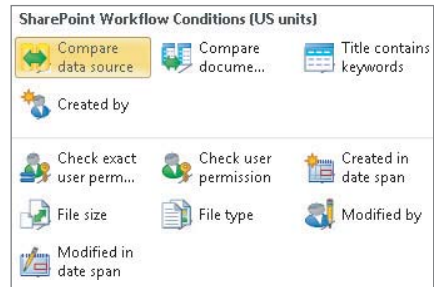


FIGURE 4-8

All the Other Actions

This categorization represents everything else that is not a terminator or a condition. Some actions relate to SharePoint-specific activities such as assigning tasks and logging to the history list, whereas other tasks are more general-purpose, such as performing a calculation or stopping the workflow. The workflow actions available in Visio are shown in Figure 4-9.

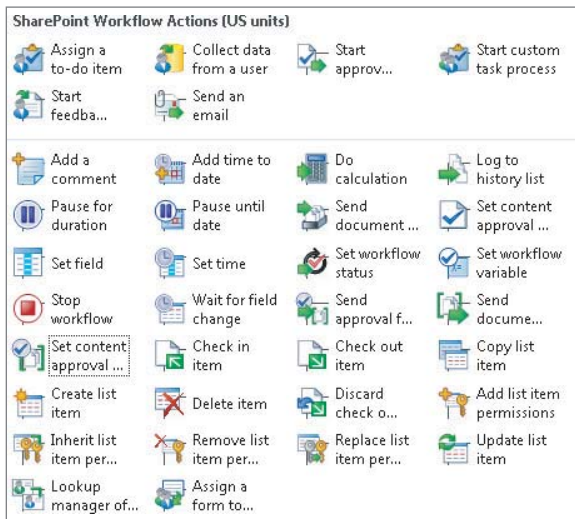


FIGURE 4-9

Again, the goal of defining the process in Visio is to simply lay out the steps in order, and no more. There is no in-depth configuration of actions. For example, when assigning a task to a user, the workflow as defined in Visio does not specify to whom the task is assigned. Similarly, the "Set workflow status" action is not actually configured in Visio to specify what the workflow status should

be when that particular action is executed. The workflow developer in SharePoint Designer will be responsible for filling out details such as these. As mentioned, the labels on each action provide some indication about what a particular action does.

Like the conditions, what these actions represent is generally evident from their name; actions like “Pause until date,” “Send an email,” and “Update list item” are clear. However, some actions should be defined explicitly for users who will be developing Visio workflows. For example, how would a designer know when to use “Collect data from user” as opposed to “Start feedback process,” or “Assign a to-do item” versus “Start custom task process”? Granted, even if the user chooses the “wrong” action for a particular scenario, the more advanced workflow designer can fix these idiosyncrasies in SharePoint Designer.

Validating a Process

Before a workflow can be exported, the process must be validated. This ensures that the workflow is actually in a workable state. A few common issues include the following:

- A workflow without a Start or Terminate action
- A workflow that has actions which are not connected to the flow with connectors
- A condition action that does not have two connectors, one labeled Yes and one labeled No

The Process Ribbon tab has a checkbox that enables an Issues window that is displayed at the bottom by default. This Issues window indicates any validation problems that were found after the Check Diagram button, which is also found in the Process tab, is clicked. If there are no active issues, the process is ready to be exported.

Exporting a Process

To see how this whole process works, in this section you create a simple process in Visio. The goal of this workflow is to acquire a piece of information from a user, their birthday, and then update the list item on which the workflow is running with this date. Finally, the workflow should send an e-mail to the person who started the workflow, the originator. This workflow can be built with the following actions, which should be connected in this order:

- 1.** Start.
- 2.** Compare document field. This is used to check if a user was specified. If no user was specified the workflow simply ends.
- 3.** Collect data from a user.
- 4.** Update list item.
- 5.** Send an e-mail.
- 6.** Terminate.

The process at this point should look something like Figure 4-10.

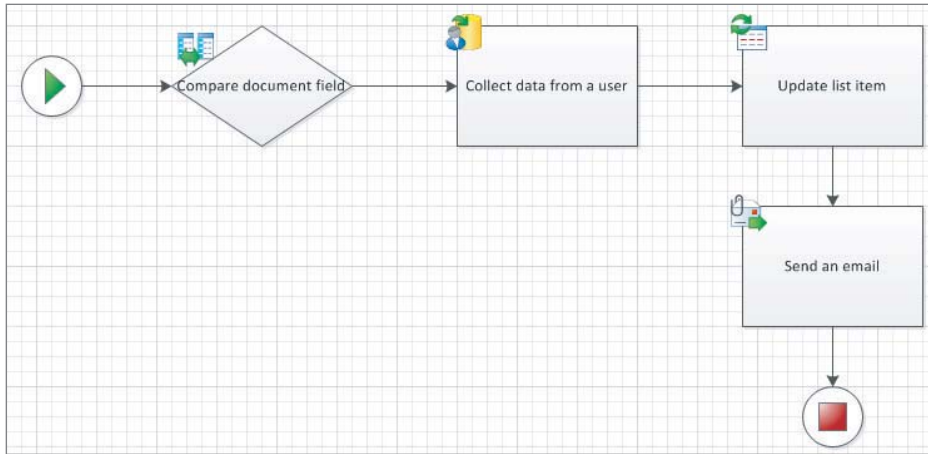


FIGURE 4-10

Aside from connecting these actions sequentially as you would in any other Visio drawing, the only configuration needed is to create two branches out of the “Compare document field” action. Because this action is intended to satisfy a Yes/No decision, it needs two connectors: one labeled Yes and one labeled No. The one labeled Yes should obviously be connected to the “Collect data from a user” action, and the one labeled No should be connected to the final Terminate action. To label these connectors, you can double-click each one and type the words **Yes** and **No** or you can right-click on each one and select **Yes** or **No** from the context menu. Also, each action should be edited so that the inner text reflects what action is occurring in your process. Figure 4-11 represents what the process should look like after making these changes.

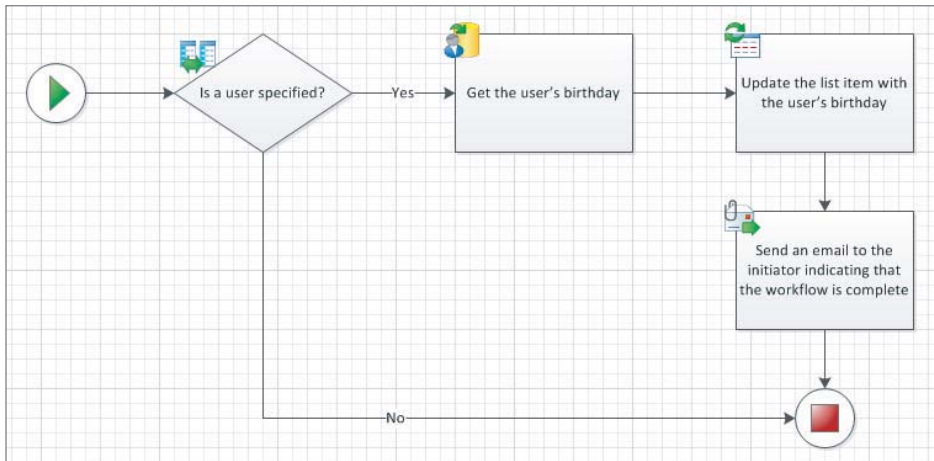


FIGURE 4-11

After performing these steps, the process is now ready to be exported. When you export the file using the **Export** command in the **Process** Ribbon tab, it has an extension of **.vwi**, which stands for

Visio Workflow Interchange. This file, like a lot of other Office file types, including DOCX and PPTX files, is just a zip file. In fact, you can rename the .vwi file to have a .zip extension. The zip file contains several files needed for SharePoint Designer and SharePoint itself to consume this workflow. If the workflow you just developed in this example were exported to a file called `GetUsersBirthday.vwi` and renamed to a .zip file, the contents of the file would match what is shown in Figure 4-12.

| Name | Type |
|---------------------|-------------------------|
| [Content_Types].xml | XML Document |
| workflow.vdx | Microsoft Visio Drawing |
| workflow.xoml | Windows Workflow File |
| workflow.xoml.rules | RULES File |

FIGURE 4-12

The files included in the zip file are similar to the files that a SharePoint Designer workflow produces behind the scenes. Similarly, this workflow type is also a declarative workflow. The heart of the logic lives in the .xoml file. This file contains a reference to each workflow action in order, as well as some configuration information about each action and the workflow as a whole. If you examine an .xoml file that is exported from Visio, most of the actions' properties will be null because, as mentioned earlier, Visio does not allow the workflow author to configure individual actions aside from giving them useful labels.

The .rules file is an XML representation of .NET CodeDOM objects. Basically, the CodeDOM allows .NET code to be represented as objects. With these objects you can represent a statement like the following as a graph of objects. These objects can then be serialized as XML and passed around in a file.

```
if (i > 5) DoSomething();
```

Importing a Process

Referring back to Figure 4-10, you will notice that workflows can also be exported from SharePoint Designer and imported back into Visio. This can be a very useful way for the original Visio designer and the SharePoint Designer author to collaboratively iterate on a workflow definition. Remember that SharePoint Designer, which is discussed shortly, is capable of actually finishing off and fully configuring a workflow, whereas Visio is not. Thankfully, any extra work that is done in SharePoint Designer is persisted in the .vwi file when exported and survives any work subsequently done in Visio. Furthermore, a workflow that is initially developed in SharePoint Designer can still be exported, viewed, and edited in Visio. If nothing else, this feature can be useful for documentation purposes, as well as process review sessions with business users.

SharePoint Designer Workflows

As mentioned earlier, SharePoint Designer enables users to create declarative (i.e., no-code, defined with XML) workflows using a text-based designer. SharePoint Designer is targeted toward the power user who is comfortable creating but is not necessarily a full-fledged .NET developer. SharePoint Designer does many other things aside from creating and managing workflows (e.g., look-and-feel, site administration, etc.), but only the workflow aspects of the tool are covered here.

Improvements in 2010

One of the biggest workflow improvements in SharePoint 2010 over SharePoint 2007 is SharePoint Designer workflow. Therefore, before diving deeply into SharePoint Designer (SPD) workflow, the following sections cover some of the most notable improvements and differences.

Authoring Experience

The biggest change most users notice initially in SharePoint Designer 2010 is the addition of the Ribbon. Although users new to the Ribbon UI paradigm might be overwhelmed at first by the new way of accessing commands, Microsoft argues that it is an innovative way to cleanly display many commands that would have otherwise been buried in menus or various options windows. Figure 4-13 shows a sample Ribbon from SPD when the user navigates to the Workflows area in the Site Objects pane for a specific site.

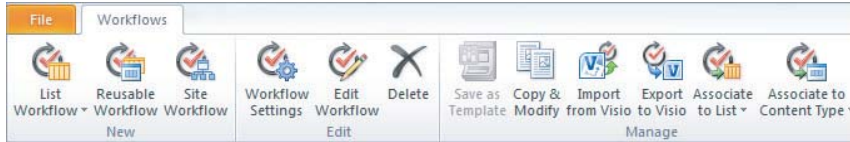


FIGURE 4-13

In addition, the workflow designer itself is improved by enabling users to type the name of the action they wish to insert. This allows for a more natural approach to defining a process, and probably eases the learning curve for new SPD users who are not yet familiar with all the available actions. For example, if a process called for a condition that compares the user who created a document to some other user, the workflow developer could simply type *if* in the designer and be presented with a list of workflow actions that they might find useful. Figure 4-14 illustrates this feature.

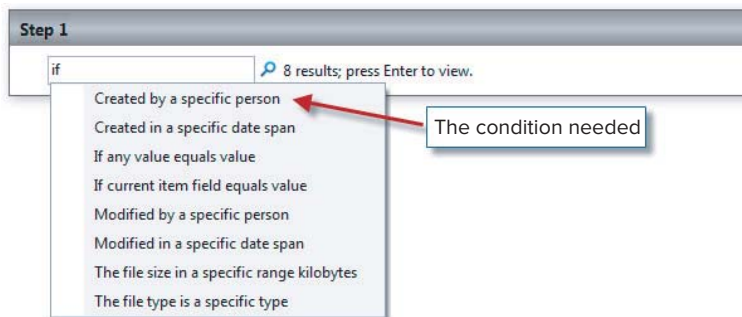


FIGURE 4-14

Finally, the new designer makes it easy to understand how items are nested in blocks. This is important when dealing with a lot of actions that may be inside of if-else and parallel blocks.

Reusable Workflows

In SPD 2007, workflows were bound to a specific list at creation time and this list could not be changed later. In addition, the workflow definition, once completed, could not even be copied and used in a different location. This severely limited the use of this tool for many enterprises. Among other reasons, this limitation was an issue because it did not allow for a standard development life cycle whereby an initial version of the workflow could be developed in a sandbox (i.e., a development environment), moved to a testing environment, and finally promoted to production.

Reusable workflows in SPD 2010 allow a workflow definition to be associated to any list or content type in the site in which the reusable workflow was defined. There are a few caveats to this statement:

- The workflow definition must be created as a reusable workflow.
- If the workflow is limited to a specific content type when created, then the list or content type to which an association is attempted must not conflict with this content type.
- If the workflow is limited to a specific content type when created and a list association is being attempted, the list must have content type management enabled so that the appropriate content type can be selected during association.

In addition to associating reusable workflows to any list or content type in the site in which it was defined, a workflow definition can be promoted globally. Global promotion means that the workflow is reusable in any site in the current site collection. To promote a reusable workflow to be globally reusable, open the workflow in SPD and then click the Publish Globally button in the Workflow Ribbon tab.

The Task Process

The task process is a new action provided in SharePoint Designer that simplifies and consolidates what Microsoft identified as very common patterns when dealing with the life cycle of a task or tasks in cases of multiple users. While it is a singular action, it provides a much more robust designer that offers a great deal of control over how user tasks behave. In addition, there are three ways, via specific actions, to use this task process feature:

- The Start Approval Process action
- The Start Custom Task Process action
- The Start Feedback Process action



The Start Approval Process and Start Feedback Process actions are simply pre-configured versions of the Start Custom Task Process action.

Because the task process enables you to do so much, the following list describes its features:

- Allows each task assigned to contain a customizable list of fields.
- Each task can have a specific outcome that can be chosen from a predefined list. The user is presented with a button for each task outcome. A common example would be “Approved” or “Rejected.”
- Settings for each task:
 - Secure each task so that only the assignee or workflow owner can see/edit a task (or not).
 - Allow tasks to be reassigned (or not).
 - Allow task assignees to request a change (or not).

- Specify the conditions which determine when a task process is completed. For example, if any of the task process participants approve or reject a document, then the task process completes. Another process might end when all of the participants have approved or rejected the document.
- Specify the behavior of a single task. This enables the workflow developer to create a flow at specific points in the life cycle of an individual task. These life cycle events are as follows:
 - Before a task is assigned
 - When a task is pending — that is, right after it is created
 - When a task expires — that is, when it is incomplete after its due date
 - When a task is deleted
 - When a task completes
- Specify the behavior of the overall task process. This enables the workflow developer to handle task process life cycle events.
 - When the task process starts — that is, when the task process action is reached in the workflow but before any tasks are assigned.
 - When the task process is running — that is, before any tasks are assigned. This is useful for managing events raised on the document or item itself, such as when it is deleted or changed.
 - When the process is canceled.
 - When the process completes. This could happen either when the last task is completed or when the End Task Process action is called from elsewhere.

Clearly, there is a lot to configure for one action! This is why the Approval and Feedback actions were included out of the box, to provide a starting point where a workflow developer can simply tweak what Microsoft believes to be a common task use case.

High-Privilege Workflows

In SPD 2007 workflow, there were issues related to security and under which security context a workflow instance was running. In SPD 2010, it is simplified and enhanced. By default, the workflow always is running under the security context of the workflow initiator. This could very well be a nonprivileged user who may not have the appropriate rights to perform crucial workflow steps (copying documents, creating sites, provisioning security, etc.). For this reason, the *user-impersonation step* was added.

Actions run inside this step are run under the security context of the workflow author; more specifically, the last person who saved the workflow definition. In addition, a couple of condition actions are only available inside the user-impersonation step:

- Check List Item Permissions
- Check List Item Permission Levels

Customizing Out-of-the-Box Workflows

In addition to reusable workflows, which greatly boost productivity by offering reuse of already created logic, the Copy & Modify feature provides an excellent way to start a new process that might be fairly similar to one already created. This feature not only enables you to copy and modify workflows created in your SharePoint installation, but out-of-the-box workflows as well. This will likely be a very popular feature. For example, many scenarios require an Approval workflow, which was discussed in some detail earlier in this chapter. For many of these cases, 80% of what is needed for a particular solution can be served by the provided workflow, with minor tweaks.



This feature applies to declarative workflows only. Custom Visual Studio workflows or the Three-state workflow (which is not a declarative workflow), for example, cannot be customized using the Copy & Modify feature.

Workflow Visualization

Another really nice feature that was added to declarative workflows in 2010 is the capability to view a workflow's progress visually. This was a very common need in SharePoint 2007 with no easy answer, especially using standard approaches. Whether or not a declarative workflow begins its life in Visio, a Visio Services visualization can be automatically generated and displayed on the workflow status page along with the tasks and history related to a particular workflow instance.

To generate this visualization, a checkbox needs to be enabled on the Workflow Settings page for the particular workflow. It can be found in the Settings section of that page and is labeled "Show workflow visualization on status page."

If this visualization is an important part of a particular solution, then it might be advisable to either start the workflow in Visio or at least export it from SharePoint Designer to Visio so that useful labels for each shape can be applied. Otherwise, the workflow shapes in the visualization will have very generic labels that only indicate the type of action. Figure 4-15 shows an example of a simple visualization representing a workflow in progress that has also been given useful labels in Visio prior to being deployed to SharePoint. Note the checkmarks and circle icons on each shape; these indicate workflow progress.



Even though workflows can display useful text on each action's shape, the layout that is defined in Visio will not carry through to the visualization. Rather, a standard layout algorithm is applied, which overrides any layout performed by the workflow author.

Finally, note that everything in the workflow is featured in the visualization. Therefore, even the more tedious actions (logging to history, performing date calculations) are displayed to the user. This could potentially lead to confusion if it is a rather involved workflow with a lot of actions. This is another reason why it is advisable to annotate each action using Visio prior to deploying an SPD workflow with the visualization feature turned on.

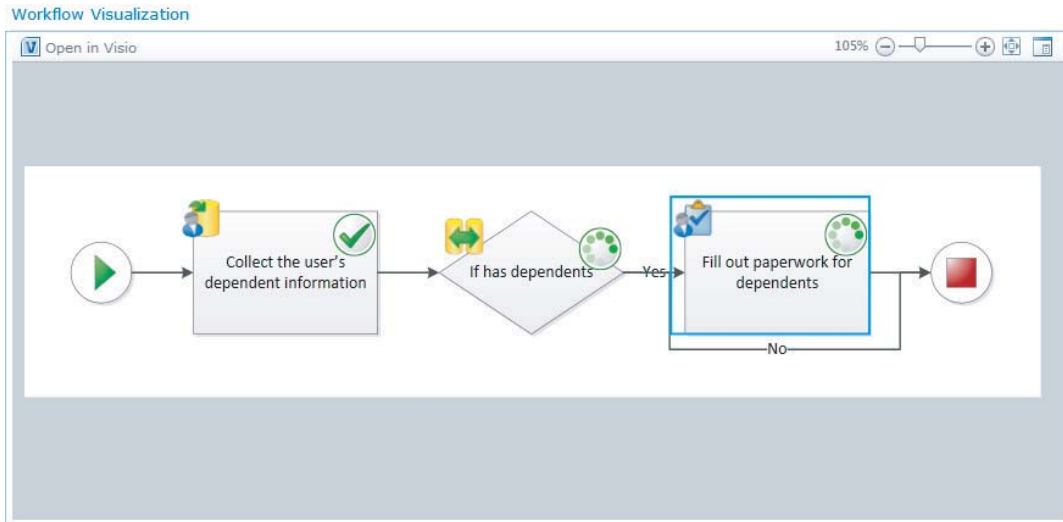


FIGURE 4-15

Association Columns

Association columns are a way to enforce that a particular set of columns exists on the list or library in which an association is created. Figure 4-16 shows the dialog for configuring association columns. In this example, two columns, Candidate Name and SSN, will be added to any list or library whenever an association is created.

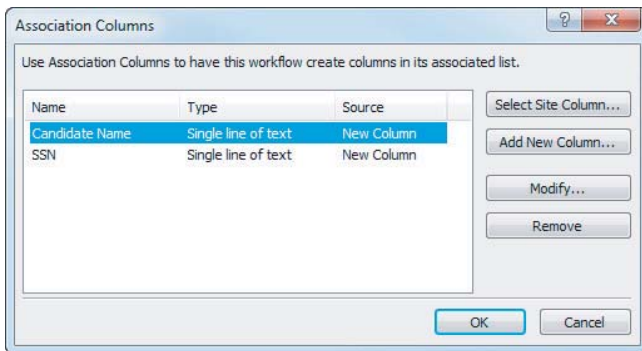


FIGURE 4-16

Another way to enforce columns is to only allow a workflow definition to be tied to a specific content type that contains the columns needed by the workflow. If the workflow is more ad-hoc or smaller scale in nature, or if it should be able to be associated with more than one content type, then using association columns could be a good choice. However, if the option exists to enforce content type association, then that option should be considered. This provides for a more consistent enterprise experience in terms of findability and consistency of metadata.

Workflow Actions in SharePoint Designer

The out-of-the-box actions in SharePoint Designer are a superset of those found in Visio. It is not necessary to enumerate all these actions here because they are well documented online. There are two good pages on Microsoft's Office site that review declarative workflow actions:

- <http://bit.ly/spdactions> — This page documents each SPD action and how they can be used together.
- <http://bit.ly/spdvisioactions> — This page provides an overview of the Visio actions and how they relate to SPD actions.

Custom Actions

Another strength of SharePoint Designer workflow is that it is extensible — new actions can be developed and exposed for use in SPD workflows. Although the process of developing a custom SharePoint Designer action is outside the scope of this book, following is a brief overview.

It is important to understand that creating a custom SPD workflow action requires .NET development as well as the ability to deploy assets to the file system of the farm's web front-end servers. The first step is to write the workflow logic, inheriting from the appropriate classes in the SharePoint and/or workflow namespaces. Because this assembly must be deployed to the GAC, it needs to be strongly named.

Next, an entry must be added to the `web.config` file for the appropriate web application to indicate that the activity in question is safe. Finally, a file with a `.actions` extension containing specific XML must be added to the file system on the web front end; specifically, in the `14\TEMPLATE\1033\Workflow` folder (for the 1033 locale which is United States English). This file points to the assembly located in the GAC, and provides the textual information that ends up being displayed in SharePoint Designer when the user is creating the workflow.

Authoring SharePoint Designer Workflows

This section walks the reader through the development of a SharePoint Designer workflow. The purpose of the example workflow is to process an incoming invoice. For this scenario, imagine that an enterprise, in an attempt to reduce paper usage and improve process visibility, has implemented a document imaging system that routes invoices to the appropriate people for approval and payment. The workflow will have the following steps:

1. Match the invoice to the purchase order (PO) and obtain a purchase order number.
 - If it cannot be matched to a PO, then e-mail a manager and end the workflow.
2. If the invoice amount is greater than \$5,000, then obtain approval from the CFO.
 - If the invoice is rejected, then e-mail a manager and end the workflow.
3. Assign a task to a manager to release payment of the invoice.

To get started, create a new workflow in SharePoint Designer by opening a site of your choosing, navigating to the Workflows page in the Site Objects menu, and clicking the Reusable Workflow button in the New group of the Workflow Ribbon tab. This will present the dialog shown in Figure 4-17. Leave the content type restriction set to All. In a complete solution, a new Invoice content type would probably be created for consistency of metadata. However, for this simple walkthrough, the association columns feature discussed earlier will be used to hold invoice metadata.

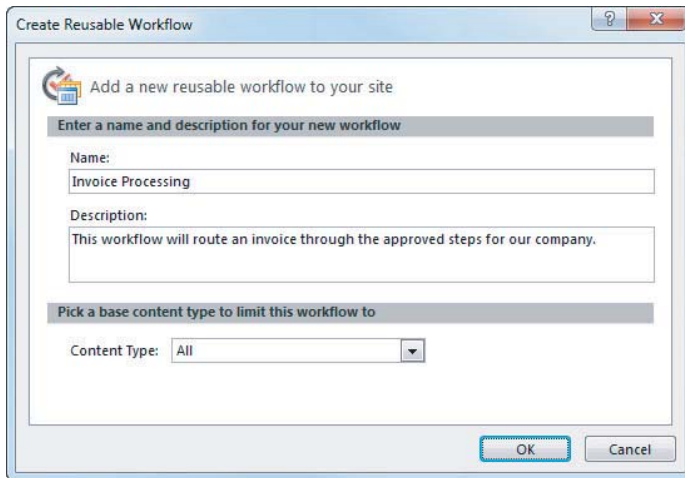


FIGURE 4-17

At this point the SPD workflow designer and the Workflow Ribbon tab are displayed. The Workflow Ribbon tab contains all the standard commands that are needed to author an SPD workflow, including the following:

- Saving/publishing functionality — A workflow in development can be saved so work is not lost, but it needs to be published before it can be used.
- Actions — The capability to add conditions and actions is found here. In addition, steps and other blocks (parallel, impersonation) can be inserted.
- Other workflow management functions:
 - Globally publish to all sites in a site collection
 - Export to Visio
 - Access workflow settings
 - Configure initiation/association form field
 - Manage workflow variables
 - Manage association columns

Next, add two new association columns to represent the PO number and the invoice amount. The PO number can be a single line of text, and the invoice amount should be currency. In the Association Columns dialog, choose to add new columns, rather than select existing site columns. Figure 4-18 shows the configured columns.

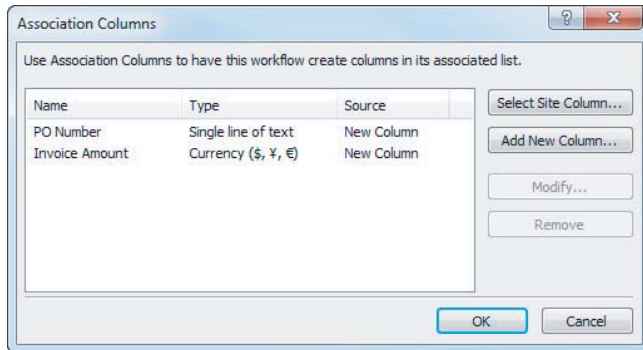


FIGURE 4-18

The workflow logic should now be added by using the designer. There are two options for adding actions to the designer: by typing in an action’s name or by selecting an action from the Ribbon. Feel free to use either approach. In both cases, the following actions should be added:

- An “If any value equals value” action. This will be used to ensure that the invoice amount has been entered; make sure it is greater than 0.
- A “Collect Data from a User” action. Add it inside of the previously added “if” action. This will be used to obtain a PO number. Make sure the PO number can be left blank to account for cases in which a PO cannot be found. This action works by collecting data from a user and storing said data in a task list item. Therefore, this action sets a workflow variable (of type “List Item ID”). This variable will be used in the next step.
- Another “If any value equals value” action. This action will be used to ensure that the PO number captured in the task is not empty. Because we have the task’s list item ID, a lookup against the task list can be performed in order to obtain the PO number. Figure 4-19 shows how the lookup should be configured. It’s actually fairly simple to configure these lookups, but the dialog might be a little confusing to a first-time user.
- Add an “Else-If” branch from the Ribbon after the previously added “if” action. Now there are two blocks in which to perform the appropriate actions.
- Keep adding actions to meet the requirements outlined earlier. The result should look something like Figure 4-20. Here are a few tips in case you get stuck:
 - To obtain the approval status, use a “Collect Data from a User” action and add a column of type option with two choices: Approve and Reject.
 - Use an “Assign a To-do Item” action in order to assign the release payment task.

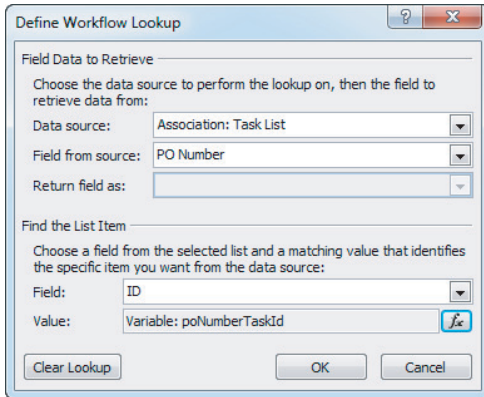


FIGURE 4-19

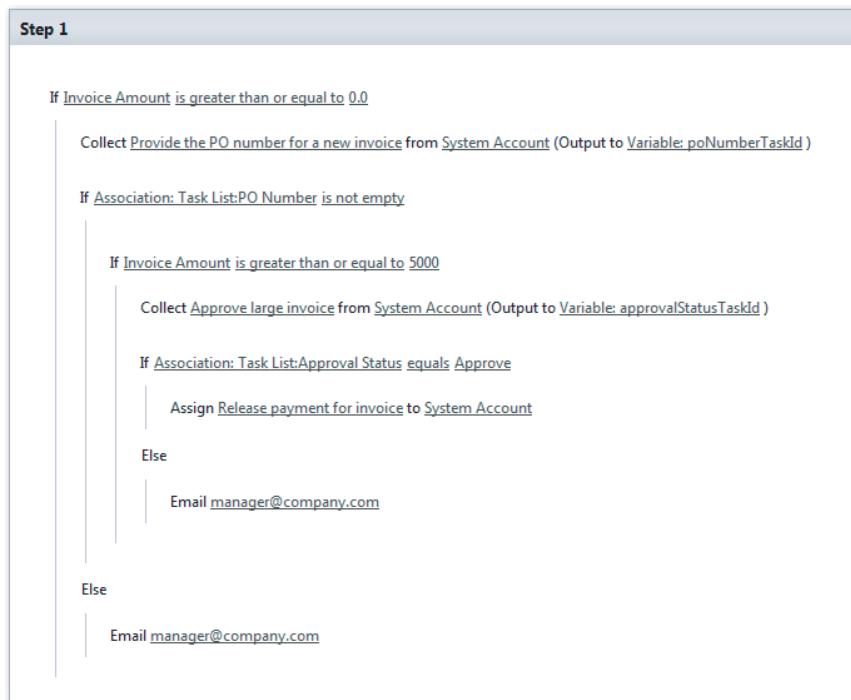


FIGURE 4-20

Visual Studio Workflows

The following sections discuss custom workflow development using Visual Studio. This is the final development approach discussed and available in SharePoint 2010. Visual Studio workflows are definitely the most powerful in terms of sheer flexibility because the developer has the full use of the .NET framework at hand. With great power comes great responsibility, meaning that now code

has to be maintained and thoroughly tested before using it in a production environment. SharePoint Designer workflows and out-of-the-box workflows do not let you proverbially “shoot yourself in the foot” as easily.

The following sections will take you through some of the improvements in Visual Studio 2010 workflows, including a greatly enhanced and first-class development environment for SharePoint. In addition, an in-depth exercise will introduce advanced concepts required to develop Visual Studio workflows. This includes correlation tokens, workflow pages, and managing the workflow task lifecycle.

Improvements

Visual Studio 2010 offers a massive improvement over Visual Studio 2005 or 2008 plus the downloadable Visual Studio Extensions for SharePoint. It is obvious that development tooling was a huge area of investment for Microsoft in 2010; SharePoint is now a first-class product, rather than an afterthought.

Most notable is the addition of the feature and solution designers. Features and the “package” (the SharePoint solution) now show up as nodes in Solution Explorer. Feature properties and items can be edited in a visual designer, rather than by hand-editing XML files. (Don’t worry; if you like the XML route or need extra control, this approach is still supported.)

In addition, many new project and item templates are available. Figure 4-21 shows the list of project templates available when creating a new project.

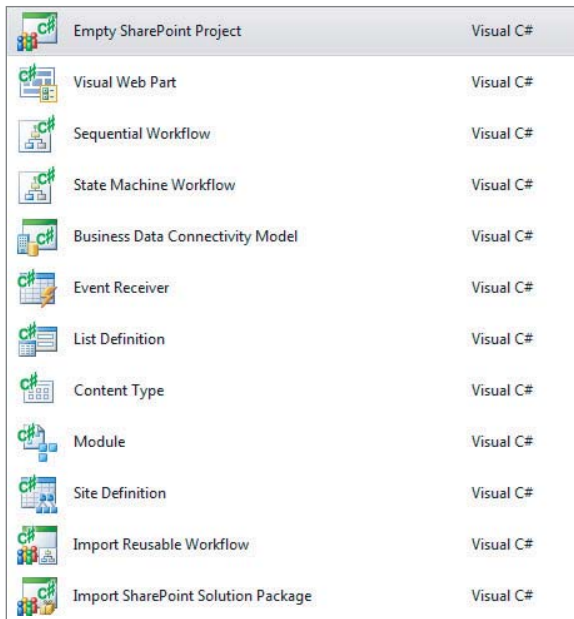


FIGURE 4-21

In addition, there are now item templates for things like workflows, Web Parts (visual and traditional), application pages, content types, event receivers, and so on.

The debugging experience for SharePoint in Visual Studio 2010 is also greatly improved. For workflow development, as shown later in this chapter, you can debug against a development site, a list, and so on. The developer then can simply press F5 and everything, including solution deployment, feature activation, and workflow association, is taken care of automatically, and debugging can immediately commence.

Creating a Workflow in Visual Studio: An Exercise

In this section you will get hands-on experience by walking through the development of a workflow in Visual Studio. The logic of this workflow will be kept fairly simple in order to focus on some of the more important technical aspects of getting up and running with workflow development in SharePoint. The logic was chosen purposefully so that key pieces of functionality are covered.

The purpose of the workflow you will develop in this section is to collect feedback from a group of people regarding a document. It has the following requirements:

- Given a list of user names, assign each of them a task in parallel in order to obtain the desired feedback.
- Once all feedback has been received, generate a summary string of all feedback.
- Finally, e-mail the person who started the workflow a status update, including the summary string mentioned above.

First, open Visual Studio 2010 and create a new Sequential Workflow project by navigating to File ⇨ New ⇨ Project. In the New Project dialog, name the project something like “FeedbackWorkflow” as shown in Figure 4-22.

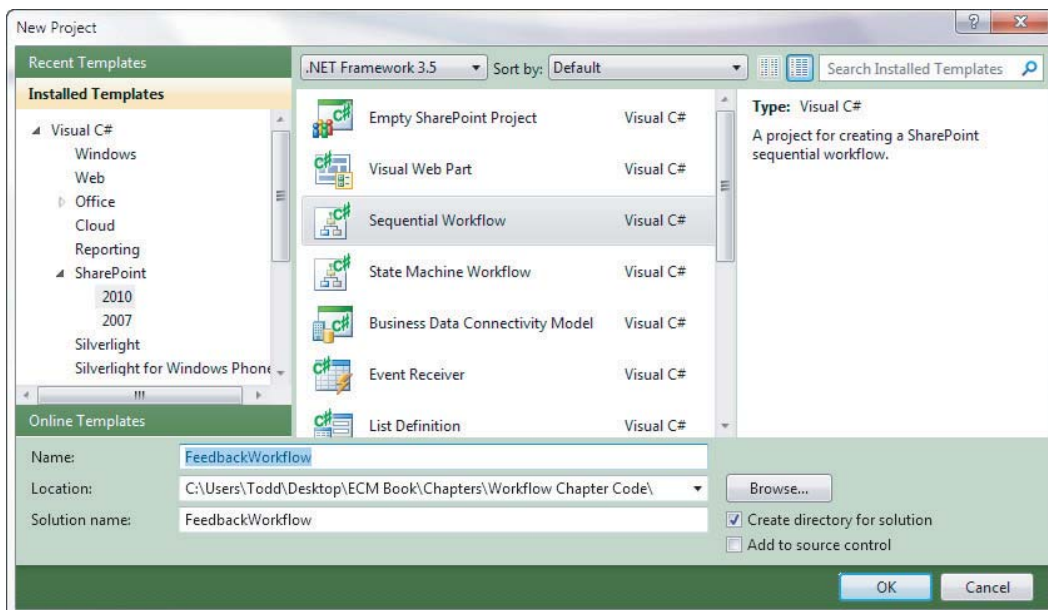


FIGURE 4-22

Visual Studio provides a useful wizard that helps users configure the local development environment for debugging. Follow the instructions for the four screens in this wizard:

- 1.** The first screen of the wizard asks the user for the SharePoint site to use for debugging purposes. Simply enter your local site or the appropriate site you wish to deploy to for development purposes.

You are also asked if you would like to create a sandboxed solution or a farm solution. It is beyond the scope of this book to explain the differences between sandboxed and farm solutions, but it is useful to know that farm solutions are exactly the same as all solutions in SharePoint 2007. Certain features in SharePoint 2010 can only be deployed as a farm solution, and workflow is one of these features. Therefore, this screen does not allow you to choose sandboxed solutions as an option.
- 2.** The second screen asks you to name the workflow and choose whether to create a list or a site workflow. For the purposes of this exercise, because the workflow runs against a specific document, you should choose List Workflow.
- 3.** Screen number three is all about workflow association. Here you can indicate whether Visual Studio should perform a workflow association or not (leave this checkbox checked), as well as the libraries and lists that are relevant to the workflow, including the list with which the workflow itself will be associated and the related task and history lists. The default values will suffice just fine for the purposes of this exercise. However, you can choose to create a new document library if you want to keep things separate from other testing you might be doing in a site.
- 4.** The last screen, also related to workflow association, enables configuration of the workflow's start options. The default values of manual start and start on create are fine for this exercise. Click Finish to create the project.

Now you can begin implementing the workflow logic outlined earlier. First, you'll lay out the workflow activities in the graphical designer before writing any code or doing any real configuration. This will give you a feel for what the workflow is going to do before attempting anything too complicated.

If you look at the newly created workflow, you'll notice that the default workflow created with the project already contains an `OnWorkflowActivated` activity. Every SharePoint workflow needs to have this as its first activity. It performs important startup functions and provides a context object to the workflow instance that includes crucial information about the environment in which the workflow is running. This context object is of type `SPWorkflowActivationProperties` and includes properties such as `AssociationData`, `InitiationData`, `Originator`, as well as properties representing the item on which the workflow is currently running, the list it is running in, and the related task and history lists. By default, the activation activity is configured to bind the context object to a variable in the code-behind of the workflow.

Logically, the first thing you need to do in this workflow is create tasks for each of the users specified by the person who starts the workflow. In this case, `ReplicatorActivity` is a perfect fit for what you need to do. `ReplicatorActivity` merely houses child activities and ensures that these children are executed n times, one time for each item in a specific collection. In this case, the child activities will be those responsible for managing tasks, and the collection will be the list of users from whom you want to collect feedback.

Begin by dragging a `ReplicatorActivity` from the Toolbox to the workflow designer. Next, immediately drop a `SequenceActivity` from the Toolbox into the `ReplicatorActivity` you just added; this will make the `SequenceActivity` its child. Adding the `SequenceActivity` as a child is necessary because `ReplicatorActivity` can contain only one child, whereas `SequenceActivity` can contain many children. In fact, the only reason `SequenceActivity` exists is to run a series of children activities.

Now you are ready to add the activities to manage the tasks inside the `SequenceActivity`. Add the following activities in the following order:

- A `CreateTask` activity
- A `While` activity
 - An `OnTaskChange` activity. Add this activity as a child to the `While` activity that was just added.
- A `CompleteTask` activity

Now, outside the `ReplicatorActivity`, add the following activities:

- A `Code` activity. This will generate a summary string and save it to a class member.
- A `SendEmail` activity

After performing these actions and perhaps giving each activity a meaningful name by using the standard Visual Studio properties window, the workflow should look similar to the one in Figure 4-23.

Notice there are a bunch of red dots with exclamation points on them. You can probably guess that these activities are not fully configured or are not configured correctly. Fixing these issues is the next order of business.

First, we will work on configuring the three task activities. If you highlight the `CreateTask` activity, its properties will be displayed in the Visual Studio Properties window. From this window it is evident that one of the properties, `CorrelationToken`, is not correctly configured, as shown in Figure 4-24.

Configuring Correlation Tokens

Correlation tokens in workflow are very important because they enable the workflow engine, which is running inside of SharePoint, to correctly associate an external event (such as the changing or completion of a task) to a particular instance of a workflow activity. Consider the workflow you are currently building. There are several activities for task management (create, monitor change, and complete). However, these activities are going to be cloned at runtime for each person participating in the workflow. Therefore, each set of task activities needs a unique correlation token so that the workflow engine can uniquely identify the task in question.

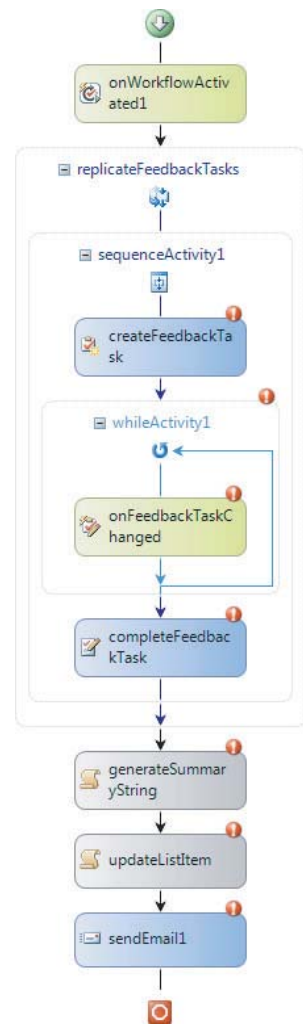


FIGURE 4-23

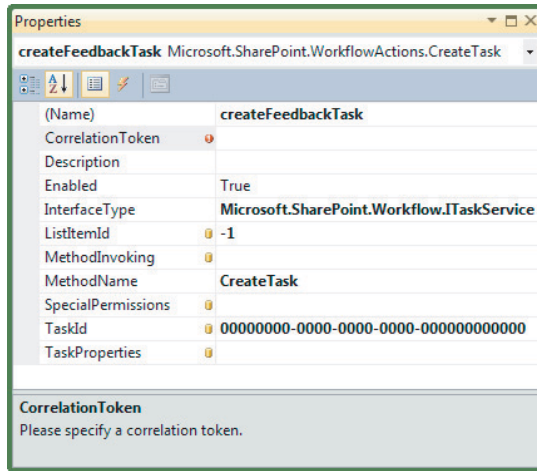


FIGURE 4-24

You don't generally need to worry about the inner workings of correlation tokens beyond having a solid understanding of what they are used for and how to configure them; the workflow engine takes care of the heavy lifting for you. In this case, you need to configure the `CorrelationToken` property on each of the three task-related activities. To do this, enter a name for this property for the `CreateTask` activity. Any name will do, but in this case something like `FeedbackTaskToken` would be appropriate.

After naming the token for the first task-related activity (which initially creates the token, by the way), the `CorrelationToken` property becomes expandable. Expanding the property exposes a subproperty called `OwnerActivityName`. Because you are creating this token to group the task-related activities together, a good choice for the parent activity would be the `SequenceActivity` in which the task activities are placed. Next, set the correlation tokens on the `OnTaskChanged` and `CompleteTask` activities as well; this enables you to select the correlation token via a drop-down, rather than type the name in.

Notice that the red bubbles are no longer visible on the task activities. This is good, but you are not fully done configuring these activities. You haven't yet described the tasks or indicated to whom they should be assigned. We will get to this in a little bit.

Workflow Pages

At this point it would be useful to digress a little and examine how this workflow will interact with the user. Remember that, based on the requirements for this workflow, you need to collect feedback from a specified list of users. An obvious question would be how you are going to get this list of users. The workflow infrastructure in SharePoint 2010 provides a great capability for interacting with users: *workflow pages*. These are implemented using ASP.NET web forms. Visual Studio provides support for two types of workflow pages out of the box: *association pages* and *initiation pages*.

Association pages are used to provide a user interface when a workflow definition is bound to — or, more appropriately, associated with — a list, library, or content type. Therefore, any data captured on an association page is not related to a specific instance of a running workflow.

Rather, the data is global to the workflow association, applying to all workflow instances. Conversely, initiation pages are displayed when a workflow instance is beginning. Therefore, all data captured is specific to one workflow instance only.

Because you need to capture a list of users when a new workflow instance is starting, you need to utilize a *workflow initiation form* as part of the solution. To add a new form, right-click the workflow in Visual Studio's Solution Explorer window and select Add ⇨ New Item. From the dialog that appears, select Workflow Initiation Form and give it a name like `WorkflowInitiationForm.aspx`. The result will look similar to the Solution Explorer shown in Figure 4-25.

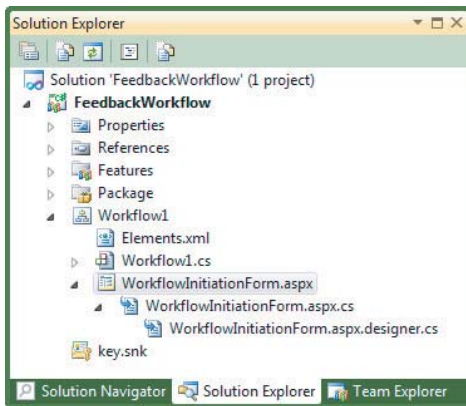


FIGURE 4-25

The base initiation page includes some default ASP.NET markup, including a SharePoint master page and several content placeholders. This gives you a good starting point to create your initiation page. In addition, the code-behind of the web form contains a decent amount of boilerplate code. In most cases you won't need to modify this boilerplate code, but it is certainly good to understand what it is doing in case you need to troubleshoot or potentially modify it down the road. Essentially, this code does three things:

- `InitializeParams` is called from the `Page_Load` event handler and parses a few parameters from the URL's query string.
 - The association ID for the workflow
 - The ID of the list with which the workflow is associated (if it is a list workflow, not a site workflow)
 - The ID of the list item the workflow is running against (if this is a list workflow)
- `HandleStartWorkflow` is called from the Start Workflow button click event handler. It first determines whether a site or list workflow is being invoked, and then starts a new workflow instance accordingly.
- `GetInitiationData` is another method that is automatically added to the code-behind class. This is where you would prepare and return any data, in the form of a string, that you want to be able to pass to your workflow. This is where you prepare the list of users who will be interacting with your workflow as feedback providers.



A detailed discussion of dependency properties is beyond the scope of this book, but dependency properties are “enhanced” properties managed by .NET. They enable capabilities like property inheritance (child activities can expose properties from parent activities).

You can obtain the workflow’s initiation data in the `OnWorkflowActivated` event handler, created earlier. The following code shows this event handler, as well as some of the pertinent class level members. Obtaining the initiation data is as easy as referencing the `InitiationData` property of the `workflowProperties` member. Because you know that the initiation data is simply a comma-delimited string, the `String.Split` method is needed to set the class member called `people`, which is a string array:



Available for
download on
Wrox.com

```
public SPWorkflowActivationProperties workflowProperties =
    new SPWorkflowActivationProperties();
public string[] people = null;

private void onWorkflowActivated1_Invoked(object sender, ExternalDataEventArgs e)
{
    string peopleString = this.workflowProperties.InitiationData;
    people = peopleString.Split(new char[] { ',' },
        StringSplitOptions.RemoveEmptyEntries);
}
```

Code snippet FeedbackWorkflow.cs

Now that you’ve collected the actors via the initiation page and can access this list in the workflow, it is time to configure the `Replicator` activity that was added to the workflow designer earlier. The `people` array, which is defined in the workflow code-behind, needs to be bound to the `Replicator`’s `InitialChildData` property; anything that implements `ICollection` (`Array` implements this interface) can be bound to this property. The data that ends up in this property will be used to replicate the child of the `Replicator` activity. Therefore, if the initiator of the workflow types in the names of three users, the feedback tasks will be replicated and run in parallel three times.

To bind the child data property to the `people` array, click the ellipsis in the `InitialChildData` property. The resulting dialog, shown in Figure 4-26, will allow you to choose a workflow class member to which to bind.

Workflow Task Properties

At this point you’re ready to configure the task-related activity properties. Because this workflow is using the `Replicator` activity, it is crucial to understand how the workflow definition (what is seen in Visual Studio) relates to what happens to the workflow at runtime. The workflow definition in this example application says that for each item in the `Replicator`’s child data list, execute the children as defined in the workflow definition. Therefore, at runtime, the direct child of the `Replicator`, a `Sequence` activity in our example, is created as a new executable instance of that

activity tree once per item in the collection. Because of this fact, you must treat an instance of any child of the `Replicator` as a unique instance even though the definition only defines each activity once. If you have not done any workflow development before, this might sound a little confusing at first, but after working through this exercise and considering why this would be true, it should make more sense soon.

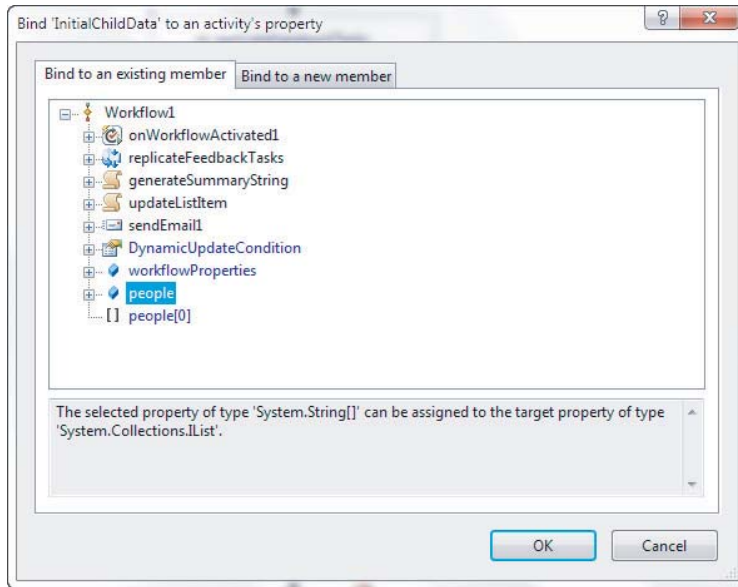


FIGURE 4-26

Start by wiring up an event handler to the `MethodInvoking` event of the `CreateTask` activity; it is in this event handler that the relevant task properties will be set on the field that was just created. To better understand how the activities are executed at runtime, the following code snippet demonstrates how *not* to set the task properties in this event handler:



Available for
download on
Wrox.com

```
private void createFeedbackTask_MethodInvoking(object sender, EventArgs e)
{
    createFeedbackTask.TaskProperties = new SPWorkflowTaskProperties();
    createFeedbackTask.TaskProperties.DueDate = DateTime.Today.AddDays(2);
    // more property setting code here...
}
```

Code snippet FeedbackWorkflow.cs

This is wrong because the code is referencing the `CreateTask` activity from the workflow definition directly (`createFeedbackTask`). Remember that if there are three items in the `Replicator`'s child data list, there will be three unique instances of the `CreateTask` activity as well! Therefore, the correct way to set properties on the `CreateTask` activity is to reference the `sender` parameter of this event handler, which is the current unique instance of the `CreateTask` activity. In addition,

you need to obtain the current instance of the `Replicator`'s child data, which represents a username that will be assigned a new task. The following code snippet does both of these tasks, and sets the pertinent properties needed to create the task with sufficient information:



Available for
download on
Wrox.com

```
private void createFeedbackTask_MethodInvoking(object sender, EventArgs e)
{
    // get a reference to the current CreateTask activity
    CreateTask taskActivity = sender as CreateTask;

    // get the current child data instance
    ReplicatorActivity replicator =
        taskActivity.Parent.Parent as ReplicatorActivity;
    int i = replicator.DynamicActivities.ToList().IndexOf(taskActivity.Parent);
    string assignee = (string)replicator.CurrentChildData[i];

    // create a new task id
    taskActivity.TaskId = Guid.NewGuid();

    // set the task properties
    taskActivity.TaskProperties = new SPWorkflowTaskProperties();
    taskActivity.TaskProperties.Title = "Feedback needed!";
    taskActivity.TaskProperties.AssignedTo = assignee;
    taskActivity.TaskProperties.DueDate = DateTime.Today.AddDays(2);
    taskActivity.TaskProperties.SendEmailNotification = true;
}
```

Code snippet FeedbackWorkflow.cs

Next, finish configuring the three task-related activities (`CreateTask`, `OnTaskChanged`, and `CompleteTask`) by simply creating and wiring up each activity's `TaskId` property to a class field called `taskId`. This is needed because each activity instance needs to know with which task it is associated.

The next step is to implement the `CodeActivity` instance to generate the summary string that will be e-mailed to the workflow originator. Do this by creating an event handler for the `ExecuteCode` event and implementing the following code:



Available for
download on
Wrox.com

```
StringBuilder sb = new StringBuilder();
foreach (SPWorkflowTask task in this.workflowProperties.Workflow.Tasks)
{
    sb.Append(task["Description"].ToString());
}

this.summaryString = sb.ToString();
```

Code snippet FeedbackWorkflow.cs

This code is simply enumerating each task associated with this workflow instance and building a string based on each task's `Description` field. It then sets the concatenated string to a class level member, which is used in the next and final step of the workflow.

It is now time to configure the last activity of the workflow: the `SendEmail` activity. There are two steps. First, set the activity's `CorrelationToken` property to the same token used in the workflow's first activity, the `OnWorkflowActivated` activity. If this same token is not used, the e-mail activity will not function correctly. Next, create an event handler for the activity's `MethodInvoking` event. In this handler, implement the following code:



```
SendEmail email = sender as SendEmail;
if (email != null)
{
    email.To = this.workflowProperties.OriginatorEmail;
    email.Subject = "Feedback cycle completed";
    email.Body = this.summaryString;
}
```

Code snippet FeedbackWorkflow.cs

This code first obtains a reference to the `SendEmail` activity that is being executed and then sets various properties. Notice that the `To` property is set to the originator's e-mail address, which is handily available from the `workflowProperties` member. Finally, the body of the e-mail is simply the summary string that was built in the previous step.

That's it! Using Visual Studio's deployment capabilities, testing the workflow is now as easy as pressing F5. The only steps needed to run the workflow from here are to manually start the workflow using the SharePoint UI and then enter the comma-delimited list of users in the initiation screen, created earlier in this exercise.

INFOPATH

While InfoPath itself has broader implications for the ECM space in SharePoint, it plays a very specific role in SharePoint workflow. InfoPath augments the workflow UI in several ways, as enumerated in the following sections. This general-purpose forms tool offers workflow developers, especially those developing workflows in SharePoint Designer, a very handy yet powerful method for creating custom user interfaces with which workflow participants can interact.

Out-of-the-Box Workflows

Out-of-the-box SharePoint workflows utilize InfoPath in several areas. Take the Approval workflow, for example. The association, initiation, and individual task forms are all designed and rendered using InfoPath and InfoPath Forms Services, respectively. Practically, however, this has little implication for either end users or developers, other than being aware of this fact.

SharePoint Designer Workflows

This is where the InfoPath integration in 2010 starts to get interesting for workflow. There are a number of places where InfoPath forms are automatically generated and used to gather data from end users. Moreover, these automatically generated forms are enumerated in SharePoint Designer in the Workflow Settings screen and are editable in InfoPath Designer.

Association and initiation forms are two of the automatically generated form types. These are generated based on the fields configured in the Initiation Form Parameters dialog. In reality, only one InfoPath form is generated, with multiple views for the separate initiation/association fields, which are not shown on both form types. Task forms are also automatically generated for you. For example, if a Collect Data from a User action is configured to collect several different fields, a form will be generated for this purpose.

As mentioned, either of these two form types can be edited in InfoPath for a more controlled user experience; simply double-click a form in the Forms section of the Workflow Settings screen in SharePoint Designer. At this point, InfoPath Designer will open and editing can commence. Note that once a form is edited manually, any updates to the fields from inside the workflow will not be reflected without further manual intervention in InfoPath Designer. However, if the previous customizations are not needed going forward, the customized form can be deleted and then regenerated in SharePoint Designer. This option for customization increases SharePoint Designer's potential to offer "no code" solutions to more people.

Visual Studio

Aside from how they are configured, InfoPath forms in the context of Visual Studio workflows are not all that different from SharePoint Designer workflows. InfoPath can be used for a workflow's association, initiation, and task forms. These forms are wired up to a workflow by configuring specific XML in the workflow deployment element. Obviously, workflows can be run against InfoPath items in a forms library as well.

PLUGGABLE WORKFLOW SERVICES

The following sections cover a new feature in SharePoint 2010: pluggable workflow services. Pluggable workflow services are also referred to as local services, external data exchange services, or simply workflow services. This feature was available in WF-proper since .NET 3.0, but it hasn't been available in SharePoint until now.

Why You Need Workflow Services

Before diving into the technical details and how to develop workflow services, it would be helpful to understand why these services exist and why they might prove useful in certain scenarios. A common use case is one in which the SharePoint workflow needs to interact with an external line of business system by sending it messages and, very importantly, receiving notifications from this external system. For example, your organization might have a document imaging solution that processes invoices with SharePoint and workflows. The workflow might need to notify various systems throughout the enterprise about new invoices and receive information back into the workflow once individual systems have performed any processing work needed.

Aside from using unorthodox methods, this was simply not possible in SharePoint 2007. (One method might have been to place tasks in a SharePoint list that was not intended for humans, but rather external systems, to update; and therefore provide the workflow with the necessary data and event it would need to proceed.) Workflow services provide a more natural way to interact with

external systems or to simply provide workflows with events from external sources. In addition, workflow services can be used to provide an inter-workflow messaging mechanism.

It is important to understand that using workflow services is an entirely custom proposition, meaning Visual Studio is a requirement here.

Authoring Custom Workflow Services

This section walks through an example service not unlike the one covered earlier. This workflow service will allow for the creation of an invoice and then fire an event whenever the invoice is updated. Normally the invoice would be created in an external system, but for the purposes of this example the invoice will be created and processed in code within the workflow. The following list provides the overall context for the steps required to implement a workflow service:

1. Create any event arguments classes needed, inheriting from `ExternalDataEventArgs`.
2. Create the service interface using the `ExternalDataExchange` attribute.
3. Create the service, inheriting from `SPWorkflowExternalDataExchangeService` and implementing the interface from step 1.
4. Configure the service in SharePoint in `web.config`.

To meet the needs of the service described here, the service should provide one method called `CreateInvoice` and one event called `InvoiceProcessed`. The following code shows what would be required to meet steps 1 and 2:



```
[Serializable]
public class InvoiceProcessedEventArgs : ExternalDataEventArgs
{
    public string InvoiceNumber { get; set; }
    public bool Approved { get; set; }

    public InvoiceProcessedEventArgs(Guid instanceId)
        : base(instanceId)
    {
    }
}

[ExternalDataExchange]
[CorrelationParameter("invoiceNumber")]
public interface IInvoiceService
{
    [CorrelationInitializer]
    void CreateInvoice(
        string invoiceNumber, string customerNumber, decimal invoiceTotal);

    [CorrelationAlias("invoiceNumber", "e.InvoiceNumber")]
    event EventHandler<InvoiceProcessedEventArgs> InvoiceProcessed;
}
```

Code snippet IInvoiceService.cs

First take a look at the `InvoiceProcessedEventArgs` class; based on the description in step 1 there isn't anything too complicated here. However, notice that this class is decorated with a `Serializable` attribute. This class allows the `InvoiceProcessed` event to pass back an approval status along with the invoice number of the invoice being processed.

Now take a look at the `IInvoiceService` interface. Note that this interface is decorated with the `ExternalDataExchange` attribute. This simply flags the interface as being eligible to be a workflow service. Note also the series of attributes on the interface, method, and event relating to correlation tokens. At runtime, these attributes tell the workflow engine how to correlate incoming events to specific activities in specific workflow instances. In this example, the invoice number is the correlation token, which will tie everything together; it is essentially the service's "primary key" in database parlance.

The `CorrelationParameter` attribute on the interface specifies that there is a correlation parameter for this service and that it is called `invoiceNumber`. The `CorrelationInitializer` attribute on the `CreateInvoice` method indicates that this is the method on which the correlation parameter (`invoiceNumber`) is created. Notice that the method indeed has a property called `invoiceNumber`. If there were no parameter to this method with this name, things would not work. Finally, notice that the `InvoiceProcessed` event is decorated with the `CorrelationAlias` attribute. This attribute essentially says, "I am handling the correlation parameter, but I need to tell you how to find it." Thus, it points the engine to `e.InvoiceNumber` — `e` being a representative instance of the `InvoiceProcessedEventArgs` class.

The following code shows the implementation of the `IInvoiceService` class (step 3):



Available for
download on
Wrox.com

```
public class InvoiceService
    : SPWorkflowExternalDataExchangeService, IInvoiceService
{
    public void CreateInvoice(string invoiceNumber, string customerNumber,
        decimal invoiceTotal)
    {
        Console.WriteLine(String.Format("Creating {0}", invoiceNumber));

        SPWorkflowExternalDataExchangeService.RaiseEvent(
            CurrentWorkflow.ParentWeb,
            WorkflowEnvironment.WorkflowInstanceId,
            typeof(IInvoiceService),
            "InvoiceProcessed",
            new object[]
            {
                invoiceNumber,
                // if the total is greater than 100 automatically reject
                invoiceTotal > 100 ? false : true
            });
    }

    public event EventHandler<InvoiceProcessedEventArgs> InvoiceProcessed;

    public override void CallEventHandler(Type eventType, string eventName,
        object[] eventData, SPWorkflow workflow, string identity,
        IPendingWork workHandler, object workItem)
```

```

    {
        if (eventName == "InvoiceProcessed")
        {
            InvoiceProcessedEventArgs args =
                new InvoiceProcessedEventArgs(workflow.InstanceId)
            {
                InvoiceNumber = (string)eventData[0],
                Approved = (bool)eventData[1]
            };

            if (InvoiceProcessed != null)
                InvoiceProcessed(null, args);
        }
    }

    public override void CreateSubscription(
        MessageEventSubscription subscription)
    {
        // TODO: if this were real, track correlation info now!
    }

    public override void DeleteSubscription(Guid subscriptionId)
    {
        // TODO: if this were real, delete the correlation info now!
    }
}

```

Code snippet IInvoiceService.cs

Note a few important things here. First, the service class inherits from `SPWorkflowExternalDataExchangeService`, which is an abstract class. This class has three methods, which must be implemented: `CallEventHandler`, `CreateSubscription`, and `DeleteSubscription`. Also notice that this class is obviously responsible for giving `IInvoiceService` its implementation details (in the form of `CreateInvoice` and `InvoiceProcessed`).

In this example, the `CreateInvoice` method simply instantly “processes the invoice” and raises the `InvoiceProcessed` event. Notice the business rule that any invoice greater than \$100 is automatically rejected and anything is else approved. In a real implementation, this method would call out to the remote system and pass any relevant data needed.

`CreateInvoice` does not actually invoke the `InvoiceProcessed` event directly. Rather, it makes a call to the `SPWorkflowExternalDataExchangeService.RaiseEvent` method, which in turn calls `CallEventHandler`. This is because in a real scenario, the workflow service itself would not be raising the event; it would be some other code outside the context of the workflow.

The `CallEventHandler` is fairly straightforward. It simply checks the event name (which was passed to the `RaiseEvent` method) and then raises the event in question with a newly created `InvoiceProcessedEventArgs` instance with the data passed from the `RaiseEvent` method. That’s about it as far as implementing the service.

The only other point of discussion is the `CreateSubscription` and `DeleteSubscription` methods. These two methods are indispensable when actually interfacing with an external system. That's because the external system somehow needs to keep track of the correlation token data, as well as the workflow instance ID. These methods could write the necessary data to a database table or pass it to some externally available web service for persistence. The corresponding delete method is simply there to clean up the same correlation data.

Now that the workflow service is written, SharePoint needs to be aware of it once deployed. This is accomplished by adding a `web.config` entry. The following code shows what this looks like. As always, you should deploy things to the farm using the standard deployment techniques, rather than edit `web.config` directly.

```
<WorkflowServices>
  ... other services here ...

  <WorkflowService
    Assembly="... fully qualified assembly name ..."
    Class="namespace.class">
  </WorkflowService>

</WorkflowServices>
```

The service is now ready to be consumed. Figure 4-27 shows a sample workflow that consumes this new service.

The activity labeled `createInvoice` is a `CallExternalMethodActivity` instance and `handleInvoiceProcessed` is an instance of `HandleExternalEventActivity`. After each of these two activities is configured, the workflow can create invoices and handle the event when the invoice is processed. The workflow makes a decision based on whether or not the invoice was approved, and writes an appropriate message to the workflow's history list.

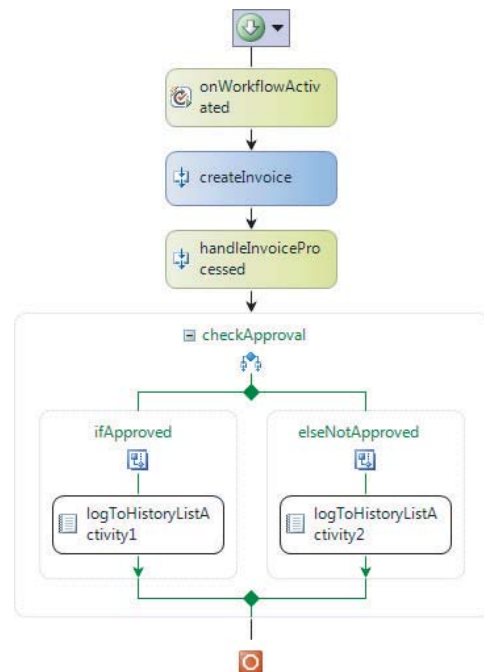


FIGURE 4-27

WORKFLOW EVENT RECEIVERS

Another way to extend workflow in SharePoint 2010 is with the brand-new workflow event receiver, which is implemented with the `SPWorkflowEventReceiver` class. Event receivers themselves are not new to SharePoint. In previous releases, developers were able to handle events such as when list items were being created, deleted, and so on. However, new in 2010 is the capability to handle workflow-related events.

The events exposed as overridable methods in the aforementioned class are as follows:

- `WorkflowCompleted`
- `WorkflowPostponed`

- WorkflowStarted
- WorkflowStarting

To create a workflow event receiver, simply use the Visual Studio project template Event Receiver and select the workflow events as shown in Figure 4-28.

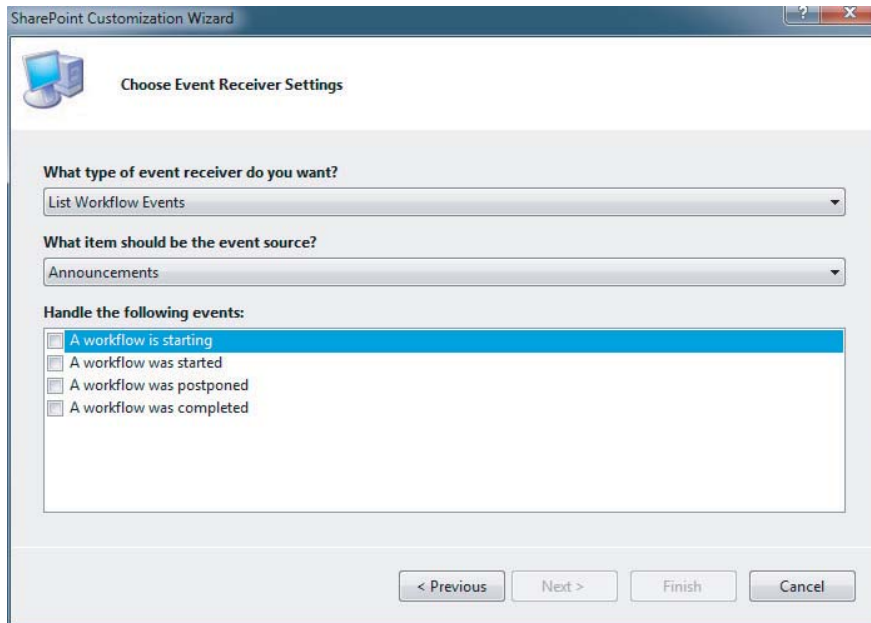


FIGURE 4-28

These events can be extremely useful for creating full solutions around several potential workflow definitions. For example, a workflow event receiver could be developed to capture the `WorkflowCompleted` event and then kick off an entirely different workflow. This would effectively chain two non-related workflow definitions to meet the solution's requirements. Being able to capture these events can also enable much more robust logging and reporting functionality as well.

SUMMARY

In the context of ECM, workflow is a very important topic in terms of managing pieces of content. From approval to content disposition, workflow is the driver of content through the system.

SharePoint workflow offers a continuum of options for workflow solutions, including a set of out-of-the-box workflows, Visio and SharePoint Designer for declarative workflows, and Visual Studio for custom-developed workflows. These various workflow types offer a range of capabilities for diverse target audiences, and accommodate the different levels of technical prowess of the workflow author.

SharePoint also offers numerous hooks to interact with the workflow system, including pluggable data services and workflow event receivers. These hooks allow a natural integration with external systems, as well as mechanisms to handle workflow-related events.

5

Collaboration

WHAT'S IN THIS CHAPTER?

- ▶ Understanding the role of collaboration in enterprise content management
- ▶ Exploring Microsoft SharePoint's powerful feature set for collaboration
- ▶ Administering and using Microsoft SharePoint's collaborative features
- ▶ Interacting with collaborative features through Microsoft SharePoint APIs

Collaboration is the act of working together to achieve a common goal. Business does not happen without collaboration. Whether it is collaboration between individuals, between individuals and an organization, or between organizations, business goals are achieved through the interaction of these entities. Organizations are well aware that collaboration is critical to their operations, and this has spurred these companies to employ collaborative software that promotes better structure, coordinated interactions, and improved management of the resulting output of collaborative efforts.

Throughout recent years, the world has seen the power of collaborative software at work. With online companies such as Facebook, MySpace, Friendster, Twitter, Yammer, Wikipedia, LinkedIn, Stack Exchange, YouTube, and countless others, the result of people collaborating together to create a world of interaction and knowledge is unprecedented. Organizations are attempting to harness the same types of tools that these online companies are leveraging to better manage and promote team collaboration both internally and externally. Through the use of weblogs, micro-blogs, social networking, wikis, and team sites, groups of individuals can collaborate effectively, and these collaborative efforts can be recorded and actively managed.

ECM AND COLLABORATION

The Association for Information and Image Management (AIIM) defines collaboration as “a practice whereby individuals work together to a common purpose to achieve business benefit.” According to AIIM, collaboration consists of eight conceptual stages that make up the collaborative life cycle:

- **Awareness** — We become part of a working entity with a shared purpose.
- **Motivation** — We drive to gain consensus in problem solving or development.
- **Self-synchronization** — We decide as individuals when things need to happen.
- **Participation** — We participate in collaboration and we expect others to participate.
- **Mediation** — We negotiate and we collaborate together and find a middle point.
- **Reciprocity** — We share and we expect sharing in return through reciprocity.
- **Reflection** — We think and we consider alternatives.
- **Engagement** — We proactively engage rather than wait and see.

Collaborative software facilitates the effective flow through the various stages of collaboration and enables collaboration to happen in an agile but orderly manner. Collaborative software also enables organizations to track the efforts and capture the output of collaboration, which can be especially useful when organizations are required to keep communications and recorded interactions as business records.

SharePoint Is Collaboration

SharePoint has always been centered on collaboration. In fact, SharePoint was originally created to promote effective collaboration throughout the enterprise, and it was first released at a time when corporate portals were gaining popularity among organizations. As collaborative software concepts have evolved and social networking has entered into the picture, SharePoint has similarly evolved in terms of the collaborative features that it brings to the table.

With each release of SharePoint, new collaborative features have been integrated into the platform and the existing features continue to be enhanced. SharePoint 2010 provides numerous collaborative features, such as social tagging, user profiles with social networking features, blogs, wikis, and the capability to integrate with SharePoint from collaborative desktop applications such as Microsoft Outlook and Microsoft SharePoint Workspace.

SOCIAL TAGGING

Social tagging refers to the capability that enables users to dynamically categorize information through the use of specialized attributes with the purpose of enhancing the representation and discoverability of the information in a way that is both meaningful and useful to the users themselves.

Using SharePoint Server, users can participate in social tagging through the use of multiple types of attributes, such as tags, ratings, and notes. SharePoint also provides the concept of *bookmarklets*, controls that can be used to enable web browsers with the functionality to participate in SharePoint social tagging outside of the SharePoint environment.

Through the use of social tagging, collaboration between users is greatly enhanced, as users are sharing information about content in a very organic way. As content is assigned various social tagging attributes, the content immediately becomes more meaningful because of these interpretive descriptors. These descriptors aid not only in the discoverability of content, but also in the visibility of high-quality content or content pertinent to a particular set of users.

Tags

Tags are words or phrases that are provided by users in an informal manner to describe existing content. The term given to the classification of items through tags is called *folksonomy*. Folksonomy differs from taxonomy in the sense that it is accomplished collaboratively and informally, whereas taxonomies are predefined and their usage is much more constrained, as they are typically of a more fixed nature. Folksonomies are particularly useful because they are created by the users themselves, which enables their knowledge, expertise, and usage to be leveraged to enhance the classification of items and to evolve over time.

SharePoint enables users to create tags for pages, documents, list items, and external websites — all of which aids users in remembering, locating, and recalling content that is pertinent and meaningful to them personally. As tags are created, they are surfaced within the user's My Site, which is discussed in depth later in this chapter.

How to Create Tags

Creating tags within SharePoint is accomplished through the I Like It and the Tags and Notes buttons located on the SharePoint Ribbon. Depending on whether you are tagging a list item, a document, or a page, the buttons appear in their applicable tab. The I Like It button is special button that enables you to tag an item quickly with one click and will create a tag with the phrase “I Like It.” The Tags and Notes button opens a dialog that enables users to provide individualized tags with words or phrases of their choosing. As items are tagged, previously used tags are automatically suggested in order to promote a consistent and speedier tagging experience.

Tagging a Page

In order to create a tag for a page within SharePoint, simply perform the following actions:

1. Navigate to the page that you want to tag.
2. Within the Browse Tab of the Ribbon, click Tags & Notes.
3. The Tags dialog will open for the current page, as shown in Figure 5-1.
4. Enter any number of tags, separated by a semicolon, or choose from Suggested Tags.
5. Click Save.

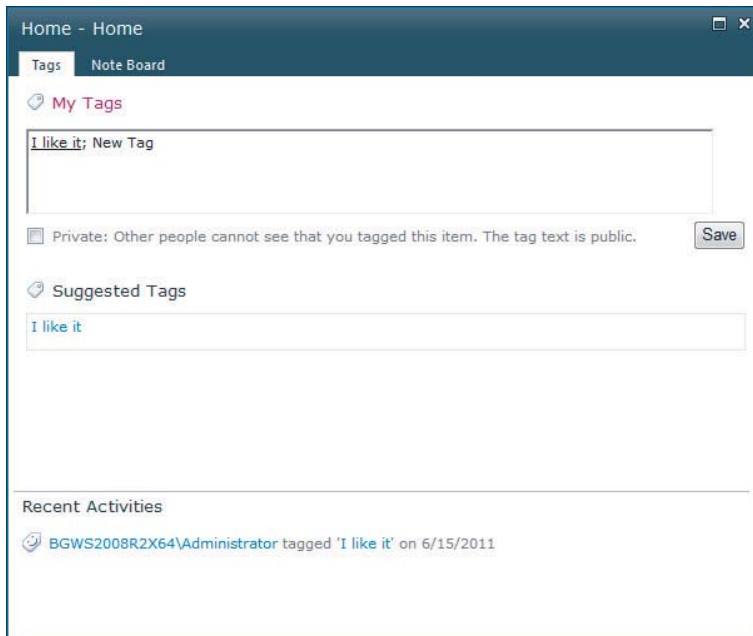


FIGURE 5-1

Tagging a Document or List Item

Tagging a document or list item in SharePoint is very similar to tagging a page. In order to tag a document or list item within SharePoint, simply perform the following actions:

1. Navigate to the document library or list for which you want to tag items.
2. Click the Documents or Items tab depending on whether you are in a document library or a list, respectively.
3. Select the item that you want to tag.
4. Within the Browse tab of the Ribbon, click Tags & Notes.
5. The Tags dialog will open for the current item, as shown in Figure 5-1.
6. Enter any number of tags, separated by a semicolon, or choose from Suggested Tags.
7. Click Save.



When tagging items, it is possible to select multiple items within a document library or list. If you select multiple items, the Tags and Notes button will be disabled. However, the I Like It button remains enabled so that you can tag all selected items with a single click.

Tag Cloud

A *tag cloud* provides a visualization of tags that have been created by the current user, specific groups, or all users. Tag clouds enable users to see which tags are used with the most frequency, and they can be filtered by date and language. By default, the tag cloud is included on a user's My Site, but it can also be added to any page by using the Tag Cloud Web Part, as shown in Figure 5-2.

The screenshot shows a SharePoint page with a navigation bar at the top. Below the navigation bar is a search box labeled "Search this site...". The main content area features a "Welcome to your site!" message and a list of documents. A "Tag Cloud" web part is highlighted with a red box, displaying the following tags and counts:

- Another Tag (I like it, My First Tag, My)
- Second Tag (My Tag, One More Tag)

The document list below the Tag Cloud web part is as follows:

| Name | Modified | Modified By |
|---|--------------------|-----------------------------|
| 6251 Invoice | 6/10/2011 9:59 AM | BGWS2008R2X64\administrator |
| 6251 Packing_Slip | 5/10/2011 10:49 AM | BGWS2008R2X64\administrator |
| 6251 PO | 5/10/2011 11:34 AM | BGWS2008R2X64\administrator |
| AES128 | 5/4/2011 2:43 PM | BGWS2008R2X64\administrator |
| CAPITAL | 5/10/2011 11:34 AM | BGWS2008R2X64\administrator |
| Content_modifications_2011-06-10T095933 | 6/10/2011 9:59 AM | BGWS2008R2X64\administrator |
| Content_modifications_2011-06-10T100105 | 6/10/2011 10:01 AM | BGWS2008R2X64\administrator |
| Content_modifications_2011-06-10T100148 | 6/10/2011 10:01 AM | BGWS2008R2X64\administrator |
| Copy of 6251 PO | 5/10/2011 11:28 AM | BGWS2008R2X64\administrator |
| Image Only PDF | 5/4/2011 2:43 PM | BGWS2008R2X64\administrator |
| NonAES128 | 5/4/2011 2:43 PM | BGWS2008R2X64\administrator |
| Searchable PDF (Image on T... | 5/4/2011 2:43 PM | BGWS2008R2X64\administrator |

FIGURE 5-2

Notes

Notes within SharePoint enable users to express comments regarding pages, documents, list items, or external sites. The creation of notes is accomplished using the Note Board, which is a term given to one of the tabs within the Tags and Notes dialog. This tab provides the ability to enter any comments that a user may want regarding a piece of content. Users can also leverage the Note Board to create comments within their own My Site page or the My Site page of another user. Notes are useful for relaying information to users without the need for using e-mail, instant messaging, or another form of communication. By leveraging notes, others users can see your comments, and it enables the tracking of these comments in an organized and methodical fashion.

How to Create Notes

Creating notes within SharePoint is accomplished through the Tags and Notes button located on the SharePoint Ribbon. Depending on whether you are tagging a list item, a document, or a page, the buttons will appear in their applicable tab. Clicking the Tags and Notes button will open a dialog giving access to the Note Board that enables the user to provide comments pertaining to the selected item. After notes are created, they will be surfaced within the user's My Site, which is discussed in detail later in this chapter. Figure 5-3 shows the Note Board tab of the Tags and Notes feature.

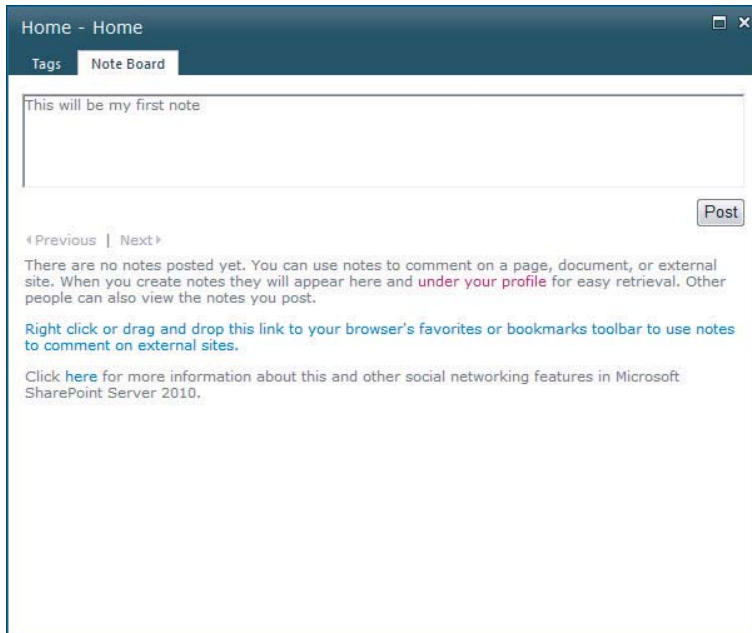


FIGURE 5-3

Commenting on a Page

In order to create a note for a page within SharePoint, simply perform the following actions:

1. Navigate to the page that you want to comment on.
2. Within the Browse tab of the Ribbon, click Tags & Notes.
3. A dialog will open for the current page (refer to Figure 5-3).
4. Select the Note Board tab.
5. Enter the comment that you would like to post.
6. Click Post.

Commenting on a Document or List Item

Commenting on a document or list item in SharePoint is very similar to commenting a page. In order to comment on a document or list item within SharePoint, simply perform the following actions:

1. Navigate to the document library or list for which you want to provide a comment.
2. Click the Documents or Items tab depending on whether you are in a document library or a list, respectively.
3. Select the item that you want to comment on.
4. A dialog will open for the current item (refer to Figure 5-3).
5. Select the Note Board tab.
6. Enter the comment that you would like to post.
7. Click Post.

Ratings

Ratings within SharePoint enable users to rank items as a means of assessing how well the content meets the need for which it is intended. By enabling ratings for a document library or list, two special field types are added to the lists; these enable users to rate individual items, as shown in Figure 5-4.

The first field added to the list is the Rating field, which is rendered as a rating control that enables users to rate items on a scale of 0–5. The second field added to the list is the Number of Ratings field, which tracks the number of ratings recorded and is incremented per each unique user’s rating for an item. When a user provides a rating for an item through the rating control, the value is recorded asynchronously and stored in the social tagging database. Once a rating is recorded, it is not immediately visible to the user. The User Profile Service - Social Rating Synchronization SharePoint timer job, which runs on an administrator-configurable interval, computes the average rating among the ratings recorded to display the actual item rating. The Number of Ratings field displays the total number of ratings recorded, which accounts for the overall average.

| Type | Name | Modified | Modified By | Rating (0-5) |
|------|-------------------|--------------------|-----------------------------|--------------|
| | 6251 Invoice | 6/10/2011 9:59 AM | BGWS2008R2X64\Administrator | |
| | 6251 Packing_Slip | 6/14/2011 1:32 PM | BGWS2008R2X64\Administrator | |
| | 6251 PO | 5/10/2011 11:34 AM | BGWS2008R2X64\Administrator | |

FIGURE 5-4

Enabling Ratings for a Document Library or List

In order to enable ratings for a document library or list, the following actions should be performed:

1. For a list, navigate to List Settings page.
2. Select Rating Settings under the General Settings section (see Figure 5-5).
3. Select the Yes radio button to Allow Items in this List to be Rated.
4. Click OK.

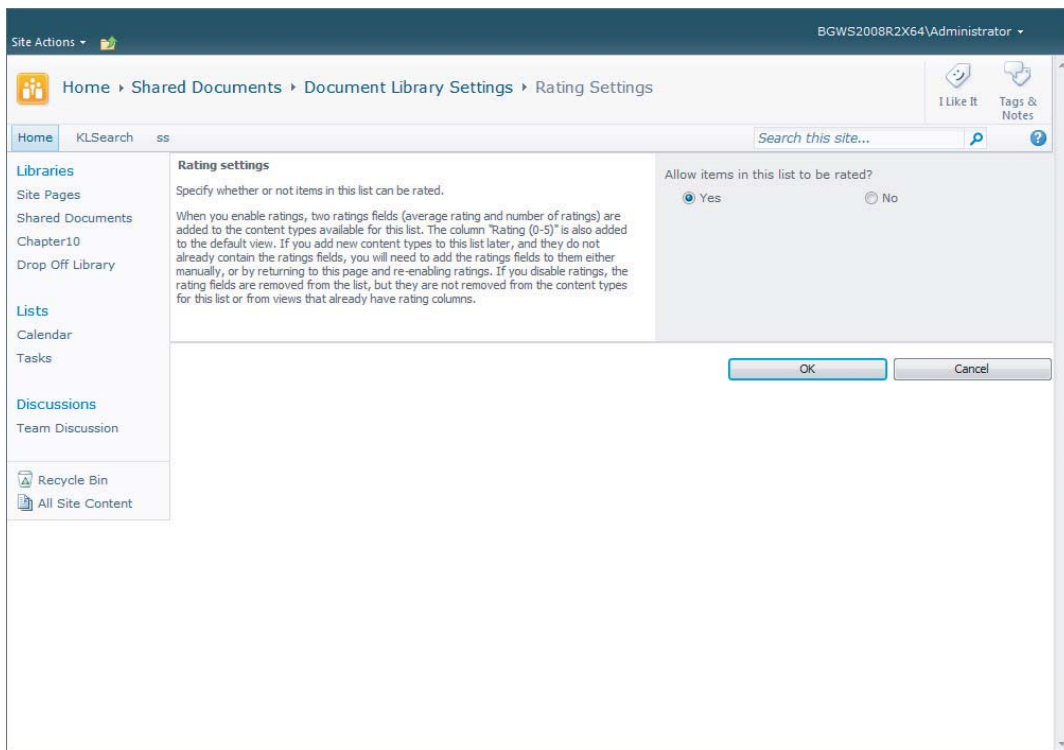


FIGURE 5-5

After enabling ratings for a document library or list, the fields displayed in Figure 5-6 will be available for use.

How to Rate an Item

You can rate an item by navigating to a view that has these columns enabled and then clicking the desired rating for the item in the list. After rating an item, a tooltip will be displayed notifying the user that the input has been accepted and will be available after the timer job has executed, as shown in Figure 5-7.

Columns

A column stores information about each document in the document library. Because this document library column settings, such as whether information is required or optional for a column, are now specified by following columns are currently available in this document library:

| Column (click to edit) | Type | Used in |
|-----------------------------------|------------------------|----------|
| Address | Multiple lines of text | Document |
| Number of Ratings | Number of Ratings | Document |
| Rating (0-5) | Rating (0-5) | Document |
| Title | Single line of text | Document |
| Created By | Person or Group | |
| Modified By | Person or Group | |
| Checked Out To | Person or Group | |

[Create column](#)
[Add from existing site columns](#)
[Indexed columns](#)

FIGURE 5-6

The screenshot shows a SharePoint document library interface. The top ribbon includes 'Library Tools' with tabs for 'Documents' and 'Library'. The 'Library' tab is active, showing a list of documents. The list has columns for 'Type', 'Name', 'Modified', 'Modified By', and 'Rating (0-5)'. The 'Rating (0-5)' column shows a star rating for each document. A notification box is visible on the right side of the list, stating 'Thank you. Your rating has been submitted and will be processed soon.'

| Type | Name | Modified | Modified By | Rating (0-5) |
|------|-------------------|--------------------|-----------------------------|--------------|
| | 6251 Invoice | 6/10/2011 9:59 AM | BGWS2008R2X64\Administrator | ☆☆☆☆☆ |
| | 6251 Packing_Slip | 6/14/2011 1:32 PM | BGWS2008R2X64\Administrator | ☆☆☆☆☆ |
| | 6251 PO | 5/10/2011 11:34 AM | BGWS2008R2X64\Administrator | ☆☆☆☆☆ |

FIGURE 5-7

Bookmarklets

Bookmarklets are snippets of JavaScript that can be saved as a bookmark within most web browsers. When saving a bookmarklet as a bookmark, the URL portion of the bookmark contains the

JavaScript snippet, rather than an actual URL. Selecting a bookmark within the browser causes the snippet of JavaScript to execute within the context of the currently viewed web page, which will ultimately carry out the specified action.

SharePoint leverages bookmarklets to support the tagging of web pages that are external to the SharePoint environment. By registering the bookmarklet as a bookmark, it is possible to navigate to any web page within the browser, then click the bookmarklet, which will launch the SharePoint Tags and Notes dialog. Through this dialog, it is possible to create both tags and notes that are associated with the current web page, just as you would with pages within SharePoint using the Tags & Notes button on the SharePoint Ribbon Menu.

Registering the Tags and Notes Bookmarklet

Registering the Tags and Notes bookmarklet is accomplished through the following actions:

1. Navigate to your My Site page or to the Tags and Notes dialog if you have not yet created any tags.
2. Right-click the hyperlink that contains the text “Right click or drag and drop this link to your browser’s favorites or bookmarks toolbar to tag external sites” and choose Add to Favorites, as shown in Figure 5-8.
3. Click Add.

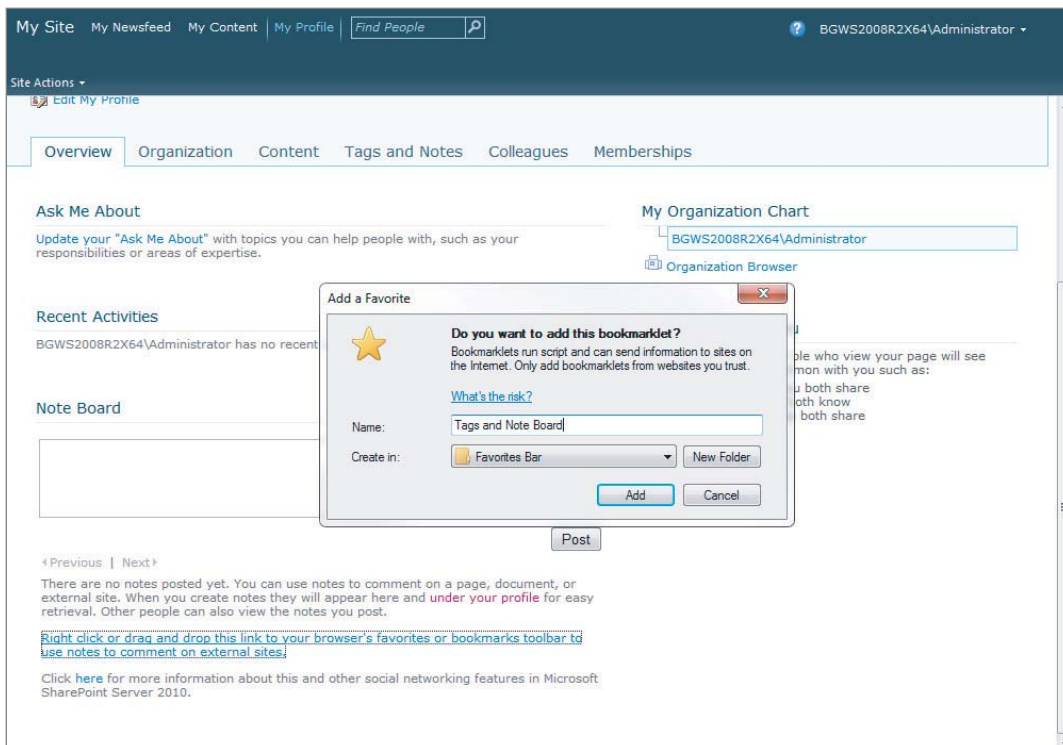


FIGURE 5-8

Creating Tags and Notes Using Bookmarklets

Once the bookmarklet is added, simply navigate to any web page and click the saved bookmark. Clicking this bookmark will open the Tags and Notes dialog for the current web page, as shown in Figure 5-9.

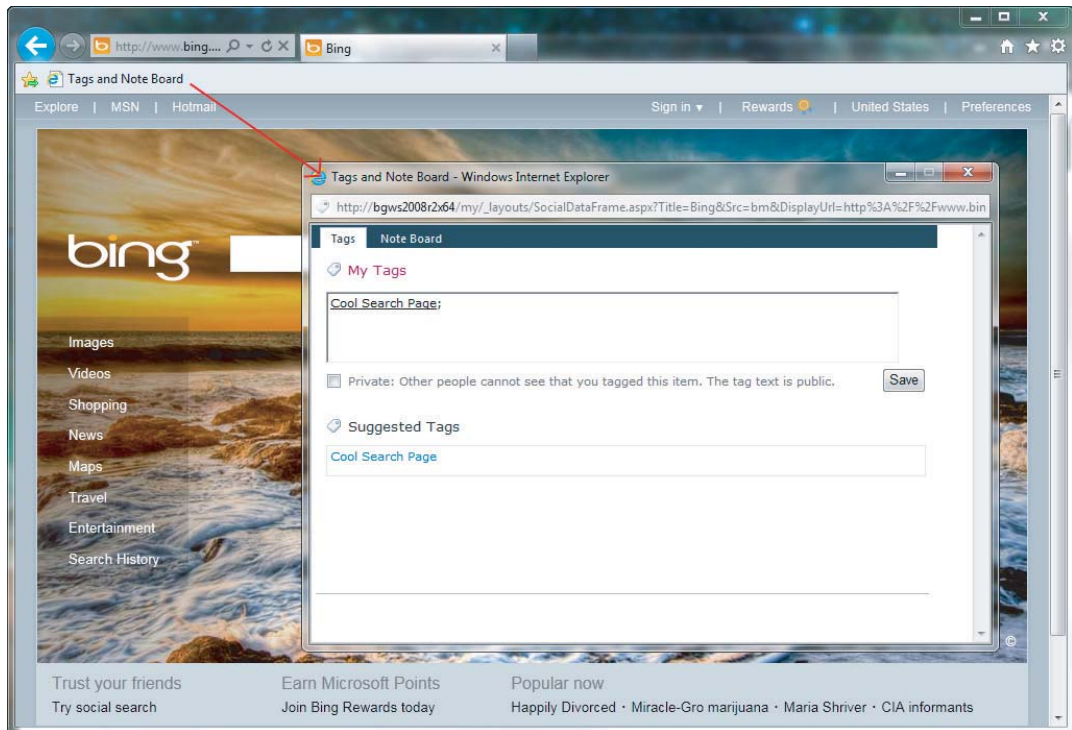


FIGURE 5-9

Privacy and Security Concerns

When using SharePoint's social tagging capabilities, administrators and users must understand the potential issues regarding privacy and security. It is important to be aware that as tags, notes, and ratings are created, SharePoint creates a record referred to as an *activity*. These activities not only appear in the user's personal My Site, they are also broadcast to any user who has registered the tagging user as a colleague on their My Site or who has flagged the specific tag as an interest in their user profile. In order to prevent users from seeing specific tags, SharePoint provides the capability to mark tags as private. Notes, however, cannot be marked as private. It is important to educate users about which social data is public and which data is private in order to prevent users from unintentionally exposing data that should remain private.

It is equally as important to understand that SharePoint security is strictly enforced throughout the social tagging infrastructure. If users do not have permissions to the items associated with an activity, the activities will not be broadcast to these users. This is because social activities participate in SharePoint security trimming.



Although social data is security trimmed, security trimming is based on the SharePoint Search Service. This means that when data is crawled, the permissions for each item are recorded and subsequently used by the security trimmer for filtering data. Therefore, security information will not be up to date until the SharePoint Search Service has crawled the data.

Tagging Programming Model

SharePoint provides for the tagging of data programmatically through the SharePoint object model. When creating tags, each tag is backed by a term in the term store. These terms must first be created in the default keywords term store and term set. SharePoint uses this term set to provide suggestions to users when they are creating social tags. The common classes involved in programming tagging data, described in Table 5-1, can be found in the `Microsoft.Office.Server.SocialData` namespace.

TABLE 5-1: Common Classes Used in the Tagging Programming Model

| CLASS | DESCRIPTION |
|-----------------------------------|---|
| <code>DeletedSocialComment</code> | Represents a comment that was deleted from the social database |
| <code>DeletedSocialData</code> | Abstract base class for classes representing data that was deleted from the social database |
| <code>DeletedSocialRating</code> | Represents a rating that was deleted from the social database |
| <code>DeletedSocialTag</code> | Represents a tag that was deleted from the social database |
| <code>SocialComment</code> | Represents a comment |
| <code>SocialCommentManager</code> | Provides the methods used to manage a <code>SocialComment</code> |
| <code>SocialData</code> | Abstract base class used by other classes that represent social data |
| <code>SocialDataManager</code> | Abstract base class for other classes that manage social data |
| <code>SocialRating</code> | Represents a rating |
| <code>SocialRatingAverage</code> | Represents the average of all ratings created for a specific URL |
| <code>SocialRatingManager</code> | Provides the methods used to manage a <code>SocialRating</code> |
| <code>SocialTag</code> | Represents a tag |
| <code>SocialTagManager</code> | Provides the methods used to manage a <code>SocialTag</code> |
| <code>SocialTerm</code> | Represents a term in the term store that is available for creating a <code>SocialTag</code> |
| <code>SocialUrl</code> | Represents a URL that has been associated with social data |

Working with Tags Programmatically

Using the SharePoint object model, it is possible to programmatically manipulate tags within SharePoint. The following examples demonstrate this capability.



Normally, the code listings in this chapter would be running within the web context of the SharePoint server, in which case the application pool account is given full control permissions to make calls to the User Profile Service Application. Because these sample listings are written within a command-line application that must run on the SharePoint Server, they will run in the context of the current user. Even if this user is a farm administrator, this user will not have access to make calls to the Service Application. In order for these listings to work, you must navigate to the Manage Service Applications page in Central Administration, select the User Profile Service Application, and within the permissions option in the Ribbon menu, grant the appropriate user Full Control permissions.

Programmatically Creating a Tag

Listing 5-1 demonstrates the creation of a tag programmatically. In this example, the first action that takes place is getting a reference to a `SPSite` object. Once this object is obtained, the `SPServiceContext` for this site is returned, which allows you to gain access to the service applications for the site. This context is passed into all calls to return the various manager objects. The next step is to get access to the `SocialTagManager`, which is used to create the actual tag. As each tag created is backed by a `Term` in the term store, a term is created that will be used within the tag. Once the `Term` is created, it is used by the manager object to create a `SocialTag` that references the indicated URL.



LISTING 5-1: Creating a Tag

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.SocialData;
using Microsoft.SharePoint.Taxonomy;

namespace Listing0501
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008r2x64"))
            {
```

continues


```

    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008r2x64"))
            {
                //Get access to the service application
                //context for the current SPSite
                SPServiceContext context = SPServiceContext.GetContext(site);

                //Get a reference to the social tags for the
                //current user for a specific URL
                SocialTagManager socialTagManager = new SocialTagManager(context);
                SocialTag[] tags =
                    socialTagManager.GetTags(new Uri("http://www.bing.com"));

                //Delete tags
                foreach (SocialTag tag in tags)
                {
                    socialTagManager.DeleteTag(tag.Url, tag.Term);
                }
            }
        }
    }
}

```

Working with Notes Programmatically

Using the SharePoint object model, it is possible to programmatically manipulate notes within SharePoint. The following examples demonstrate this capability.

Programmatically Creating a Note

Listing 5-3 demonstrates the creation of a note for the current user. In this example, the first action that takes place is getting a reference to a `SPSite` object. Using the `SPServiceContext` for the current `SPSite` object, a reference to the `SocialCommentManager` object is obtained. Using the manager object, the `AddComment` method is called to create a note containing the text specified along with a reference to the URL indicated.



LISTING 5-3: Creating a Note

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.SocialData;

namespace Listing0503
{
    class Program
    {

```

continues

LISTING 5-3 (continued)

```

static void Main(string[] args)
{
    using (SPSite site = new SPSite("http://bgws2008r2x64"))
    {
        //Get access to the service application
        //context for the current SPSite
        SPServiceContext context = SPServiceContext.GetContext(site);

        //Get a reference to the social tags manager
        //for the current service context
        SocialCommentManager socialCommentManager =
            new SocialCommentManager(context);

        //Create comment for associated URL
        socialCommentManager.AddComment(new
            Uri("http://www.bing.com"),
            "This is a great search site!");
    }
}

```

Programmatically Retrieving and Deleting a Note

Listing 5-4 demonstrates the retrieval and deletion of an existing note for the current user. In this example, the first action gets a reference to a `SPSite` object. Using the `SPServiceContext` for the current `SPSite` object, a reference to the `SocialCommentManager` object is obtained. Using the manager object, an array of `SocialComment` objects is returned by calling the `GetComments` method and passing in the specified URL. You then iterate through the array calling the `Delete` method of each comment to execute the deletion of the note.

**LISTING 5-4: Retrieving and Deleting a Note**

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.SocialData;
using Microsoft.Office.Server.UserProfiles;

namespace Listing0504
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008r2x64"))

```


LISTING 5-5 (continued)

```

    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008r2x64"))
            {
                //Get access to the service application
                //context for the current SPSite
                SPServiceContext context = SPServiceContext.GetContext(site);

                //Get social rating manager for current service context
                SocialRatingManager ratingManager =
                    new SocialRatingManager(context);

                //Set rating of 5 stars for document within a Document Library
                ratingManager.SetRating(new Uri("http://bgws2008r2x64/
                    Shared%20Documents/6251%20PO.tif"), 5);
            }
        }
    }
}

```

Programmatically Retrieving and Deleting a Rating

Listing 5-6 demonstrates the retrieval and deletion of an existing rating for the current user. In this example, the first action that takes place is getting a reference to a `SPSite` object. Using the `SPServiceContext` for the current `SPSite` object, a reference to the `SocialRatingManager` object is obtained. Using the manager object, a `SocialRating` object is returned for a specific document by calling `GetRating` and passing in the appropriate URL. The rating is then deleted by calling the `Delete` method of the `SocialRating` object.

**LISTING 5-6: Retrieving and Deleting a Rating**

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.SocialData;
using Microsoft.Office.Server.UserProfiles;

namespace Listing0506
{
    class Program
    {

```

```

static void Main(string[] args)
{
    using (SPSite site = new SPSite("http://bgws2008r2x64"))
    {
        //Get access to the service application
        //context for the current SPSite
        SPServiceContext context = SPServiceContext.GetContext(site);

        //Get social rating for the current user and URL specified
        SocialRatingManager ratingManager =
            new SocialRatingManager(context);
        SocialRating rating = ratingManager.GetRating(
            new Uri("http://bgws2008r2x64/
                Shared%20Documents/6251%20PO.tif"));

        //Delete rating
        rating.Delete();
    }
}
}
}

```

MY SITES

SharePoint Server provides a specialized web application infrastructure for hosting individual sites that are specific to an individual user. When the My Sites functionality is enabled, users are provided with the option to navigate to their personal My Site. Upon navigating to the site for the first time, the site is provisioned with a number of capabilities for managing personal content, such as a profile page for managing information about yourself within the organization as well as your social tagging activity and a personal content store for creating your own document libraries and lists.

An individual My Site consists of three major sections that are navigable from a menu at the top of each My Site page. The three sections are My Profile, My Content, and My Newsfeed. Content within these sections is managed by the user with whom the My Site is associated.

My Profile

This section of a user's personal My Site contains a multitude of information about the current user. The information is displayed with a common header that can include an avatar representing the current user as well as any user-provided details. The header information displayed is governed by configuration settings within the User Profile Service Application, which is discussed later in this chapter. This common header can also include a status message from the user indicating what they are doing at the moment. This is analogous to a status message in an instant messenger client or similar application. The My Profile section contains several subsections represented by various tabs, as shown in Figure 5-10 and described in Table 5-2.

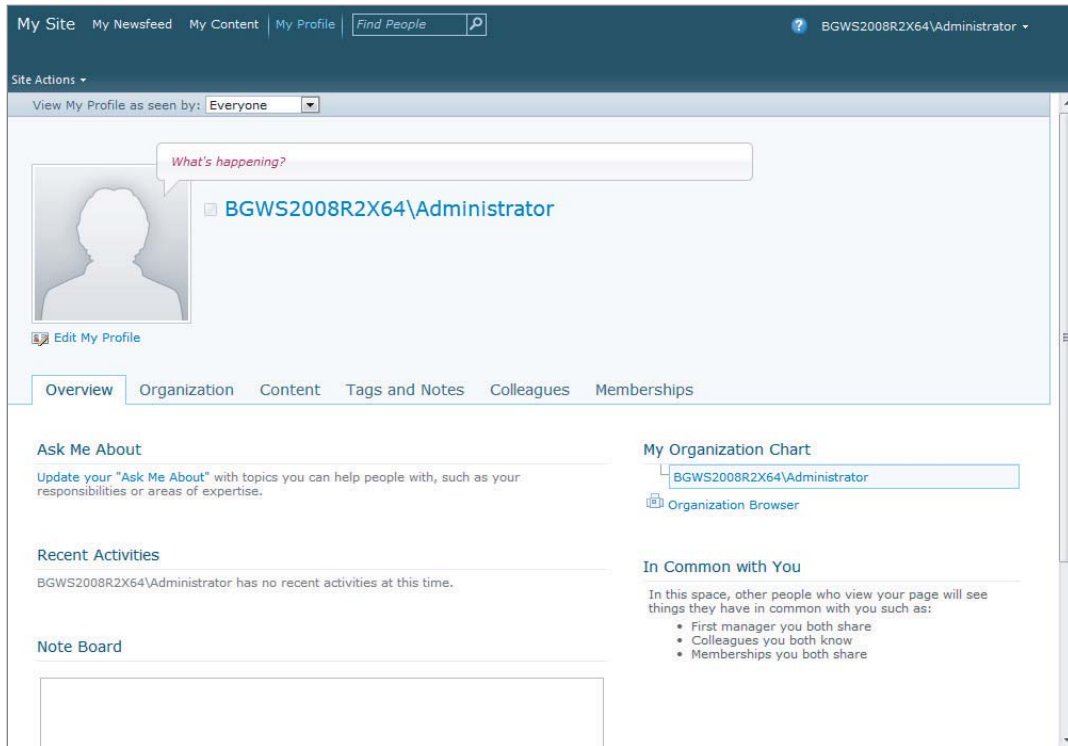


FIGURE 5-10

TABLE 5-2: My Profile Tabs

| TAB | DESCRIPTION |
|----------------|---|
| Overview | Displays a summary of information about the user as well as recent activities. It also provides access to the user's Note Board for posting and viewing comments for the current user. |
| Organization | Displays an organizational chart in a hierarchical fashion and how the current user fits into that chart. |
| Content | Displays content that has been shared by the current user such as documents, lists, or blog entries. |
| Tags and Notes | Displays the tags and notes created by the current user. From this tab, users can view/filter the tags and notes, manage the existing tags, and view the current tag cloud. |
| Colleagues | Displays the users whom the current user has added to his or her colleague list. The social activities performed by the users in this list are surfaced with the current user's newsfeed. |
| Memberships | Displays the groups that the current user is a member of. |

My Content

This section of the My Site is a Web Part page that acts like the user's personal team site. Users use this page to create their own document libraries and lists for storing personal content. By default, when a My Site is created, three lists are created: Shared Pictures, Shared Documents, and Personal Documents. Lists with the Shared prefix are visible to other users viewing the My Site, whereas the Personal Documents content is available only to the user associated with the My Site. Users can create any number of new lists within this site and can even add Web Parts to this site.

It is also possible for a user to create a personal blog from the My Content section. When a personal blog is created, a subsite is created that uses the Blog Site template, which is discussed later in this chapter. When blog entries are created here, the entries are surfaced in the activity feed for the current user.

An example of the My Content section is shown in Figure 5-11.

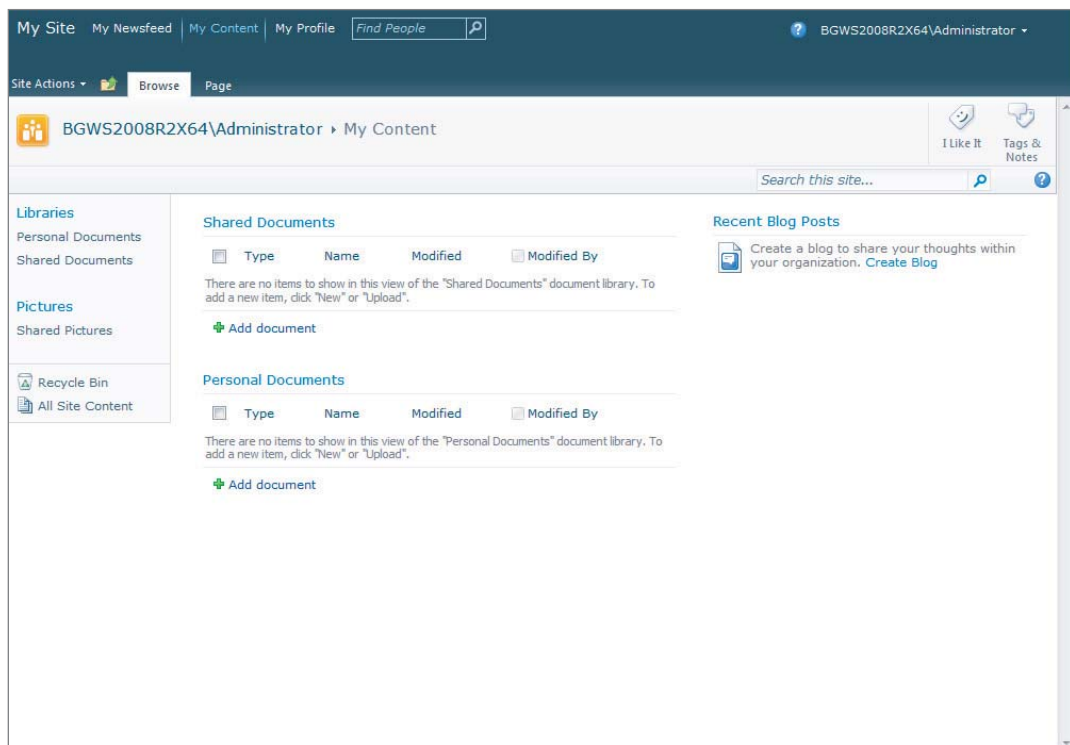


FIGURE 5-11

My Newsfeed

This section of the My Site displays a list of the social activities that have taken place within SharePoint for the user's specified colleagues and configured interests. When users first navigate to their My Site, this is the default page that is displayed. As users tag items, make comments, or share information, records of these activities are presented within this page. An example is shown in Figure 5-12.

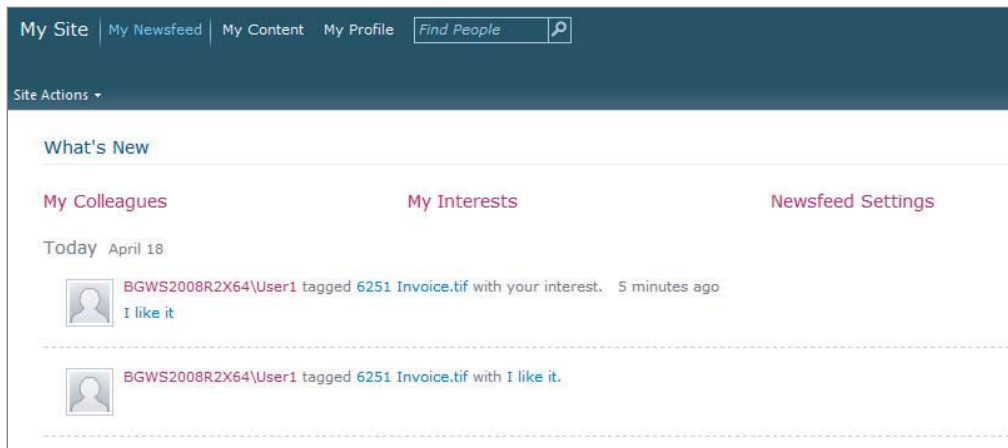


FIGURE 5-12

My Sites Architecture

The My Sites infrastructure is composed of a Web Application that is used to host the My Site Host site collection. SharePoint allows this Web Application to be any existing Web Application, but recommends that you create a dedicated Web Application for better backup, restore, security, and overall manageability.

Within the Web Application specified to host the My Site infrastructure, a special site collection is created using the My Site Host site template. This site collection is shared by all users and contains the My Newsfeed and My Profile portions of a user's My Site.

When users navigate to their My Site, another site collection specific to the current user is provisioned below the My Site Host site collection. This site collection hosts the My Content portion of a user's My Site.

Table 5-3 outlines the various site collections and URLs that comprise the My Site infrastructure.

Configuring My Sites

The My Site infrastructure relies on the User Profile Service Application discussed later in this chapter. Much of the My Site infrastructure also leverages the Managed Metadata Service Application and the Search Service Application. These are not required, but they are highly recommended for fully enabling My Site functionality. The following section walks you through the process of configuring the My Site infrastructure.

TABLE 5-3: My Site Infrastructure

| URL | DESCRIPTION |
|--|--|
| <code>http://hostname/default.aspx</code> | Represents the default location for the My Newsfeed portion of the My Site and is the location that users are presented with when they first navigate to their My Site. The information in this page is specific to the current user but is contained in the My Site Host site collection. |
| <code>http://hostname/person.aspx</code> | Represents the default location for the My Profile portion of the My Site and is the location that users are presented with when they view their personal profile. When users navigate to another user's profile page, they are brought to the same page but the URL is <code>http://hostname/person.aspx?accountname=account</code> . This page is contained in the My Site Host site collection. |
| <code>http://hostname/personal/account/default.aspx</code> | Represents the default location for the My Content portion of the My Site. The pages in this location are hosted within a site collection underneath the My Site Host site collection and the content herein applies to each individual user. |



In order to configure the My Site infrastructure, you must be a member of the Farm Administrators group on the computer running Central Administration.

Configuring the My Site Host Site Collection

In order to configure the My Site Host site collection, perform the following actions:

- 1.** Within Central Administration, under the Application Management section, click Create Site Collections.
- 2.** Select the Web Application that will be used to host the My Site Host Site Collection. It is recommended that a new Web Application be used.
- 3.** Enter a Title and Description for the site collection.
- 4.** Enter the Web Site Address that will be used to host the site collection. This is simply specifying the URL that will be used to access the site collection.
- 5.** Select the My Site Host template from the Enterprise tab.

6. Enter the Site Collection Administrators for the site collection.
7. Click OK to create the My Site Host site collection.
8. Within Central Administration, under the Application Management section, click Manage Web Applications.
9. Select the Web Application that is hosting the My Site Host site collection. Then, within the Ribbon menu, click Managed Paths.
10. Add a new path that will be used to host personal My Site locations and ensure that the type of Wildcard Inclusion is selected. The path "my/personal" is recommended. Adding this managed path will allow sites to be created using the URL specified as a prefix when My Sites are dynamically generated.
11. Click Add Path and then click OK.
12. Ensure that the Web Application is still selected within the Manage Web Applications screen and click Self-Service Site Create in the Ribbon menu.
13. Select the On option to enable Self-Service Site Creation.
14. Click OK.

Configuring My Site Settings in the User Profile Service Application

In order for the My Site infrastructure to function, the User Profile Service Application must be configured to point to the My Site Host site collection. The following actions outline this procedure:

1. Within Central Administration, under the Application Management section, click Manage Service Applications.
2. Select the User Profile Service Application. Then, within the Ribbon menu, click Manage.
3. Under the My Site Settings section, click Setup My Sites.
4. Enter a URL in the Preferred Search Center section, which will be used when users search for people or documents from the My Profile page. This URL should point to the location where a Search Center Site has been created.
5. Enter a URL in the My Site Host section that points to the location that was created for the My Site Host site collection. This is the URL that was provided in step 4 of the "Configuring the My Site Host Site Collection" section earlier in this chapter.
6. In the Personal Site Location section, select the managed path that was specified for the Wildcard Inclusion when configuring the My Site Host site collection. It was recommended that "my/personal" be used.
7. Select the Site Naming Format, which can be configured to avoid naming conflicts when personal My Sites are provisioned.

8. Select whether to allow users to use their own language settings.
9. Enter the users or groups who will have read access to personal My Sites when they are created. By default, all authenticated users are specified.
10. Enter an e-mail address to be used as the sender for My Site e-mail notifications. This does not need to be an actual working e-mail address.
11. Click OK.

Enabling the Activity Feed Timer Job

Once the My Site infrastructure has been configured, the Activity Feed Timer Job must be enabled in order for the My Newsfeed section to be updated within each user's My Site. The following procedure outlines the actions involved in this configuration:

1. Within SharePoint Central Administration, click Monitoring to navigate to the Monitoring section.
2. Within Monitoring, click Review Job Definitions.
3. Click User Profile Service - Activity Feed Job to edit the timer job settings.
4. Under the Recurring Schedule section, enter the interval specifying how often the timer job should execute. This job should run as often as possible so that user newsfeeds are updated, but be aware that this can place a large load on a system with many users and updates. The proper interval needs to be determined based on usage.
5. Click Enable.

USER PROFILES

The personal data that is displayed within the My Profile section of a user's My Site is referred to as a user profile. User profiles consist of a collection of properties that describe an individual user. SharePoint has a default set of common properties for each user profile, each with a default policy specifying how they can be used within the system. However, SharePoint provides the capability to create, delete, or modify existing properties to meet your organization's need. SharePoint even provides the capability to import this profile data from disparate systems such as Active Directory or a line-of-business application that may contain user data.

When managing the properties that comprise a user profile, it is possible to specify profile subtypes that allow different sets of properties to be applied to different users in cases where properties are not applicable to all users.

Figure 5-13 shows an example of editing a user profile from the My Profile section of a user's My Site.

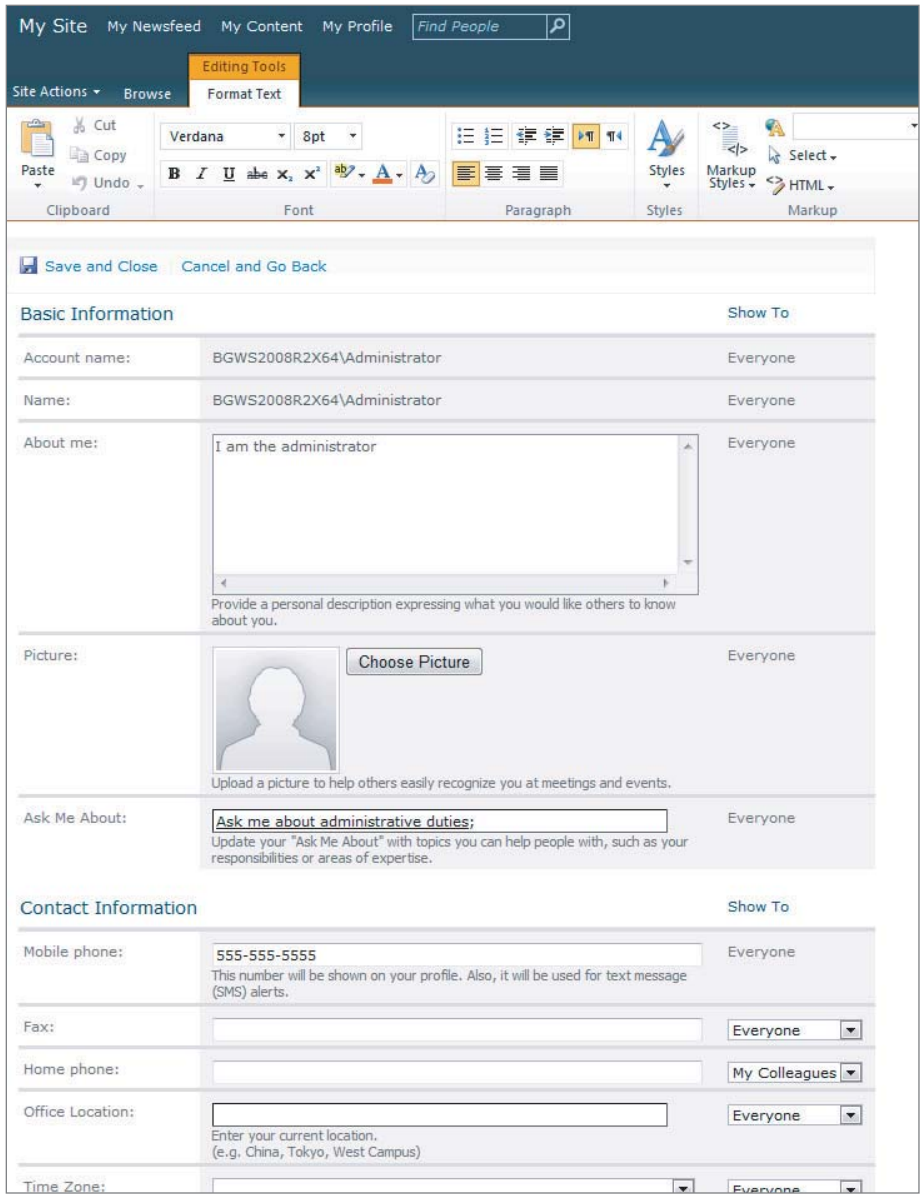


FIGURE 5-13

User Profile Policies

For each property that exists within a user profile, an administrator can configure a policy that governs the privacy settings for the property and whether users have the option to override these settings. Along with these settings, the policy also specifies whether the individual property will be replicated across all sites when user profile synchronization takes place.

User Profile Programming Model

SharePoint provides for programmatic management of user profile data through the SharePoint object model. The common classes involved in programming tagging data can be found in the `Microsoft.Office.Server.UserProfiles` and `Microsoft.Office.Server.ActivityFeed` namespaces and are described in Table 5-4.

TABLE 5-4: Common Classes Used in the User Profile Programming Model

| CLASS | DESCRIPTION |
|---|--|
| <code>ActivityEvent</code> | Represents an activity within an activity feed |
| <code>ActivityEventCollection</code> | Represents a collection of <code>ActivityEvent</code> objects |
| <code>ActivityManager</code> | Provides the methods to manage an activity feed |
| <code>ActivityPreference</code> | Represents the types of <code>ActivityEvent</code> objects that a user wants to see in an activity feed based on <code>ActivityType</code> |
| <code>ActivityPreferenceCollection</code> | Represents a collection of <code>ActivityPreference</code> objects |
| <code>ActivityPreferencePerType</code> | Represents an <code>ActivityType</code> and a boolean value indicating a set preference |
| <code>ActivityTemplate</code> | Represents the formatting of an <code>ActivityEvent</code> based on locale |
| <code>ActivityTemplateCollection</code> | Represents a collection of <code>ActivityTemplate</code> objects |
| <code>ActivityType</code> | Represents the <code>ActivityEvent</code> and its associated <code>ActivityTemplate</code> object |
| <code>ActivityTypeCollection</code> | Represents a collection of <code>ActivityType</code> objects |
| <code>Entity</code> | Represents the owner of an <code>ActivityFeed</code> |
| <code>EntityTypeIds</code> | Provides the constants for types of <code>Entity</code> objects |
| <code>Link</code> | Represents the URL associated with an <code>ActivityEvent</code> object |
| <code>List</code> | Represents a list of <code>Link</code> objects along with display information for an activity feed |
| <code>MinimalPerson</code> | Specifies the minimum information required to create a <code>SyndicationPerson</code> object that represents the author of a activity feed |
| <code>ProfileBase</code> | Abstract base class for all profile objects |

continues

TABLE 5-4 (continued)

| CLASS | DESCRIPTION |
|-------------------------------|--|
| ProfileLoader | Provides a way of obtaining user profiles |
| ProfileManagerBase | Abstract base class for all profile manager classes |
| ProfilePropertyManager | Provides the methods for managing property schemas for profiles |
| ProfileSearchManager | Provides the methods for managing and executing profile search queries |
| ProfileSubtype | Represents a profile subtype |
| ProfileSubtypeProperty | Represents a profile subtype property definition |
| ProfileSubtypePropertyManager | Provides the methods for managing profile subtypes |
| ProfileTypeProperty | Represents a profile type property |
| ProfileTypePropertyManager | Provides the methods for managing profile type property definitions |
| ProfileValueCollectionBase | Abstract class representing the values for a profile property |
| Property | Represents a user profile property definition |
| PropertyBase | Abstract base class for a profile property |
| PropertyBaseManager (T) | Abstract base class for classes that manage profile properties |
| PropertyConstants | Contains the constants for referencing default profile properties |
| PropertyDataType | Represents the type of a profile property |
| UserProfile | Represents a user profile |
| UserProfileConfigManager | Provides the methods for managing a user profile configuration |
| UserProfileManager | Provides the methods for managing a user profile |

Working with a User Profile Programmatically

Using the SharePoint object model, it is possible to programmatically manipulate user profile data within SharePoint. The following examples demonstrate this capability.



Normally, the code listings in this chapter would be running within the web context of the SharePoint server and as such, the application pool account is given full control permissions to make calls to the User Profile Service Application. Because these sample listings are written within a command-line application that must run on the SharePoint Server, they will run in the context of the current user. Even if this user is a farm administrator, this user will not have access to make calls to the service application. In order for these listings to work, you must navigate to the Manager Service Applications page in Central Administration and select the User Profile Service Application. Then, within the permissions option in the Ribbon menu, the appropriate user must be granted Full Control permissions.

Programmatically Retrieving a User Profile and Its Properties

Listing 5-7 demonstrates the retrieval of a user profile and its properties programmatically. In this example, the first action that takes place is constructing a `UserProfileManager` object from the appropriate `SPSite` and `SPServiceContext`. Using the `UserProfileManager`, a `UserProfile` for the indicated user is obtained. Once you have a reference to the profile, you retrieve a property using the `PropertyConstants` class by printing its value to the console. You then iterate through each property outputting each property's value to the console.



Available for
download on
Wrox.com

LISTING 5-7: Retrieving a User Profile and Its Properties

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.UserProfiles;

namespace Listing0507
{
    class Program
    {
        static void Main(string[] args)
        {
            string format = "{0}: {1}";

            using (SPSite site = new SPSite("http://bgws2008r2x64"))
            {
                //Get access to the service application
                //context for the current SPSite
                SPServiceContext context = SPServiceContext.GetContext(site);
```

continues


```

static void Main(string[] args)
{
    string format = "{0} [{1}]: {2}";

    using (SPSite site = new SPSite("http://bgws2008r2x64"))
    {
        //Get access to the service application
        //context for the current SPSite
        SPServiceContext context = SPServiceContext.GetContext(site);

        //Get a reference to the UserProfileManager
        //for the current service context
        UserProfileManager userProfileManager =
            new UserProfileManager(context);

        //Get user profile for a specific user
        UserProfile profile =
            userProfileManager.GetUserProfile(false);

        //Get activity manager for the current user
        ActivityManager activityManager =
            new ActivityManager(profile, context);

        //Get activities by the current user profile
        ActivityEventsCollection activityEvents =
            activityManager.GetActivitiesByMe();

        //Iterate through events to create syndication
        //items that can be displayed in a web page
        foreach (ActivityEvent activityEvent in activityEvents)
        {
            SyndicationItem syndicationItem =
                activityEvent.CreateSyndicationItem(
                    activityManager.ActivityTypes, ContentType.Html);
            Console.WriteLine(string.Format(format,
                syndicationItem.Authors[0].Name,
                syndicationItem.PublishDate,
                syndicationItem.Summary.Text));
        }

        Console.ReadLine();
    }
}
}
}

```

Programmatically Creating an Activity Feed Event for a User Profile

Listing 5-9 demonstrates the programmatic creation of an activity feed event for a user profile. In this example, you first get access to a `UserProfile` that represents the owner of the activity feed. You then get access to the current user's `UserProfile`. From these profiles, you construct the appropriate `Entity` object, which will be used for publishing to the feed. Using these entities, you use the `ActivityManager` object to create an `ActivityEvent` of the indicated `ActivityType`, populate its values, and then publish the feed event by calling `commit` on the event, which saves the changes to the database.



Available for
download on
Wrox.com

LISTING 5-9: Creating an Activity Feed Event for a User Profile

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.Server.UserProfiles;
using Microsoft.Office.Server.ActivityFeed;

namespace Listing0509
{
    class Program
    {
        static void Main(string[] args)
        {
            string format = "{0} [{1}]: {2}";

            using (SPSite site = new SPSite("http://bgws2008r2x64"))
            {
                //Get access to the service application
                //context for the current SPSite
                SPServiceContext context = SPServiceContext.GetContext(site);

                //Get a reference to the UserProfileManager for
                //the current service context
                UserProfileManager userProfileManager =
                    new UserProfileManager(context);

                //Get user profile for a specific user who will
                //be the feed owner
                UserProfile ownerProfile =
                    userProfileManager.GetUserProfile("bgws2008r2x64\\user1");

                //Get user profile for the current user who will be the publisher
                UserProfile publisherProfile =
                    userProfileManager.GetUserProfile(false);

                //Get activity manager for the publisher profile
                ActivityManager activityManager =
                    new ActivityManager(publisherProfile, context);

                //Create entities for publishing to an activity feed
                Entity owner =
                    new MinimalPerson(ownerProfile).
                    CreateEntity(activityManager);
                Entity publisher =
                    new MinimalPerson(publisherProfile).
                    CreateEntity(activityManager);

                //Create a noteboard activity event in the owner's feed
                ActivityType activityType =
                    activityManager.ActivityTypes["NoteboardPosts"];

                ActivityEvent activityEvent =
                    ActivityEvent.CreateActivityEvent(
```


People

The People section of the User Profile Service Application enables administrators to manage all aspects of user profiles within SharePoint, the social networking permissions for each user, and audience compilation for content targeting. Table 5-5 describes the configurable options in this section.

TABLE 5-5: People Management Options

| SECTION | DESCRIPTION |
|-------------------------------|---|
| Manage User Properties | Provides the capability to create, delete, and modify the properties for a user profile. When user properties are created, they can be associated with user subtypes. |
| Manage User Profiles | Provides the capability to manage individual user profiles |
| Manage User Sub-Types | Provides the capability to manage the subtypes that can be used to group user properties |
| Manage Audiences | Provides the capability to create the rules that are mapped to user profile property values, reporting structure, or group membership, which are used to indicate what users make up an audience. Audiences can be used to target content to specific set of users. |
| Schedule Audience Compilation | Provides the capability to specify a schedule by which the audience targeting rules will executed |
| Manage User Permissions | Provides the capability to enable or disable permission to use a personal feature, create a personal site, and use social features for individual users or groups |
| Compile Audiences | Provides the capability to immediately execute the rules for audience targeting |
| Manage Policies | Provides the capability to manage the policy settings for an individual property |

Organizations

The Organizations section of the User Profile Service Application enables administrators to manage all aspects of organization profiles within SharePoint. Table 5-6 describes the configurable options in this section.

TABLE 5-6: Organization Management Options

| SECTION | DESCRIPTION |
|--------------------------------|--|
| Manage Organization Properties | Provides the capability to create, delete, and modify the properties for an organization profile. When user properties are created, they can be associated with organization subtypes. |
| Manage Organization Profiles | Provides the capability to manage individual organization profiles |
| Manage Organization Sub-Types | Provides the capability to manage the subtypes that can be used to group organization properties |

My Site Settings

The My Site Settings section of the User Profile Service Application enables administrators to manage the My Site hosting infrastructure, as well as social tags and notes. Table 5-7 describes the configurable options in this section.

TABLE 5-7: My Site Settings Management Options

| SECTION | DESCRIPTION |
|---|--|
| Setup My Sites | Provides the capability to connect the User Profile Service Application to the My Site Web Application infrastructure. |
| Configure Trusted Host Locations | Provides the capability to configure which User Profile Service Application users should use for their My Site Host based on their audience. |
| Configure Personalization Site | Provides the capability to configure the links that appear within the site navigation section on the left side of the My Site and enables them to be targeted by audience. |
| Publish Links to Office Client Applications | Provides the capability to configure links to lists and sites that will appear when opening and saving documents in Office. These links appear in the My SharePoint tab and can be targeted by audience. |
| Manage Social Tags and Notes | Provides the capability to manage social tags and notes that exist through SharePoint. It is useful for ensuring that invalid or inappropriate tags and notes are properly deleted. |

Synchronization

The Synchronization section of the User Profile Service Application enables administrators to manage user profile synchronization. User profile synchronization provides SharePoint Server with the capability to synchronize user profile data with data from an external system such as Active Directory and line-of-business systems throughout the organization. This synchronization can be configured to execute in a recurring or nonrecurring manner by executing synchronization on-demand. Table 5-8 describes the configurable options in this section.

TABLE 5-8: Synchronization Management Options

| SECTION | DESCRIPTION |
|---------------------------------------|---|
| Configure Synchronization Connections | Provides the capability to configure connections to external systems for use in user profile synchronization |
| Configure Synchronization Timer Job | Provides the capability to configure the interval for executing the SharePoint Timer Job used to perform user profile synchronization |
| Configure Synchronization Settings | Provides the capability to configure profile synchronization settings for users and groups |
| Start Profile Synchronization | Provides the capability to immediately execute user profile synchronization |

ENTERPRISE WIKIS

SharePoint Server provides a publishing site template for creating an enterprise wiki (see Figure 5-15). This Enterprise Wiki site template enables users to participate in a collaborative, free-form fashion to create a common repository of information targeting a specific topic or business need. Typically, all users within an organization or subdepartment are given the permissions to contribute to this type of site to create and modify individual content pages for recording and sharing knowledge, which can be easily linked to one another. Along with this, each content page can be rated using social ratings, and tags and notes are supported throughout.

The major benefit of an enterprise wiki is that it enables users to contribute knowledge that may be not recorded anywhere else, perhaps due to its informal or assumed nature. For example, a wiki may target tips and tricks or internal company culture.

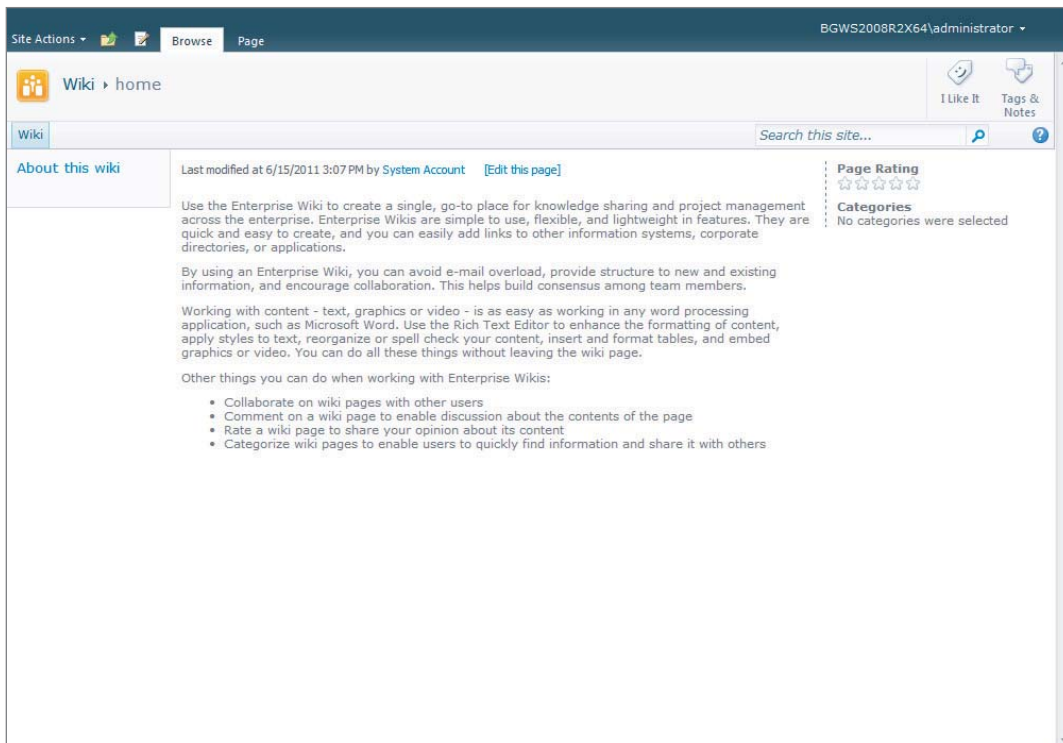


FIGURE 5-15

BLOGS

SharePoint Server provides a site template for creating a SharePoint site with blogging capabilities. This site template enables users to share ideas, observations, or expert guidance as posts, which visitors to the site can comment on. This same site template is leveraged when a user chooses to create a personal blog site from their My Site page. Figure 5-16 shows an example of a site created using the Blog site template.

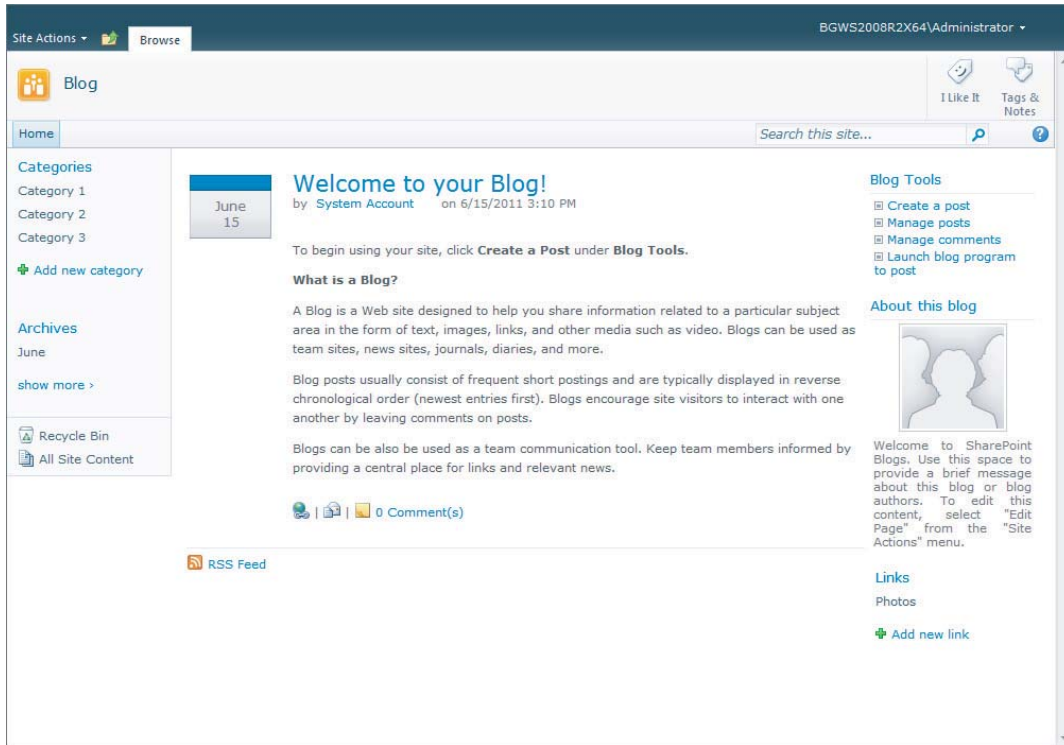


FIGURE 5-16

MICROSOFT OFFICE INTEGRATION

Microsoft Office is tightly integrated with SharePoint to support collaboration. The main collaborative applications in the Microsoft Office stack that support collaboration in SharePoint are SharePoint Workspace and Outlook.

SharePoint Workspace

SharePoint Workspace 2010 is a desktop client application that is included with Microsoft Office Professional Plus 2010, which replaces Microsoft Groove 2007. SharePoint Workspace provides the capability to create two types of entities, called *workspaces*, on the local desktop system. The two types of workspaces supported are the Groove workspace and the SharePoint workspace.

A Groove workspace works on a peer-to-peer basis, so sharing in this sense is between two individual systems with no centralized server. A SharePoint workspace, conversely, points to a centralized site hosted in Microsoft SharePoint. Essentially, this type of workspace enables users to point to a SharePoint Site in order to interact with its document libraries and lists locally on the desktop. It also enables working with the SharePoint library and list data offline, so changes can be synchronized later when a connection is restored.

Figure 5-17 demonstrates a connection to a SharePoint site using SharePoint Workspace.

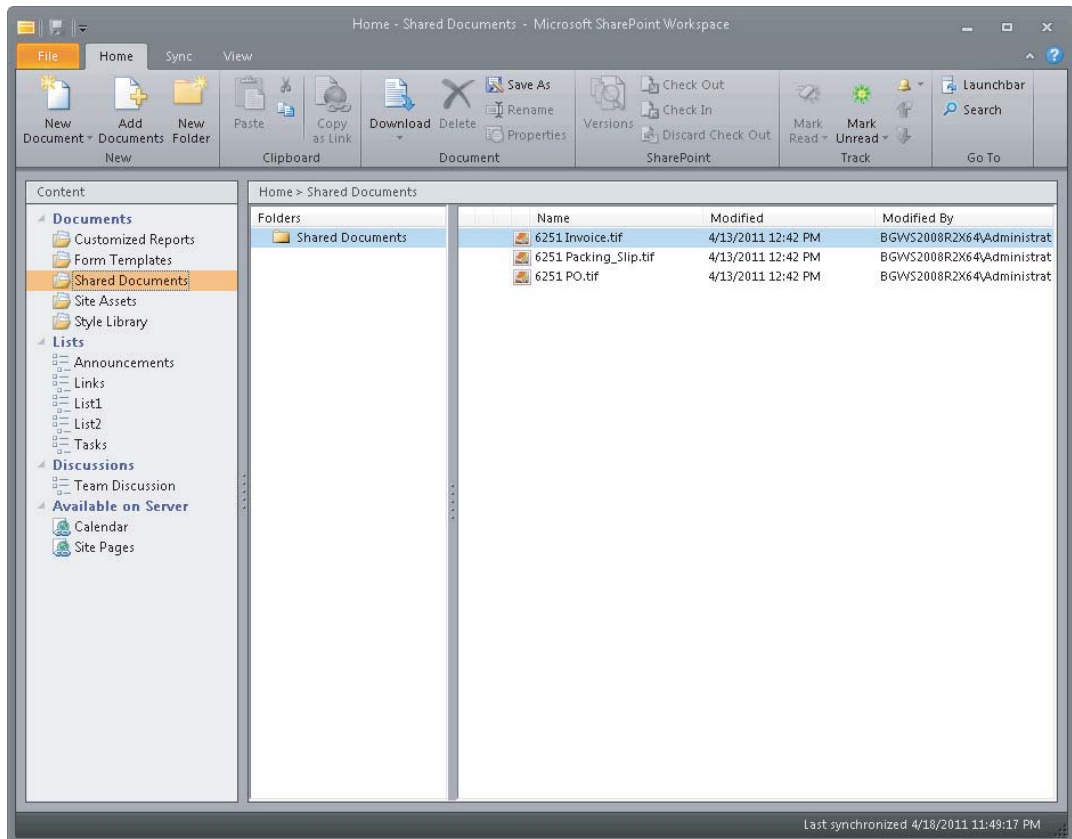


FIGURE 5-17

Outlook Integration

Microsoft Outlook 2010 is the e-mail client included with Microsoft Office 2010. It provides extensive capabilities for managing e-mail, calendars, contact lists, tasks, and notes. Along with these capabilities, Outlook provides a number of integration points with Microsoft SharePoint that enable collaboration from a user's local desktop.

Outlook can be integrated with SharePoint by allowing users to synchronize with libraries, lists, calendars, contacts, discussion boards, RSS feeds, and meetings workspaces. It is even possible for Outlook to connect to a user's My Site through its built-in Social Connector Plugin in order to stay up-to-date with status and activity feed updates from with Outlook.

Follow these steps to connect Outlook to the newsfeed of a user's My Site:

1. Open Outlook 2010 and navigate to the reading pane with the Mail section.
2. By default, at the bottom of the screen, the Social Connector Plugin is displayed. Expand it.

3. Once expanded, click the Add button to add a social network connection.
4. Choose the My Site provider. It is possible to install other providers to connect to other social networks — such as LinkedIn, for example.
5. Enter the URL to the My Site location along with credentials for this site.
6. Click Connect, and then click Finish.

Once Outlook has been connected to the My Site, the Social Connector will display status and activity feed updates for the colleagues configured within the My Site for the current user.

SUMMARY

Collaboration is a critical component of doing business, and therefore a critical component of enterprise content management. AIIM, the worldwide ECM association, recognizes this significance and has outlined eight conceptual stages that encompass the collaborative life cycle in order to provide an overview of how collaborative software enables users to collaborate in an agile yet orderly manner. It also recognizes the need to effectively track and record collaboration, especially when collaboration involves items that require strict management.

Social networking is enhancing the way in which users collaborate within an organization in many new and effective ways. SharePoint provides extensive collaborative capabilities and has incorporated social networking as a central component of this functionality. Although SharePoint has always been engineered with collaboration in mind, it continues to enhance and extend its offerings with each release.

A business cannot thrive without effective collaboration, and SharePoint provides for extensive and effective collaboration, including the capability to manage and track these efforts. As organizations leverage SharePoint's features, they begin to understand the powerful influence that these features can have over how they conduct their collaborative efforts on a daily basis, which enables them to be much more productive and efficient in terms of coordinating and sharing knowledge throughout the organization.

6

Search

WHAT'S IN THIS CHAPTER?

- Profiling the corpus
- Understanding scalable search solutions
- Designing a scalable search architecture
- Monitoring and tuning search performance

In many ways, search is one of the most important subsystems in any ECM platform. Even the best content storage and management solution is useless if end users can't reliably locate the right content at the right time. Unfortunately, it can be a difficult task to get users to move away from the tried-and-true method of hierarchy-based folder navigation toward using the search center or other powerful search tools.

This chapter dives into the underlying technology and search concepts that enable end users to locate and retrieve ECM content consistently and reliably. It also discusses the functional components of SharePoint Enterprise Search and FAST for SharePoint 2010 as a foundation for architecting a scalable search solution that will drive user adoption. Finally, sample search topology architectures are identified, along with the performance factors that drive them.

INTRODUCTION

It doesn't matter whether SharePoint is being used as a public-facing portal driven by content management, a collaboration portal driven by organization employees, or a document and records management system driven by regulatory compliance policies; the system's ability to crawl, index, and return query results in an accurate and efficient manner is always a core requirement.

It's pointless to implement a highly scalable system if users can't locate content when the search subsystem is called upon. The concept of content search has rapidly evolved in recent years. Back in the days when paper was king and trees were afraid for their lives, interns and runners were the "search technology" of choice. The boss would send someone to a big room loaded with filing cabinets and have them manually dig through records (alphabetized, it is hoped) to locate a customer record. This was a risky proposition. What if there were a fire or some other natural disaster?

To solve this problem, an entire service industry was born with the sole mandate of protection and timely destruction of paper documents. Customer records would be stored in massive, environmentally controlled warehouses. Corporations would pay a hefty fee to ensure that records could be stored and, when necessary, destroyed in a safe and consistent manner. However, with such a huge repository of paper documents, it could take days or even weeks to locate a customer record. Often, these manual "searches" cost money too! Executing content retrieval was essentially a last resort that caused a significant bottleneck in the customer service process.

As corporate networks and file servers began to proliferate, it was obvious that digital storage was the wave of the future. The corporate "share" drive became commonplace, as did hierarchical folder structures comprised of simple metadata. Users continued to draw on the file cabinet mentality to help with organization. Various navigation schemes employing year, department, and customer name would help users to locate content; but this unstructured technique is quite painful for new employees who don't know to which department or managed account a customer file belongs. New users would be inefficient for some time while they learned the system. The larger the content set, the longer it would take to ramp up new employees.

Seeing the need for a better way to locate content, software vendors began to develop and sell document management solutions. These systems were successful in that they added structure to the content and leveraged early search concepts to locate documents. The biggest drawback was solution cost and the specialized skills necessary to maintain such systems. Many of these implementations are viable even today, but at an increasingly heavy maintenance cost.

Similarly, internal development shops would often create homegrown search solutions using Microsoft SQL Server and/or Microsoft Index Server. These solutions have varying degrees of scalability and overall success. Depending on the needs of the each organization, they may be perfectly functional for long-term use.

But times change and technology continues to evolve. Antiquated document management systems are becoming too expensive for the limited functionality they provide, and homegrown solutions usually can't continue to scale as businesses strive to add customers. As a result, a powerful new technology emerged to provide a solution to the rapidly accelerating growth of document and record storage.

Retrieval: The Key to User Adoption

Only in the last few years has the corporate portal evolved into *the* central location for information and content. Largely thanks to low *total cost of ownership* (TCO) of Microsoft SharePoint, IT shops far and wide began to see the value of the SharePoint portal. Starting with Microsoft Office SharePoint Server 2007 (MOSS), enterprise search in the corporate portal started to gain traction. SharePoint could crawl all those old file shares and Exchange Public Folders in which users had stored content. More important, SharePoint could be used as a powerful content and records management solution for tens of millions of documents. Also, thanks to MOSS search, content could be queried

regardless of the SharePoint site or library in which it was stored. Content types and site columns could be used to describe documents in a much more structured way, enabling a rich user experience.

Not all corporate intranets are properly adopted by the organization's user community. This can happen for several reasons. Perhaps the portal information architecture and taxonomy was poorly designed and deployed. Perhaps the portal does not drive new content, giving users a reason to return. But given the new ECM capabilities of the SharePoint-based corporate portal, a big issue is that users can't find content after it's uploaded. To mitigate this, the natural reaction is to return to the hierarchical nature of sites, libraries, and folders so that users can just "navigate" to content. As mentioned earlier, this is not a scalable solution. It's very difficult to navigate to a document when there are 70 million documents to weed through.

Search is the key. Users must be motivated to access the search center, and when searches are executed they need to "just work." Results need to be relevant and usable every time. Quite often, an IT department will deploy a SharePoint farm with a well thought out information architecture and even drive users to submit content. Initially, the system seems fantastic; but over time, search results begin to degrade. Query latency increases, results aren't as fresh as they should be, or they simply aren't as relevant as they should be.

Just as SharePoint requires regular monitoring and maintenance, it is beneficial to have someone specifically dedicated to the *search administrator* role. This person performs several functions:

- Monitors query latency and indexing latency to ensure that results are fresh and returned in a timely manner
- Monitors query rate to ensure that queries per second remain within the capabilities of the existing topology
- Monitors indexing rate to ensure that crawled documents per second is keeping up with new or changed content
- Monitors frequently performed searches and adjusts keywords and best bets to bubble up hot topics to the top of search results
- Ensures that managed properties are properly mapped to crawled properties to facilitate exact relevance searches against structured metadata
- Monitors the number of documents in an index partition to ensure that it does not exceed 10 million
- Creates new index partitions to ensure that the 10 million documents per partition recommendation is not exceeded or creates a fault-tolerant index partition
- Creates search scopes and targeted search results pages that help users narrow down the content that is being queried
- Manages new content sources as web applications are added to the farm

These are just a few of the responsibilities of a search administrator. The important point is that if the search subsystem is properly maintained, end user queries will consistently return relevant results in a timely manner. This increases confidence in the SharePoint farm as a viable ECM solution that will help make life a little easier for the end user. In the end, satisfied users will accept the technology, and widespread user adoption will improve business efficiency.

The Corpus Profile

SharePoint administrators often look to books, blog sites, or even Microsoft TechNet for help with SharePoint topology recommendations, particularly with respect to search architecture. Even this chapter includes sample topologies that provide a general idea of what hardware resources might be needed in various circumstances.

While these are excellent resources to start the information-gathering process, it's imperative for researchers to understand that these information sources are, by nature, very general. Myriad variables affect the ability of a search subsystem to provide accurate and relevant results. When considering the example scenarios, the administrator must view the guidance being provided in light of the organization's *content corpus*.

Think of the corpus as the size and shape of the content that will be stored in SharePoint or some other content source. Before a search topology can be properly architected, the administrator must understand the profile of the corpus and how it affects crawl and query processing. You can define the profile of the corpus by answering several questions based on currently available information and future projections.

What Types of Documents Will Be Crawled?

This question has a profound impact on the size of the content index. In many cases, SharePoint will be implemented primarily as a content collaboration solution. In this scenario, most documents will usually be Microsoft Word, Excel, or PowerPoint documents. Because these documents tend to be primarily text, SharePoint full-text indexes these documents by default. This results in a content index that is fully populated based on the sizing expectations set forth in standard Microsoft documentation.

Consider a second scenario in which you have some collaborative data but SharePoint is primarily used as a content repository for a document imaging system. In this scenario, a high volume of TIFF documents is loaded into SharePoint on a regular basis. Out of the box, SharePoint does not include an IFilter that performs any type of optical character recognition (OCR) on TIF images. An IFilter is a Microsoft search technology plugin that extracts full-text data from specific file types. In the case of a TIF document, the only way to extract text data is to perform OCR in real-time during a crawl, but it's a very expensive operation that does not scale well. Because there is no IFilter for TIF images, the NULL IFilter is used. The document metadata (column data) will be added to the property index, and, if specified, to the content index. Because the document is not crawled for full-text indexing, very little data is added to the content index. This results in a faster indexing rate and a smaller overall content index. The bottom line? While the recommended calculations for the property index are still valid, the guidelines for predicting the size of the content index would result in a significantly overestimated index size.

Is an IFilter Available for Full-text Crawling All Document Types?

Microsoft SharePoint includes several IFilters out of the box. An installation prerequisite is the Microsoft Office Filter pack, which installs IFilters for legacy Office file types, the newer "Metro" Office file types, as well as a few others such as Zip, OneNote, Visio, Publisher, and Open Document Format.

As alluded to in the previous profile question, TIF is not handled by SharePoint out of the box. Another important document type not handled natively by SharePoint is Adobe PDF. If an organization has a significant number of full-text PDF documents included in the corpus, then a PDF IFilter should be acquired and installed. If the organization does not wish to use SharePoint search to query a given third-party document type, then the IFilter should not be installed.

When a third-party IFilter is needed, it is important to determine the impact of the IFilter on index rate performance. Some IFilters are still based on 32-bit core code or do not handle multithreading well. SharePoint 2010 is 64-bit, and crawl operations are multithreaded, so this is an important consideration. For example, some PDF IFilters perform significantly better than others. In order to estimate the performance impact, the IFilter should be tested. Then the administrator should weigh the organization's need for full-text search of a given file type against the performance cost of full-text indexing the file type.

How Many of Each Document Type Will Be Crawled?

The number of existing documents is often a tough value to identify. It is of vital importance that every effort be made to determine document counts by document type for the major document types in the corpus. Without these counts, it is very difficult to predict the storage requirements for property database(s), content database(s), and index partition(s).

What Is the Average File Size By Document Type?

Once a document count by document type is available, the next step is to determine the average document size, preferably by document type. Again, average document size is instrumental in calculating the raw storage requirements for documents and ultimately the estimated content database size.

SharePoint only crawls documents that are 16MB or less. This is the default value of the `MaxDownloadSize` property of the Search service application. When determining average file size, also pay attention to the number of documents that are larger than 16MB. If this number is significant, try to find a sweet spot that will enable most of the larger documents to be crawled with minimal negative impact to index rate. The maximum value of the `MaxDownloadSize` property is 64MB. You can adjust the value of the `MaxDownloadSize` property using PowerShell:

```
$ssa = Get-SPEnterpriseSearchServiceApplication
$ssa.SetProperty("MaxDownloadSize",<new size in MB>)
$ssa.Update()
```



After using PowerShell to change the `MaxDownloadSize` value for the search service application, the `OSearch14` service application will need to be restarted on each crawl server for it to take full effect.

How Often Are Existing Documents Changed?

The corpus change rate has a significant impact on search topology and hardware requirements. For example, a very large organization that makes extensive use of SharePoint as a collaborative

document and records management portal may have tens of millions of documents (or more) that are frequently changed. This high volume of changes to process will cause significant pressure on the crawl subsystem in order to keep indexes fresh. This can be a significant factor in determining how many crawl servers are necessary to service the corpus. Conventional guidance may project an underestimated number of crawl servers.

Conversely, if an organization with the same large corpus uses SharePoint as a document archive but the documents rarely change, then the number of crawlers required falls back to the SLA requirements for the time it takes to execute a full crawl should it be necessary. Perhaps both solutions service 70 million documents, but the crawler requirements would not be the same due to vastly different corpus change rates.

How Much New Content Will Be Added During a Specific Period of Time?

Like the corpus change rate, the rate at which new documents are added to the corpus is very important. In this case, it doesn't necessarily matter if the documents are collaborative or archive in nature. However, predicting the number of documents is again an important factor in order to size the search topology properly. Equally important is the period of time over which documents are added.

Consider the document load that an accounting firm may generate over the course of a year. Throughout the year, the document load may be fairly stable. However, as a tax deadline approaches, a rapid spike in new content is common. When determining the search profile, potential load rate spikes must be accounted for so that search topologies can be properly architected to handle expected loads throughout all phases of the business season.

Impact of the Corpus Profile

As with any topology guidance, mileage will always vary from organization to organization, and performance testing is always the best way to fine-tune an implementation; but every firm must begin with initial recommendations or consultative advice from best practices resources. After that, understanding how the corpus profile affects those recommendations can mean the difference between having to make drastic changes to the topology versus minor tuning tweaks after testing.

SEARCH SOLUTIONS

SharePoint search technology has significantly evolved since the early days of SharePoint Portal Server 2003. Back then, shared services were but a shadow of what they would become. It was possible to get decent full-text results, but property-based search was definitely in its infancy. The only real way to execute a property-based search was to formulate a Collaborative Application Markup Language (CAML) query. While this was effective for a single library, it wasn't possible to execute cross-library or cross-site property-based searches. This lack of metadata-based search was a significant limitation that inhibited the adoption of SharePoint Portal Server 2003 as an ECM solution.

Fast forward a few years to the introduction of Microsoft Office SharePoint Server 2007 (MOSS) and WSS 3.0. WSS heralded an evolution of CAML known as SiteDataQuery. Using SiteDataQuery enabled users to perform property-based searches across libraries and sites as long as they existed in the same content database; but the new capabilities didn't stop there. MOSS also introduced

the concept of *managed properties*. The MOSS Enterprise Search subsystem was able to extract library column data related to a given document. These *crawled properties* could then be mapped to a managed property, either automatically or manually. When a crawled document has a column value (crawled property) that is mapped to a managed property, that value is stored in the *property catalog*, sometimes called the *property index*. The property catalog is a special table in the search database that stores name/value pairs. Essentially, the property catalog becomes a superset of all managed metadata serviced by the *shared service provider*. The property index can then be queried by advanced search pages or object model code in order to perform property-based searches across any site collections or web applications that are serviced by the shared service provider.

Even with the significant advancements of MOSS Enterprise Search, two extremely important drawbacks remained. Both drawbacks are related to the fact that a given shared service provider can be associated with only one index server. In other words, the drawbacks are crawl performance and availability. Not only is crawl performance physically limited to the boundaries of a single crawl server, but if that server fails, no other server can take over crawl operations. Fortunately, the content index can be stored on other query servers, which allows end users to search previously crawled content even if the index server fails. Because the index server can be rebuilt and data won't technically ever be lost, the impact on the business is recovery time, during which search results are stale.

Crawl performance was always a limiting factor. Even with a massive index server with a large number of physical cores and a significant amount of RAM, MOSS imposed a 50 million document recommended maximum limit to corpus size. Limited search database flexibility also had a big impact on the recommended limit. The same database used to store the property index, a superset of all managed metadata in the farm, was also used to manage crawl processing. With a 50 million document corpus, this database could easily be 500GB or more in size. The size and I/O performance of this massive database can limit crawl performance. Combined with the fact that a single index server is crawling all content, a full crawl of a 50 million document corpus might take weeks. Obviously, this is not an optimal scenario, particularly given the single-point failure possibility of the index server.

In January 2008, Microsoft acquired FAST Search and Transfer (FAST). FAST Enterprise Search (ESP) was the technology that Microsoft needed to overcome the limitations of MOSS Enterprise Search. FAST ESP can be flexibly scaled out to support *billions* of documents! It is a repository-agnostic search solution. That is, it can crawl SharePoint content just as easily as it can crawl web content. It also has the advantage of an advanced content processing pipeline. More important, it can be architected into a highly available solution. It arrived late in the MOSS 2007 life cycle, but it has been implemented, quite successfully, in some very large MOSS farms.

Microsoft rolled the FAST team into the Enterprise Search team as development of SharePoint Server 2010 kicked into high gear. The influence of the FAST team can be clearly seen in the updated architecture of the SharePoint 2010 Enterprise Search solution. The property database and crawl databases were separated. Multiple property and crawl databases are now possible. Crawl servers are now stateless and there can be more than one. If one of the crawl servers fails, crawl processing can still proceed on other crawl servers. Several architectural concepts from FAST have found their way into the SharePoint 2010 Enterprise Search platform, enabling it to scale up to a 100-million-item corpus.

Even though SharePoint Enterprise Search has dramatically improved, there is still a need for extreme scale search solutions. For this reason, the Microsoft Enterprise Search team developed

FAST Search Server 2010 for SharePoint, which scales out to support a corpus of up to 500 million items! FAST Search Server 2010 for SharePoint is external to the core SharePoint Server 2010 code base, but it does directly integrate with the Application Service model in SharePoint 2010 through a Content Search service application and a Query Search service application. This close integration has several benefits. First, end users who are used to the advanced search capabilities of MOSS 2007 can have a similar experience. Second, third-party vendors who have developed search applications that take advantage of the MOSS Search API layer only need to make minor changes to support FAST. Finally, all the power and flexibility that FAST Search Server 2010 for SharePoint provides is available in a new FAST Search Center site template.

While the FAST team has had a significant impact on SharePoint 2010 Enterprise Search, there are definitely reasons why, in some cases, FAST Search Server 2010 for SharePoint might be a better fit. In order to determine which search platform is best for a given organization, it's important to understand the solution components of each platform.

SharePoint Server 2010 Enterprise Search

The Enterprise Search platform has been significantly re-architected in SharePoint Server 2010. Gone are the scalability and availability limitations of a single index server and a single search database. The new topology is significantly more flexible, at the cost of some complexity.

Topology Components

SharePoint implementers and administrators need to gain a solid understanding of the changes to the topology options for SharePoint 2010 Enterprise Search. It is important to grasp the purpose of each component so that appropriate topology configuration decisions can be made, implemented, and tested to ensure that performance and availability requirements are met.

As you review each of the following components, understand that they are all interconnected, so it is difficult to describe one without also mentioning its relationship to other components, which may not yet be described. All the components are eventually covered, however, so it may be beneficial to read this section on topology components a couple times to gain a better understanding of each component and how it relates to the other components in the search subsystem.

Index Partition

In MOSS 2007 Enterprise Search, the content index was stored on both the index server and each query server. A single, related search database contained the property index. The “index,” which consisted of both the content index and the property index, could support a maximum of 50 million content items.

In SharePoint 2010 Enterprise Search, the *index partition* is the unit of measure. Multiple index partitions can be created, each supporting a recommended maximum of 10 million content items. Each index partition is associated with a single property database that contains a superset of name/value pairs for all content items serviced by the index partition. In the most basic deployment, a single query partition can be deployed with a single query component on a single query server. In this scenario, the entire full-text content index exists in a single index partition (see Figure 6-1)

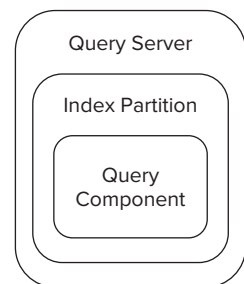


FIGURE 6-1

Query Component

Each index partition will contain one or more *query components*. Each query component contains at least a portion of the content index. Typically, an additional query component is added to an index partition in order to “mirror” the content index on another query server for the purpose of fault tolerance (see Figure 6-2). The query architecture in Figure 6-2 could support a corpus of up to 10 million items with fault tolerance.

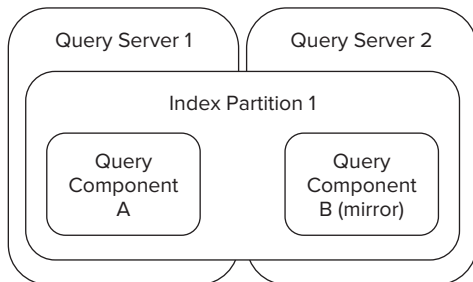


FIGURE 6-2



Adding a mirrored query component is typically recommended. However, because each query component contains a full copy of the content index, adding a mirrored query component doubles the content index storage requirement.

It is also possible to have more than two query components in the same index partition, but it is not recommended. If improved query performance is desired, it is better to add an additional index partition. For example, a query architecture with two index partitions, each with a primary and a mirrored query component, deployed to two query servers could support a corpus of up to 20 million items (see Figure 6-3).

Query Server

Each query component will be hosted on a specific *query server*. A query server can host multiple query components, each of which is associated with a specific index partition.

Each active query component on the query server consumes RAM as it serves query requests.

Microsoft recommends that for each active query component on a query server, 33% of the content index serviced by the query component should fit into RAM.

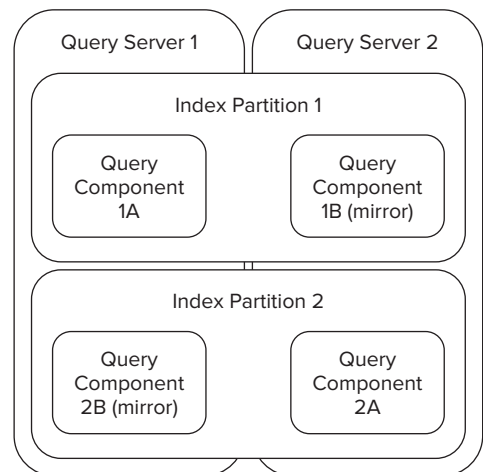


FIGURE 6-3



You should regularly monitor the amount of disk space consumed by a content index for a given index partition associated with a given active query component. The content index grows over time at a rate that is directly proportional to the rate at which content is added to the farm. In order to maintain Queries per Second (QPS) targets, it is important to ensure that at least 33% of the content index can fit into available RAM on the query server as the content index grows.

Regardless of whether a query component is active or mirrored, the associated content index will consume disk I/O during both full and incremental crawl operations. Microsoft recommends that 2,000 IOPS be available per pair of query components (active/mirrored) on a given server. That doesn't necessarily mean that the active/mirrored pair is associated with the same query partition. Figure 6-3 illustrates this concept.

Microsoft also recommends that two physical CPU cores be dedicated to each active query component, and that one physical CPU core be dedicated to each mirrored query component. As long as RAM, disk I/O, and CPU recommendations are accounted for, it is possible to have more than two query components and their associated content indexes deployed on a given query server. However, because each index partition services up to 10 million items, query component deployment should be limited to one active and one mirrored query component per query server. Following this best practice will ensure that a suitable QPS is achieved and that end users experience responsive search operations.

Property Database

The *property database* contains the property store for index partitions. An index partition can be associated with only one property database but multiple index partitions can be associated with the same property database. Therefore, any given crawled item may have managed property values stored in only one property database. For any given crawled item, the metadata properties of that item will be stored in the property store. Microsoft recommends that the database server that hosts the property database have enough RAM to hold 33% of the property store in memory. In addition, the database server storage needs to accommodate 2,000 write-heavy IOPS per property database.

The property store is a very important component of the search subsystem with respect to SharePoint ECM solutions. Advanced, property-based searches use the property store and managed properties to produce exact-relevance search results.

Crawl Database

In MOSS 2007, the database tables that managed crawl processing had a profound impact on the performance of the search database. By adding the capability for multiple crawl databases that are separate from property store databases, Microsoft has significantly enhanced the scalability of SharePoint 2010 Enterprise Search.

A *crawl database* is used by an associated crawl component to manage crawl operations for a subset of the corpus served by the associated search service application. One or more crawl components can be associated with a single crawl database. Crawl processing for a given host or web application

can only be managed by a single crawl database. Microsoft has established a supported threshold of 25 million items per crawl database. Because only one crawl database can be used to crawl a given host or web application, it is important not to exceed 25 million items in any web application that will be crawled. This is particularly relevant for large-scale document archive type ECM solutions.

Crawl database performance is highly dependent on the I/O subsystem of the database server. During crawl operations, a single crawl database consumes a minimum of 3,500 (read heavy) IOPS. It can utilize as much as 6,000 IOPS if available. For this reason, it is recommended that a maximum of two crawl databases be deployed to a given database server. Under a load of tens of millions of items, two crawl databases will significantly strain the I/O subsystem of the database server, so it is imperative that the I/O subsystem be extremely high performance.

When a new host is added to the farm, SharePoint attempts to assign the host to the crawl database that has the fewest items. It is possible to control the distribution of hosts to crawl databases by adding and managing the host distribution rules for the search service application (see Figure 6-4).

Crawl Server

A *crawl server* is the physical server machine that hosts crawl components. A crawl server can host multiple crawl components regardless of the crawl database with which the crawl component is associated. However, crawl components take full advantage of up to four physical CPU cores during crawl processing. During crawl operations, crawl components build a temporary local index before propagating the index changes to the index partitions. Interacting with the temporary index causes a load of approximately 300 to 400 IOPS on the crawl server. Before adding additional crawl components to a crawl server, ensure that you have enough resources to service the new crawl component without negatively impacting the existing crawl components.

One crawl server hosting one crawl component that is associated with one crawl database can support approximately 10 million items (see Figure 6-5).

Crawl Component

A *crawl component* (also called an indexer) extracts content from a content source in order to build a content index and a property store. A temporary content index is constructed locally on the crawl server just before it is propagated to an index partition. The index partition that will contain the index for a given item is chosen by SharePoint in such a way that the index partitions remain balanced as much as possible. The temporary index content is propagated to both the active index partition and, if it exists, the mirror index partition.

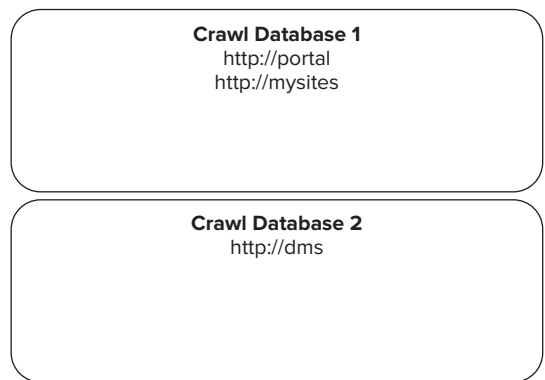


FIGURE 6-4

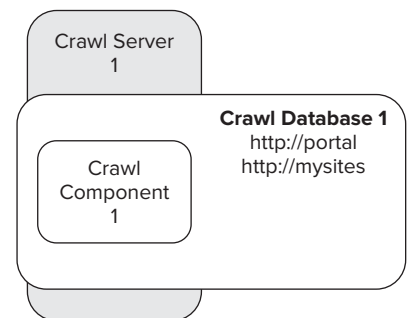


FIGURE 6-5

Any crawled properties of a given item are stored in the property database. If more than one property database exists, the property values for the item are stored in the property database associated with the index partition that was chosen by SharePoint to contain the content index for the item.

A crawl component can only be associated with a single crawl database. Multiple crawl components can be associated with the same crawl database. Multiple crawl components can be created on different crawl servers to scale out crawl processing and improve overall crawl speed. Two crawl servers, each hosting one crawl component that is associated with the same crawl database, can support approximately 20 million items (see Figure 6-6).

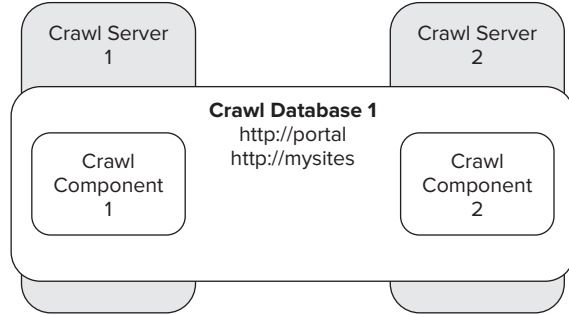


FIGURE 6-6

Adding an additional crawl database and additional crawl components can further improve scalability as long as the crawl servers can handle the load of additional crawl components. Two crawl servers each hosting two crawl components that are connected to two different crawl databases can handle approximate 40 million items (see Figure 6-7).

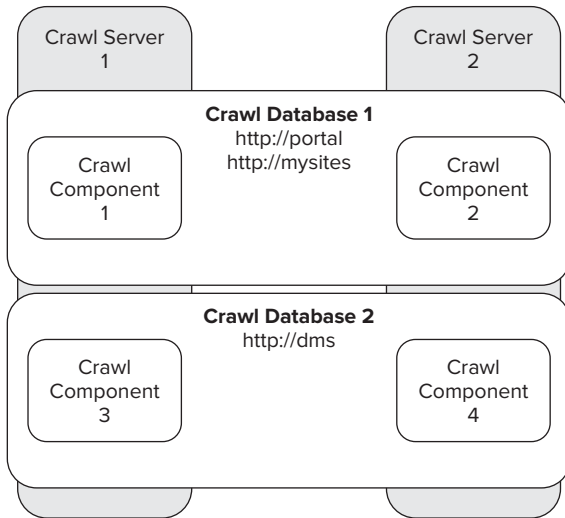


FIGURE 6-7

Configuration Components

In many ways, the SharePoint 2010 Enterprise Search configuration points are quite similar to the configuration points of MOSS 2007. For the purposes of ECM in SharePoint 2010, the following configuration points are relevant.

Content Sources

A *content source* enables the administrator to determine what content will be crawled and when. From an ECM perspective, it is important to determine how fresh index results need to be. For example, a large document repository may need very fresh results so that customer service representatives can locate newly added customer documents as soon as possible. In this case, the content source should be scheduled for incremental crawls that are separated by just a few minutes each throughout the day. In contrast, a long-term document archive content source may receive very few queries, in which case a single daily incremental crawl during off-peak hours would be sufficient (see Figure 6-8).

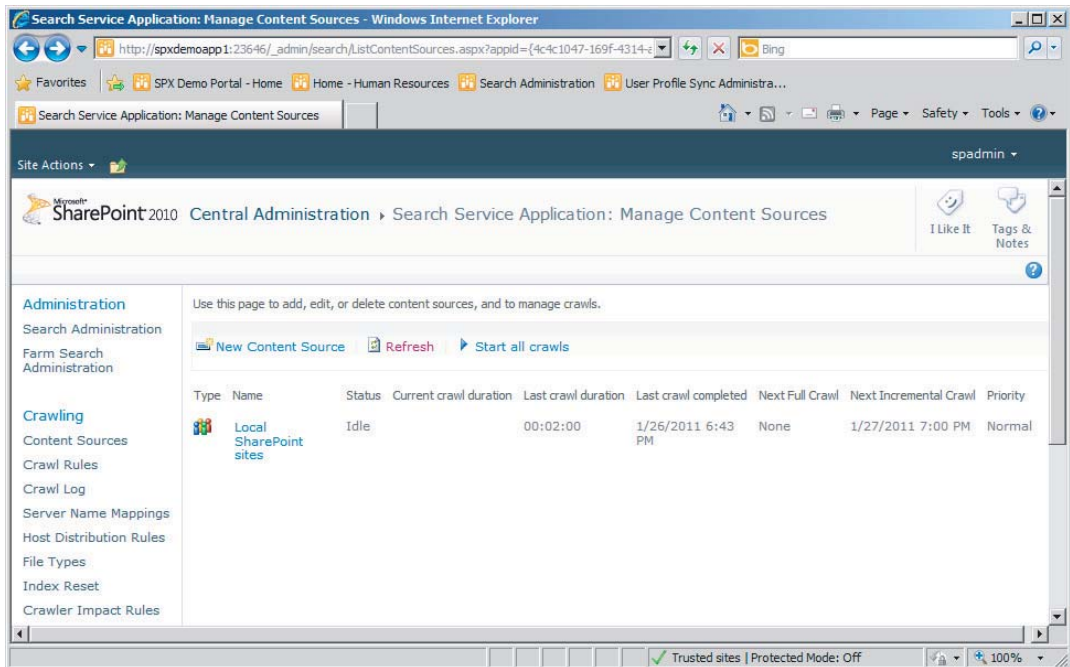


FIGURE 6-8

Crawl Rules

Crawl rules in SharePoint 2010 are essentially unchanged from MOSS 2007 with one big exception: They now support regular expressions. This can be particularly helpful when users have a tendency to embed sensitive information in filenames. Crawl rules continue to be very useful when items need to be excluded from search results. With regular expression support, crawl rules are a much more powerful tool.

Host Distribution Rules

A *host distribution rule* can be used to assign a specific crawl database to a specific host. Because a crawl component is associated with a specific crawl database, and a crawl component is deployed

on a single server, it is possible to designate a specific crawl database, crawl component, and crawl server for the purpose of crawling a specific host. This can be very useful, particularly when there are multiple hosts with significantly different types of content.

Consider a scenario in which an organization has a collaboration portal that is serving their search needs well. The same organization wants to add a new document management system for high-volume invoice processing, storage, and retrieval. When added to the farm, this new web application will create significant pressure on a crawl subsystem that otherwise performs quite well when crawling just the collaborative content. To avoid the possibility of the collaboration portal experiencing stale search results, the organization can add a dedicated crawl database, crawl server, and crawl component for crawling the document management system content. A crawl rule can then be used to associate the new crawl database with the new document management system web application hostname (see Figure 6-9).

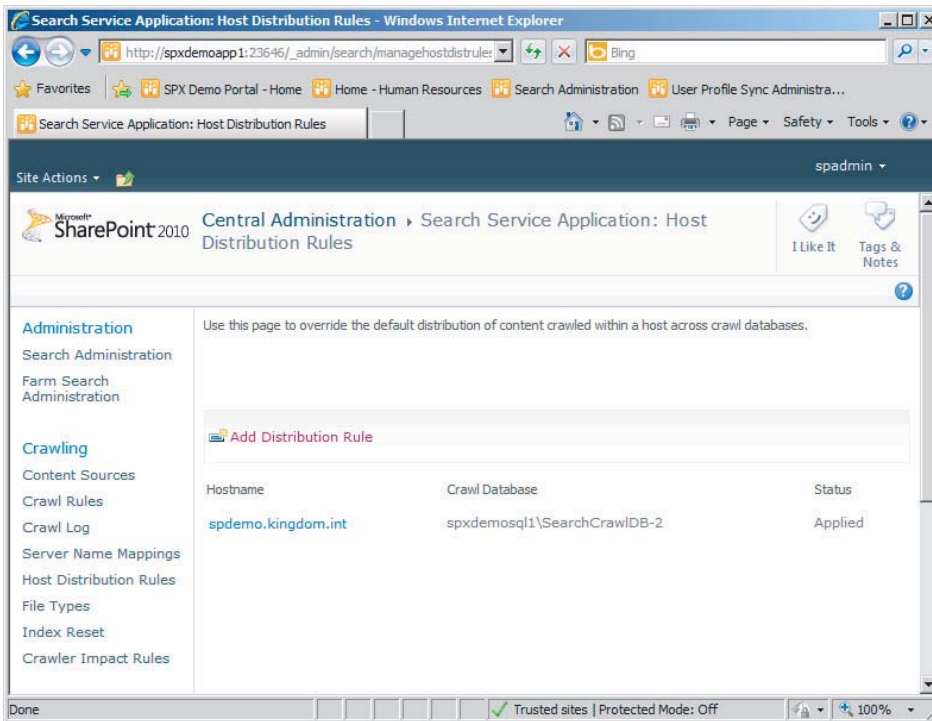


FIGURE 6-9

Crawled Properties

Crawled properties are metadata values related to a specific item (see Figure 6-10). When an item is crawled, the metadata values associated with that item are stored in the property database. Crawled properties are considered during query processing when a user submits a full-text query. Advanced property-based searches can be executed against a crawled property if it has been mapped to a managed property.

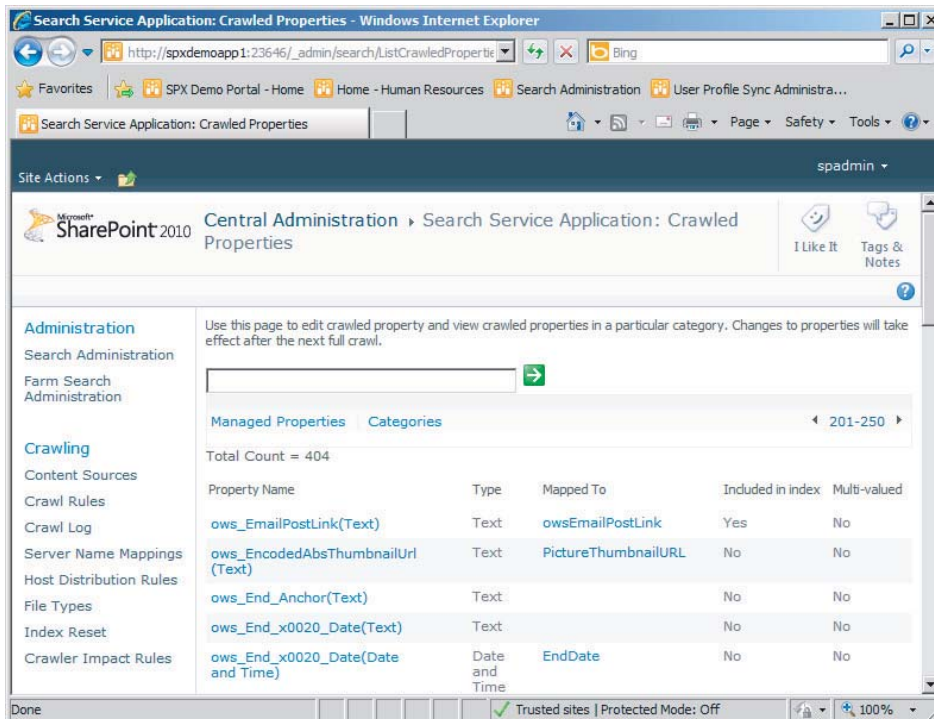


FIGURE 6-10

Managed Properties

One or more crawled properties can be mapped to a *managed property* (see Figure 6-11). After a crawled property has been mapped to a managed property, users can execute structured metadata-based queries that return exact relevance results. Managed properties are very important to both collaboration and document management type sites. The capability to describe the content that is stored in a site or library is one of SharePoint's most powerful features. Executing managed property-based queries takes advantage of this structured metadata in order to return predictable query results.

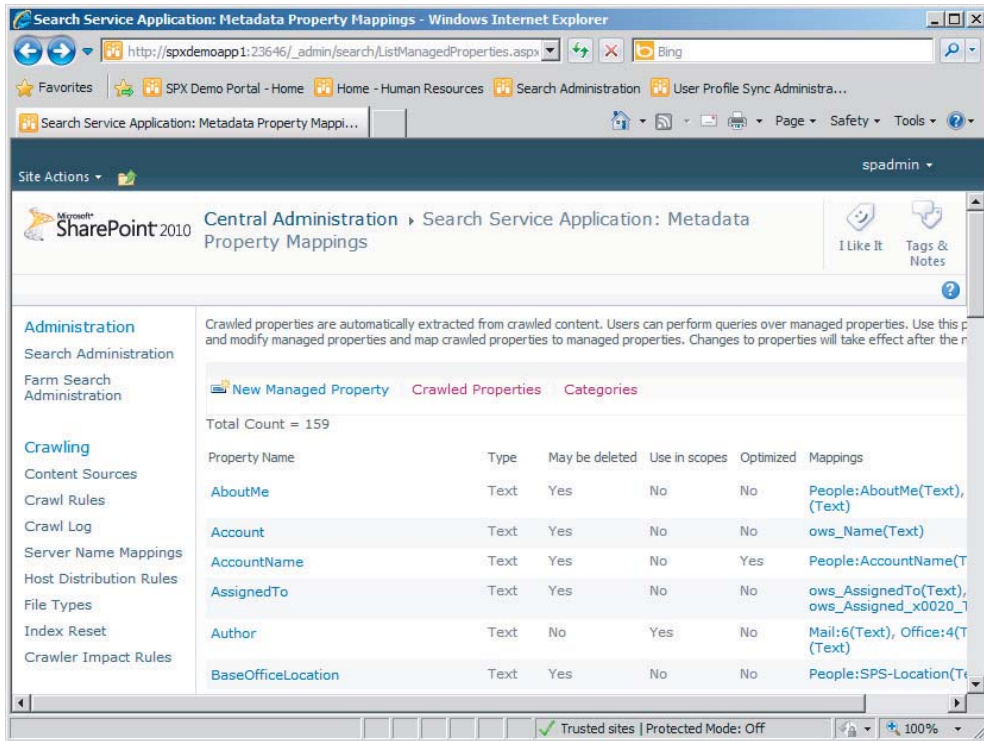


FIGURE 6-11

Search Scopes

Search Scopes are very powerful in that they provide a mechanism for defining a subset of the index that can be searched. The scope is defined by a web address, property query, or content source. The property query is particularly useful because it can define a subset of content that shares the same property value while the actual content is scattered throughout a content source. Consider a scenario in which a large staffing organization in Europe receives many new consultant applications per day. Due to the wonders of telecommuting, applicants could be located anywhere across the continent. When a prospective customer requests possible candidates for consulting work, it might be easier to search for only those candidates located in the same country as the prospective client. By tagging all candidate records with a `CountryCode` property, it is possible to create a search scope for each country based on a property query (see Figure 6-12). Once a search scope has been created, it can be associated with a search results page such that it is possible to search for skilled candidates within the scope of the relevant country.

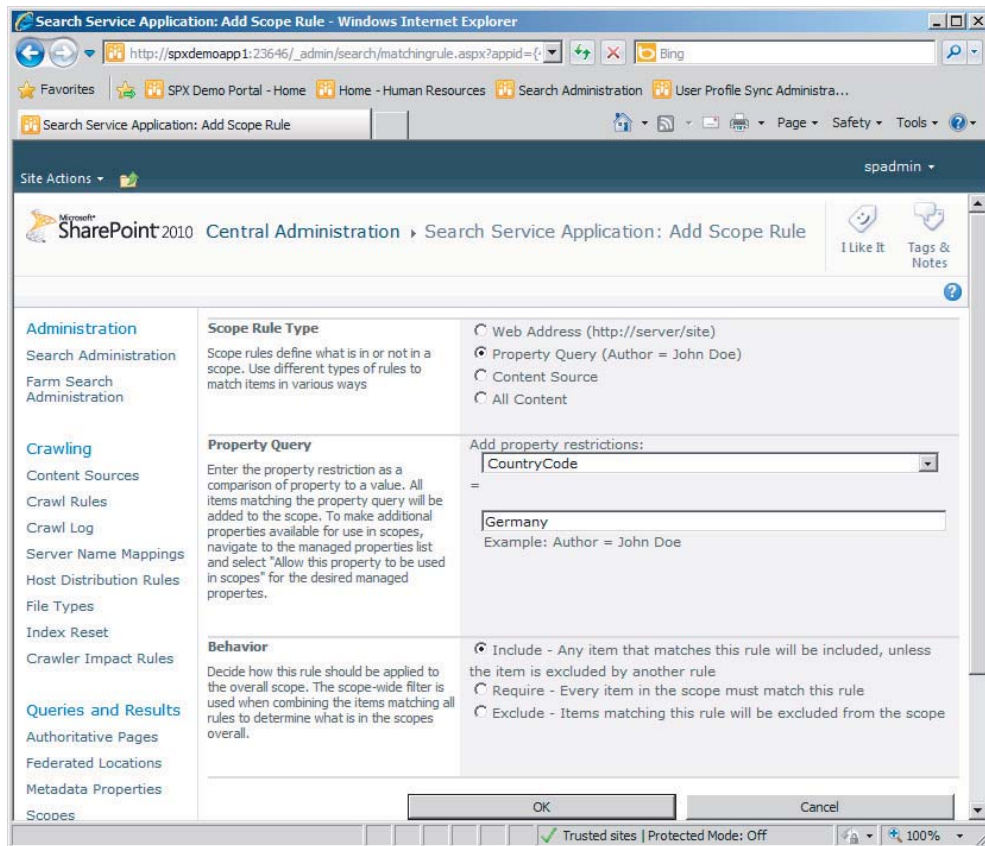


FIGURE 6-12

The Search Center

The *search center* is a SharePoint site that provides an interface for end users to execute queries and refine results in order to identify a particular subset of items based on a full-text keyword or item metadata. The search center is particularly important in the realm of SharePoint ECM. In any ECM system, content will be added and hopefully tagged using keywords or structured metadata.

The SharePoint Enterprise Search subsystem also “crawls” many document types in order to extract additional keywords. The purpose of manual and automated full-text and metadata extraction is to provide a way for end users to retrieve content. After all, if users can’t reliably locate content that was added to the repository, then the solution loses credibility and the goal of user adoption is lost.

Microsoft has significantly enhanced the search experience in SharePoint 2010 in order to further empower users to execute queries that will yield reliable results. The following topics discuss the new features available to the SharePoint 2010 search center and how they relate to ECM.

Improvements to Keyword Search

Several functional improvements have been made to keyword search in SharePoint 2010. Keyword queries now support Boolean operators such as AND, NOT, and OR. Keyword queries also support using a prefix word followed by an asterisk (“*”) character in order to execute a prefix-driven wildcard search. Another new feature is support for the NEAR operator for keyword proximity searches (see Table 6-1).

TABLE 6-1: Keyword Search Examples

| NEW FEATURE | QUERY TEXT | QUERY RESULT |
|-----------------|-----------------------|---|
| Boolean AND | apple AND pear | Any item that contains both the “apple” keyword and the “pear” keyword will be returned. |
| Boolean OR | lemon OR orange | Any item that contains the word “lemon” or the word “orange” will be returned. |
| Boolean NOT | strawberry NOT banana | Any item that contains the keyword “strawberry” will be returned <i>unless</i> it also contains the keyword “banana.” |
| Wildcard Prefix | prof* | Returns items that contain keywords such as “profile,” “professional,” or “proficiency.” |
| Proximity NEAR | imaging NEAR solution | Returns all items that contain the word “solution” within the eight keywords that follow the word “imaging.” |

Finally, a new refinement panel takes advantage of managed properties to provide a quick way to narrow down search results and improve relevancy. When a user executes a search, the refinement panel, located just to the left of the search results, is populated with a list of properties that can be used to filter or “refine” the result set with a simple click of the mouse (see Figure 6-13). In addition to the refiners that SharePoint automatically generates, you can also add custom refiners to the refinement panel.

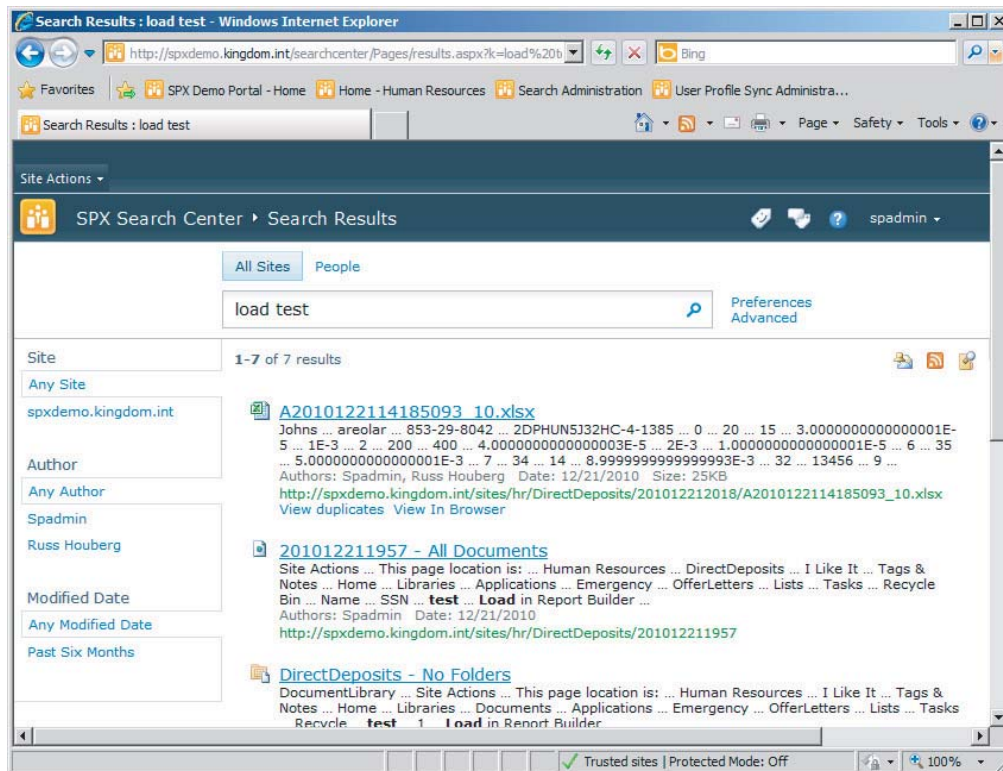


FIGURE 6-13

Metadata Search versus Keyword Search

The value of a metadata-based query can't be overstated in an ECM solution. Considering the fact that Microsoft is investing in new features like metadata-driven refiners, it is clear that queries that take advantage of structured metadata are very powerful.

Consider a scenario in which a nationwide financial institution uses SharePoint as a document management solution for loan application processing. Each application might consist of a primary form document and perhaps between 5 and 10 supporting documents, all of which will be stored in SharePoint. Hundreds of loan applications are received every day and they may take up to a week to process. Loan processors work with a workflow-driven "inbox" that prompts them to begin work

on a new loan. The first step in that process is to conduct a review that ensures that all required documents have been submitted by the applicant.

What is the most effective way for the loan processor to retrieve these documents? First of all, due to the volume of content, it is not reasonable to expect that a loan processor could just “navigate” to all the documents for a specific user. One might assume that it would be possible to just enter the applicant’s name — say, “Bill Johnson” — as a free-text query. That’s a problem. Perhaps 30 “Bill Johnson” applicants have gone through the loan process in the last year alone. Is the loan processor stuck? Nope. It turns out that a unique loan number barcode sticker was added to each application document by the loan originator at the branch office. As the documents were captured into SharePoint, an automated solution automatically scanned the barcode and populated a “Loan Number” metadata field during upload to SharePoint. That means the loan processor can just use the unique loan number found in the workflow inbox to search for all documents related to the application. Better yet, the workflow inbox Web Part could be designed such that the loan number is hyperlinked to the search center with the property-based search pre-populated in the search center page URL!

Because of the globally unique metadata value and the fact that a metadata property-based search was executed, every single document that has been tagged with the unique loan number will always be returned (as long as the loan processor has the rights to view the document). Perhaps more important, any document that is not directly tagged with the loan number will *not* be returned, cluttering up the search results. This is the concept of using metadata properties to drive queries that return exact-relevance search results. This concept is crucial to making the loan processor’s job easier and thus driving user adoption.

Controlling the Scope

There are many scenarios in which an end user might want to begin a search in the context of a specific subset of the overall corpus. SharePoint facilitates this with search scopes. Returning to the earlier scenario regarding the staffing organization in Europe, it is possible to create a search scope that would enable a staffing specialist to search for a skilled candidate in a specific country. You could also create additional search pages in the search center that take advantage of specific scopes. The following steps can be used to create a customized search experience based on a specific scope:

1. Open SharePoint Central Administration and the General Application Settings link.
2. On the General Application Settings page, under the Search section, select Farm Search Administration.
3. On the Farm Search Administration page, under Search Service Applications, select the search service application.
4. On the Search Administration page, under the Queries and Results Section, click Scopes.
5. On the View Scopes page, click New Scope.
6. On the Create Scope page, shown in Figure 6-14, populate the title of the scope. Then click OK.
7. On the View Scopes page, in the Update Status column of the new scope, click Add rules.
8. On the Add Scope Rule page, populate the Scope Rule Type and related scope restriction values. In the Behavior section, select Require to ensure that the scope is limited to the defined restriction (see Figure 6-15). Click OK.

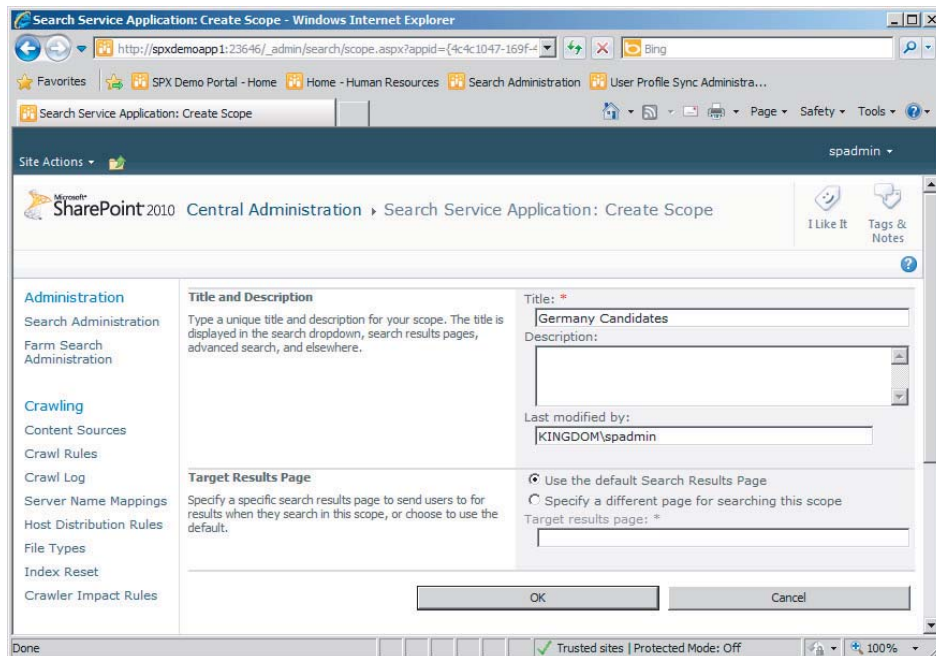


FIGURE 6-14

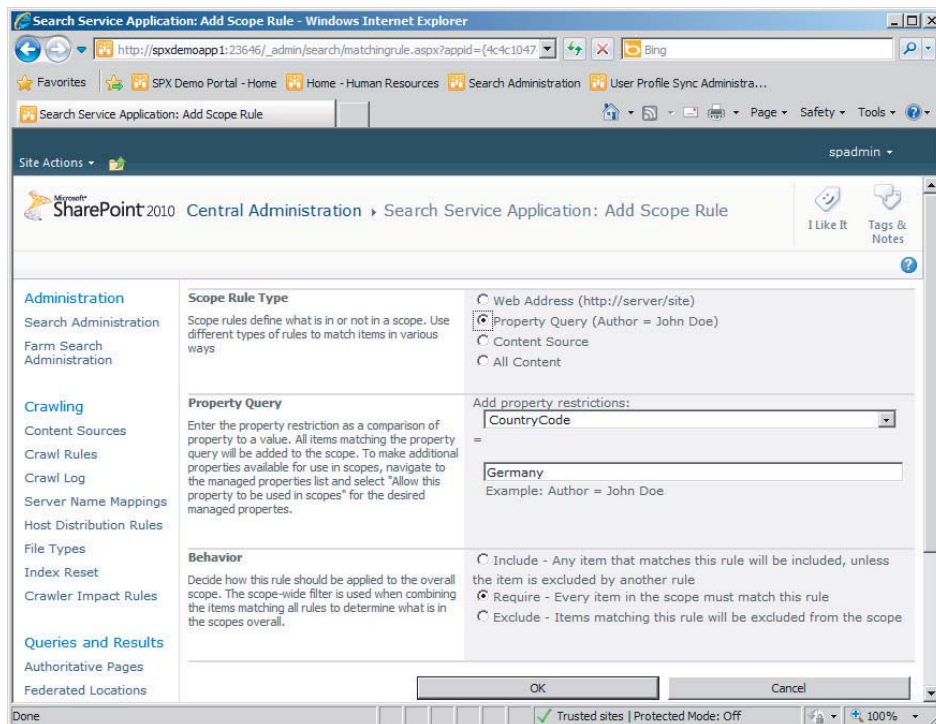


FIGURE 6-15



Scope changes are applied during an update process that executes every 15 minutes. Therefore, depending on when the scope change is saved, it may take up to 15 minutes for the change to be available for queries.

9. Navigate to the search center site.
10. Select Site Actions ⇨ View All Site Content.
11. On the All Site Content page, under the Document Libraries Section, select the Pages document library.
12. In the Pages document library, select the Documents tab ⇨ New Document ⇨ Page.
13. On the Create Page page, give the search results page a title and a URL name. In the Page Layout section, select (Welcome Page) Search results (see Figure 6-16). Click the Create button.

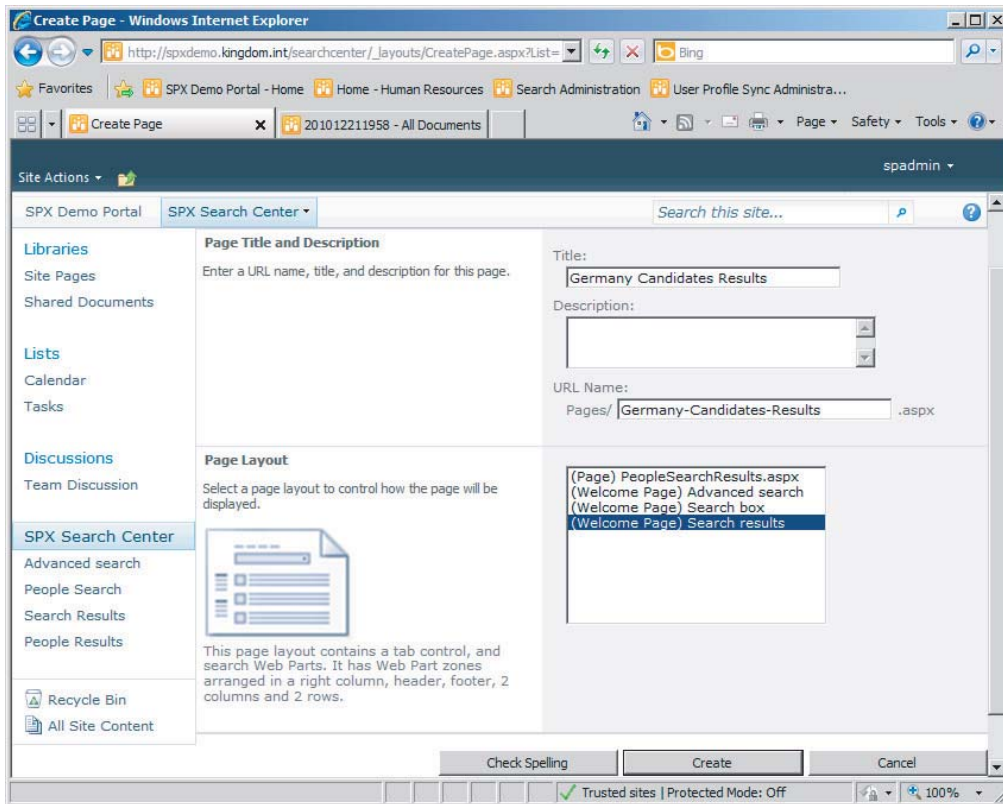


FIGURE 6-16

14. In the Pages document library, click the name URL for the new results page just created.
15. On the new search scope results, click Site Actions ⇨ Edit Page.

16. Near the top of the page, click the Add New Tab link.
17. On the Tabs in Search Results - New Item page, provide a name for the new tab. In the Page section, enter the name of the page, including the file extension (see Figure 6-17). Add a tooltip description if desired. Click the Save Button.

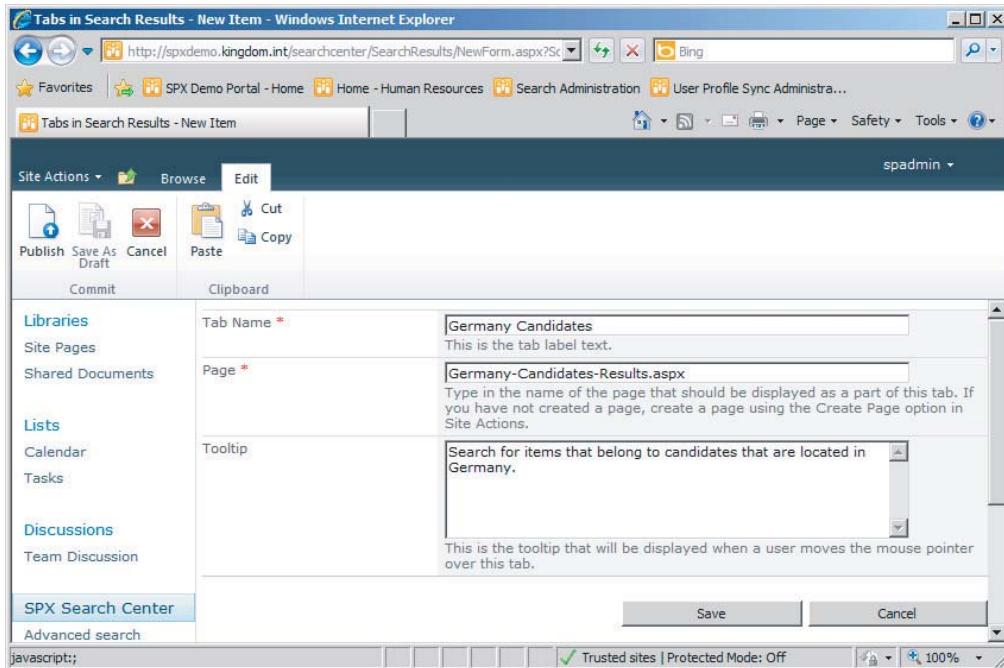


FIGURE 6-17

18. On the new search scope results, again click Site Actions ⇄ Edit Page.
19. In the Search Core Results Web Part, click the configuration dropdown menu ⇄ Edit Web Part.
20. Expand the Web Part configuration Location Properties section. In the Scope field, enter the exact name of the scope that was created in step 6 (see Figure 6-18). Click the OK button.
21. On the new search scope results page, click the Publish tab ⇄ Publish menu button. Enter comments if desired and click the Continue button.
22. Select Site Actions ⇄ View All Site Content.
23. On the All Site Content page, under the Document Libraries Section, select the Pages document library.
24. In the Pages document library, select the Documents tab ⇄ New Document ⇄ Page.
25. On the Create Page page, give the search page a title and URL. In the Page Layout section, select (Welcome Page) Search box (see Figure 6-19). Click the Create button.

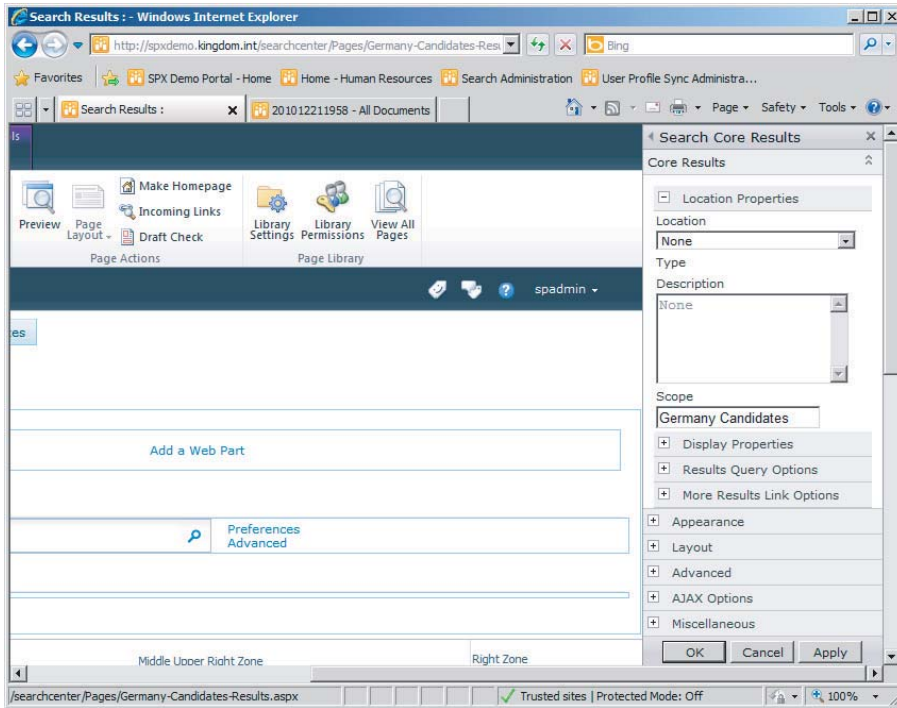


FIGURE 6-18

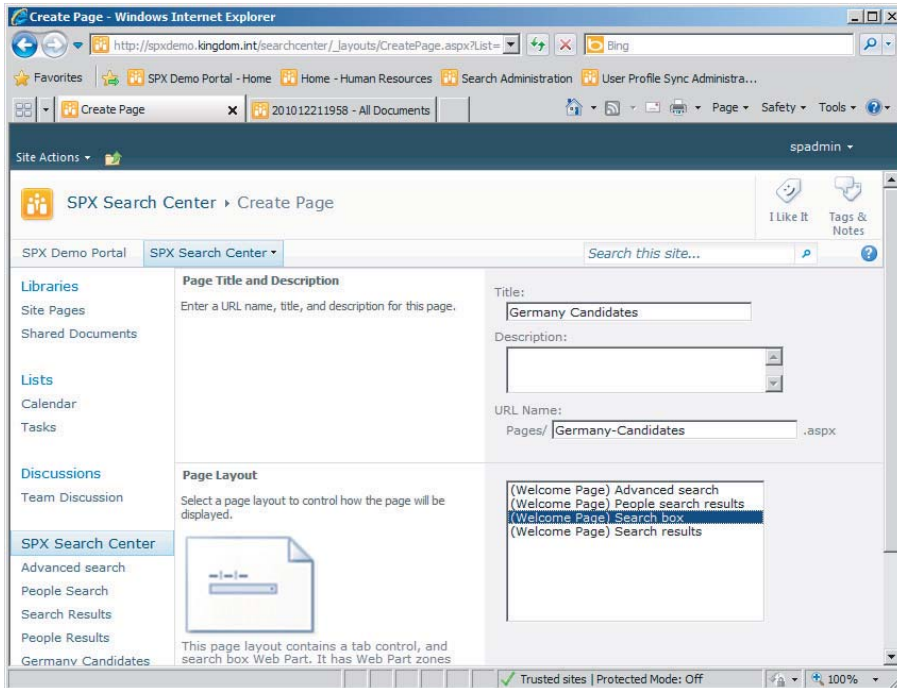


FIGURE 6-19

26. In the Pages document library, click the name of the URL for the new search page that was just created.
27. On the new search page, click Site Actions ⇄ Edit Page.
28. Near the top of the page, click the Add New Tab link.
29. On the Tabs in Search Pages - New Item page, provide a name for the new tab. In the Page section, enter the name of the page, including the file extension (see Figure 6-20). Add a tooltip description if desired. Click the Save Button.

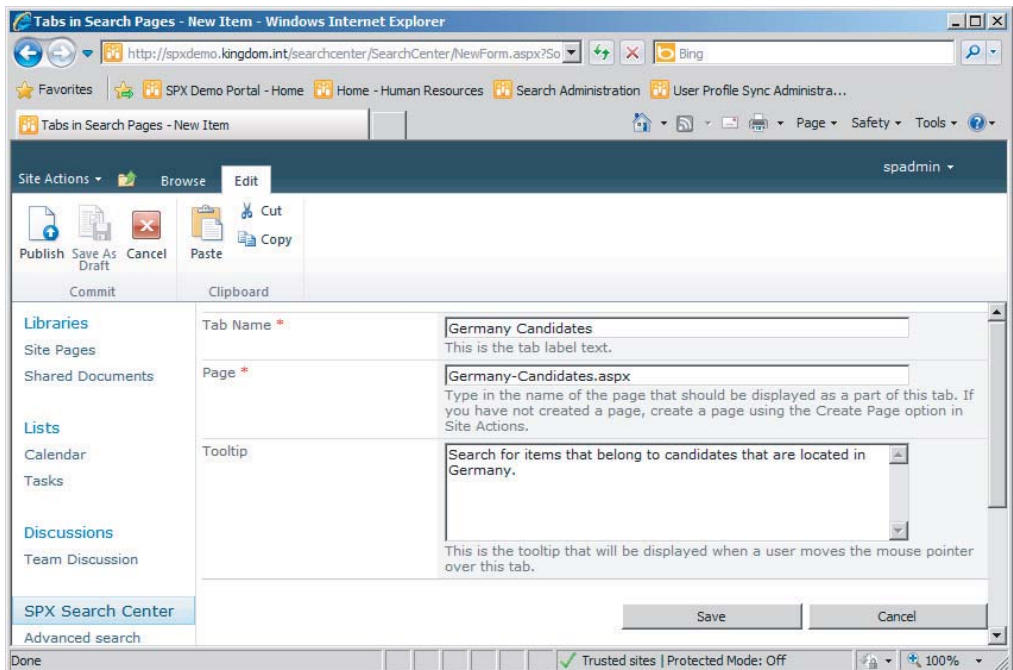


FIGURE 6-20

30. On the new search page, again click Site Actions ⇄ Edit Page.
31. In the Search Box Web Part, click the configuration dropdown menu ⇄ Edit Web Part.
32. Expand the Web Part configuration Miscellaneous section. In the Target search results page URL field, enter the search results page name (with file extension) that was determined in step 13 (see Figure 6-21). Click OK.
33. On the new search page, click the Publish tab ⇄ Publish menu button. Enter comments if desired and click the Continue button.

At this point the search scope should be properly configured. It should be possible to compare the search results using the All Sites scope against the search results that are configured to take advantage of the new scope (see Figure 6-22).

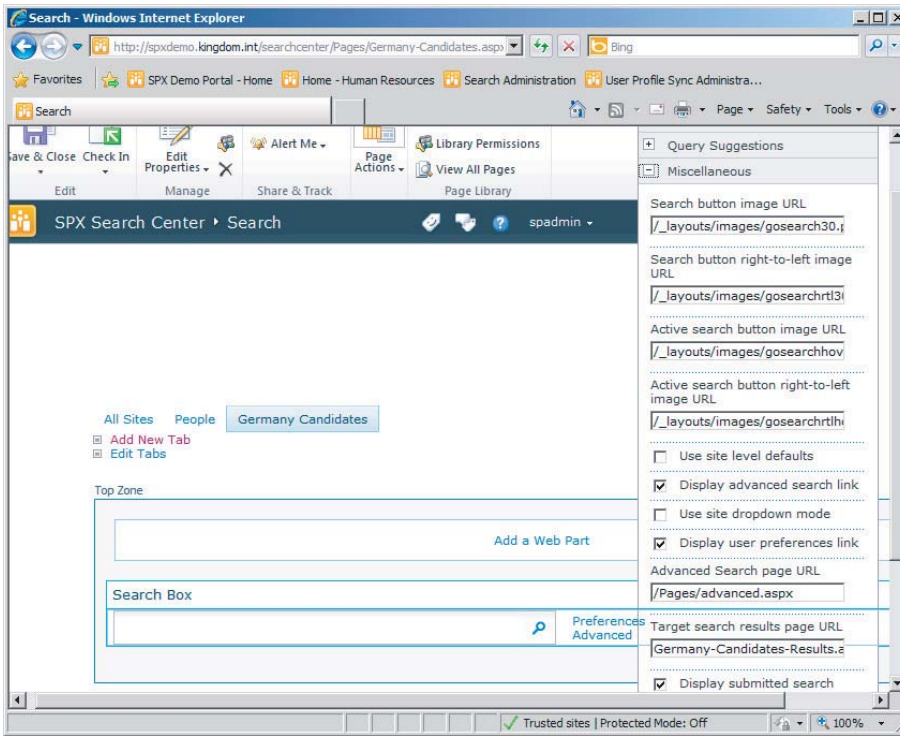


FIGURE 6-21

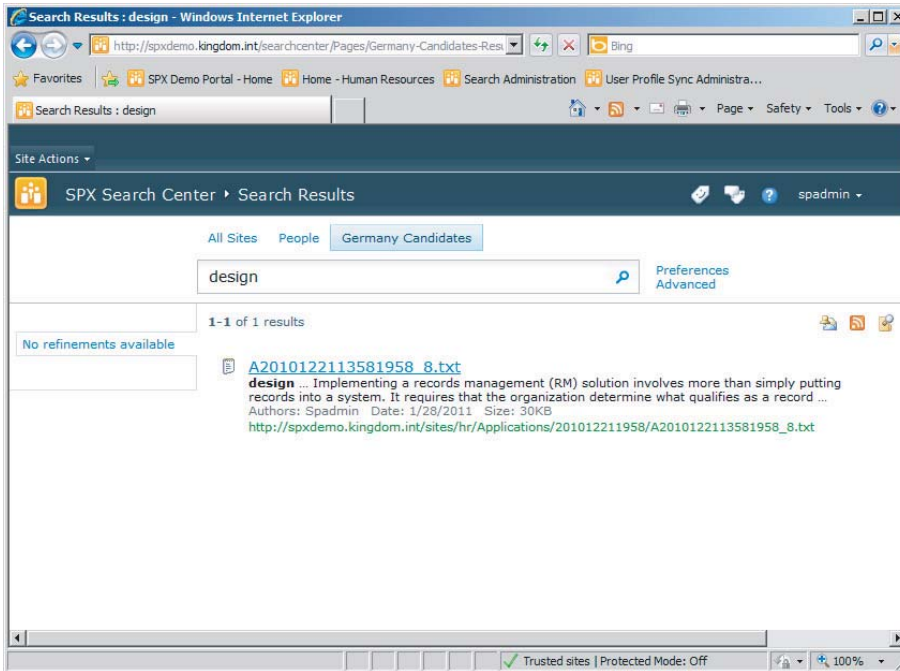


FIGURE 6-22

Advanced Property-Based Search

This chapter has regularly referenced the capability to execute property-based searches. Consider again the scenario in which a user wants to search for documents that originated in a particular country. For small or medium search environments, a search scope may be overkill. However, it is possible to use that managed property directly in a query. How can end users do this? Well, unless a third-party solution is purchased to facilitate dynamically driven managed metadata queries, a bit of manual effort is required. The following instructions identify the steps required to add a searchable managed metadata property to the advanced search results page in the search center:

1. Using the browser, navigate to the search center.
2. On the search center page, click the Advanced link to the right of the search box.
3. On the Advanced Search page, under the Add property restrictions section, click the (Pick Property) dropdown box. Notice that it does not automatically include all managed properties (see Figure 6-23).

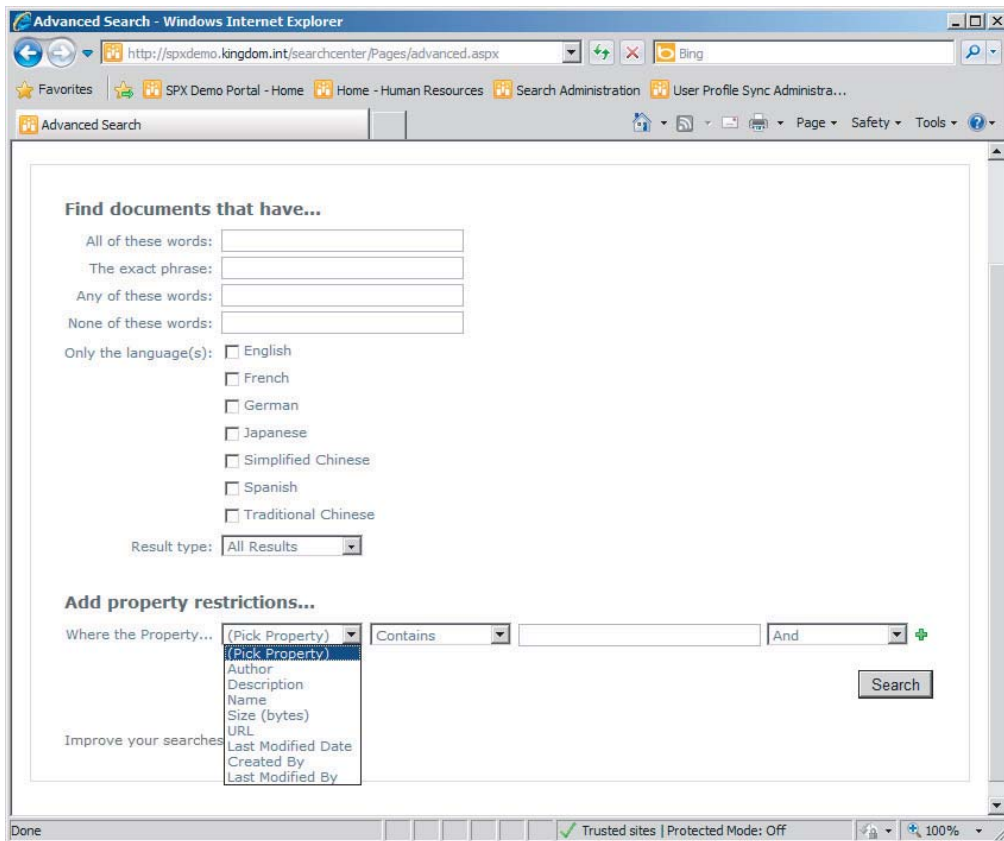


FIGURE 6-23

4. On the Site Actions menu, click Edit Page.

5. In the Advanced Search Box Web Part, click the configuration dropdown menu ⇌ Edit Web Part.
6. In the Properties section of the Advanced Search Box Web Part configuration pane, click the ellipsis button next to the Properties field.



It is much easier to edit the property's XML if the XML text is copied from the property's XML edit box, pasted into a standard text editor, modified, and then placed back in the property's XML edit box.

7. In the property's XML text, insert a new `<PropertyDef />` element as a child under the `<PropertyDefs />` element (see Figure 6-24).

```

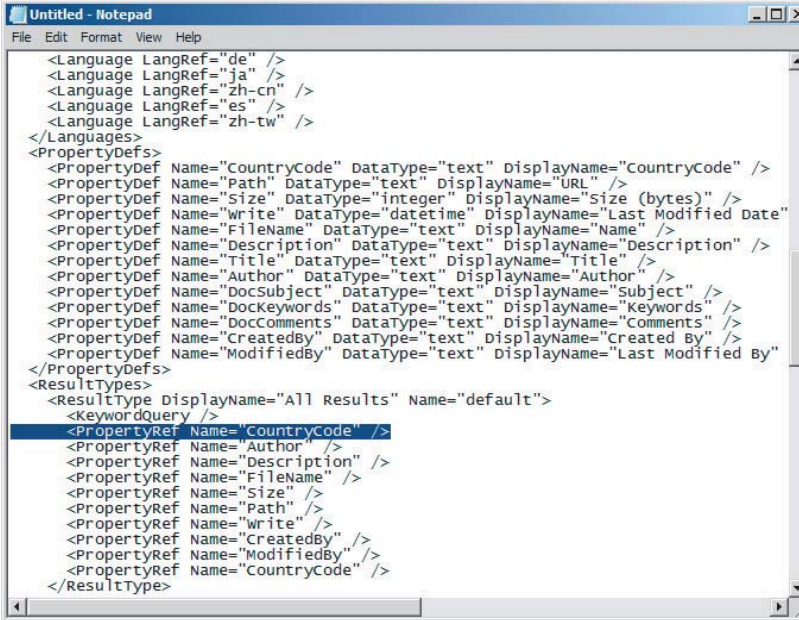
Untitled - Notepad
File Edit Format View Help
<LangDef DisplayName="ukrainian" LangID="uk" />
<LangDef DisplayName="urdu" LangID="ur" />
<LangDef DisplayName="vietnamese" LangID="vi" />
</LangDefs>
<Languages>
  <Language LangRef="en" />
  <Language LangRef="fr" />
  <Language LangRef="de" />
  <Language LangRef="ja" />
  <Language LangRef="zh-cn" />
  <Language LangRef="es" />
  <Language LangRef="zh-tw" />
</Languages>
<PropertyDefs>
  <PropertyDef Name="CountryCode" DataType="text" DisplayName="CountryCode" />
  <PropertyDef Name="Path" DataType="text" DisplayName="URL" />
  <PropertyDef Name="Size" DataType="integer" DisplayName="size (bytes)" />
  <PropertyDef Name="write" DataType="datetime" DisplayName="Last Modified Date" />
  <PropertyDef Name="FileName" DataType="text" DisplayName="Name" />
  <PropertyDef Name="Description" DataType="text" DisplayName="Description" />
  <PropertyDef Name="Title" DataType="text" DisplayName="Title" />
  <PropertyDef Name="Author" DataType="text" DisplayName="Author" />
  <PropertyDef Name="DocSubject" DataType="text" DisplayName="Subject" />
  <PropertyDef Name="DocKeywords" DataType="text" DisplayName="Keywords" />
  <PropertyDef Name="DocComments" DataType="text" DisplayName="Comments" />
  <PropertyDef Name="CreatedBy" DataType="text" DisplayName="Created By" />
  <PropertyDef Name="ModifiedBy" DataType="text" DisplayName="Last Modified By" />
</PropertyDefs>
<ResultTypes>
  <ResultType DisplayName="All Results" Name="default">
    <KeywordQuery />
    <PropertyRef Name="Author" />
    <PropertyRef Name="Description" />
    <PropertyRef Name="FileName" />
    <PropertyRef Name="Size" />
  </ResultType>
</ResultTypes>

```

FIGURE 6-24

8. Populate the new `<PropertyDef />` element with the following attributes:
 - `Name` — This attribute must match the exact name of the managed property.
 - `DataType` — This attribute must match the data type of the managed property.
 - `DisplayName` — This attribute should be the display name of the property, as it should be displayed in the (Pick Property) dropdown box.

9. In the <ResultTypes /> element, locate a result type with a DisplayName attribute of “All Results” and a Name attribute of “default.” Insert a new <PropertyRef /> element just after the <KeywordQuery /> element.
10. Populate the new <PropertyRef /> element with a Name attribute that matches the exact name of the managed property (see Figure 6-25).



```

Untitled - Notepad
File Edit Format View Help
<Language LangRef="de" />
<Language LangRef="ja" />
<Language LangRef="zh-cn" />
<Language LangRef="es" />
<Language LangRef="zh-tw" />
</Languages>
<PropertyDefs>
<PropertyDef Name="CountryCode" DataType="text" DisplayName="CountryCode" />
<PropertyDef Name="Path" DataType="text" DisplayName="URL" />
<PropertyDef Name="Size" DataType="integer" DisplayName="Size (bytes)" />
<PropertyDef Name="write" DataType="datetime" DisplayName="Last Modified Date" />
<PropertyDef Name="FileName" DataType="text" DisplayName="Name" />
<PropertyDef Name="Description" DataType="text" DisplayName="Description" />
<PropertyDef Name="Title" DataType="text" DisplayName="Title" />
<PropertyDef Name="Author" DataType="text" DisplayName="Author" />
<PropertyDef Name="DocSubject" DataType="text" DisplayName="Subject" />
<PropertyDef Name="DocKeywords" DataType="text" DisplayName="Keywords" />
<PropertyDef Name="DocComments" DataType="text" DisplayName="Comments" />
<PropertyDef Name="CreatedBy" DataType="text" DisplayName="Created By" />
<PropertyDef Name="ModifiedBy" DataType="text" DisplayName="Last Modified By" />
</PropertyDefs>
<ResultTypes>
<ResultType DisplayName="All Results" Name="default">
<KeywordQuery />
<PropertyRef Name="CountryCode" />
<PropertyRef Name="Author" />
<PropertyRef Name="Description" />
<PropertyRef Name="FileName" />
<PropertyRef Name="Size" />
<PropertyRef Name="Path" />
<PropertyRef Name="write" />
<PropertyRef Name="CreatedBy" />
<PropertyRef Name="ModifiedBy" />
<PropertyRef Name="CountryCode" />
</ResultType>

```

FIGURE 6-25

11. With the new property references added to the property’s XML, click the OK button on the text editor dialog.
12. Click the OK button on the Advanced Search Box Web Part configuration pane.
13. On the Advanced Search page, click the Publish tab ⇔ Publish menu button. Enter comments if desired and click the Continue button.
14. Notice that the (Pick Property) dropdown box now has an option for CountryCode in the property list. Populate a known value in the Add property restrictions search box and click the Search button (see Figure 6-26).
15. Notice that a new query text has emerged and appropriate search results have been returned (see Figure 6-27).

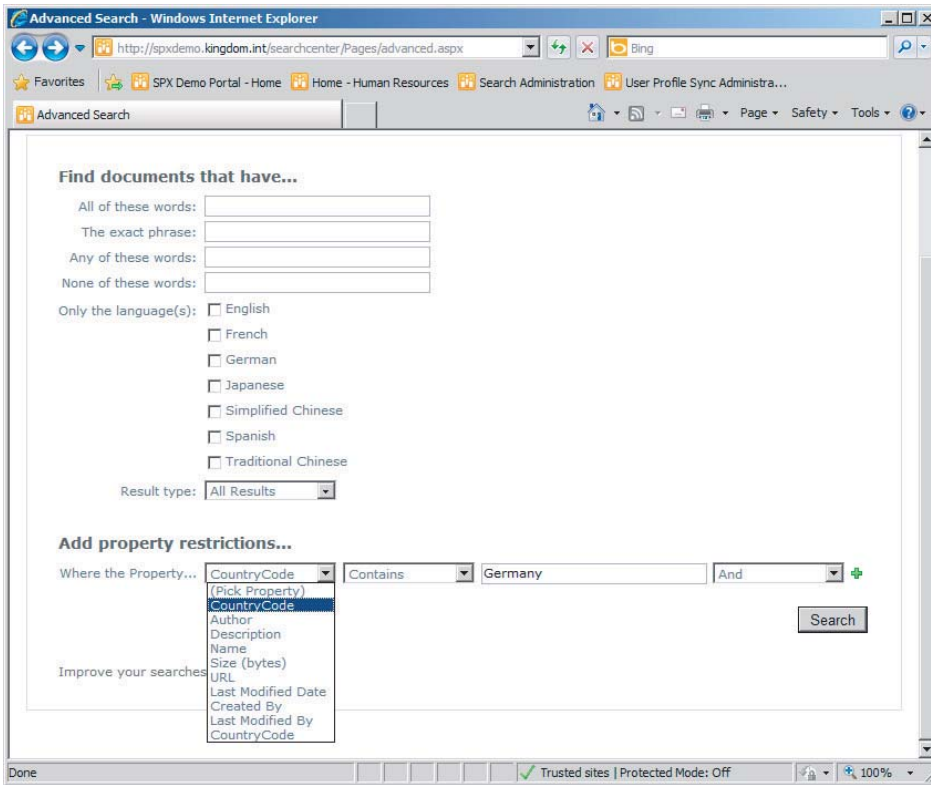


FIGURE 6-26

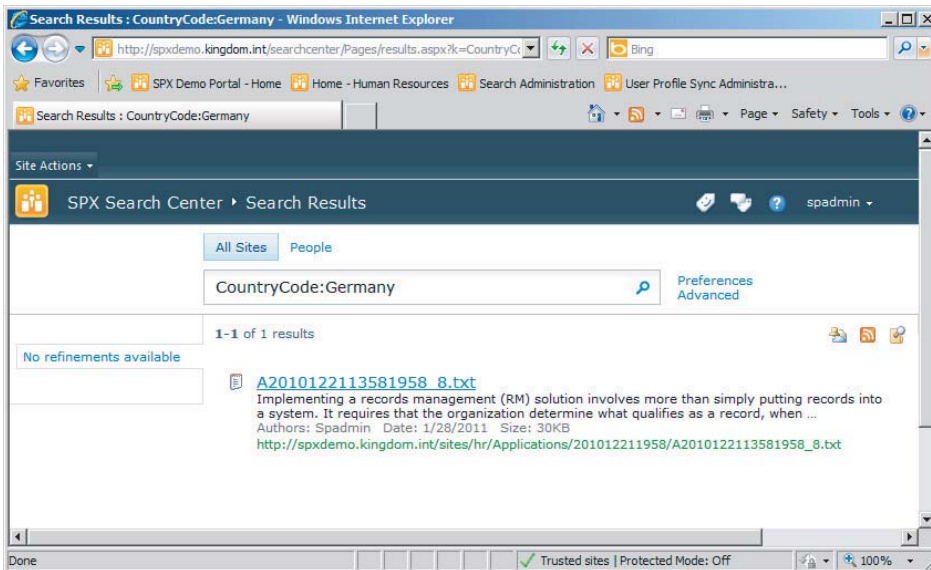


FIGURE 6-27

Calling the Search API

There are several ways to automate query execution in SharePoint 2010. The simplest way is to take advantage of the search center by using a browser to launch a query results page that has query string parameters already provided. It is possible to execute both keyword- and property-based searches in this way. It is even possible to specify a search scope as a parameter. In order to directly link to a search results page, an automation tool can launch a browser instance using the URL of the `results.aspx` page. In order to automatically execute the desired search, provide a keyword value for the “k” parameter and a scope value for the “s” parameter (see Figure 6-28).

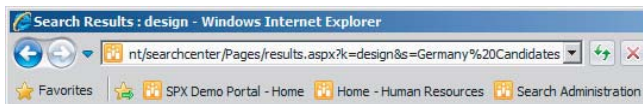


FIGURE 6-28

Another way to execute queries programmatically is via the SharePoint Server 2010 Search web service. Using the `QueryService` class, it is possible to query the search subsystem for properties and search scopes that can be used in subsequent queries. Queries are then assembled using XML query packets. Executing the query yields either an XML or DataSet result that can be consumed by the calling application. Credentials are also provided by the calling application, which facilitates proper security trimming in the search results. This technique is an excellent option for line-of-business solutions that are external to SharePoint.

The final method for executing API based queries is to code directly against the SharePoint Enterprise Search and FAST Search Server for SharePoint class libraries. This method is by far the most powerful and most flexible way to interact with the search subsystem at a lower level; but with flexibility comes complexity. Third-party software vendors and advanced development shops can dig deeply into this API layer and develop powerful content and people search solutions for ECM solutions developed on the SharePoint platform.

FAST Search for SharePoint 2010

After the acquisition of the Fast Search & Transfer company, Microsoft quickly merged the tremendously talented staff and their knowledge into the Enterprise Search group. The result was a well-executed integration that paid immediate dividends in the SharePoint 2010 search architecture. The architecture of the Enterprise Search subsystem has clearly been influenced by the design of FAST. The result is a significant improvement in scalability, as well as full support for fault tolerance, which was a highly demanded feature.

Even with influences from the FAST architecture, there are clear advantages to FAST Search for SharePoint 2010, particularly for ECM solutions. The advanced document processing capabilities, powerful FAST Query Language (FQL) integration opportunities, and extreme architectural scaling possibilities make FAST Search for SharePoint 2010 the ideal search subsystem for large-scale ECM solutions. FAST for SharePoint 2010 can support a corpus of at least 500 million items.

Functional Overview

Although FAST Search for SharePoint 2010 is actually a brand-new product that was built from the ground up and closely integrates with SharePoint 2010, it was born of a rich FAST heritage and

its architecture is deeply rooted in proven FAST search technology. Traditional FAST features and functions are available throughout the FAST Search Server for SharePoint 2010 platform.

Advanced Content Processing

An advanced content processing pipeline is able to perform property extraction on each item. In addition to facilitating end user queries, these properties can be used to drive query refinement, define search scopes, and tune relevancy.

FAST content processing is highly extensible. Developers can write code modules that can be inserted into the content processing pipeline to customize the property extraction.

Deep Refiners

As previously mentioned, extracted metadata can drive the refinement panel for easier end user navigation. Refiners in FAST also show document counts related to each of the refinement navigation links.

Tunable Relevance and Rank Profiles

Items can be promoted and demoted based on metadata properties. This feature is used to ensure that relevant documents receive a higher rank in the query results and less relevant documents are given a lower rank.

FAST Query Language Search

FAST Query Language (FQL) syntax can be used by developers to execute precise queries. In addition to the ability to execute specific queries based on keywords or metadata properties, FQL facilitates advanced proximity queries and queries that can control ranking at query time. FQL also allows certain query operators to be nested to ensure the proper execution order of the operators.

Extreme Scale

While SharePoint 2010 Enterprise Search solutions should be limited to a corpus size of 100 million documents, FAST Search for SharePoint 2010 will scale to 500 million items and possibly more than that! The same search architecture technology that enables FAST ESP to scale to billions of documents has been baked into FAST Search for SharePoint 2010.

FAST Search for SharePoint 2010 operates on the concept of *index columns*, which are similar to SharePoint Enterprise Search index partitions. *Indexer rows* populate the index and *search rows* perform query matching using the index (see Figure 6-29). This enables the index to be extremely scalable.

FAST Search for SharePoint 2010 can function in a normal density mode that facilitates a higher number of queries per second (QPS) as compared to an implementation configured for high-density mode. In normal density mode, each index/search node will contain approximately 15 million items. In high density mode, each index/search node can contain approximately 40 million items, but QPS will be lower.

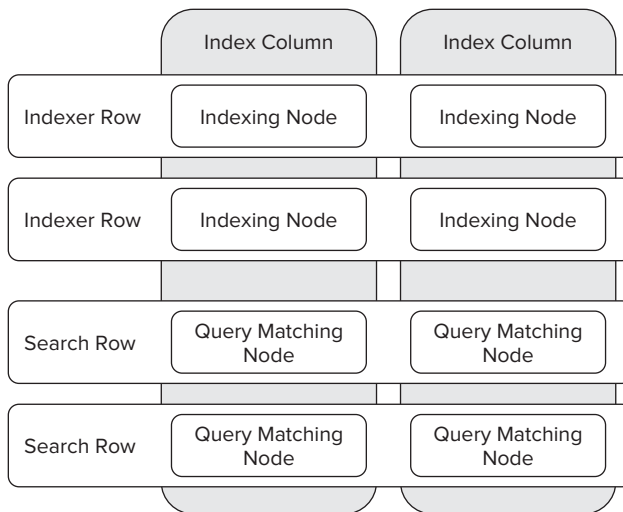


FIGURE 6-29

Crawl processing for a SharePoint 2010 web application (hostname) can be serviced by one, and only one, crawl database. It is not possible to gather all the web application items that need to be crawled and divide them among multiple crawl databases. The recommended limit of items in a SharePoint 2010 Enterprise Search crawl database is 25 million. Due to the relationship between web applications and crawl databases, a single web application is therefore also limited to 25 million items. When FAST for SharePoint 2010 is deployed as the search subsystem, this limit is much higher. Because most of the index and query processing happens in the FAST server farm, the Content SSA crawl database can contain at least 50 million items. Thus, the web application item limit grows to 50 million items when FAST for SharePoint 2010 is deployed as the search subsystem.

Index and Query Processing Path

FAST for SharePoint 2010 is able to crawl all of the same content that SharePoint 2010 can crawl plus a bit more. A Content Search Service Application (Content SSA), also called the FAST Search Connector, gathers traditional SharePoint, Exchange, BCS, and file share content and feeds it to FAST for processing. Using non-SharePoint connectors, FAST can crawl standard websites as well as Java Database Connectivity (JDBC) and Lotus Notes content sources. The Content SSA operates in the context of the SharePoint Enterprise Search subsystem before content is passed to the FAST farm for processing. Content items are passed through the document processing pipeline for property extraction and then passed through the Indexing Dispatcher to an indexer that indexes the document and populates the content index (see Figure 6-30).

A second Search Service Application (Query SSA) accepts user queries. People search queries are handled directly by the Query SSA but content queries are handed off to Query Processing in the FAST farm. Query Processing performs query parsing and linguistic processing on the inbound query. It also interacts with the FAST Search Authorization component to add additional filters to

the query that security trim results based on user and group membership information in the querying user's claims-based authentication token. The query is then forwarded to Query Matching processes that retrieve items matching the query from each index. Query Processing merges the Query Matching results into a single result set. Finally, the results are passed through the Query SSA back to the user.

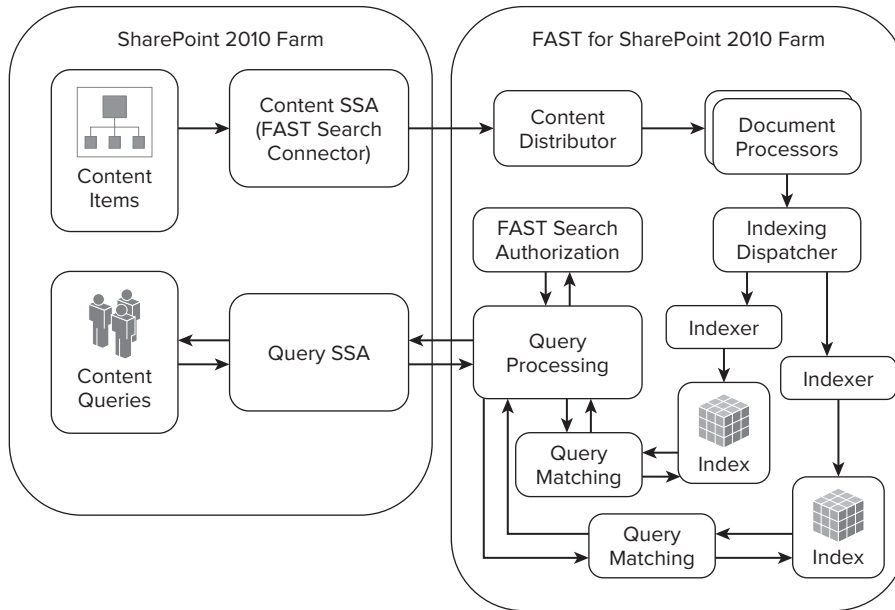


FIGURE 6-30

SEARCH ARCHITECTURES FOR SHAREPOINT ECM

Designing an adequate SharePoint 2010 Enterprise Search architecture without overspending on required hardware is a delicate balancing act. It can be done as long as the architect understands the size and shape of the corpus (the corpus profile) as well as the queries per second (QPS) requirements of the business. Armed with these requirements, an architect can properly estimate the initial hardware requirements for the search topology. Once the topology is implemented, and as content is continuously loaded, monitoring is the key to ensuring that the QPS target is met.

When designing the search topology, a few key concepts drive hardware requirements. First, the larger the corpus, the higher the crawl server and crawl database server requirements will be. Second, the higher the QPS requirement, the higher the query server requirements will be. While these requirements may seem obvious, in some use scenarios there will be a generally small corpus but a high number of users; or, conversely, a very large corpus but a small number of users.

Consider a scenario in which a medium-size organization uses SharePoint as the document management system for a workflow-driven accounts receivable department. While they will have a rapidly growing corpus of invoices, purchase orders, and receipts, they may only have a few accounts receivable processing employees. In this case, the corpus could grow to 10 or 20 million documents in just

a few years but the QPS requirement will be relatively low. Index freshness is also important because workflow processes may include search operations for related documents. In this case, it would make sense to dedicate available search servers to crawl operations. The SharePoint Web Front End (WFE) servers should be more than sufficient to also function as query servers for this small number of users.

Now consider a scenario in which an organization of 5,000 employees uses SharePoint primarily as a collaboration portal. User adoption is good and item content is regularly added and updated. Even at a high rate of 2,000 new documents per day, that only results in 504,000 new documents per year. After four years there are just over 2 million documents in the corpus, but users have become comfortable with SharePoint and frequently use it, as intended, to locate previously contributed items. QPS could be as high as 50 and all queries need to be processed in two seconds or less. In this case, even though a single index partition on a single query server can handle up to 10 million documents, dedicating an additional query server to query processing will add redundancy and ensure that query throughput and latency requirements are met.

These two scenarios are two sides of a balancing act for which hardware requirements are the fulcrum (see Figure 6-31). Given a specific set of available servers, this diagram illustrates how to best utilize the resources of the farm. For example, given a farm topology of one web front-end server and two application servers, the following statements apply:

- Shifting the application servers to function as crawl servers would allow the search subsystem to support a corpus of 20 million items with a relatively fresh index at the cost of a low QPS and lack of index partition redundancy.
- Shifting one of the free application servers to function as a crawl server and the other free application server as a query server, while also promoting the web front end to be a query server, would allow the search subsystem to support a corpus of 10 to 20 million items.

In this solution, QPS would be higher and the index partition would be fault tolerant, but the crawl server may be overworked, resulting in a less fresh index. But what if the solution must support a corpus of 20 million items, a relatively fresh index, and a reasonable QPS? In this case, the hardware resources must increase. The fulcrum concept can be applied to virtually any search topology paradigm.

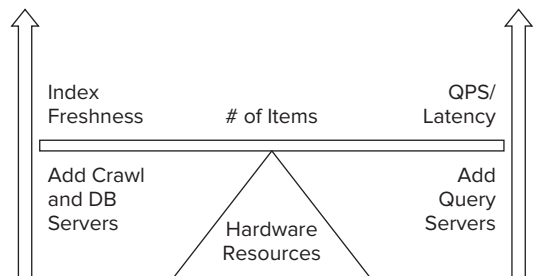


FIGURE 6-31

Sample Architectures

The goal is always to balance index freshness and query throughput requirements for a given corpus size and profile with reasonable hardware requirements. The sample architectures provided in this section are indeed just samples, but the concepts will help with the design of a search topology based on predicted size estimates. It is important to understand that while sample architectures are a good place to start, they will never be a perfect fit. The fulcrum example illustrates the variable nature of every environment. No amount of prediction and estimation will ever be more accurate than performance testing and monitoring.

3-Million-Item Corpus

These days, 3 million documents is a walk in the park for SharePoint 2010. The trick at this level is that hardware and software budgets may be limited. Often, while the solution is mission critical, it can tolerate some downtime if it means purchasing fewer servers and software licenses. While it may be possible to force some fault tolerance, it may come at the cost of performance.

The solution shown in Figure 6-32 is essentially the smallest recommended solution for a SharePoint “farm” that could handle 3 million items. It is likely that a single-server solution could contain 3 million items but it would come at a cost of poor performance. In this solution, the application server will be spending CPU cycles by crawling content. The index partition will be hosted on the WFE, which also serves query results. The database server will contain all content databases, a single crawl database, and a single property database, along with all other SharePoint and service application databases. There is no fault tolerance, but index freshness should be good and query performance should be reasonable. In Figure 6-32, the diagram is meant to convey the concept that the hardware provided strikes a balance between Index Freshness and QPS/Latency while providing support for a specific corpus capacity. As larger architectures are addressed later in this chapter, the concept of the relationship of available hardware to the corpus size supported will also continue to evolve.

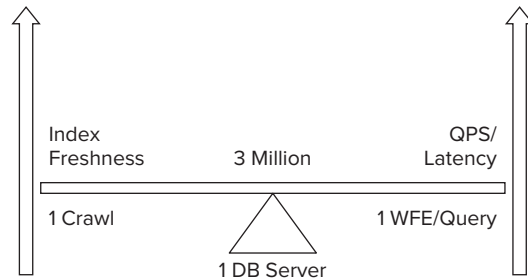


FIGURE 6-32

10-Million-Item Corpus

A topology that would support 10 million items would consist of at least four servers (see Figure 6-33). Notice that the crawl server facilitates better index freshness and the 2 WFE/Query servers facilitate a higher QPS and lower query latency. With four servers, performance is nicely balanced and the additional WFE/Query server provides fault tolerance for the WFE/Query role and thus the index partition. Fault tolerance does not yet exist for the crawl server but because crawlers are stateless, if the crawl server fails, it will be relatively easy to recover by building a new (or temporary) server to serve in the crawler role. As the corpus approaches 10 million, adding a second index partition and planning for an additional crawl server and crawl database server would be advisable.

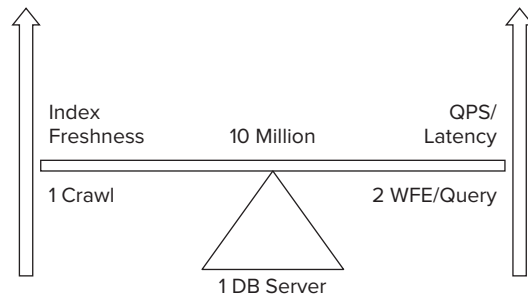


FIGURE 6-33

40-Million-Item Corpus

At 40 million items, the topology supports fault-tolerant WFEs, query servers, and crawlers. Any one of these servers can go down and the farm will still function. Reasonable index freshness and query throughput/latency can be expected (see Figure 6-34).

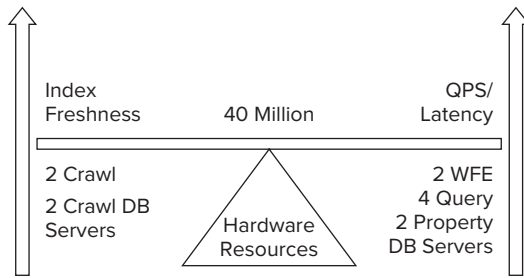


FIGURE 6-34

At 40 million items, the corpus profile and query use cases can have a significant impact on search subsystem topology. For example, if the bulk of the corpus consists of file types that are not associated with an available IFILTER, such as scanned TIF images, then indexed items per second will trend higher than normal. This is because only the metadata properties need to be extracted for these items. Crawler processes will have little else to do because, by default, they won't be able to OCR the images in order to extract keyword data. Conversely, if the corpus profile consists mostly of Microsoft Office–type documents or searchable PDF documents, then full crawl operations may take longer than SLA requirements allow. In this case, it may be necessary to add an additional crawl database server and crawl server to improve indexed items per second and index freshness. Conceptually speaking, the heavy index freshness requirements might cause the hardware resources fulcrum to shift toward adding additional resources to service Index Freshness requirements (see Figure 6-35).

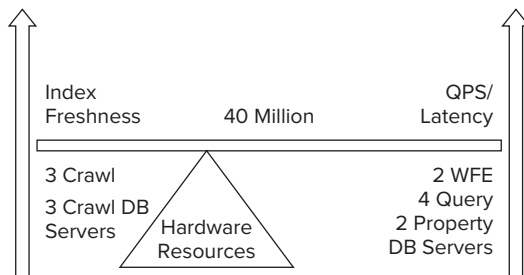


FIGURE 6-35

If the 40 million item corpus profile consisted of primarily static, archived documents that are rarely searched, the query servers may experience relatively light load levels. Conversely, if the corpus profile consists of mostly searchable documents that are queried frequently by a large number of users, the query servers may begin to experience high resource utilization levels, which will increase query latency and cause end user frustration. Also, as the corpus threatens to push beyond 40 million documents, query latency can also increase. In both of these situations, adding an additional query server will reduce query latency. Conceptually speaking, the increased query latency pressure might cause the hardware resources fulcrum to shift toward adding additional resources to service the QPS/Query Latency requirements (see Figure 6-36).

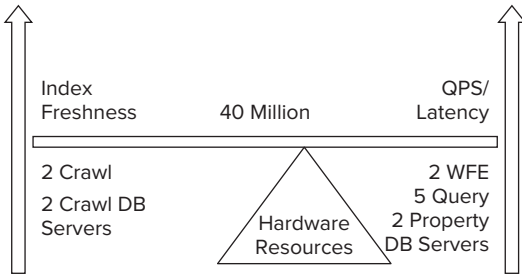


FIGURE 6-36

100-Million-Item Corpus

At the time this book was written, Microsoft says that SharePoint 2010 Enterprise Search can handle at least a 100 million item corpus; but hardware technology is constantly evolving and software service packs regularly improve performance. Therefore, depending on the corpus profile, it may be possible to one day squeeze more than 100 million items into Enterprise Search. Currently, as the corpus approaches 100 million items, it is time to begin thinking about FAST Search for SharePoint 2010; but if 100 million items is the targeted ceiling, SharePoint search topology can be architected to handle it.

Microsoft suggests a 10 million item recommended maximum per index partition and a 25 million item recommended maximum number of items in a crawl database, so a minimum of 10 index partitions is necessary, serviced by 10 query servers. A minimum of four crawl databases is required. If content is not evenly distributed across four web applications, then an additional crawl database may be necessary. Microsoft recommends that no more than two crawl databases be hosted on a single crawl database server, so a minimum of two crawl database servers is required. However, in order to meet indexed item per second and index freshness requirements, as many as three or possibly even four crawl database servers may be necessary, depending on how well resourced the crawl database servers are provisioned. Taking advantage of those four-plus crawl databases will mean a minimum of four crawl servers, each with two crawlers. At this high level of topology architecture, predicting how additional resources will affect index freshness or query latency is not really a valid exercise, so the example provided is certainly just a starting point (see Figure 6-37). With a projected ceiling of 100 million documents, the solution simply won't be successful without proper testing and monitoring practices.

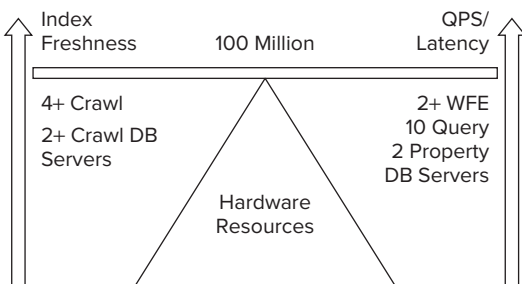


FIGURE 6-37

500 Million Documents

Microsoft claims that a FAST for SharePoint 2010 farm can support a corpus of at least 500 million items. At this level, it would not be advisable to use sample architecture from a book to define a server topology. That said, for the sake of a thought experiment, it is interesting to contemplate the magnitude of the hardware required to store and process this volume of content.

In high-density mode, FAST for SharePoint 2010 supports 40 million items per index column, so an absolute minimum of 13 index columns would be needed just to contain the entire index. Deploying two rows per column ensures that redundancy is built into the entire index. Therefore, with 13 columns and two rows, a minimum of 26 servers would be necessary for a fault-tolerant index. In addition to that, five servers are needed to fulfill admin, content distribution, web analyzer, and document processor roles. On the SharePoint side of the house, two crawl database servers, 10 crawl servers, and a minimum of two WFEs to facilitate inbound queries would be necessary. Adding it all up results in a grand total of 45 servers.

If it becomes necessary to increase the indexed items per second or queries per second, or to reduce query latency, the server count increases from there. In addition, assuming an average document size of 512KB, it's going to take approximately 238TB of storage space to contain 500 million documents, and 126TB of storage space for all crawl, index, and query resources. The bottom line boils down to a few simple statements. One, it is possible to architect SharePoint to support a corpus of 500 million documents. Two, thanks to FAST for SharePoint 2010, it is possible to index that content and return search results in a reasonable amount of time. Finally, it is going to be quite an expensive undertaking.

The Impact of Virtualization

Conventional wisdom suggests that hosting a virtualized server on a host that has appropriate resources available for the new server will result in performance that is degraded approximately 10% to 20% compared to the same server deployed on bare metal. Fortunately, SharePoint 2010 is extremely flexible when it comes to topology requirements.

In most cases, a virtual server will be limited to four virtual processor cores. This may not be the optimal configuration for some servers in the SharePoint farm. For example, the SQL Server and crawl servers will definitely benefit by having more than four CPU cores. The good news is that SharePoint 2010 allows for a very flexible topology.

The samples in this section are for an approximate item count. When virtual servers are involved in the solution, it is best to estimate downward a bit for these numbers. While it is impossible to provide specific guidance, it should be possible to estimate the hardware upward by a server or two in the strategic topology roles to offset the decreased performance of virtualization.

Tuning Search Performance

For many large organizations for which SharePoint is a mission-critical application, search relevancy tuning and performance monitoring often warrants a full-time position. It would be unfortunate if an organization spent the money to architect and implement a high-quality search solution and then didn't maintain that investment by provisioning for regular monitoring and tuning. The concepts

presented in this section discuss important techniques for ensuring that the search system remains healthy over the long term.

Health Monitoring

The SharePoint 2010 Health Analyzer does provide a couple of rule definitions that help ensure optimal property and crawl database performance, but that is about the extent of the rules that are defined specifically for the search subsystem. A better option for much more comprehensive health monitoring is to use System Center Operations Manager (SCOM) 2007.

With the SharePoint 2010 Products management pack, SCOM is able to detect and diagnose issues identified by agents installed on various servers in the SharePoint farm. With this management pack, SCOM is able to do the following:

- Check whether the search service is running
- Search database out-of-space errors
- Search indexer failure
- Query component mirror failure
- Query component failure
- Identify low disk space on query or crawl components
- Query crawl propagation errors
- Query index error
- Search crawler disk full warnings
- Identify host unavailable errors

Together these monitors ensure that the search subsystem is healthy and operational.

Performance Monitoring

As mentioned throughout this chapter, there are a few key metrics for the search subsystem:

- Indexed items per second (crawl rate)
- Queries per second (throughput)
- Query latency (the amount of time it takes the search subsystem to return query results when an end user executes a query)

The best way to execute performance testing on SharePoint 2010 and the search subsystem is to use Visual Studio 2010. From within Visual Studio 2010 it is possible to record test operations, modify the tests to use random data (for query execution), and finally generate and execute load tests. When executing the tests, additional metrics that track CPU, memory, and disk I/O utilization for each

server in the farm can be recorded as well. Armed with this information, a SharePoint administrator can identify any performance bottlenecks and take appropriate action.

Improving Crawl Performance

Poor crawl performance presents itself in the form of low indexed items per second and results in stale query results. Table 6-2 identifies several bottlenecks that inhibit crawl performance, and the solutions to relieve the bottleneck.

TABLE 6-2: Crawl Bottlenecks and Solutions

| BOTTLENECK | SOLUTION |
|---|---|
| Crawl database server disk I/O is queuing requests | Add an additional crawl database stored on a separate physical disk that provides 3,500 to 7,000 IOPS. Then assign new crawlers to the new crawl database and add a host distribution rule to facilitate crawl item balancing. |
| Crawl database server CPU frequently reaches or maintains 100% utilization | Add an additional crawl database server and an additional crawl database stored on a separate physical disk that provides 3,500 to 7,000 IOPS. Then assign new crawlers to the new crawl database and add a host distribution rule to facilitate crawl item balancing. |
| Crawl database server is managing the crawl processing of more than 25 million items and database performance is poor | If the crawl database server is hosting only one crawl database server, add an additional crawl database stored on a separate physical disk that provides 3,500 to 7,000 IOPS. If the crawl database server is hosting two crawl databases, then add an additional crawl database server and an additional crawl database stored on a separate physical disk that provides 3,500 to 7,000 IOPS. After the new crawl database is provisioned, assign new crawlers to the new crawl database and add a host distribution rule to facilitate crawl item balancing. |
| Crawl server CPU frequently reaches or maintains 100% utilization | Crawl components can utilize four CPU cores. Add an additional CPU to the server if an open socket is available or add an additional crawl server to the farm and move one or more crawl components to the new server. |

Improving Query Performance

Poor query performance presents itself in the form of low query throughput. Ideally, when an end user executes a query, results should be returned in less than one second. If this is routinely not the case, then queries per second will suffer and end users will become frustrated. Table 6-3 identifies several bottlenecks that inhibit query performance, as well as their solution.

TABLE 6-3: Query Bottlenecks and Solutions

| BOTTLENECK | SOLUTION |
|---|---|
| One or more index partitions contains more than 10 million documents. | Add an additional index partition and, if possible, an additional index partition mirror. If all query servers already contain an active and a mirrored index partition, then add one or more additional query servers. Then add an additional index partition and, if possible, an additional index partition mirror. |
| One or more query servers are memory bound and/or paging virtual memory on disk. | Add additional memory to the query server. Ensure that the query server has enough RAM to store 33% of <i>each</i> index partition (present on the query server) in memory. |
| Query performance suffers during the first few queries after the server is rebooted or during crawl processing and index propagation. Disk I/O on the query server is queuing requests. | Ensure that the physical disk where the index partition is stored is capable of providing 2,000 IOPS for <i>each</i> index partition. |
| Query latency is high but disk I/O, memory, and CPU resources are available on all query servers. | Ensure that the property database server has enough RAM available to store 33% of the property store tables in memory. Ensure that the property database server is not CPU or disk I/O bound. If necessary, add an additional property database server and an additional property database. Then add an additional index partition and, if possible, an index partition mirror. |

SUMMARY

In this chapter, the search subsystem was identified as a very important component of SharePoint 2010 because it is central to the end user experience. If queries process slowly or yield unpredictable or stale results, users will lose trust in the search system and user adoption of the solution will suffer. To avoid this possibility, it is important to properly design and deploy the search topology according to real business requirements with respect to the corpus profile. Whether the search solution is provided by SharePoint 2010 Enterprise Search or FAST Search for SharePoint 2010, it is vital to monitor and tune the search system to ensure that the search solution performs well as it matures.

7

Web Content Management

WHAT'S IN THIS CHAPTER?

- Understanding how the foundational features of SharePoint 2010 enable web content management solutions
- Learning how content is related to rendered web pages in a user's browser
- Knowing page components and SharePoint reusability

WCM OVERVIEW

Web content management describes the tools and processes that facilitate the overall life cycle of web content. Web content can be something as obvious as HTML and CSS, or richer content such as videos and Silverlight files. Managing web content is, in many respects, similar to managing other content such as documents. For example, many types of content need to go through approval and disposition processes. In addition, many different types of content require check-in/check-out and versioning functionality. Although there are many similarities between managing web content and documents, there are differences as well.

Whereas documents are generally self-contained, web content is spread out across many discrete files. In addition, web pages themselves often require a fixed or semi-fixed format in order to adhere to design standards. By taking advantage of both the core functionality provided in SharePoint 2010 and the WCM-specific functionality it offers, you can use SharePoint to create a fully functional WCM system.

More often than not, WCM in the context of SharePoint is thought of as a system for public-facing websites. However, this is not necessarily the case. The WCM features of SharePoint

provide the capability to enable controlled authoring and publishing in an intranet scenario as well. This could be for an internal news site, which employees visit to learn what is happening in the organization, or even an enterprise wiki where large numbers of people can collaboratively develop content for consumption by the masses.

Improvements in 2010

Before diving deeply into the core areas of WCM in SharePoint, it is worthwhile to briefly look at the major improvements available in the 2010 version of the product. These features are also covered in more detail in their respective areas of discussion throughout the chapter.

Authoring

Just in the area of authoring alone, there are several very important improvements over SharePoint 2007. First, and what many people probably think of first when considering authoring, the “what you see is what you get” (WYSIWYG) editor is vastly improved. SharePoint 2010 includes an out-of-the-box editor that generates clean XHTML, more like what a designer might craft by hand, and is cross-browser compatible.

In addition, page creation and management is much easier. For example, you can create a page with one click now, and only one field is required, the title. Authors can also easily change the page layout for an already created page on-the-fly using the Ribbon UI.

Speaking of the Ribbon UI, this base platform improvement is taken advantage of in a big way in the area of SharePoint WCM, especially in terms of authoring. Just as with the Ribbon (or fluent UI) in the Office suite of applications, features are presented to users in a context-sensitive manner, by displaying the most important options only when they are needed, thereby improving feature discovery and user efficiency.

AJAX

This improvement is related to authoring but it’s slightly broader than that. While not a specific WCM feature per se, and really just a technology, the use of AJAX and other client-side code (read JavaScript) greatly improves the user’s experience when navigating around SharePoint and performing tasks — from site administration to content authoring. To be clear, mentioning AJAX here is specifically related to the out-of-the-box SharePoint UI (e.g., the Ribbon, dialogs, etc.), not necessarily what users of a public-facing site or otherwise would experience. Developers of sites implemented using the SharePoint WCM features still need to make their own custom features leverage AJAX technologies if this is a requirement.

Accessibility

Investments focusing on accessibility were made by Microsoft in this version of the product, enabling SharePoint 2010 to qualify for the AA level of the Web Content Accessibility Guidelines (WCAG) 2.0. According to the W3C’s website, the WCAG guidelines “make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity, and combinations of these.”



For more information on the WCAG guidelines and the associated levels, please visit the recommendations page at www.w3.org/TR/WCAG20.

Markup Standards

Piggybacking on the accessibility standards just mentioned, the SharePoint team also placed an emphasis on rendering standard markup — namely, the XHTML standard. XHTML requires certain conventions such as tags that are always closed (including self-closing tags), quoted attribute values, and doctype declarations (among many other things).

Content Query Web Part

The Content Query web part was an essential tool when implementing WCM features in SharePoint 2007, and that is still the case in SharePoint 2010. This web part is indispensable, as its sole purpose in life is to query, or roll up, content and display the results in a visually effective manner on a page. For example, the front page of a website might require that the five most recent articles be displayed in the right navigation area.

In SharePoint 2010, there are some major upgrades to this web part's functionality. The first area of improvement is related to querying data. The web part is now capable of using values in the URL's query string or the current page's metadata to query for content. Another new feature is related to displaying content found with a query. When configuring the web part's properties, you can now directly type in the fields desired for rendering purposes, rather than muck around with XSLT. This is a welcome change for users who are not comfortable dealing with something as technical as XSLT.

Cross-browser Support

In SharePoint 2010, more effort was placed on supporting multiple browsers. In addition to supporting Internet Explorer versions 7 and 8 (and soon IE 9), 2010 natively supports Firefox 3.6 and Safari 4.04. Cross-browser support is certainly a welcome feature as SharePoint adoption continues to grow.

Rich Media

Given that rich media online has exploded over the past several years, it only makes sense that Microsoft make investments in this area. The two most notable areas of improvement in rich media are the Asset Library and the customizable Silverlight media player. For more information on digital assets and rich media, see Chapter 9 of this book.

Metadata

As one of the main areas of improvement for SharePoint 2010 in general, metadata support carries over into the WCM feature set in particular. Managed metadata enables a set of consistent terms to be applied to content across an entire SharePoint farm or even across multiple farms. Consequently, web content can be tagged with terms from specific term sets, which are consistent across the entire WCM site or set of sites. This type of consistent, managed tagging allows for a variety of search and

navigation solutions. Users can navigate using the existing site structure or content can be sliced and diced based on the metadata applied to it. Managed metadata is also covered in depth in Chapter 3.

Spectrum of WCM in 2010

When most people consider SharePoint WCM, they automatically think of public-facing or .com websites. This is fair perception, as public-facing websites are the most common use for SharePoint WCM. Many high-profile companies and organizations have taken the SharePoint plunge when representing themselves on the public Internet.

However, the WCM features in SharePoint support a spectrum of usage scenarios, with public-facing websites being just one point on this spectrum. Another point is an internal-facing company intranet. If you think about it, a lot of the goals of a public-facing site spill over into a company's employee-only intranet. Both site types are typically meant to educate an audience, both can offer self-service functions, and both generally require content to be fresh and discoverable, as well as authored by a select few for a wide range of readers. Although an intranet and a public-facing website are definitely not the same thing, the same functionality and infrastructure can be leveraged to support both. In addition, a site of any size can be placed in this category; it does not have to be a large company-wide intranet. Divisional or departmental portals can also be supported using the WCM features of SharePoint.

So far, the site types discussed, whether internal or external, are characterized by being tightly controlled in terms of authoring, content available, and structure. Again, these sites generally have very few authors and a relatively much larger audience as well. Another type of site supported by the SharePoint WCM features is a *wiki*. Most readers are likely familiar with the concept of wikis thanks to the wildly popular and revolutionary site Wikipedia. Compared to the other WCM site types that have already been discussed, wikis are much looser in nature with regard to controls and what content is available when. Wikis also generally have a much more balanced author-to-reader ratio.

Wikis typically allow “anyone” (of course, anyone doesn't always mean anyone — it means whatever is appropriate in a given scenario) to author and/or edit content. The type of content available on a wiki can really be anything, but most wiki content is meant to preserve knowledge for future use. Examples include best practices within an organization or department, jargon that new employees might need to know, system documentation in the form of tips and tricks, and other informational content. Even though wikis are drastically different in nature from public-facing websites or corporate intranets, there is enough overlap of functionality that the WCM infrastructure supports all these capabilities.

THE SHAREPOINT SERVER PUBLISHING INFRASTRUCTURE

The following sections cover the infrastructure that supports web content management in SharePoint 2010. Without the features described in this section, WCM in SharePoint would not exist!

Templates

To support creating new site collections and sites that use the SharePoint WCM features, several site templates are included out the box; these templates are described in Table 7-1. The qualities that make these templates “publishing” templates are related to their intended usage as well as the technology that is available within provisioned sites. This includes, but is not limited to, activated

SharePoint features (WCM features are discussed in the next section), default permissions, lists/libraries provisioned by default, other default settings, and so on.

TABLE 7-1: Site Templates for WCM

| SITE TEMPLATE | DESCRIPTION |
|-------------------------------|--|
| Publishing Portal | This template is designed to provide a starting point for a public-facing Internet site or a larger or more formal corporate intranet. It includes a sample structure, including press releases and search. It also enforces content publishing with workflow out of the box. In addition, this template supports anonymous users, who are restricted from viewing SharePoint application pages. |
| Publishing Site | This template includes some core functionality for publishing web content (pages and images). It does not use workflow out of the box in order to approve content for publication. Rather, it utilizes drafts and major versions to show content to contributors and viewers, respectively. |
| Publishing Site with Workflow | This is similar to the Publishing Site template except that the Approval workflow is used to control content publication. |
| Enterprise Wiki | This template is appropriate for creating new sites that are used to capture organizational knowledge. |

These different templates reflect the variety of purposes for which different sites are intended, enabling you to select a site template appropriate for your scenario. Obviously, if you are pursuing a knowledge capture and dissemination solution for usage internal to your organization, an enterprise wiki would be the place to start.

As for the “few authors, many readers” model of SharePoint WCM, you’ll use one of the other three site templates if you are starting with what is available out of the box. The Publishing Portal template is the only one of these three that can be used to create a new site collection, so this is a good place to start when creating your root site. Creating subsites is simply a matter of your requirements. The Publishing Site with Workflow template is a good fit for sites that require an element of distributed content approval. However, if this is too heavy for your particular project, then the Publish Site template might be a good fit. Whatever your situation, it is important to consider things like security and governance requirements when planning a site structure.

Features

Features in SharePoint can be thought of as a unit of deployment for a particular piece of functionality or part of a solution. They can do things like provision content types, create lists, add content to the site, deploy workflows, and more. SharePoint 2010 utilizes features heavily to deploy out-of-the-box functionality, and that includes the publishing functionality. This section takes an in-depth look at the SharePoint 2010 publishing features and their associated functionality.

You can see only two features in the SharePoint user interface that are specifically related to provisioning publishing functionality. The SharePoint Server Publishing Infrastructure feature is scoped

to a site collection, and the SharePoint Server Publishing feature is scoped to a site. Unsurprisingly, these two features are activated in the appropriate spots when you use the out-of-the-box site templates described in the previous section.

Although only two publishing-related features can be activated via the UI, a total of 13 features are activated when both the site and site collection features are enabled. This chain of events happens because of a feature in SharePoint features (no pun intended) called *activation dependencies*. Activation dependencies are rules that define the features which must be activated before another feature can be activated. While it is not necessary to memorize the list of features that are activated because of the main publishing features, you can glean a lot from studying these dependencies.

Table 7-2 describes the features that participate in the publishing feature activation tree. Again, the goal here isn't to memorize these features. Rather, this table provides a handy reference to the publishing features and what actually happens when they are activated.

TABLE 7-2: Publishing Feature Tree

| FEATURE ID | FEATURE FOLDER | SCOPE | SHORT DESCRIPTION |
|--------------------------------------|--------------------------|-------|--|
| F6924D36-2FA8-4f0b-B16D-06B7250180FA | Publishing Site | Site | Main, visible site collection scoped feature |
| 94c94ca6-b32f-4da9-a9e3-1f3d343d7ecb | Publishing Web | Web | Main, visible site scoped feature |
| 22A9EF51-737B-4ff2-9346-694633FE4416 | Publishing | Web | Calls PublishingFeatureHandler receiver, controls, custom actions, Ribbon UI |
| A392DA98-270B-4e85-9769-04C0FDE267AA | Publishing Prerequisites | Site | Calls PublishingPrerequisitesFeatureHandler receiver |
| AEBC918D-B20F-4a11-A1DB-9ED84D79C87E | Publishing Resources | Site | Web parts, content types, XSL files, Silverlight (for media) |
| 89E0306D-453B-4ec5-8D68-42067CDBF98E | Navigation | Site | Calls NavigationFeatureHandler receiver, publishing navigation controls/custom actions |
| D3F51BE2-38A8-4e44-BA84-940D35BE1566 | Publishing Layouts | Site | Styles, master pages, page layouts (article, splash, etc.) |
| 4BCCCD62-DCAF-46dc-A7D4-E38277EF33F4 | AssetLibrary | Site | Asset library list template and associated artifacts |

| FEATURE ID | FEATURE FOLDER | SCOPE | SHORT DESCRIPTION |
|--------------------------------------|----------------------------|-------|---|
| 068BC832-4951-11DC-8314-0800200C9A66 | Enhanced Theming | Site | Theming controls |
| A942A218-FA43-4d11-9D85-C01E3E3A37CB | Enterprise WikiLayouts | Site | Calls <code>EnterpriseWikiSiteFeatureHandler</code> receiver, wiki content types, and pages |
| 915c240e-a6cc-49b8-8b2c-0bff8b553ed3 | Ratings | Site | Calls <code>RatingsFeatureReceiver</code> receiver, ratings fields |
| 0C8A9A47-22A9-4798-82F1-00E62A96006E | Document Routing Resources | Site | Calls <code>DocumentRoutingResourcesFeatureReceiver</code> receiver, routing fields, and content type |
| 5BCCB9A4-B903-4fd1-8620-B795FA33C9BA | Record Resources | Site | Policy fields, custom actions |

Security

When the publishing features are enabled, some publishing-specific permission levels and roles are added to the site collection in which the features are activated. The permission levels that are added are as follows.

Approve Permission Level

The Approve permission level is self-explanatory. An approver is someone in charge of reviewing fresh content and eventually either approving content for publishing, rejecting content, or requesting further changes to the content in question.

In order to facilitate these goals, this permission level grants permissions such as the following (this is not an all-inclusive list, but it gives you an idea of what the Approve permission level enables):

- View items
- Approve items
- Open items
- View versions
- Create alerts (so the approver can see when new content is available)

This permission level also allows the user to create content, but it does not allow any site administration tasks to be performed.

Manage Hierarchy Permission Level

The Manage Hierarchy permission level is designed with a site manager in mind. This person can create new sites, manage which users can do what within sites, monitor site usage, and so on. This person can also create content. However, this permission level is not, by default, given the ability to approve content. That task is left to those with the Approve permission level.

Restricted Read Permission Level

This permission level allows only a specified subgroup of permissions to users. The goal of this permission level is to allow assigned users access to a specific area only. For example, perhaps you want to allow certain users to read a document in a library but not to actually navigate to the library itself. The permissions available in this level are as follows:

- View items
- Open items
- View pages
- Open

Groups

In addition to the three permission levels already discussed, six groups are added when the publishing features are enabled:

- Approvers
- Designers
- Hierarchy Managers
- Quick Deploy Users
- Restricted Readers
- Style Resource Readers

From their names, three of the preceding groups are self-explanatory as far as which permission levels are associated with which group. However, the other three are worth explaining briefly.

Designers are assigned the Design and Limited Access permission levels. The purpose of this group is to allow its members to create visual artifacts such as master pages, page layouts, and style sheets. They are also able to assign these visual artifacts to be used in a site.

Quick Deploy Users is a group that has no permission levels assigned to it by default. This group exists solely to identify the users who are allowed to schedule “quick deploy” content deployment jobs. Content deployment and quick deploy are features discussed later in this chapter.

The Style Resource Readers group is assigned only the Limited Access permission level. Its members, by default, are automatically given the Read permission in the master page gallery and the Restricted Read permission in the Style Library. Another default for this group is that all authenticated users are members. This default exists so that all users can leverage the visual artifacts necessary to actually render pages in the browser.

Content Types

Content types are obviously an important concept in SharePoint in general, especially in terms of document management (see Chapter 3 for more details on document management). However, content types play a huge role in SharePoint WCM as well. One of the most powerful aspects of SharePoint WCM is that the content itself is completely separated from any elements of look and feel. This enables content to be used from many different form factors, such as PCs and phones, and it enables a complete site redesign without necessarily “losing” any content in the process. The publishing features discussed earlier can deploy several content types that are key to the WCM features in SharePoint.

“Content” Content Types

Of all the publishing content types, if one had to be chosen as the most important, it would probably be the Page content type. This content type, which inherits (albeit indirectly) from Document, represents the core metadata that is needed by the publishing infrastructure in order to adequately capture, manage, and present web content. Table 7-3 describes the unique columns that are defined as part of this content type.

TABLE 7-3: Page Content Type Columns

| COLUMN | TYPE | DESCRIPTION |
|------------------------|--------------------------------|--|
| Comments | Multiple lines of text | A placeholder for optional comments about a page instance |
| Scheduling Start Date | Publishing Schedule Start Date | The date on which a page should “go-live” |
| Scheduling End Date | Publishing Schedule End Date | The date after which a page will no longer be available |
| Contact | Person or Group | An optional reference to a person who is responsible for a page’s content |
| Contact E-Mail Address | Single line of text | An alternative to the Contact column |
| Contact Name | Single line of text | An alternative to the Contact column |
| Contact Picture | Hyperlink or Picture | An alternative to the Contact column |
| Rollup Image | Publishing Image | An image representing a page instance. This image is meant to be small and displayed in a list format such as in search results. |
| Target Audiences | Audience Targeting | The target audiences of a particular page |

While the Page content type is very important and defines base functionality, you couldn’t really create interesting content with just this content type. Therefore, SharePoint ships with several other content types, which inherit from Page. In addition, users of SharePoint publishing are obviously

encouraged to create their own content types, to represent the types of content that are expected in a particular deployment.

The other Page content types are as follows:

- Article Page
- Redirect Page
- Welcome Page
- Enterprise Wiki Page
- Project Page

The first three pages are geared toward standard web content, while the last two are specific to wiki implementations. Article Page is probably the simplest a content type could be while still representing some kind of meaningful content. Aside from the columns inherited from Page, Table 7-4 shows its columns.

TABLE 7-4: Article Page Content Type Columns

| COLUMN | TYPE | DESCRIPTION |
|---------------|---------------------|---|
| Page Image | Publishing Image | A singular image that would exist on a page |
| Page Content | Publishing HTML | Represents the content that would exist on a page |
| Summary Links | Summary Links | Free-form links that can point to wherever the author desires |
| Byline | Single line of text | The page's author's byline |
| Article Date | Date and Time | The article's associated date |
| Image Caption | Publishing HTML | The image's (from the Page Image column) caption |

When implemented in a page instance, the Redirect Page content type simply adds a URL indicating to where the user should be redirected. The Welcome Page content type adds some columns to the Page content type's columns and represents a welcome, or landing, page. The Enterprise Wiki Page content type adds columns for content, categorization, and ratings, while the Project Page content type adds a link to a project site and a status column.

Infrastructural Content Types

Whereas the previous section discussed content types that represent content a user would likely see, this section covers content types that are more related to site structure. There are really only two content types of interest here: Page Layout and Publishing Master Page.

The Page Layout content type represents the information necessary to store and utilize a publishing page layout (page layouts are discussed later in the chapter). In order to represent a page layout, this content type contains columns for the layout's associated content type (each page layout is associated to one and only one content type inheriting from Page) and variations (variations are also discussed later in this chapter).

The Publishing Master Page content type contains columns that represent who is responsible for the master page instance in question, a preview image, comments, and compatible UI versions to distinguish which master pages are compatible with SharePoint 2007 and/or 2010.

Site Content

A good chunk of content is needed by default in a publishing site in order to support the requirements of WCM. Content in this context namely refers to lists and libraries, as well as items and documents contained within these same lists and libraries. The content in question is broken down into six main lists or libraries, as described in Table 7-5.

TABLE 7-5: Publishing Site Content

| LIST/LIBRARY TITLE | DESCRIPTION |
|------------------------------------|--|
| Pages Library | Contains draft and published web content. This library can utilize content approval and/or workflow to ensure quality content. |
| Style Library | Contains style sheets (CSS), images, and XSLT used to give a site its look and feel. In addition, it contains a file called <code>AlternateMediaPlayer.xaml</code> that is a skinable Silverlight media player. |
| Site Collection Documents Library | Contains documents that are meant to be accessible across the entire site collection. |
| Site Collection Images Library | Contains images that are meant to be accessible across the entire site collection. |
| Content and Structure Reports List | The items in this list define reports available to users in the site's Content and Structure Tool view. Figures 7-1 and 7-2 show this feature in action. |
| Reusable Content List | Houses small pieces of content that can be created, managed, and reused from this central location. For example, an address could be maintained here and referenced from many pieces of content. One update could then take place to keep all content fresh. Figure 7-3 shows this list. |

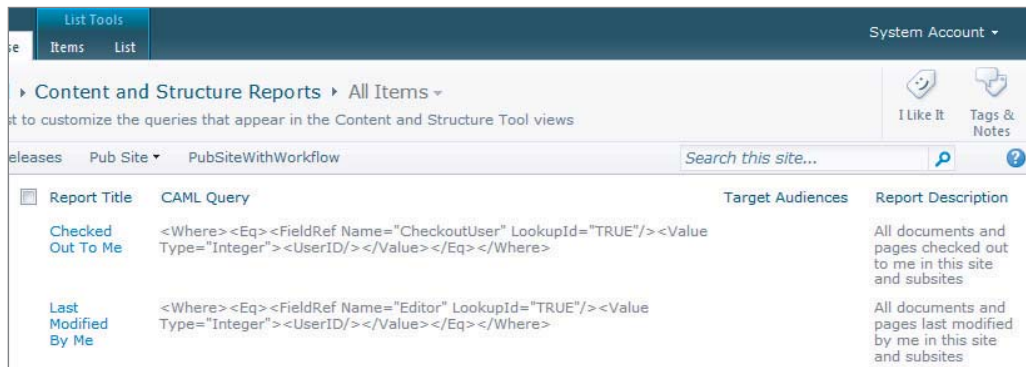


FIGURE 7-1

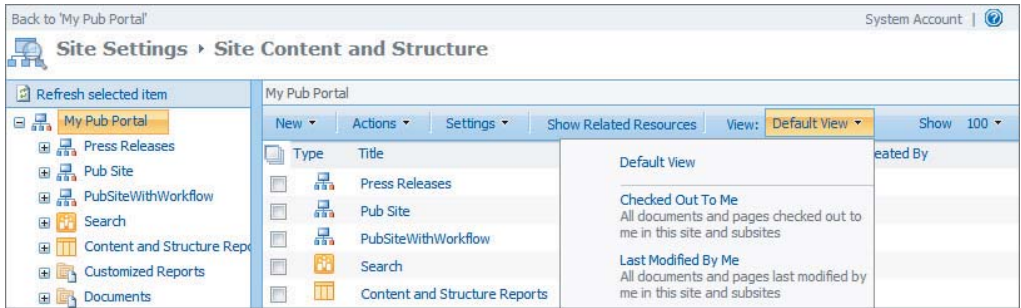


FIGURE 7-2

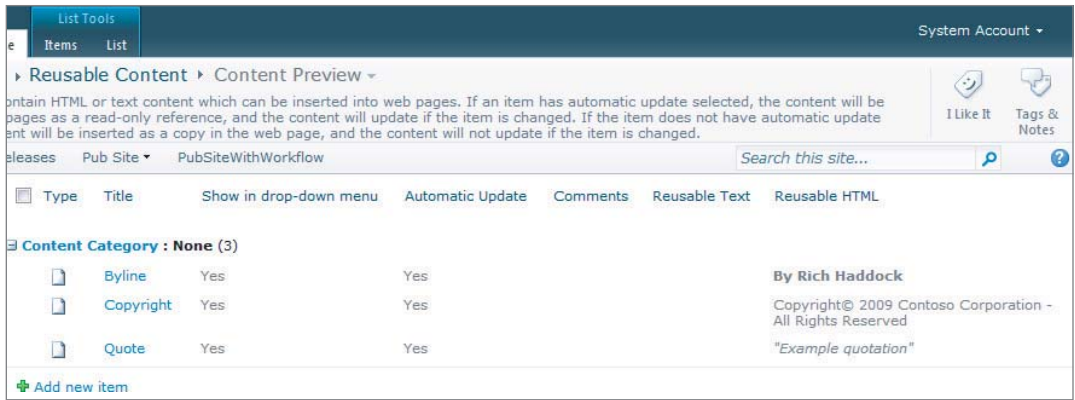


FIGURE 7-3

THE ANATOMY OF A PAGE

In terms of providing a consistent user experience, the SharePoint WCM features provide both control and flexibility. One of the ways this is accomplished is through the way a page, meaning the content delivered to the user, is constructed and subsequently rendered. The following sections cover the various mechanisms that SharePoint uses to make this level of control possible.

Master Pages

Master pages play a pivotal role in SharePoint WCM with regard to a site’s appearance. These special files dictate core layout and functionality, which is inherited by any other referencing page. Note that master pages are certainly not a SharePoint-only feature. They were originally introduced into ASP.NET in the 2.0 era and were a boon to developers and designers who struggled with creating consistent and reusable functionality across a site. Master pages in SharePoint are not extremely different from standard ASP.NET master pages. However, there are some items related to master pages that are specific to SharePoint.

Page Layouts

While master pages are global to a site and are not content-specific, page layouts take over when it comes down to presenting content-specific information. More specifically, one page layout is associated with one, and only one, content type. However, more than one page layout can exist for a given content type. This makes sense, as there should be potentially more than one way to present homogeneous types of content. Consider the example of a product. A product content type would likely have fields such as product name, color, size, and so on. In addition, a product may or may not have an image associated with it. Therefore, it would be appropriate to have one page layout for products that have images, and another layout for products without images.

Page layouts define which pieces of metadata from a content type will be available on a page and where/how they will be displayed. Page layouts can also be as restrictive or loose as the requirements call for. For example, a very liberal and generic page layout might simply provide a rich HTML field control with nothing locked down. This would essentially allow users to create whatever content they want on the page, and however they please — kind of like using Microsoft Word with the full range of fonts, styles, and so on.

In addition, page layouts can contain web parts. The benefits here are fairly obvious, as web parts can provide endless custom functionality — ranging from simple tasks to complex integration with line-of-business systems. A page layout designer can “hard-code” a specific web part of a set of web parts into a page layout. To provide greater flexibility, the page layout designer can also allow authors to add their own web parts. However, this flexibility has a cost. If end users are allowed to add any web parts they wish, it could be difficult to ensure a reasonably consistent corpus of web content.

AN EXERCISE WITH TAXONOMY AND LAYOUTS

This section walks through the basic steps of creating a simple taxonomy with an associated layout. To accomplish this, a simple content type will be created in Visual Studio, SharePoint Designer will be utilized to create the page layout, and finally Visual Studio will be used to bundle everything up in a reusable solution package.

Start out in Visual Studio by creating a new project using the Empty SharePoint Project template. Call the project something like **SimpleTaxonomyAndLayout**, as shown in Figure 7-4. Because everything we are doing is compatible with sandboxed solutions, go ahead and select that option.

Next, create a new feature by right-clicking the Features node in Solution Explorer and selecting Add Feature. Immediately rename this feature to something like **Simple Taxonomy Content Types** and change its scope to Site.

The content type we are going to create is intended to showcase a company’s products, so it needs fields like product name, product description, product number, size, and an optional image. Create this content type by right-clicking on the project node in Solution Explorer and selecting Add ⇨ New Item. Select Content Type from the resulting dialog and call it **Product**. Because this content type is representing a product page, it needs to inherit from the Page content type; therefore, select Page on the next dialog and click Finish.

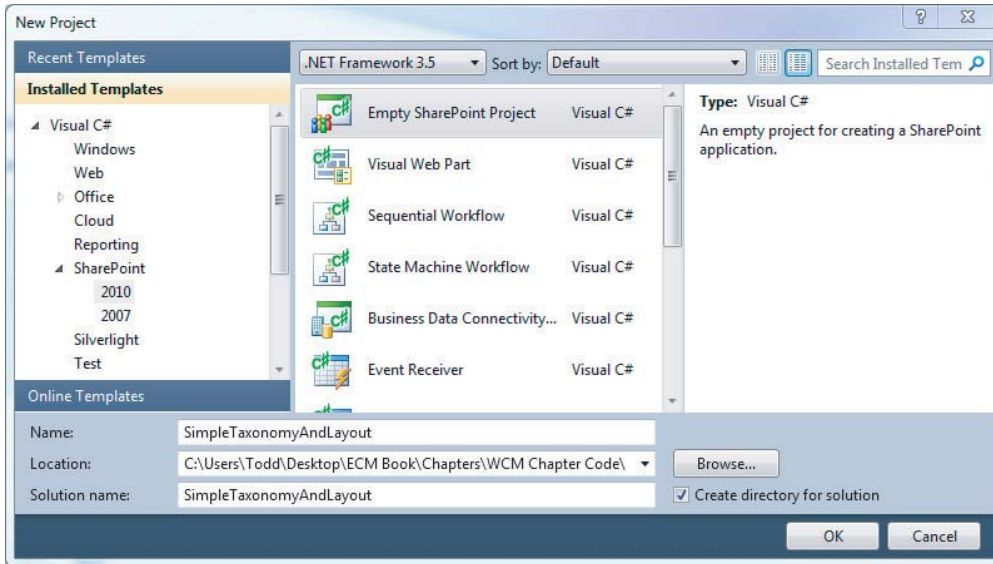


FIGURE 7-4

In the resulting `Elements.xml` file, notice the `ID` attribute of the `ContentType` node. It looks extra-long right? Well, content type inheritance works by appending new characters to an already existing content type definition's `ID`. The `ID` of the `Page` content type is itself fairly long, and then more characters must be appended to the end to ensure uniqueness; Visual Studio does this for you here.

Next, you need to create the site columns that will be a part of the `Page` content type. To do this, create a new item of type `Empty Element` in the same manner that the new content type was created (Add ⇨ New Item ⇨ Empty Element); call this item `SiteColumns`. Some of the fields previously mentioned as needed already have site columns created out of the box that can be used; we'll get to that in a moment. However, there are a few fields for which new site columns must be created. These are product number and product size. Using CAML, define these fields in the new `Elements.xml` file. The resulting XML should look similar to the following code:



Available for
download on
Wrox.com

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Field ID="{2C3CEE2F-AA31-461B-9E5D-4871C7583141}"
    Type="Text"
    Name="ProductNumber"
    DisplayName="Product Number"
    Group="Custom Columns">
  </Field>

  <Field ID="{7B000768-08CF-4EFF-9CDD-6AB1E8FC3428}"
    Type="Choice"
    Name="ProductSize"
    DisplayName="Product Size"
    Group="Custom Columns">
    <CHOICES>
      <CHOICE>XS</CHOICE>
      <CHOICE>S</CHOICE>
    </CHOICES>
  </Field>
</Elements>
```



```

    <CHOICE>M</CHOICE>
    <CHOICE>L</CHOICE>
    <CHOICE>XL</CHOICE>
    <CHOICE>XXL</CHOICE>
  </CHOICES>
</Field>
</Elements>

```

Code snippet Elements.xml

Notice that the `ProductNumber` field was simply created as a single line of text, whereas the `ProductSize` field was created as a set of options, such as S, M, and L (perhaps this website sells clothing).

The next step is to associate all desired site columns with the previously created Product content type. This includes the two site columns that were just created, as well as those representing product name, description, and an image. The following code shows what the content type's CAML should look like after doing this:



Available for
download on
Wrox.com

```

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ContentType ID="<a really long ID here...>"
    Name="Product"
    Group="Custom Content Types"
    Description="A content type for the product page."
    Inherits="TRUE"
    Version="0">
    <FieldRefs>
      <FieldRef ID="{F55C4D88-1F2E-4ad9-AAA8-819AF4EE7EE8}"
        Name="PublishingPageContent"
        DisplayName="Product Description"
        Required="TRUE" />
      <FieldRef ID="{3de94b06-4120-41a5-b907-88773e493458}"
        Name="PublishingPageImage"
        DisplayName="Product Image"
        Required="FALSE" />
      <FieldRef ID="{2C3CEE2F-AA31-461B-9E5D-4871C7583141}"
        Name="ProductNumber"
        DisplayName="Product Number"
        Required="TRUE" />
      <FieldRef ID="{7B000768-08CF-4EFF-9CDD-6AB1E8FC3428}"
        Name="ProductSize"
        DisplayName="Product Size"
        Required="TRUE" />
    </FieldRefs>
  </ContentType>
</Elements>

```

Code snippet Elements.xml

Notice a few things here. First, there is no definition for, or reference to, a product title field. This is because the parent content type, Page, already has a field defined, `Title`, which meets the needs of this field. Second, this content type definition is referencing two fields, `PublishingPageContent`

and `PublishingPageImage`, which were not defined in our solution. These fields are defined in SharePoint's publishing features and they meet our needs just fine, so we are using them here. We are also able to provide custom display names and define whether or not the fields are required. Finally, the two fields that were created earlier are referenced using the same GUIDs with which they were created.

That completes the taxonomy for this exercise! Take a look at the feature designer by double-clicking the feature in Solution Explorer. You will see that the new elements were automatically added to the "items in this feature" area of the designer. Visual Studio was nice enough to take care of this for you. Go ahead and deploy the solution by right-clicking on the project and selecting Deploy. Deploying the site columns and content type now will set the stage for our next step: creating the product page layout.

Because Visual Studio lacks a nice designer for creating page layouts, we will leverage SharePoint Designer for this task. Start by launching SharePoint Designer and opening the site you have been working with for this exercise. Notice that Page Layouts is an option in the left-hand navigation. Select this option and then click New Page Layout in the context-aware Ribbon at the top. The Product content type, which was created as part of the feature from the previous step, is selectable on the resulting dialog. Fill out the fields in this dialog as shown in Figure 7-5 and click the OK button.

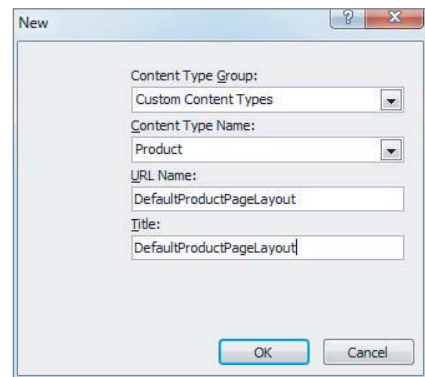


FIGURE 7-5

SharePoint Designer creates the page layout and then presents a new tab with the default markup for the new layout. Start by switching to code view and ensuring that the Toolbox is open. To open the Toolbox, select the View tab from the Ribbon and select Task Panes ⇄ Toolbox. The Toolbox contains a section called SharePoint Controls that has a subsection for Page Fields and Content Fields. Notice that the custom fields created in previous steps are displayed here, as shown in Figure 7-6.

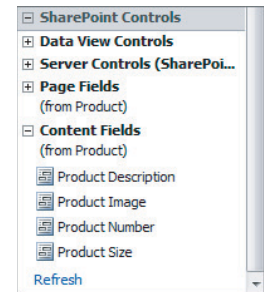


FIGURE 7-6

In the content section with the ID of `PlaceHolderMain`, you need to add references to the fields created in the Product content type. This is done by dragging a field from the Toolbox into the code editor; no typing needed! The following code shows what the completed page layout should look like:

```
<asp:Content
  ContentPlaceholderID="PlaceHolderPageTitle" runat="server">
  <SharePointWebControls:FieldValue id="PageTitle"
    FieldName="Title" runat="server" />
</asp:Content>
<asp:Content ContentPlaceholderID="PlaceHolderMain" runat="server">
  Product &#35;&#58;
  <SharePointWebControls:TextField
    FieldName="2c3cee2f-aa31-461b-9e5d-4871c7583141"
    runat="server"></SharePointWebControls:TextField>
```

```

&nbsp;Size&#58;
<SharePointWebControls:DropDownChoiceField
  FieldName="7b000768-08cf-4eff-9cdd-6ab1e8fc3428"
  runat="server"></SharePointWebControls:DropDownChoiceField>
<br /><br />
<PublishingWebControls:RichImageField
  FieldName="3de94b06-4120-41a5-b907-88773e493458"
  runat="server"></PublishingWebControls:RichImageField>
<br /><br />
<PublishingWebControls:RichHtmlField
  FieldName="f55c4d88-1f2e-4ad9-aaa8-819af4ee7ee8"
  runat="server"></PublishingWebControls:RichHtmlField>
</asp:Content>

```

First, notice that the actual HTML layout is nothing special here; the goal is simply to show the reader how to use the tools, not necessarily how to design awesome-looking websites. Each field control was selected according to its field type. For example, simply by dragging the `Product Size` field into the editor, a `DropDownFieldChoice` was automatically created for you. Each field control references its underlying field by either name or ID. You can verify this by comparing the GUIDs used in the markup with the GUIDs used to define the field controls in the Visual Studio project. By switching to design view, you can quickly verify that the layout at least roughly approximates what was intended.

Now that the page layout has been created, it needs to be transferred to Visual Studio so that it can be properly integrated into any application lifecycle management (AML) tooling, such as Team Foundation Server (TFS), that might be used by your organization, and deployed in a reusable manner as part of the already created SharePoint solution.

Switching back to Visual Studio, create a new item of type `Module` in the project and call it **masterpage**. The moniker `masterpage` is being used here because page layouts are deployed to a site's master page gallery. Next, remove the default file created with the module; the page layout created in SharePoint Designer is now ready to be imported into the project.

Visual Studio does not have an item template for page layouts, so the `Text File` item template can be used instead. Right-click on the `masterpage` module in Solution Explorer and create a new text file called `DefaultProjectPageLayout.aspx` and paste the contents of the page layout from SharePoint Designer into this newly created file.

After creating the page layout `aspx` file, it now needs to be referenced into the module you just created. When the `aspx` file was added to the module, a reference to it was automatically added to the `Elements.xml` file. However, a little more work needs to be done before it is completed. The following code shows what the completed file should look like:

```

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Module Name="masterpage" Url="_catalogs/masterpage">
    <File Path="masterpage\DefaultProjectPageLayout.aspx"
      Url="masterpage/DefaultProjectPageLayout.aspx">
      <Property Name="Title" Value="Default Product Page Layout" />
    <Property
      Name="ContentType"
      Value="$Resources:cmscore,contenttype_pagelayout_name;" />
    <Property

```



Available for
download on
Wrox.com

continues

continued

```

    Name="PublishingPreviewImage"
    Value="~SiteCollection/_catalogs/masterpage/$Resources:core,Culture;/
    Preview Images/CustomPageLayout.png, ~SiteCollection/_catalogs/
    masterpage/$Resources:core,Culture;/Preview Images/
    CustomPageLayout.png" />
  <Property
    Name="PublishingAssociatedContentType"
    Value=" ;#Product;#0x010100C568DB52D9D0A14D9B2FDCC96666E9F200794813
    0EC3DB064584E219954237AF390045db144ae16049968d9233165cdd89dc;#" />
</File>
</Module>
</Elements>

```

Code snippet Elements.xml

The big thing to notice here is the addition of several `Property` nodes. The nodes that are included here are expected by SharePoint for new page layouts. The following list explains what each node is for:

- `Title` is simply a short description of this page layout.
- `ContentType` specifies that this file is a page layout (as opposed to a master page). The value shown in the code is exactly what is needed each time a page layout is created.
- `PublishingPreviewImage` points to an image to be displayed when selecting this page layout during authoring.
- `PublishingAssociatedContentType` defines the content type to which this page layout is bound. The format is `;#<Content Type Name> ;#<Content Type ID> ;#.`

The solution is almost ready to be deployed. Before deploying, perform two small cleanup steps. First, there is a property on the Visual Studio project that indicates the project should not deploy the associated assembly. Because there is no custom code in this project, this property should be set accordingly, as shown in Figure 7-7.

Next, if you saved the page layout that was created using SharePoint Designer, go ahead and delete it now. Remember, SharePoint Designer was used simply for an easier and more enjoyable design experience. The real page layout will be deployed in a moment using Visual Studio. Finally, the solution is ready to deploy. Right-click the project in Solution Explorer and select `Deploy`; at this point, the page layout is ready to be used to create new pages.

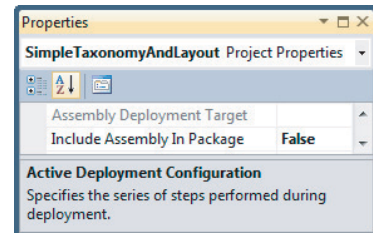


FIGURE 7-7

METADATA

To put it mildly, the metadata features in SharePoint 2010 are a welcome addition — especially from an enterprise content management perspective. Although other chapters of this book, such as Chapter 3, cover the metadata features in more detail, a quick review here is called for before covering these features from a WCM perspective.

“Metadata” in SharePoint 2010 can mean many things. However, it typically refers to the keyword taxonomies that are available in the form of terms. Terms are stored in the term store and can be considered either *managed terms* or *enterprise keywords*. The main difference between the two is that managed terms are generally more locked down as far as who can create or modify the terms, whereas enterprise keywords can be created and manipulated by users as they see fit. Another big difference is that managed terms can be stored in a hierarchy to model various business entities. This is in stark contrast to enterprise keywords, which are stored in a flat list.

Managed terms can be grouped into an entity called a *term set*. Term sets, not surprisingly, contain a grouping of related terms. A term set is what is used to model terms in a hierarchical manner. In addition, the aforementioned enterprise keywords can be “promoted” in a sense to managed terms in a term set. Term sets can exist at one of two scopes: *global* or *local*. Local term sets are visible and usable only at the scope of a site collection. Global term sets are defined in a managed metadata service application instance. Although a discussion of service applications is beyond the scope of this chapter, service applications allow specific functionality to be available globally to a farm, across many farms in an enterprise, or even to farms external to an organization. This central and global way to define term sets is a boon for anyone who previously had issues defining and applying a consistent taxonomy and tagging scheme.

Applying metadata to web content makes it much easier to link related pages together semantically or to simply group similar pages together in any manner desired. One way to take advantage of this metadata that content authors and others work so hard to apply is to utilize the Swiss Army knife of WCM: the Content Query web part, which is discussed in the next section. For example, a product page might have a metadata field that enables an author to indicate related products. These related products might then be displayed as a list on the main product’s page. Another example might be showing a list of people associated with a geographical region or some other business entity.

CONTENT QUERY WEB PART

As mentioned earlier in this chapter, the content query web part (also referred to as CQWP) is one of the more powerful and useful tools in the SharePoint WCM Toolbox. At a high level, the CQWP is designed to aggregate site content of varying types from varying scopes and to render results in a specific manner. In addition, it is designed to do these tasks in an efficient manner.

Web Part Options

The CQWP’s options are broken down into query and display options.

Query Options

The first section of options for this web part is related to the query that will retrieve relevant content. There are several categories of sub-options within the query options. First, the source of the content must be selected. The content source specifies the size of the scope used when looking for content. Possible options include all sites in the current site collection, a specific site and all of its subsites, or simply a specific list. Therefore, the query scope can be quite broad or quite small depending on what you are trying to accomplish.

The next set of options related to querying content relates to the type of content that is of interest. To filter to certain types of content, the web part can be configured to pull from certain list/library types. While this could potentially be useful, for the purposes of a WCM site, the Pages Library list type is typically selected. The actual SharePoint content type of the content of interest can also be chosen. This content type can be as specific or general as desired, based on where in the content type inheritance tree the selected type resides, as well as whether or not child content types of the selected type are allowed. The web part can be configured to consider any already configured audience targeting settings.

Finally, the query options allow for general-purpose query filtering. These filters can be very powerful when you want to reduce the number of results returned to a very specific set of content. For example, the home page of a corporate intranet might have a requirement to display all news articles in the past week that have an article category of company-wide news; these filters would enable that type of query. Essentially, the user configuring the web part is presented with a set of controls that represent a column, an operator, and a value. These can also be “AND’d” and “OR’d” to achieve the desired result. Figure 7-8 shows what these controls look like when editing the CQWP. In this example, the query will return items that have a created date in the last week, or, more specifically, items that have a created date that is greater than or equal to the current date minus seven days.

FIGURE 7-8

Presentation

The next set of options of the CQWP relate to presentation of the query results. Options here include things like how to group and sort the results, as well as specifying a maximum number of results. This comes in handy when you have a requirement such as displaying the 10 most recently added product pages.

Of course, the web part can also be configured to return what the results actually look like — that is to say, the style of the rendered items. There are several group and item styles included out of the box. These styles are defined in XSLT files, which can be found in their default location, the Style Library. The item styles are defined in `ItemStyle.xsl` and the group styles are defined in `Header.xsl`. Of course, as with most things in SharePoint, custom styles can be developed by someone who is comfortable with XSLT.

The presentation options also define which fields are actually displayed in the rendered UI. Figure 7-9 shows this portion of the web part’s properties. Note that the display slots vary according to the styles that were chosen earlier. The available fields to choose from are only limited by the content queried for. To put it another way, the content that is returned by the query determines the available columns

FIGURE 7-9

that can be configured here. Note that the fields are configured manually — that is, the column names are typed in, rather than being chosen from a list of options provided by the UI. Therefore, special care must be taken to input the correct values.

THE CONTENT AUTHORING PROCESS

Content authoring is not just about typing copy into a textbox. Although generating the actual words and images that will make it onto finished web pages is a large part of the content authoring process, it is not the only thing to consider. To underscore this point, consider the following questions and then review the topics in this section:

- How is content reviewed for quality?
- How does content end up in the right place (right environment, right library, etc.)?
- How is content look-and-feel standardized?

Authoring Web Content

In order to have content to display in a web page, there first has to be a way to get the content into the system. The most common way to do this in a SharePoint WCM environment is to use the browser itself to create and manage content. Creating a new page in SharePoint 2010 is as easy as selecting New Page from the Site Actions menu. This will trigger a very simple web dialog that contains only one input field for the new page's title, as shown in Figure 7-10.

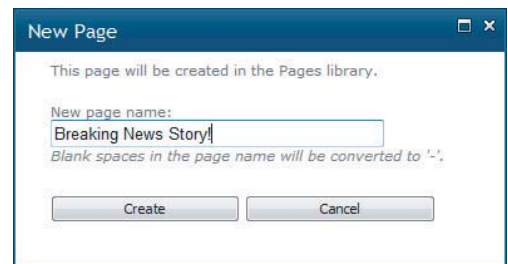


FIGURE 7-10

If you have any experience with WCM authoring in SharePoint 2007, you will notice right away that the page creation process is extremely streamlined. Rather than force users to enter quite a bit more information about the page to be created, they can focus on just getting the page in place. Configuring can be done later. This approach should reduce confusion and anxiety for content authors.

After clicking the Create button, a new page is created and ready to be configured and authored. Figure 7-11 shows what a new page might look like.

Note a few important things about this authoring experience. First, the Ribbon is present, which is a welcome addition to WCM authoring because of the sheer number of features available to content authors. Second, notice the main content area labeled “Page Content” in Figure 7-11. This is the new and improved rich-text edition in SharePoint 2010. This text editor provides a familiar authoring environment to anyone accustomed to modern word processors such as Microsoft Word. The rich-text editor works hand-in-hand with the aforementioned Ribbon to provide users with an easy way to manipulate and style web copy.

You can customize the rich-text editor to allow users to style content in any way desired or to allow only predefined styles. As a blanket statement, something closer to the latter approach is

better as long as thought has been given to the styles that should be included in an organization's style guide.

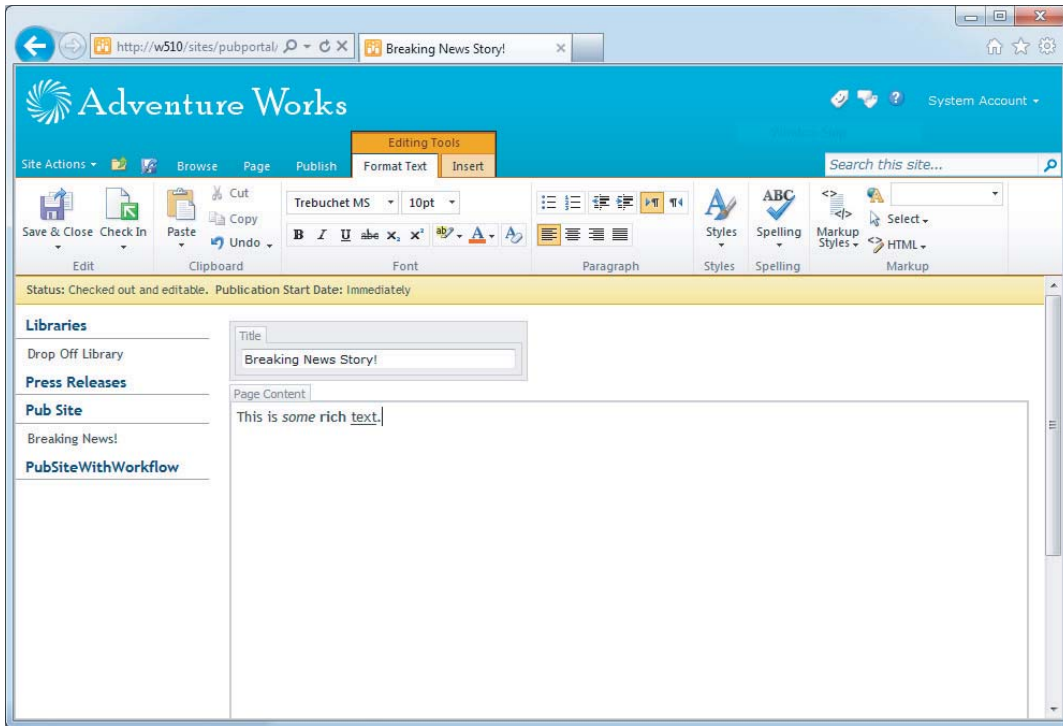


FIGURE 7-11

The rich-text editor is actually just an ASP.NET web control because it ultimately inherits from `System.Web.UI.Control`. This control has several properties that can be used to open up or restrict authoring capabilities. For example, setting the `AllowFonts` property to false disables the “Font” area of the Ribbon. In a tightly controlled website, this is likely a good idea, as you don’t want someone adding 72-point Comic Sans to an otherwise great-looking site. Another useful property in this vein is `AllowTextMarkup`. Setting this property to false disallows the author from adding, no matter the method, content with specific markup tags (e.g., ``, ``, `<i>`, etc.). This is useful because authors might be pasting text copied from a source that contains undesirable formatting. In addition to disabling what users are allowed to do with the rich-text editor, a predefined list of styles can be created so that authors can select the target text and then select one of the predefined styles from the Ribbon control. This, along with author training, goes a long way toward ensuring consistency.

Another great feature that is surfaced in the authoring Ribbon is the capability to change a page’s layout on-the-fly. You can change the page layout from the Page Ribbon tab; the Ribbon menu is specifically found in the Page Actions section (see Figure 7-12).

In addition to the great features mentioned already, the Ribbon also enables users to perform the following operations:

- Manage the page life cycle.
 - Save
 - Publish
 - Delete
 - Schedule
 - Workflow interaction
- Create and manipulate tables.
- Insert media (pictures, video).
- Utilize reusable content.
- Manage web parts on the page.

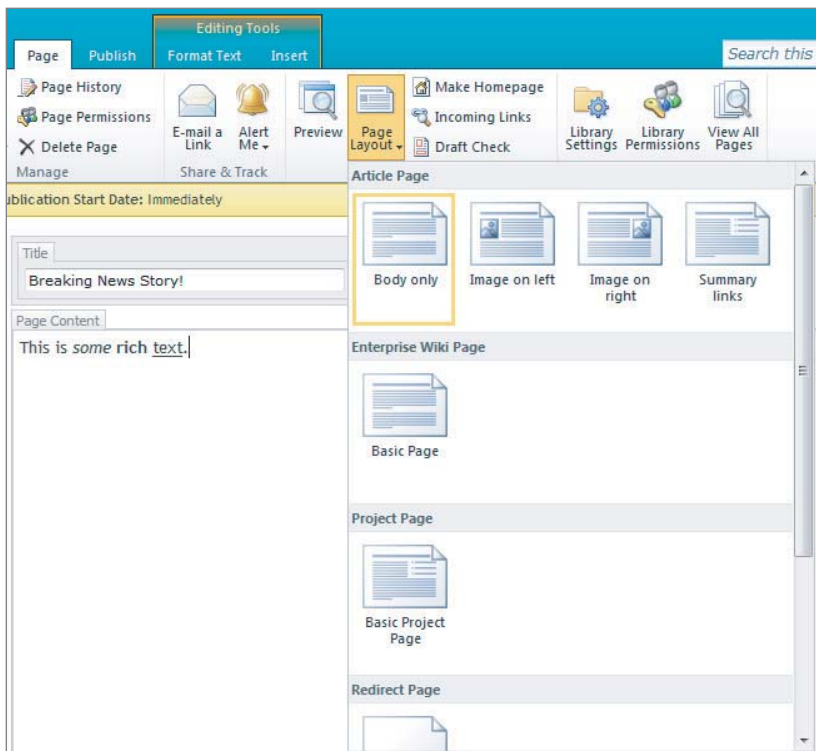


FIGURE 7-12

Using the Content Organizer

The Content Organizer, which is discussed in depth in Chapter 8, is a core feature of SharePoint 2010 and therefore has multiple uses. However, it can be greatly beneficial in web content management scenarios.

For example, a site may contain many different types of content. Perhaps there are pages and/or documents for products, customers, employees, training manuals, and so on. In this case, one management option would be to train all the users to use the correct location for each type of content. If there is a short list of content types, and not too many users, then this approach may be reasonable. However, if there is a long list of users, perhaps coupled with a long list of content types, then this approach may become hazardous to a well-organized corpus of content. This is where the Content Organizer comes in.

The Content Organizer has its roots in the SharePoint 2007 records center and essentially uses something called a Drop Off Library, along with a set of rules to route incoming content to the correct location. The Content Organizer can be configured such that users literally cannot place content in an incorrect location. Furthermore, it can be configured such that content is routed to the appropriate library only if the correct metadata has been applied.

In a public-facing website scenario, and in many other cases, it is obviously very important to ensure that content is located where users expect it. Therefore, the Content Organizer can be a boon for content stewards who may otherwise have a hard time keeping everything in the right spot.

Consider this scenario: a site reorganization effort currently has product pages going to a completely different site than the one they have been using for the past two years. Rather than having to notify all authors of this change and then having to rely on them to suddenly modify their firmly established behavior, a Content Organizer rule can be updated to send the pages to the correct spot, with no author intervention required.

In order to use the Content Organizer, you must activate the site-level feature of the same name. Once the feature has been activated, the following changes are made to the site:

- Two configuration links are added under the Site Administration section of site settings:
 - Content Organizer Settings
 - Content Organizer Rules
- A document library called “Drop Off Library” is created.

The Content Organizer is discussed in more detail in Chapter 8.

Content Deployment

Content deployment is a high-level feature in SharePoint that enables the publishing of content from one site collection (the source) to another site collection (the destination). Note that the source and destination can be, and in many cases will be, in different SharePoint farms. This is certainly not a requirement, but in a production environment there are advantages to this scenario. First and foremost, this enables the source and destination farms to be in different Active Directory domains, which reduces the attack surface of a public-facing, anonymous website.

In terms of “content,” it is probably what you’d expect: pages, list items, libraries, images, and so on. In other words, content refers to the items that end up in a SharePoint content database. Therefore, this excludes things like solution packages, files on the file system, assemblies in the GAC, and so on.

So why does content deployment exist? Basically, it is to assist in keeping the target site collection pristine and free from manual modifications made by users. In addition, there are security benefits, as previously mentioned.

Configuring content deployment consists of defining content deployment *paths* and *jobs*. Content deployment paths simply define the source and destination site collection. Content deployment jobs come in three flavors:

- Full
- Incremental
- Quick

The full and incremental jobs are schedule-based and obviously differ based on the scope of the content that is deployed from the source to the destination. Quick deployment enables users to deploy a specified page on an ad-hoc basis. It is not automatically available to end users; therefore, some configuration must take place prior to this feature being usable. First, for each content deployment path for which Quick deploy is desired, it must be enabled. Second, each user who should be able to perform this action needs to be added to the Quick Deploy Users group. Note that Quick deploy is not instantaneous; the Quick deploy job in Central Administration runs every 15 minutes by default. Therefore, if a user chooses to Quick deploy a page, it will be deployed between 0 seconds and 14 minutes and 59 seconds if the default schedule for this job is used.

Workflow

While workflow is a foundational feature of SharePoint, it plays a very important role in WCM scenarios. By default, publishing sites with workflow have content approval turned on and an associated approval workflow in the Pages library. Because content approval is being used, the workflow is started whenever a major version is being published. In addition, a workflow can be manually started by anyone with edit permissions.

By properly defining authoring and approval roles up front and leveraging SharePoint’s built-in workflow capabilities, configured by default in publishing sites, content governance can be implemented and easily monitored by appropriate parties. For more on workflow, refer to Chapter 4.

ENTERPRISE WIKIS

Far from public-facing Internet sites, at the other end of the spectrum of SharePoint WCM are enterprise wikis. Mostly due to the extreme popularity of Wikipedia, the concept of a wiki is familiar to most technologically savvy people. Wikis provide a semi-structured way to enable large numbers of participants (“large” can mean more than a few, hundreds, or beyond) to create, edit, and review user-generated content. This content distribution paradigm is generally referred to as *many-to-many*

because it reflects the many authors generating content for many consumers. This is in contrast to blogs or team sites, whose content is generally created by relatively few people for relatively many people.

Wikis in SharePoint 2010 are heavily supported by the out-of-the-box publishing features to enable page ratings, tagging with the managed metadata infrastructure, as well as provisioning the enterprise wiki content types and page layouts. In addition, unlike most of the sites discussed in this chapter, enterprise wikis are generally used in an intranet scenario for managing corporate information. This is not a requirement, however.

OTHER MAJOR CONSIDERATIONS

The following sections cover some topics that are not necessarily always specific to WCM but play an important role in providing a complete WCM solution.

Branding

Given that WCM is often used to implement an organization's public web presence, control of branding, or a site's look-and-feel, is of the utmost importance. As already discussed, SharePoint WCM leverages the master page framework, so it facilitates the use of a consistent look and layout throughout the site.

However, master pages are not the only way to implement a brand in SharePoint; themes can also be used for this purpose. SharePoint ships with several out-of-the-box themes, and it is also possible to create custom themes without too much effort. It is important to understand, however, that themes will only take you so far. Themes are used to provide a consistent color scheme and graphics to existing layouts, but because themes are implemented using CSS (and not HTML), you can only do so much to customize existing markup. Therefore, when planning a branding effort, consider the various options available, including out-of-the-box, theming, and full-on custom, which would include master pages, CSS, images, and so on.

Navigation and Search

Given that a website should offer users the content they need as quickly and easily as possible, a good approach to "findability" is of the utmost importance. Navigation and search are the two main methods for locating content of importance to users.

While search is a common way for users to find what they are looking for, content should be searchable using the "browse" approach, as a user may not even know what to search for initially. The navigation approaches provided by SharePoint make this possible. These approaches include the following:

- Top navigation for major topical areas.
- Side navigation for content local to the current site.
- Use of breadcrumbs for drilling down and keeping track of where you are in a hierarchy.
- Use of metadata for dynamic navigation based on some context. This would likely involve use of the content query web part.
- Summary links for fixed navigation on a page.

Like navigation, search is not specific to WCM in SharePoint, but it is an important component. Features like best bets and search scopes ensure that users can locate the content they are looking for in your site.

Targeting Global Users

If your website will be consumed by users whose native language is other than the site's default language, then careful planning is needed to enable this scenario. There are two major enablers of multilingual sites in SharePoint: *language packs* and *variations*.

Language packs are fairly straightforward. When Microsoft developed SharePoint, care was taken to translate each string that is displayed in an out-of-the-box screen. This refers to toolbars, settings pages, dialog boxes, and other out-of-the-box elements. To enable the creation of site collections or sites with a language other than the one in which SharePoint was installed, language packs must be downloaded from Microsoft's website and installed on each SharePoint web front end.

Installing and using the language packs are only one step to enabling a thorough global experience. The next, and much more resource-intensive, step is utilizing a feature of SharePoint called *variations*. Variations allow a site to be specified as a source for custom content as well as many destination sites to be specified — one for each desired language. Content created in the source site can then be automatically copied to each configured destination. This is where the resource-intensive part comes in.

The content that is copied from the source site to the destination sites is still in the original language. Computers are not yet proficient enough to automatically translate custom-created content to arbitrary languages without the aid of humans. Therefore, when content is copied, notifications can be sent to alert someone that a translation needs to occur. After the translation effort has occurred, pages on the destination sites can be published and consumed by users. Although variations don't do a lot of the heavy lifting required in a global web scenario, they do provide the organization required to ensure that the right content ends up in the right spot, and that the right people perform the work necessary to get content to users.

Reporting and Analytics

Out of the box, SharePoint provides some rich and insightful reporting capabilities at the site and site-collection levels. Figure 7-13 shows an example report that displays the top pages in a given site.

Other traffic reports include the following:

- Number of Page Views
- Number of Daily Unique Visitors
- Number of Referrers
- Top Visitors
- Top Referrers
- Top Destinations
- Top Browsers

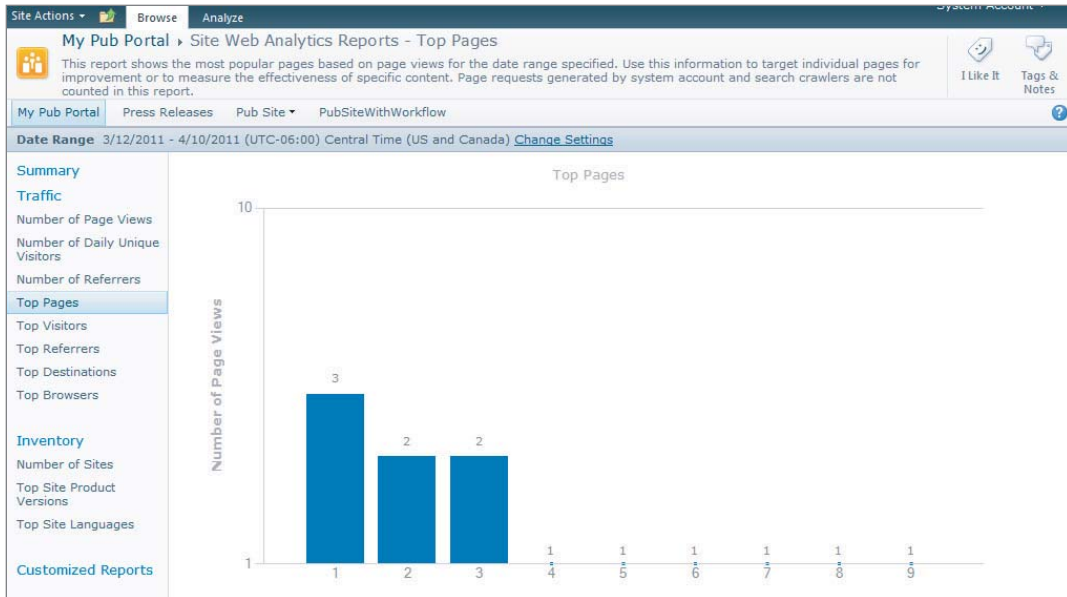


FIGURE 7-13

These reports are available in the browser in nice looking graphs and can be filtered by data points such as date. In addition to the out-of-the-box reports listed above, custom reports can be created in Excel. SharePoint will generate a workbook for you when you click the Analyze button on the Ribbon. What's great about this feature is that the data in the formatted Excel workbook doesn't become stale; it can be refreshed on demand, making this feature all the more powerful.

You also have the ability to configure a workflow which will send a report on a given interval or when a given condition is met. This feature allows you to proactively receive information about your site without having to remember to check a report page every so often.

SUMMARY

While web content management is in a class by itself with respect to the majority of other ECM topics, it requires and benefits from the same controls and features that SharePoint offers. Features such as content approval, workflow, and the Content Organizer provide enterprise-class governance and control capabilities needed in today's content management systems.

In addition, web-specific features such as page layouts and master pages round out the SharePoint WCM offering. Web content is given first-class treatment in SharePoint 2010, as demonstrated by its rapidly growing user base, who trust this platform to create, maintain, and present their public face to the world on the Web.

8

Records Management

WHAT'S IN THIS CHAPTER?

- ▶ Understanding records management and how to leverage Microsoft SharePoint to support an organization's records management process
- ▶ Exploring Microsoft SharePoint's extensive records management capabilities
- ▶ Administering and using Microsoft SharePoint's records management capabilities
- ▶ Leveraging Microsoft SharePoint APIs to programmatically interact with its numerous records management features

Over the past few decades, businesses have seen a tremendous increase in the importance of effectively managing the documents that are critical to their daily operations. In many cases, effectively managing these documents not only means being able to store them and manage their usage, but also ensure that they are managed according to strict compliance requirements imposed by government regulation, legal mandates, information disclosure legislation, and other internal or external requirements for which failing to do so is met with harsh legal penalties or other negative repercussions.

Companies are reacting to these requirements by implementing various governance, retention, and compliance policies and procedures within their organizations. While a document management system can help an organization store and manage the usage of electronic documents, these systems lack many capabilities necessary to mitigate the business risk associated with failure to meet mandated compliance scenarios. In order to achieve the compliance required, companies are relying on records management systems to enforce and support the policies and procedures they set forth.

WHAT IS RECORDS MANAGEMENT?

Records management represents the central pillar of enterprise content management; it is responsible for the methodical and consistent management of an organization's records. Many standards have been defined by various authorities to describe the field of records management. The ISO 15489 standard published in 2001 defines records management as “the field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use and disposition of records, including the processes for capturing and maintaining evidence of and information about business activities and transactions in the form of records.” The Association for Information and Image Management (AIIM) extends this definition to include records of all types, including those maintained in electronic format. Another standard defined by the United States Department of Defense (DoD) is outlined in directive 5015.2, which was first issued in 1997. This directive defines the mandatory functional requirements that a records management system must provide in order to be used in the DoD, and it has become the de facto standard for many other federal agencies.

While these definitions vary somewhat according to the source, it is clear that records management pertains to not only the management of records, but also the policies and procedures surrounding them. Another critical piece of information that must be defined is what a “record” actually is. This definition also differs slightly between various authorities, but a record is essentially any electronic or physical document that represents evidence of an organization's business activities that must be retained for a specific time period. The types of documents, their retention period, and the ability to audit them is ultimately defined and driven by various legislation, regulation, and legal requirements, as well as good business practices, all of which differ from organization to organization.

Why Records Management Is Important

The importance of records management to an organization has been demonstrated by numerous, highly publicized corporate scandals centering on the mismanagement of corporate records. Some of these examples include the shredding of key documents to hide wrongdoing by the Enron Corporation, the falsification of financial statements by WorldCom, the destruction of audit records by Arthur Anderson, and even the destruction of royalty records by Walt Disney. The result of mismanaging records can be extremely costly to a business in terms of large fines and steep penalties.

Records management systems are used to mitigate the risk associated with doing business. By providing an organization with the means to proactively manage its records, they enable a company to protect itself from the potential loss or destruction of records, eliminating the need for time-consuming, ad-hoc record gathering to support audits or litigation; and they ensure compliance with various legislative or regulatory requirements, which continue to grow. When you consider various compliance scenarios such as Sarbanes-Oxley, Turnbull, Basel II, CFR 21 Part 11, SEC, the Patriot Act, Health Insurance Portability and Accountability Act (HIPAA), and basic operational risk management, it becomes apparent that records management is critical to any organization.

MICROSOFT SHAREPOINT AS A RECORDS MANAGEMENT SYSTEM

Records management is a core function of Microsoft SharePoint Server, which provides numerous features for managing an organization's records. For an organization to best use records management features, the organization must not only understand the features that are available, but they must also spend time thoroughly planning how they will be used effectively, as the management of records is largely a process that must be followed more so than just a set of features that are simply implemented and used. Having a process outlined, formalized, and consistently reviewed for accuracy and compliance is the most critical component of all records management systems. The reason such importance is placed on the formalization of this process is that if the organization ever faces litigation surrounding their records management practices, proof that this process is in place and being actively followed will be absolutely necessary to support a valid defense of these practices.

Records Management Planning

Prior to implementing your records management process using Microsoft SharePoint Server, several planning activities must take place in order to ensure that SharePoint is leveraged to meet your organization's records management goals. The following sections describe these prerequisite tasks.

Identifying Roles

In order for a records management process to be successful, it is critical to identify key roles within the organization. This ensures that the records management process is correctly implemented and actively managed throughout the organization. Properly identifying the people involved in the process allows each person to understand what their responsibilities are in the creation and maintaining of the process and who is accountable for each area of the process.

Records Manager

A *records manager* is someone within the organization who is knowledgeable about the field of records management and whose responsibility it is to create and manage the records management process. In larger organizations, records managers may be dedicated resources whose sole responsibilities are managing the organization's records, whereas smaller organizations may assign these responsibilities to resources that have other organizational responsibilities but expertise in records management.

Determining the resource who will serve as the records manager differs according to the organization's structure and may range anywhere from a file clerk to the Central Information Office (CIO). The person designated as the records manager should have competency in the creation and use of records, record retention strategies, record categorization, the operational functions of the records management system, and knowledge of how to implement and support various compliance scenarios within the system. In an effort to create a standard designation by which records managers could be measured and accredited, the Institute of Certified Records Managers (ICRM) was founded as an international organization that specializes in the certification of professional records managers.

Compliance Officer

A *compliance officer* is someone within the organization who is knowledgeable about specific compliance scenarios that the organization faces. Their main responsibility is to oversee and provide guidance on the records management process to ensure that all issues surrounding compliance are addressed accordingly. Compliance officers are extremely important in industries that are heavily regulated, such as healthcare or financial services. Compliance officers typically work closely with records managers or may even be a records manager themselves.

The identification of compliance officers is largely dependent on industry as this is typically what governs the compliance concerns of the organization. For instance, in the financial services industry, a compliance officer may need to understand Sarbanes-Oxley for public accounting standards, SEC rules for protection of market investors, and various financial regulations, among others. Within the healthcare industry, the compliance officer may need to be intimately familiar with HIPPA for patient privacy matters and the rules of the Joint Commission on Accreditation. Many large organizations will have entire departments dedicated to compliance that are typically led by an executive-level position referred to as the Chief Compliance Officer (CCO).

In order to ensure that compliance officers meet a certified level of standard for their industry, there are organizations that serve to govern and promote this certification such as the Health Care Compliance Association (HCCA) or the International Compliance Association, among many other industry-specific certification programs.

Content Manager

A *content manager* is someone within the organization whose responsibility it is to find and control where records in the organization are stored. Content Managers manage teams within the organization that produce documents or files. They possess the knowledge of what constitutes a record according to the records management process. Content managers actively manage content and ensure that the records management process and policies are being properly followed by their teams.

Analyzing Content

Once the key roles in a company have been identified, the next step is to analyze existing content. This is a critical planning activity that must take place prior to devising a records management process. It is up to the records managers, compliance officers, and content managers to understand and evaluate the various documents that are used within an organization in order to determine what documents constitute records and how they will become records.

When analyzing content to determine what constitutes a record, the records manager will need to work closely with the compliance officer to understand the organization's compliance and legal requirements. These requirements typically provide a guideline as to what information must be maintained and for how long. Using these requirements along with the general needs of the business, the records manager is able to determine what documents constitute an actual record. Enabled with this information, the records manager can then collaborate with the content managers within the organization to determine where these records exist and how they are used.

After compiling the list of records, the records manager will then categorize these records into common groups of related records. For example, purchase orders and invoices may be grouped into

the account payable category, whereas resumes, applications, and benefit enrollment forms may be grouped into the personnel records category. These various categories will aid the records manager in developing the file plan according to the types of records that will be maintained.

Developing a File Plan

After careful analysis of the organization's records management needs, it is imperative to develop a file plan that outlines the records management process. The file plan provides the guidelines to follow when implementing the process and policies within SharePoint. It also serves as important documentation if your organization is ever faced with litigation in which records are called into question.

Plan Recordization

When building a file plan for records management, a key area that should be addressed is *recordization*. Recordization is the process by which a document becomes an official record. It is important that the file plan identify what documents will be declared as records, the categories to which each record type will belong, how and when records will be declared, and where records will be stored. It is also important to specify the people involved with the creation and usage of the records and who will act as the content manager for each record category.

Plan Retention Schedule

Along with the recordization plan, it is equally important to outline the *retention schedule* for each record as it pertains to compliance, legal, or business requirements. This is important because typically, each type of record must be actively managed for a specific or recommended period of time. Failure to properly maintain these records could have serious, negative ramifications if your organization is ever faced with legal litigation or audits. The retention schedule should also indicate how a record should be disposed of once it has met its expiration date, which enables an organization to reclaim storage and improve efficiency by reducing the number of managed records. The act of disposing of a document is referred to as *disposition* and will typically involve archival to a location for long-term storage or destruction of the file all together.

Plan Compliance Documentation

Once the file plan has been created, the next step is to expound upon it with supporting information. This will serve as documentation that can be used as verification of compliance. This supporting information should include guidelines for system usage, the design and implementation plan, and any reports that will be generated to measure plan effectiveness. Obviously, some of this information will be gathered after the solution has been properly designed and implemented. Ultimately, this documentation, along with the file plan, will be crucial should your organization be required to prove its records management competence and practices in case of litigation.

Designing a Solution

After records management planning has been completed, the next step is designing the solution within SharePoint to support your file plan and compliance needs. Because a lot of planning took place up front during plan creation, the implementation within SharePoint is fairly straightforward. This is the point at which sites, document libraries, content types, policies, routing rules, and workflows are implemented.

Using the file plan, the appropriate site structure can be created to house the document libraries that will store the records belonging to the various categories which were identified. The optimal site structure will allow for records within this site to be easily secured, using SharePoint users and groups to limit access to the records that are only applicable to the people indicated in the file plan for the identified record category. Further security restrictions can then be applied to the individual document libraries if necessary. After the document libraries have been created for record storage, content types can be added to these libraries for capturing the metadata that is applicable to each record type. Once all of this structure has been put into place, the rules and workflows for routing records to these locations can be created, and the policies that govern retention and disposition can be configured according to the plan.

During this phase, critical decisions must be made regarding the SharePoint features, capabilities, and customizations that will be leveraged. Throughout the rest of this chapter, the records management capabilities of SharePoint will be discussed and should present a good overview of what is possible and how to accomplish, carry out, and manage your file plan.

Compliance and SharePoint

With regard to records management, SharePoint does not provide support for specific scenarios directly. For instance, the Sarbanes-Oxley (SOX) Act is a piece of government regulation put into place for public accounting practices which basically consists of a set of rules that an organization must follow with regard to what specific records are kept and for how long. SharePoint does not provide a SOX module that can simply be enabled which automatically configures all of the rules set forth. For that matter, it does not do this for any specific scenario. For example, HIPPA dictates that disclosures, authorization forms, and responses to a patient are maintained for six years, but SharePoint does not provide HIPPA-specific retention periods. The Patriot Act even dictates that records are maintained for all types of software and hardware that are used by an organization and the location of all of its electronic data, but SharePoint does not provide the capability to enable the recordization of this specific information.

SharePoint views compliance requirements from a more generalized viewpoint than specific compliance scenarios may dictate. By providing general but flexible scenarios, SharePoint can offer the facilities necessary to accommodate an organization's specific needs. In this regard, the success of maintaining compliance is largely reliant on the organization, various roles previously outlined, and the defined policies and processes to ensure that the SharePoint capabilities are leveraged in such a way as to enforce certain rules in order to mitigate business risk.

The records management capabilities of Microsoft SharePoint Server were designed with the general compliance-driven scenarios outlined in Table 8-1. The records management features of SharePoint can be categorized into the corresponding scenarios from which they are derived, as shown in Figure 8-1.

TABLE 8-1: SharePoint Compliance Scenarios

| SCENARIO | DESCRIPTION |
|------------------|--|
| Managing Records | Drives the features in SharePoint that pertain to recordization and the management of the records repository |
| Retention | Drives the features in SharePoint that pertain to the life cycle of records, from retention to deletion |
| Auditing | Drives the features in SharePoint that pertain to tracking records usage |
| eDiscovery | Drives the features in SharePoint that pertain to finding and placing holds on records that will be used in litigation |

| | |
|--|--|
| <p>Managing Records</p> <ul style="list-style-type: none"> • Records Center • In-Place Records Management • Information Management Policy • Content Organizer • Workflow | <p>eDiscovery</p> <ul style="list-style-type: none"> • Searching and Holding • Hold Reports |
| <p>Auditing</p> <ul style="list-style-type: none"> • Audit Reports • File Plan Reports • Information Management Policy | <p>Retention</p> <ul style="list-style-type: none"> • Information Management Policy |

FIGURE 8-1

MANAGING RECORDS

SharePoint Server's records management features take advantage of and build on existing document management features for document taxonomy, storage, and security. Along with these features, SharePoint provides two separate ways to manage records. Records can be managed *in place*, meaning the documents are declared as records within the document library in which they are located; or within a site called a records center, which is based on a specialized site template of the same name tailored for records management. Choosing whether to manage records in-place, in a records center, or using a hybrid approach is a decision that should be made by the records manager during the creation of the file plan and solution planning activities.

Other records management features are also provided, including the capability to institute an *information management policy* for declared records, the capability to route records to their appropriate locations using the Content Organizer, and the capability to manage the recordization process using SharePoint workflows.

Recordization

Once records have been identified, the next step in the process is to make them an officially managed record. This act is referred to as recordization. Records can be declared and managed in two ways, in-place, and by adding them to a Records Center site. The way in which a record is declared can be accomplished by declaring it in-place, submitting it to a Records Center, routing them their appropriate destination using the Content Organizer, or as an activity of a SharePoint Workflow. Most recordization tasks can be performed using SharePoint's user interface or through the various programming models which SharePoint provides.

In-Place Records Management

Using SharePoint Server, you can manage records alongside of active documents without having to move them to a separate archival location. By enabling the in-place records management feature within SharePoint, administrators can specify which users have the permissions needed to manually declare or undeclare a record, and edit or delete a record.

Although managing records in this manner is often appropriate and may be easier in some situations, it is important to consider whether any compliance restrictions affect this feature as a viable option.

Configuring In-Place Records Management

In order to use in-place records management within SharePoint, perform the following actions to configure its usage:

1. Navigate to Site Collection Features from the Site Settings page.
2. Activate the In Place Records Management Feature.
3. Navigate to Record Declaration Settings (see Figure 8-2) from the Site Settings page.
4. Choose Record Restrictions options:
 - a. No Additional Restrictions: Records are not restricted
 - b. Block Delete: Records can be edited, but not deleted
 - c. Block Edit and Delete: Records cannot be edited or deleted
5. Choose Record Declaration Availability options:
 - a. Available in all locations by default
 - b. Not available in all locations by default
6. Choose Record Declaration Roles for Declaring Records:
 - a. All list contributors and administrators

- b. Only list administrators
 - c. Only policy actions
7. Choose Record Declaration Roles for Undeclaring Records:
- a. All list contributors and administrators
 - b. Only list administrators
 - c. Only policy actions

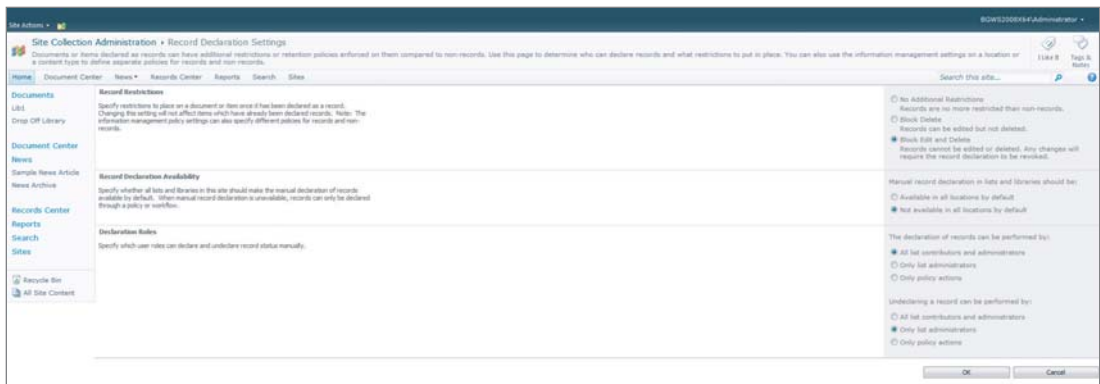



FIGURE 8-2

After in-place records management has been configured, it is possible to override records declaration settings at the document library level by navigating to the List Settings page for the library. If record declaration availability was set to not become available in all locations by default, then it must be enabled for each library where it will be used.

 From the Record Declaration Settings page for a library, you can also enable items to automatically become declared records when they are added to the library.

Manually Declaring and Undeclaring a Record

You can declare records manually by using the Compliance Details dialog (see Figure 8-3). Simply perform the following actions:

1. From the document library, select Compliance Details from the Edit Control Block (see Figure 8-4) for the document you want to declare as a record.
2. From the Compliance Details dialog, select Declare as a Record and click OK when requested to confirm the action.

You can undeclare a record in much the same way, instead of selecting Undeclare a Record.

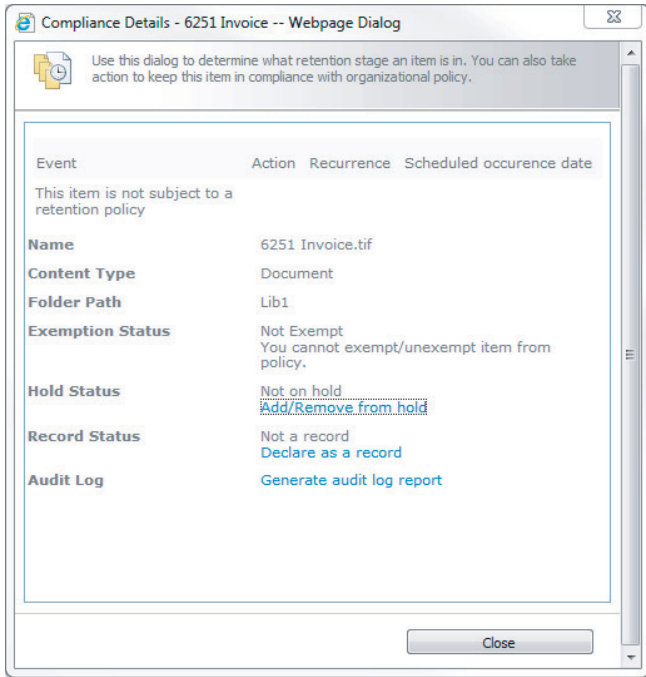


FIGURE 8-3

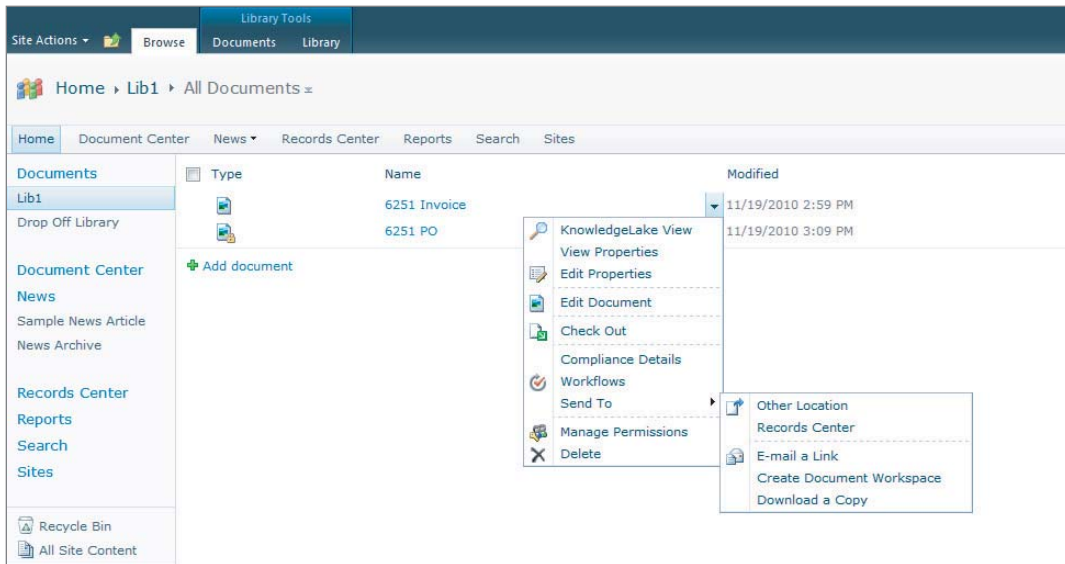


FIGURE 8-4

*You can declare multiple records simultaneously by selecting multiple documents and then choosing **Declare Record** from the Ribbon menu.*

Records Center

SharePoint Server provides a specialized site template called the Records Center that can be used to create a site tailored to the purpose of archiving and managing records. A Records Center site provides and pre-enables various features such as the Records Center landing page (see Figure 8-5), the Content Organizer feature, and the Hold and eDiscovery feature.

The Records Center landing page enables administrators to provide an overview of the records management policies. It contains links to other compliance locations, shows any records pending submission, and includes a Submit a Record button that is tied directly to the Drop-Off Library upload form for the Content Organizer.

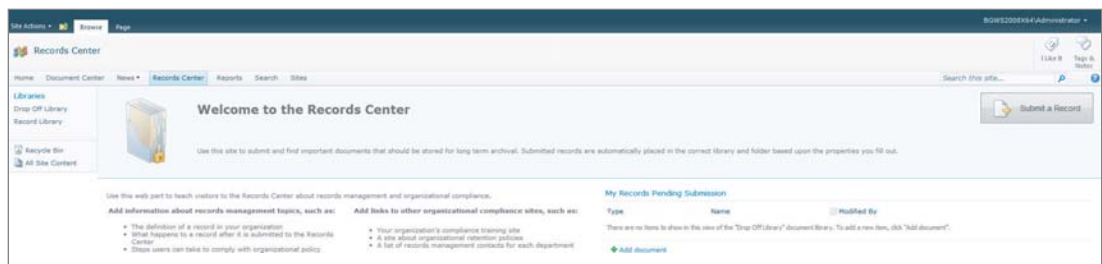


FIGURE 8-5

Records Center Management

Configuring the records center can be easily accomplished through the Records Center Management page, which is provided when a Records Center site is created. This page, which is displayed in Figure 8-6, provides step-based instructions for setting up the records center, as well as links to perform common records management tasks. One of these common tasks is the creation of a record library, which is a retention library that has automatic record declaration pre-enabled.

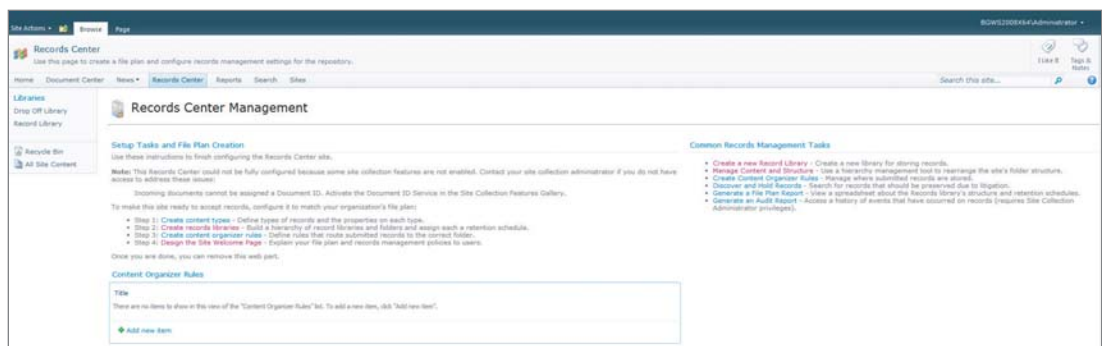


FIGURE 8-6

You can access the Records Center Management page by clicking Site Settings ⇄ Site Administration ⇄ Manage Records Center, or through the Site Actions dropdown located on the Ribbon menu.

Managing a Records Center Connection

When leveraging a Records Center site for records management, a common task is the routing of a document to the records center so that it can become an actively managed record. One approach to accomplishing this is to allow users to manually select individual documents to be moved to the records center through the Send To menu option in the Edit Control Block for a document, as shown earlier in Figure 8-4. SharePoint Server makes this possible by allowing users to actively manage the locations that appear in the Send To menu.

Creating a Connection

You can create a connection in the Send To menu by performing the following steps from Central Administration:

- 1.** Under General Application Settings, select Configure Send To Connections to go to the configuration dialog (see Figure 8-7).
- 2.** Select Web Application where Send To Connection should be displayed.
- 3.** Select New Connection from the Send To Connections list.
- 4.** Provide the Display Name.
- 5.** Provide a link to the Official File Web Service for a site where a Drop Off Library is located. This URL for this link can be found in the Content Organizer settings page discussed in the note in the Content Organizer section later in this chapter.
- 6.** Select whether the link should be displayed in the Send To menu.
- 7.** Select the desired action for when the connection is used:
 - a.** Copy: Creates a copy of the document and sends it to the routing location
 - b.** Move: Moves the document to the routing location and deletes it from the current location
 - c.** Move and Leave a Link: Moves the document to the routing location and leaves a link that will be tied to the new document location
- 8.** Click Add Connection to create the new connection.

Modifying a Connection

You can modify a connection in the Send To menu by performing the following steps from Central Administration:

- 1.** Under General Application Settings, select Configure Send To Connections to go to the configuration dialog (see Figure 8-7).
- 2.** Select the existing connection from the Send To Connections list.
- 3.** Change the desired settings for the connection.
- 4.** Click OK.

Deleting a Connection

To delete a connection in the Send To menu, perform the following steps from Central Administration:

1. Under General Application Settings, select Configure Send To Connections to go to the configuration dialog (see Figure 8-7).
2. Select the existing connection from the Send To Connections list.
3. Select Remove Connection to remove the existing connection.

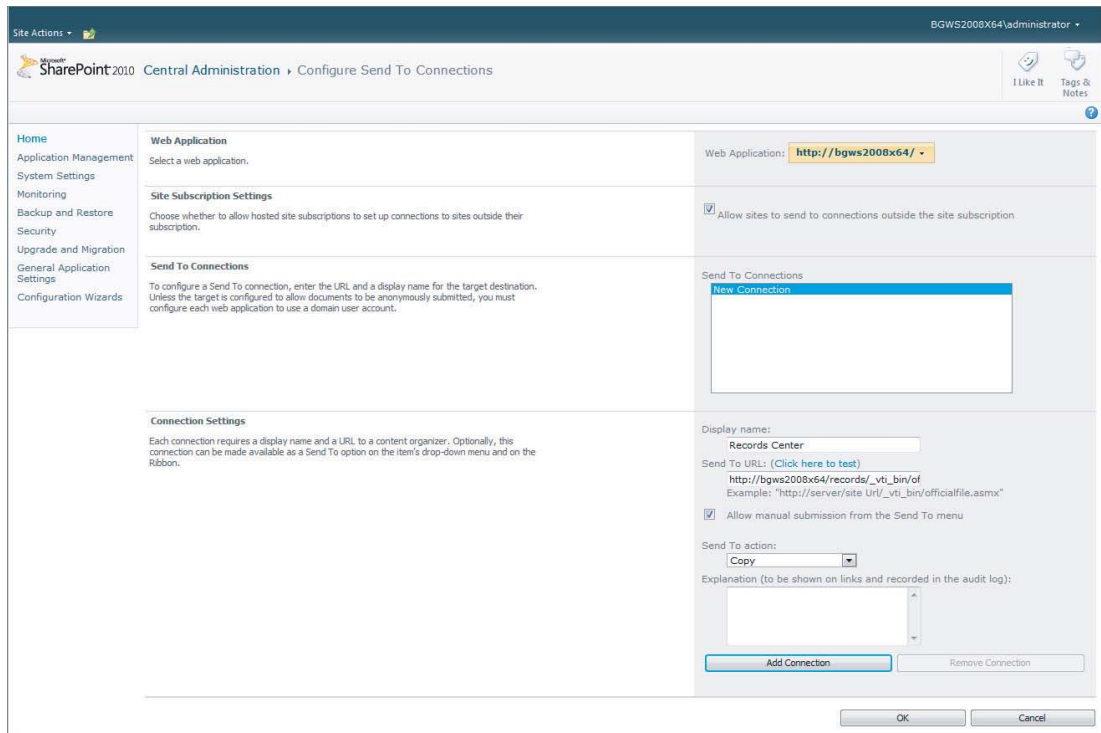


FIGURE 8-7

Content Organizer

Using the Content Organizer, you can route documents from a central location called the Drop Off Library to a target destination based on routing rules, which are configured by users with the appropriate level of permissions. While the Content Organizer can be used wherever document routing makes sense, one scenario in which the Content Organizer is extremely useful is in routing documents to a Records Center for archival purposes.

In order to use the Content Organizer, you must enable the feature from the Manage Site Features page of a SharePoint site. Once this feature is enabled, SharePoint automatically creates a special document library called the Drop Off Library and enables two links, Content Organizer Settings

and Content Organizer Rules, under the Site Administration section of the Site Settings page for the site where the Content Organizer was enabled.

Configuring the Content Organizer

After enabling the Content Organizer, you should first configure its various settings, shown in Figure 8-8. It is important to note that the Content Organizer is enabled on a per-site basis, so this configuration will need to be accomplished per each site where the Content Organizer is enabled. To configure the Content Organizer, perform the following steps:

1. Navigate to Site Settings and select Content Organizer Settings from the Site Administration section.
2. Select whether to require users to use the Drop Off Library or to allow them to upload directly to individual libraries. If this option is enabled, the upload form for each library will automatically redirect to the Drop Off Library upload form.
3. Select whether to allow rules to send documents to another site.
4. Select whether to create subfolders based on the number of items in each folder. This is referred to as *folder partitioning*.
5. Select whether to save the original audit log and properties for submitted content.
6. Specify the users who are allowed to create routing rules.

The screenshot shows the 'Content Organizer: Settings' page in SharePoint. The page is titled 'Records Center > Site Settings > Content Organizer: Settings'. Below the title, there is a search bar and navigation links. The main content area is divided into several sections, each with a description and a checkbox for configuration:

- Redirect Users to the Drop Off Library:** A checkbox is checked. Description: 'When this setting is enabled, users are redirected to the Drop Off Library when they try to upload content to libraries in this site that have one or more content organizer rules pointing to them. If this setting is disabled, users can always bypass the organizer and upload files directly to a library or folder.' A note states: 'Users will never be redirected to the Drop Off Library when organizing pages.'
- Sending to Another Site:** A checkbox is unchecked. Description: 'If there are too many items to fit into one site collection, enable this setting to distribute content to other sites that also have content organizers.'
- Folder Partitioning:** A checkbox is checked. Description: 'The organizer can automatically create subfolders once a target location exceeds a certain size.' Input fields: 'Number of items in a single folder: 2500', 'Format of folder name: Submitted after %1', and a note: '%1 is replaced by the date and time the folder is created.'
- Duplicate Submissions:** Two radio buttons are present: 'Use SharePoint versioning' (unchecked) and 'Append unique characters to the end of duplicate filenames' (checked). Description: 'Specify what should occur when a file with the same name already exists in a target location. If versioning is not enabled in a target library, the organizer will append unique characters to duplicate submissions regardless of the setting selected here.'
- Preserving Context:** A checkbox is checked. Description: 'The organizer can save the original audit logs and properties if they are included with submissions. The saved logs and properties are stored in an audit entry on the submitted document.'
- Rule Managers:** A text input field contains 'BGWS2008X64Administrator'. Description: 'Specify the users who manage the rules and can respond when incoming content doesn't match any rule. Rule Managers must have the Manage Web Site permission to access the content organizer rules list from the site settings page.' Checkboxes: 'E-mail rule managers when submissions do not match a rule' (checked) and 'E-mail rule managers when content has been left in the Drop Off Library' (checked). Input field: 'Number of days to wait before sending an e-mail: 3'.
- Submission Points:** Description: 'Use this information to set up other sites or e-mail messaging software to send content to this site.' Fields: 'Web service URL: http://bgws2008x64/records/_vti_bin/OfficialFile.asmx', 'E-mail address: recordscenter@', and a link: 'Configure the organizer's incoming email settings'.

At the bottom of the page, there are 'OK' and 'Cancel' buttons.

FIGURE 8-8



At the bottom of the Content Organizer Settings page is a reference to the Web Service URL for the Official File Web Service. This reference is the same URL that should be used in the configuration of a Send To Connection for a Records Center in order to route records to their appropriate destination.

Creating Routing Rules

Once the Content Organizer has been configured, the next step is to create the rules it should use regarding how and where to send documents. Routing rules are stored in a special site list called Content Organizer Rules, which is created when the Content Organizer is enabled. To access this list, select Site Settings ⇨ Site Administration ⇨ Content Organizer Rules. Only users who were configured as rule managers have permission to navigate to this location.

Content Organizer Rules enable documents to be routed according to the content type to which the document is set. These rules can be even further refined to also route based on property-based conditions, which are configured for the rule's configured content type. When a rule is created, the rule's configured content type is automatically added to the Drop Off Library to support the selection of this content type during uploading. However, in order for the destination library to be selected within a rule, the library must have the content type already configured. Figure 8-9 shows the dialog used to create a Content Organizer Rule, while Table 8-2 outlines the rule properties and how they are used.

TABLE 8-2: Content Organizer Rule Properties

| PROPERTY | DESCRIPTION |
|--|---|
| Rule Name | Name used to identify the rule |
| Priority | Used to set rule precedence if matching conditions are found. It is also possible to set a rule to inactive, which is very useful for proving a rule was in place even if it is no longer in use. |
| Submission's Content Type | Determines the content type for which the rule applies |
| Properties Used in Conditions | Determines the property-based conditions that apply for the selected content type |
| Aliases | Indicates the equivalent content type when it is known by another name in another location |
| Target Location | Specifies the path to the routing destination |
| Property for Automatic Folder Creation | Allows for the automatic creation and naming of folders based on unique values of a property |

Content Organizer Rules: New Rule

Rule Name *
Describe the conditions and actions of this rule. The rule name is used in reports about the content of this site, such as a library's File Plan Report.

Name:

Rule Status And Priority *
Specify whether this rule should run on incoming documents and what the rule's priority is. If a submission matches multiple rules, the router will choose the rule with the higher priority.

Status:
 Active
 Priority:
 Inactive (will not run on incoming content)

Submission's Content Type *
By selecting a content type, you are determining the properties that can be used in the conditions of this rule. In addition, submissions that match this rule will receive the content type selected here when they are placed in a target location.

Content type:
 Group:
 Type:
 Alternate names:
 This content type has alternate names in other sites:
 Add alternate name:
 Note: Adding the type "*" will allow documents of unknown content types to be organized by this rule.
 List of alternate names:

Conditions
In order to match this rule, a submission's properties must match all the specified property conditions (e.g. "If Date Created is before 1/1/2000").

Property-based conditions:

Target Location *
Specify where to place content that matches this rule.

Destination:
 Example: /sites/DocumentCenter/Documents/
 Automatically create a folder for each unique value of a property:
 Select a property (must be a required, single value property):
 Specify the format for the folder name:

 When the folder is created:
 %1 will be replaced by the name of the property
 %2 will be replaced with the unique value for the property

FIGURE 8-9

Workflow in Recordization

When discussing implementation procedures for records management in SharePoint, it is important not to overlook SharePoint workflow. Workflow is a well-suited technology for the recordization process of records management, as it provides a way to enforce procedures through processes that are automated and streamlined by the workflow engine. Workflow is discussed extensively in Chapter 4, but it is revisited here to emphasize the capabilities built into SharePoint Designer that aid in recordization when using SharePoint workflow.

SharePoint Designer provides three out-of-the-box activities for recordization. One of the activities provides the capability to send document sets to a records center. The other two activities enable documents to be declared or undeclared as records. Figure 8-10 shows these activities being used in SharePoint Designer.

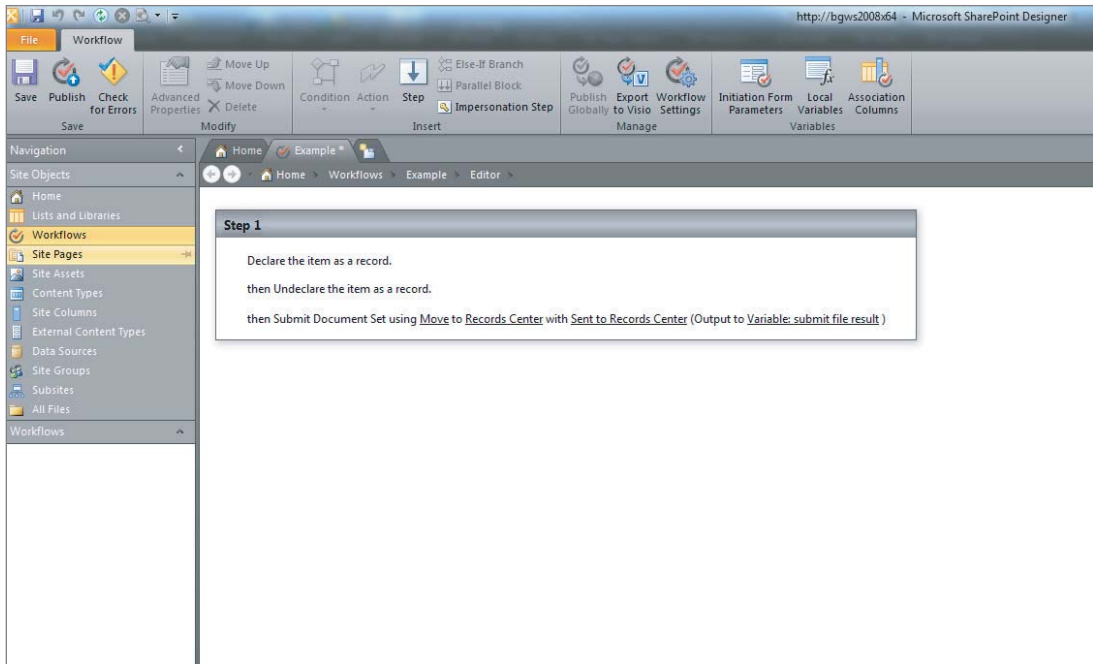


FIGURE 8-10

Programming Model for Recordization

SharePoint Server provides an extensive programming model for interacting with SharePoint’s records management functions programmatically. Table 8-3 provides an overview of some classes commonly used in recordization. These classes are found in the `Microsoft.Office.RecordsManagement.RecordsRepository` namespace, which is located in the `Microsoft.Office.Policy` assembly.

TABLE 8-3: Commonly Used Classes for Recordization

| CLASS | DESCRIPTION |
|---|---|
| <code>DocumentRouterAutoFolderSettings</code> | This class is used to configure the auto-folding setting for the Content Organizer. |
| <code>EcmDocumentRouter</code> | This class represent an instance of a Content Organizer. |
| <code>EcmDocumentRouterRule</code> | This class represents a routing rule for the Content Organizer. |

continues

TABLE 8-3 (continued)

| CLASS | DESCRIPTION |
|------------------------------|---|
| EcmDocumentRoutingWeb | This class represents a SharePoint site that has the Drop Off Library enabled and therefore exposes the <code>DropOffZoneUrl</code> property. This URL represent the end-point of the Official File Web Service, which can be used to submit documents remotely for document routing. |
| ICustomRouter | This interface provides the capability to create custom routing logic for a Content Organizer. |
| IRecordDeclarationHandler | This interface provides the capability to create custom processing logic for record declaration. |
| IRecordUndeclarationHandler | This interface provides the capability to create custom processing logic for undeclaring a record. |
| RecordDeclarationPermissions | This enumeration is used to represent the permission level required for record declaration. |
| Records | This class provides the main functionality for performing record-related actions. |

Programmatically Declaring a Record

Listing 8-1 provides an example of programmatically declaring and undeclaring a record using the `Records` object. The listing begins by getting a reference to the desired document library as a `SPList` object. After getting this reference, it is passed to the static method of `IsInPlaceRecordsEnabled` to determine if records can be declared within the library. The listing then gets a reference to an item within the library and calls `IsRecord` to determine whether the item is already a declared record. If it is, it calls the `UndeclareItemAsRecord` method to set the item to no longer be a record. Otherwise, `DeclareItemAsRecord` is called to make the item a record.



LISTING 8-1: Declaring and Undeclaring a Record

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.RecordsManagement.RecordsRepository;

namespace Listing0801
{
    class Program
    {
        static void Main(string[] args)
        {
```


LISTING 8-2 *(continued)*

```

        Assembly.GetExecutingAssembly().FullName,
        typeof(CustomRecordDeclarationHandler).FullName);

        Console.WriteLine(string.Format(
            "CustomRecordDeclarationHandler registered at {0}", site.Url));
    }
}

RecordOperationResult IRecordDeclarationHandler.OnDeclare(SPListItem item)
{
    //If item does not have a title, do not allow record declaration
    return (string.IsNullOrEmpty(item.Title)) ?
        RecordOperationResult.CancelRecordProcessing :
        RecordOperationResult.ContinueRecordProcessing;
}
}
}

```

Programmatically Managing Routing Rules

Using the SharePoint Server object model, it is possible to programmatically manage content type routing rules. This is demonstrated in Listing 8-3.

**LISTING 8-3: Creating Content Type Routing Rules**

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.RecordsManagement.RecordsRepository;

namespace Listing0803
{
    class Program
    {
        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008x64"))
            using (SPWeb web = site.OpenWeb())
            {
                EcmDocumentRoutingWeb routingSite = new EcmDocumentRoutingWeb(web);

                //Get target objects
                SPContentType targetContentType =
                    web.ContentTypes["ContentTypeName"];
                SPList targetLibrary = web.Lists["LibraryName"];

                //Ensure target library has the target contenttype for the rule
                if (targetLibrary.ContentTypes
                    .BestMatch(targetContentType.Id) == null)
            }
        }
    }
}

```


6. Select whether to Enable Retention for this policy, and configure if enabled.
7. Select whether to Enable Auditing for this policy, and configure if enabled.
8. Select whether to Enable Barcodes for this policy, and configure if enabled.
9. Select whether to Enable Labels for this policy, and configure if enabled.

The screenshot shows the 'Edit Policy' dialog box in SharePoint. The dialog is titled 'Edit Policy' and is part of the 'Policies' section. It contains several sections for configuration:

- Name and Administrative Description:** This section includes a 'Name' text box and an 'Administrative Description' text box. The description states: 'The name and administrative description are shown to list managers when configuring policies on a list or content type.'
- Policy Statement:** This section includes a 'Policy Statement' text box. The description states: 'The policy statement is displayed to end users when they open items subject to this policy. The policy statement can explain which policies apply to the content or indicate any special handling or information that users need to be aware of.'
- Retention:** This section includes an 'Enable Retention' checkbox. The description states: 'Schedule how content is managed and disposed by specifying a sequence of retention stages. If you specify multiple stages, each stage will occur one after the other in the order they appear on this page.'
- Auditing:** This section includes an 'Enable Auditing' checkbox. The description states: 'Specify the events that should be audited for documents and items subject to this policy.'
- Barcodes:** This section includes an 'Enable Barcodes' checkbox. The description states: 'Assigns a barcode to each document or item. Optionally, Microsoft Office applications can require users to insert these barcodes into documents.'
- Labels:** This section includes an 'Enable Labels' checkbox. The description states: 'You can add a label to a document to ensure that important information about the document is included when it is printed. To specify the label, type the text you want to use in the "Label format" box. You can use any combination of fixed text or document properties, except calculated or built-in properties such as GUID or CreatedBy. To start a new line, use the "\n" character sequence.'

At the bottom of the dialog, there are 'OK' and 'Cancel' buttons.

FIGURE 8-11

Creating a Policy for a Content Type

When configuring policies at the document library level, you can either select policies from the Site Collection Policies list or they must be created for the content type for which they should be applied. To configure an information management policy for a content type within a document library, follow these steps:

1. Navigate to Library Settings and select Information Management Policy Settings.
2. From the Information Management Policy Settings page, select a Content Type to create a policy (see Figure 8-12).
3. Select whether to apply an existing site collection policy or create a new policy (see Figure 8-13). If you are creating a new policy, continue. Otherwise, Click OK. The name of the policy will already be set to the content type name.
4. Provide an Administrative Description, which is visible by users with Manage Lists rights in SharePoint.

5. Provide a Policy Statement, which will displayed to users when they are accessing items that use this policy.
6. Select whether to Enable Retention for this policy, and configure if enabled.
7. Select whether to Enable Auditing for this policy, and configure if enabled.
8. Select whether to Enable Barcodes for this policy, and configure if enabled.
9. Select whether to Enable Labels for this policy, and configure if enabled.

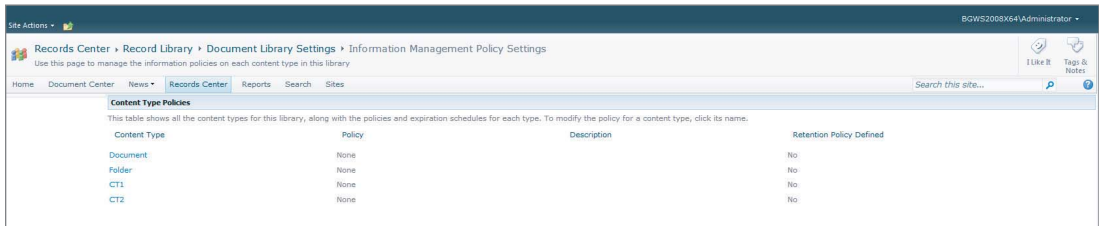


FIGURE 8-12

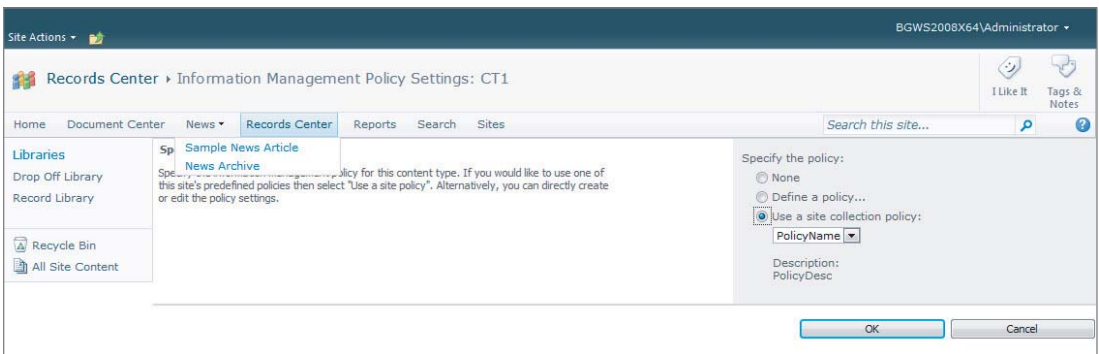


FIGURE 8-13

Creating a Retention Policy for a Document Library

SharePoint enables you to apply retention policies to an entire document library, rather than each individual content type. By enabling the Library and Folder Based Retention feature from the Site Collection Features page, the option to set policy for the entire document library is made available in the Information Management Settings page for the document library, as shown in Figure 8-14.

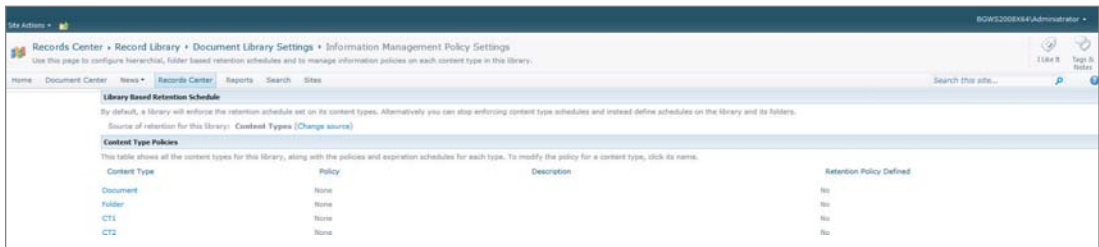


FIGURE 8-14

To set retention for an entire document library, follow these steps:

1. From the Information Management Settings page of the document library, select the Change Source option.
2. Select Library and Folders as the source of retention, as shown in Figure 8-15.
3. Configure retention schedule by clicking Add a Retention State.
4. Click OK.

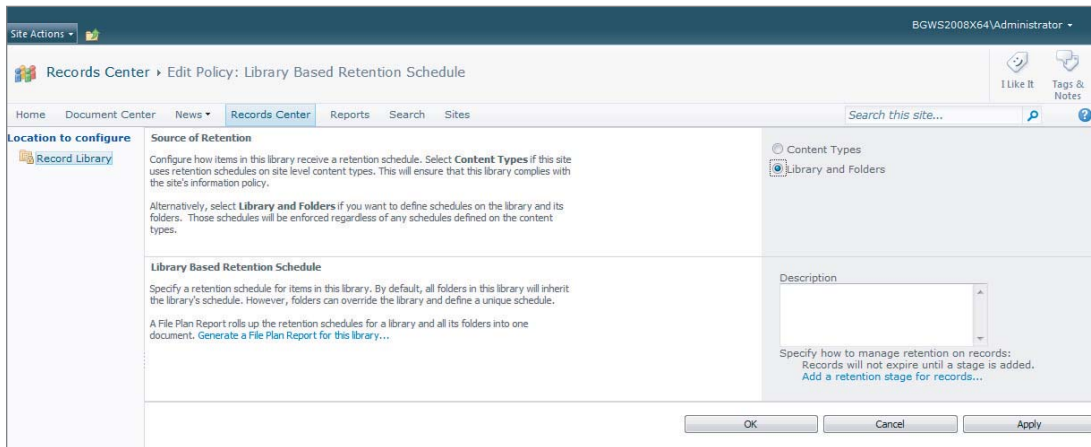


FIGURE 8-15

Exporting and Importing Policy Settings

Using SharePoint, you can export policies created at the site collection level so that they can be used in other site collections and therefore do not have to be recreated again manually.

Exporting a Policy

When policies are exported from SharePoint, they are exported in XML format. After creating a policy, it can be exported by performing the following actions:

1. Navigate to Site Settings and select Site Collection Policies under the Site Collection Administration section.
2. Select an existing policy to be exported.
3. Click Export.
4. Save the file when prompted.

Importing a Policy

When policies are imported into SharePoint, they are imported in XML format. Follow these steps to import a policy:

1. Navigate to Site Settings and select Site Collection Policies under the Site Collection Administration section.

2. Click Import.
3. Browse to an existing policy XML file, select it, and click Open.
4. Click Import.

Programming Model for Information Management Policy

SharePoint Server provides an extensive programming model for interacting with an information management policy. Table 8-4 provides a list of the commonly used classes from this model, which are located in the `Microsoft.Office.RecordsManagement.InformationPolicy` namespace of the `Microsoft.Office.Policy` assembly.

TABLE 8-4: Commonly Used Classes for Information Management Policy

| CLASS | DESCRIPTION |
|--------------------------------------|---|
| <code>ListPolicySettings</code> | Represents policy settings for list-based retention |
| <code>Policy</code> | Represents a single policy |
| <code>PolicyCatalog</code> | Represents all policy collections for a site collection |
| <code>PolicyCollection</code> | Represents a collection of policies |
| <code>PolicyFeature</code> | Represents a single feature of a policy |
| <code>PolicyFeatureCollection</code> | Represents a collection of policy features for a policy |
| <code>PolicyItem</code> | Represents the setting for an individual policy feature |
| <code>PolicyItemCollection</code> | Represents a collection of policy items in a policy |

RETENTION

Within SharePoint, document retention is managed through an information management policy by creating policies that have a retention schedule. A retention schedule specifies how long a document is retained and what actions should be taken when the document has reached the end of its retention period. The action that takes place following the retention expiration is referred to as disposition. For disposition, users can configure an action to move the document to the recycle bin, delete the document permanently, transfer the document to another location for archiving, or continue to the next stage of retention. While retention can be managed for all types of items, it is especially relevant to records management, as compliance often indicates how long a document should be kept and what actions must be performed until expiration.

Creating Retention Schedules

When creating retention schedules, SharePoint supports the concept of multi-stage retention. This means that you can specify, through a policy, multiple retention schedules for a single policy. As documents expire from one stage of retention, they move into the next stage until fully expired after the last stage of retention.

To create a retention schedule, perform the following steps (see Figure 8-16):

1. Open an existing policy or create a new policy.
2. Enable Retention by clicking the appropriate checkbox.
3. Click Add a Retention Stage for Records.
4. Select the Retention Event that indicates when the retention period should expire and the Retention Action should execute. The retention period can be based on the following items plus any number of years, months, or days:
 - A. Created: Period begins when the document is created
 - B. Modified: Period begins when the document is modified
 - C. Declared a Record: Period begins when the document is declared a record
5. Select Retention Action, which tells SharePoint what to do upon expiration.
6. Select Stage Recurrence if allowed by Retention Action. This allows the current stage to be repeated until the next stage is activated.

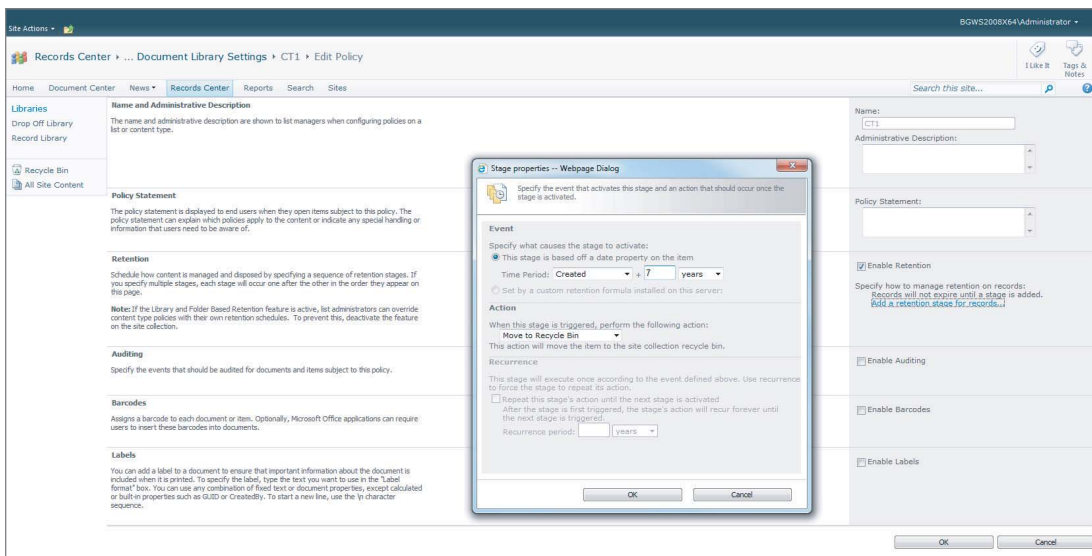


FIGURE 8-16

Programmatically Creating Retention Schedules

It is possible to create retention schedules programmatically using the programming model for an information management policy. Listing 8-4 demonstrates the creation of a policy with a specific retention schedule and the application of the policy to the content type in a document library. Notice that retention schedules are defined using XML. This XML schema is fairly easy to understand, as it closely mirrors the dialog in Figure 8-16.


LISTING 8-4: Creating a Retention Schedule

Available for
download on
Wrox.com

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SharePoint;
using Microsoft.Office.RecordsManagement.InformationPolicy;
using Microsoft.Office.RecordsManagement.PolicyFeatures;

namespace Listing0804
{
    class Program
    {
        private const string RETENTION_XML =
"<Schedules nextStageId=\"2\">
  <Schedule type=\"Default\">
    <stages>
      <data stageId=\"1\">
        <formula
id=\"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Formula
.BuiltIn\">
          <number>7</number>
          <property>_vti_ItemDeclaredRecord</property>
          <propertyId>f9a44731-84eb-43a4-9973-cd2953ad8646</propertyId>
          <period>years</period>
        </formula>
        <action type=\"action\"
id=\"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action
.Delete\" />
      </data>
    </stages>
  </Schedule>
</Schedules>";

        static void Main(string[] args)
        {
            using (SPSite site = new SPSite("http://bgws2008x64/records/"))
            using (SPWeb web = site.OpenWeb())
            {
                //Get List and ContentType
                SPList list = web.Lists["Record Library"];
                SPContentType contentType = list.ContentTypes["CT1"];

                //Attempt to get Policy
                Policy policy = Policy.GetPolicy(contentType);

                //Create new Policy if it does not exist
                if (policy == null)
                {
                    Policy.CreatePolicy(contentType, null);
                }
            }
        }
    }
}

```

continues

Site Actions BGWS2008X64\Administrator

Site Collection Administration > Policies > PolicyName > Edit Policy

Home | Document Center | News | Records Center | Reports | Search | Sites

Search this site...

Documents
Lib1
Drop Off Library

Document Center
News
Sample News Article
News Archive

Records Center
Reports
Search
Sites

Recycle Bin
All Site Content

Name and Administrative Description
The name and administrative description are shown to list managers when configuring policies on a list or content type.

Name:
PolicyName
Administrative Description:
Description

Policy Statement
The policy statement is displayed to end users when they open items subject to this policy. The policy statement can explain which policies apply to the content or indicate any special handling or information that users need to be aware of.

Policy Statement:
Statement

Retention
Schedule how content is managed and disposed by specifying a sequence of retention stages. If you specify multiple stages, each stage will occur one after the other in the order they appear on this page.

Note: If the Library and Folder Based Retention feature is active, list administrators can override content type policies with their own retention schedules. To prevent this, deactivate the feature on the site collection.

Enable Retention

Auditing
Specify the events that should be audited for documents and items subject to this policy.

Enable Auditing

Specify the events to audit:

- Opening or downloading documents, viewing items in lists, or viewing item properties
- Editing items
- Checking out or checking in items
- Moving or copying items to another location in the site
- Deleting or restoring items

Barcodes
Assigns a barcode to each document or item. Optionally, Microsoft Office applications can require users to insert these barcodes into documents.

Enable Barcodes

Labels
You can add a label to a document to ensure that important information about the document is included when it is printed. To specify the label, type the text you want to use in the "Label format" box. You can use any combination of fixed text or document properties, except calculated or built-in properties such as GUID or CreatedBy. To start a new line, use the \n character sequence.

Enable Labels

OK Cancel Export... Delete

FIGURE 8-17

Reporting

SharePoint provides the capability to generate a file plan report for documenting the structure and policies implemented within a document library, as well as a number of audit reports that detail content usage information as indicated by the information management policy.

Audit Reports

SharePoint allows audit reports to be accessed from two separate locations. One, you can select Site Settings ⇨ Site Collection Administration ⇨ Audit Log Reports. Alternately, you can use the Compliance Details dialog, which can be opened from the Edit Control Block of a document in a document library. Both of these methods take you to the same administration page, where you can access the reports listed in Table 8-5.

TABLE 8-5: Audit Log Reports

| REPORT NAME | DESCRIPTION |
|-------------------------------------|--|
| Content Modifications | Displays all events that modified content |
| Content Type and List Modifications | Displays all events that modified content types and lists |
| Content Viewing | Displays all events for which a user viewed content |
| Deletion | Displays all events that caused content to be deleted or restored from the Recycle Bin |
| Custom | Displays report based on manually specified filters |
| Expiration and Disposition | Displays all events related to the expiration and disposition of content in this site |
| Policy Modifications | Displays all events related to the creation and use of policies on content |
| Auditing Settings | Displays all events that changed the auditing settings of SharePoint Foundation |
| Security Settings | Displays all events that changed the security configuration of SharePoint Foundation |

File Plan Report

File plan reports can be accessed by navigating to the List Settings for a document library and clicking the link titled Generate File Plan Report. The file plan report is useful for documenting and validating records management policies and settings. This report shows the content types that exist in the list, list settings, record declaration settings, retentions details, folder-level details, and a lot of other useful information.

EDISCOVERY

eDiscovery, or electronic discovery, is the process that an enterprise goes through to find and preserve documents when the organization is involved in some sort of legal litigation, investigation, or audit. eDiscovery consists mainly of two acts. The act of finding the necessary records based on some criteria is referred to as *searching*, whereas the act of preserving the records in their current form as a unit is referred to as a *hold*. In order to leverage eDiscovery in SharePoint, the Hold and eDiscovery feature must be enabled within Site Features.

To find content that should be placed on hold, users can either manually locate the items and place them on hold or they can configure an eDiscovery search that can be used to automatically find and place holds on content that meets specific criteria. As shown in Figure 8-18, this is accomplished by performing the following actions:

1. Navigate to Site Settings and select Discover and Hold Content.

2. Select a site to search for content.
3. Provide a query using keyword syntax to find content within the selected site.
4. Select whether to hold the documents in place or copy them to a new location and hold them.
5. Select or create a new hold to be used.
6. Select Add Results to Hold.



FIGURE 8-18

Once holds are placed on documents, the content that is placed on hold can be reviewed through hold reports. Hold reports are accessed from Site Settings under the Hold and eDiscovery section.

SUMMARY

Records management represents the central pillar of enterprise content management, as it is responsible for the methodical and consistent management of an organization's records. An organization must leverage good records management policies and practices to ensure that it is operating according to the increasing amount of compliance scenarios, legal litigation and audit requirements, and good business practices.

SharePoint provides a strong set of foundational components that cover the general scenarios to aid in organizational compliance and the management of records. Records can be managed in place with active documents, within a centralized archival location using the records center, or using a hybrid approach. Recordization is provided through manual user actions, the Content Organizer, or built-in workflow actions. SharePoint provides robust support for creating information management policies, which not only outline which documents should be handled and how, but also form a central component for configuring and enforcing auditing and retention requirements. If an enterprise faces some sort of litigation or audit, it can easily leverage SharePoint's built-in facilities for performing eDiscovery.

While SharePoint Server provides a lot of built-in functionality for enterprise records management, it also offers extensive programming models for adapting SharePoint to meet any custom needs.



Digital Asset Management

WHAT'S IN THIS CHAPTER?

- Understanding the components
- Designing digital asset management solutions
- Implementing digital asset management taxonomy
- Managing the content storage
- Tuning performance for digital assets

A digital asset is defined as a graphic, audio, or video file or other fragment of rich content that is used by an organization. Digital asset management (DAM) is the process of managing the life cycle of this special content.

Years ago, it was believed that no single application would provide a complete content management system that could serve as the broad-spectrum solution that SharePoint 2010 has become today. Surely the same platform that provides web content management (WCM) capabilities could not also serve as a DAM system. Indeed, Microsoft has accomplished this feat with SharePoint 2010. SharePoint's infrastructure provides for storage, retrieval, governance, and life cycle management of digital assets. Going beyond the standard content management capabilities, SharePoint Server 2010 provides special out-of-the-box components such as the asset document library, site columns, content types, and Web Part components that facilitate management of graphic or multimedia assets. Of course, SharePoint workflow rounds out the feature requirements of any good DAM system.

Entire books have been written about DAM concepts. While space limitations prevent this chapter from a deep discussion of DAM theory, it does provide guidance for implementing a DAM solution using SharePoint Server 2010. After the core components are described, common DAM solution scenarios are discussed. Finally, because digital assets are often very large media files, storage architecture and performance optimization is addressed.

SHAREPOINT SERVER 2010 DIGITAL ASSET MANAGEMENT COMPONENTS

Traditional content management capabilities in SharePoint 2010 are extended to provide an out-of-the-box DAM solution. The included asset library, site columns, content types, and media Web Parts are the foundation of digital asset management in SharePoint.

The Asset Library

The SharePoint Server 2010 asset library is used to store digital assets that can be shared among users. It is a specially customized library template that is designed to use Image, Audio, and Video content types that are specifically defined for storing and cataloging rich media assets. It also takes advantage of document parsers to automatically extract metadata for image files.

The asset library supports a new preview mode that displays a thumbnail view and important metadata when the cursor is hovered over an asset. The metadata that is displayed can be modified by managing the thumbnail view in the asset library. The asset library also supports user rating metadata, which can be used to facilitate the presentation of top-rated assets.

Asset creators and managers upload, categorize, and manage assets directly in the asset library. Consumers browse the library or search for assets that can be viewed or downloaded for insertion into a given project. Microsoft Office applications such as Microsoft Word and Microsoft PowerPoint can browse and insert works from an asset library from within the client application.

Digital Asset Columns

SharePoint Server 2010 provides several built-in site columns that enable the content types used by asset libraries. Table 9-1 describes the out-of-the-box site columns available with SharePoint Server 2010.



Some of the digital asset site columns are marked as hidden and will not be visible in the site column list in site settings.

TABLE 9-1: Digital Asset Site Columns

| COLUMN NAME | DESCRIPTION |
|-------------|---|
| Name | The name of the file being uploaded or created. All document library items are required to have a name. |
| Title | The friendly name of the document being uploaded or created. This field is particularly useful when configuring a media Web Part to display a specific video. By default, the title is displayed in the media Web Part above the video. |
| Preview | A computed field for internal SharePoint use. |
| Keywords | A list of metadata words that can be used to locate the digital asset during a search operation. |

| COLUMN NAME | DESCRIPTION |
|--------------------|--|
| Thumbnail Preview | A hidden computed value used internally by SharePoint to facilitate the display of digital asset preview thumbnails. |
| Author | The name of the digital asset author. |
| Comments | Additional comment text regarding the digital asset. |
| Preview Image URL | A hyperlink to the thumbnail image for the digital asset. |
| Copyright | Text that identifies the copyright information for the digital asset. |
| Length (seconds) | The number of seconds that represent the length of the digital asset. Only applies to audio and video files. |
| Picture Size | A computed value that identifies the composite height and width of a graphic digital asset. The value is computed from the <code>ImageHeight</code> and <code>ImageWidth</code> internal field values that are automatically parsed out of the graphic file during insert and update operations. |
| Date Picture Taken | The date when the picture was taken. |
| Frame Width | The frame width value, in pixels, of a video digital asset. |
| Frame Height | The frame height value, in pixels, of a video digital asset. |

Digital Asset Content Types

SharePoint provides the Rich Media Asset, Audio, Image, and Video content types in order to facilitate the management of digital asset items. When an asset library is created, the Audio, Image, and Video content types are associated by default. Table 9-2 describes the digital asset content types.

TABLE 9-2: Digital Asset Site Columns

| CONTENT TYPE NAME | DESCRIPTION |
|-------------------|--|
| Rich Media Asset | Defines the most basic metadata fields for a digital asset. Inherits from the Document content type. |
| Audio | Defines metadata fields that are specific to audio assets. Inherits from Rich Media Asset. |
| Image | Defines metadata fields that are specific to graphic assets. Inherits from Rich Media Asset. |
| Video | Defines metadata fields that are specific to video assets. Inherits from Rich Media Asset. |

Media and Image Web Parts

SharePoint Server 2010 also provides a few very useful Web Parts and controls that enable significant no-code multimedia capabilities directly in any SharePoint Server 2010 site.

Media Web Part and Field Control

The *media Web Part* uses Silverlight to display audio and video assets. This Web Part enables a designer to add audio or video to any SharePoint page without writing any code. To deploy a media Web Part on a SharePoint 2010 page, start an edit session on the page and execute these steps:

1. Select Insert tab ⇨ Web Part.
2. From the Categories pane on the left, select Media and Content.
3. In the Web Parts pane on the right, select Media Web Part.
4. In the About the Web Part section, select the Web Part zone into which the Web Part should be inserted.
5. Click the Add button. The media Web Part will be inserted into the page (see Figure 9-1).
6. Click the media Web Part to initiate configuration. Several configuration options will become available in the Ribbon (see Figure 9-2).



FIGURE 9-1

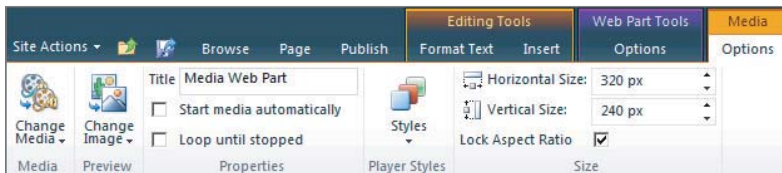


FIGURE 9-2

7. On the Media tab of the Ribbon, select Change Media ⇨ From SharePoint to choose an audio or video stored in an asset library using the Select an Asset dialog.
8. On the Media tab of the Ribbon, select Change Image ⇨ From SharePoint to choose a thumbnail graphic stored in an asset library.
9. On the Page tab of the Ribbon, select Check In to publish the page. The video may now be played by authorized users using the media Web Part (see Figure 9-3).

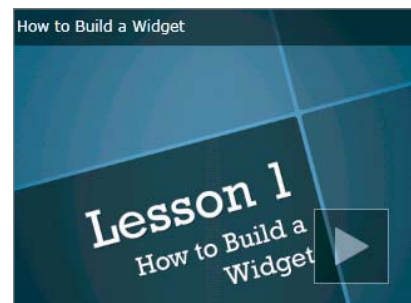


FIGURE 9-3

In addition to manually adding a media Web Part to a page, SharePoint 2010 also exposes an ECMAScript object model that enables a media player object to be constructed on-the-fly. In order to access the media player objects, it is necessary to ensure that a reference to the `mediaplayer.js` script file is included somewhere on the page. After ensuring that the `mediaplayer.js` file is included, a media player object can be constructed and a media asset can be played using just a few lines of code.

The *media field control* facilitates the playing of audio and video assets on a page. In contrast to the media Web Part, the media field control is added to a publishing page layout such that all pages created with the layout will have the media field control available. After a new page is created using the custom layout, editing the page and clicking the media field control enables the same Ribbon media configuration options that are available in the media Web Part. Table 9-3 describes the properties that can be specified as attributes of the `<PublishingWebControls:MediaFieldControl1/>` element when it is added to a layout page.

TABLE 9-3: Media Field Control Properties

| PROPERTY NAME | DESCRIPTION |
|---------------------------------|--|
| <code>AutoPlay</code> | Indicates whether or not media should play automatically after the page is loaded. |
| <code>DisplayMode</code> | Determines the <code>MediaDisplayMode</code> in which to start the media player. Options are <code>Inline</code> : the player is displayed inline with the rest of the page; <code>Overlay</code> : the player overlays the page in a pop-up; and <code>FullScreen</code> : the player is in full-screen display mode. |
| <code>Height</code> | The height of the media player when it is displayed inline with the page. |
| <code>Loop</code> | Determines whether the media should restart automatically after playing to completion. |
| <code>MediaSource</code> | The default URL reference to the media source. |
| <code>PresentationLocked</code> | Determines whether the presentation properties (<code>TemplateSource</code> , <code>Width</code> , and <code>Height</code>) are locked against editing by content authors. |
| <code>PreviewImageSource</code> | A URL reference to the media preview image. |
| <code>TemplateSource</code> | A URL reference to the XAML document containing a <code>ControlTemplate</code> that defines the media player's "skin". |
| <code>Title</code> | The title of the media. If a title value is specified by the <code>MediaFieldValue</code> , it takes precedence over this setting. |

continues

TABLE 9-3 (continued)

| PROPERTY NAME | DESCRIPTION |
|---------------|---|
| Value | The <code>MediaFieldValue</code> . |
| Width | The width of the media player when it is being displayed inline with the rest of the page. If a width value is specified by the <code>MediaFieldValue</code> , it takes precedence over this setting unless <code>PresentationLocked</code> is set to true. |

Picture Library Slideshow Web Part

The *picture library slideshow* Web Part fills a simple need but is very useful. This Web Part is able to rotate images from a picture library using a fade transition effect. After adding this Web Part to a page, a designer can control the duration of each slide and whether the slides are rotated randomly or sequentially.

Image Viewer Web Part and Field Control

The *image viewer* Web Part isn't new but it does enable the viewing of image assets. Deployment is as simple as it gets: Drop the Web Part on a page and set the URL. New for SharePoint 2010 is the capability of the image viewer Web Part to control the background color behind the image. By default, the "color" is set to transparent.

Content Query Web Part

The *content query* Web Part is a tremendously powerful tool that enables site designers to aggregate related information in an extremely flexible format. Results can be filtered, sorted, grouped, and rendered using a custom style, and can be exposed as an RSS feed. Queries can be constructed using the web interface and then further customized using SharePoint Designer.

The content query Web Part can be used to query media assets from a site. Because the results can be grouped and styled in virtually unlimited ways, it is possible to create a dynamic collection of playable media clips.

DIGITAL ASSET MANAGEMENT SOLUTION SCENARIOS

The asset library and media Web Parts, combined with content publishing, workflow, and other enterprise content management capabilities in SharePoint, provide for a wide variety of DAM solutions. The following solutions are generic scenarios that identify common real-world challenges faced by many enterprises. After reading through these examples, you should be able to modify and extend them to create personalized solutions.

Marketing and Brand Management

Marketing is a major part of any successful business. With the goal of promoting product and services to potential customers, brand awareness and thought leadership are key concepts of any marketing plan. Consider how a brand management solution could help the following entities:

- A restaurant chain that needs to standardize logos, graphics, and other marketing materials for independently owned franchises
- A department store that needs to develop and coordinate print, web, and radio broadcast advertising
- A large software vendor that needs to provide potential customers with product brochures that use a consistent theme to drive brand awareness

The marketing department for any of these organizations would be well served by a centralized DAM solution deployed as one or more SharePoint sites. SharePoint is then used to standardize the storage and identification of marketing resources (see Figure 9-4).

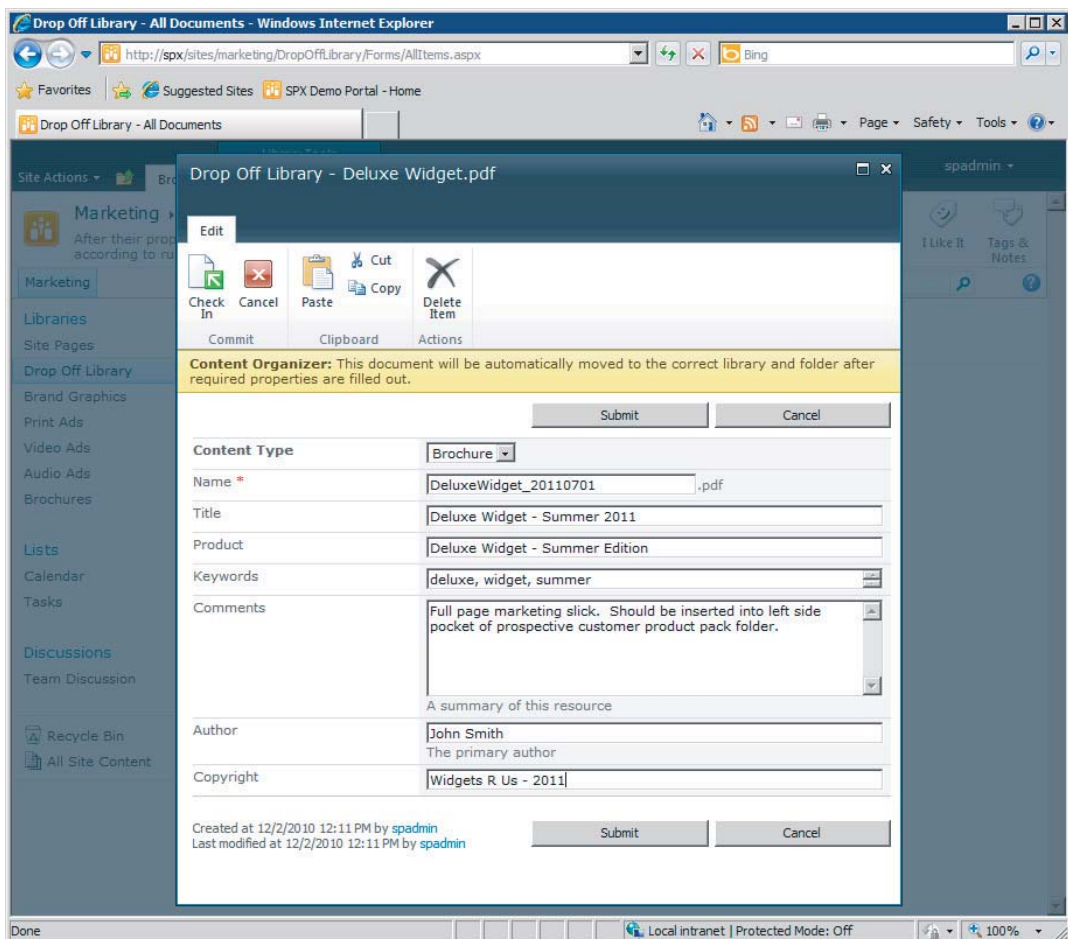


FIGURE 9-4

In this scenario, a SharePoint 2010 team site has been created for the marketing department at Widgets R Us. The site contains a Brand Graphics source library that is the approved image assets repository for all corporate logos and product graphics. Individual libraries are created to contain advertisement materials for print, audio, and video media outlets. Also, a brochures library contains product marketing materials that are provided for potential customers.

Content flows into the site through the Drop Off Library, which helps the asset manager to ensure that all metadata is present before the asset is routed to the proper asset library by the content organizer. Routing rules can be configured to use document metadata fields to place content not only in the proper asset library, but also in subfolders, perhaps based on product line.

After the creative team produces the content, assets can easily be located and retrieved from the appropriate library to be sent to the appropriate advertisement entity or, in the case of brochures, potential customers.

Media Development Project

A video game or movie production project provides a classic example of how a DAM solution can be used to compile a comprehensive digital creative product. The digital assets that are required to produce a video game are extensive:

- Storyboard graphics are created to provide a creative framework for the project.
- While custom game code will likely be kept in a specialized source control system, graphics, textures, and video cut-scene assets are created for use by the game engine.
- The game will need to be marketed, so at the very least the game developer needs to generate special graphics that can be used by the publisher to create promotional materials.
- Trailer videos are also an important part of a video game release cycle. Game footage video clips can be created and stored as media assets for later assembly as a trailer.

Obviously, this is a contrived example, but the concepts are valid. A completed media development project can use asset libraries as repositories for digital assets during the development and editing phases.

Similarly, a movie production could use SharePoint asset libraries combined with custom rich media content types to categorize and store scene content. Consider this scenario:

- Scene takes are recorded digitally throughout the day.
- At the end of the day, production assistants upload clips to a drop-off library in a movie production SharePoint site.
- Required metadata include the scene name and take number. Optional metadata includes a clip sequence number.
- After required metadata has been provided, the Content Organizer routes the video assets to an asset library.

With the daily scene takes stored in SharePoint, the production assistant launches a custom developed “build” utility on a high-powered rendering server that also happens to be another SharePoint

server in the farm. The production assistant selects the scenes that should be compiled and initiates the build process. The build application is able to use scene and clip sequence metadata to download clips and build new daily “rough cut” assets that are then automatically uploaded back to another SharePoint asset library.

Although this is a contrived scenario, it demonstrates how you can combine SharePoint’s development platform with its extensive out-of-the-box capabilities to move beyond a standard DAM application — even to create a media production management solution.

Online Training Center

Another common SharePoint asset management scenario is for training videos. Using SharePoint as a training delivery mechanism has broad implications in terms of mining SharePoint’s value. In this case, the obvious benefit is an online training center that could be created as a fee-based customer service offering.

Another similar example, and perhaps one more widely implemented, would be an online training system for new employees. Consider the scenario in which a new IT administrator is hired. Even if the new administrator is certified in specific technologies, every organization has unique procedures for managing everyday tasks. A trip through the online training center could provide that “easy first day” that the new employee is looking for (see Figure 9-5).

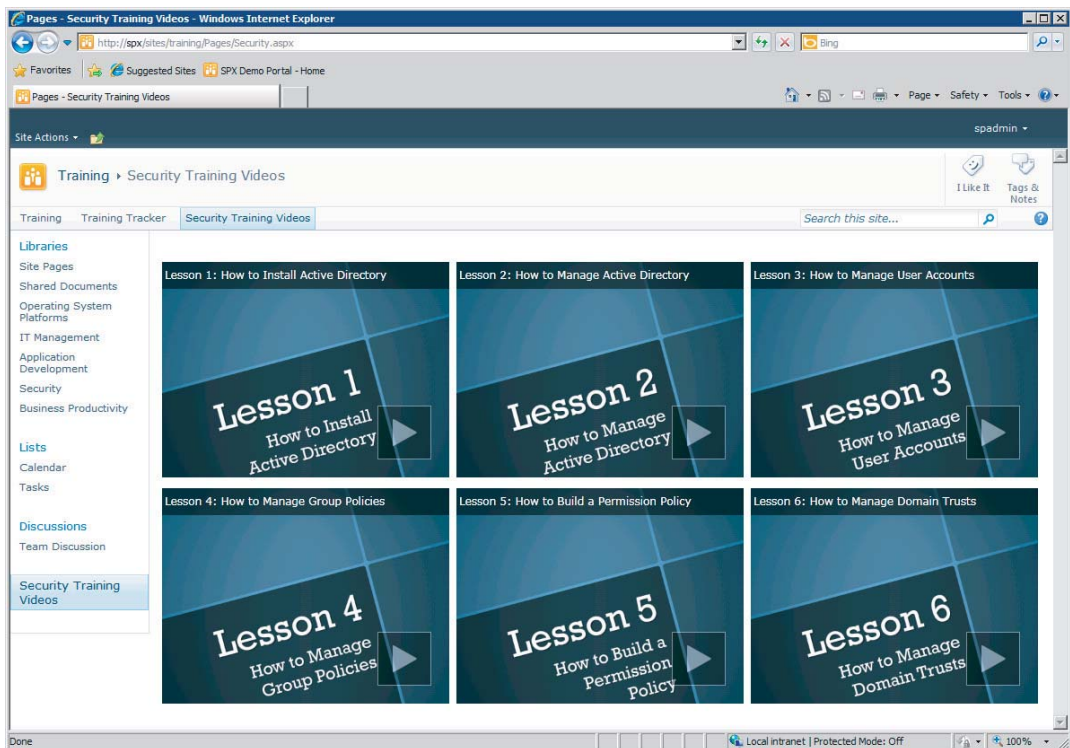


FIGURE 9-5

When a new IT administrator is hired, chances are good that the existing administration team is overworked and has little time to train the new employee. After an initial time investment with a screen capture application, however, the new employee can get up to speed on the basics quickly with little additional hand-holding.

Similarly, all new hires can view human resources and benefits videos before diving into department-specific orientation. For example, new application developers might sit through a video covering the organizational policies regarding source control, or new quality assurance testers might view a video on proper testing procedures.

Every organization has training needs at some point. If training topics have the potential to be repeated, then creating and reusing media assets and taking advantage of SharePoint DAM capabilities can provide an easy return on investment.

Audio or Video Podcasting

SharePoint social media capabilities have been significantly enhanced in SharePoint 2010. It is reasonable to expect that corporate websites that use SharePoint 2010 will expose blog and podcasting subsites. Audio and video blogs (podcasts) can be a powerful tool to both reach new customers and keep existing customers informed.

Asset libraries, like all SharePoint libraries, generate an RSS feed that can be consumed by feed readers and podcast software. In addition to providing fresh content on the company website, which is good for search engine optimization (SEO), customers can stay connected and informed about new or updated product demonstrations.

Media Resource Library

Perhaps the easiest scenario to overlook is the use of asset libraries for storing content resources that are intended to directly enhance the website on which they are hosted. Image assets and Silverlight applications most commonly fit this paradigm. By taking advantage of the publishing capabilities of SharePoint, content authors can ensure that only authorized personnel are allowed to manipulate assets that are visible to the public.

TAXONOMY CONSIDERATIONS

Taxonomy, as the term is used with respect to SharePoint architecture, is essentially the classification and organization of SharePoint containers. The SharePoint taxonomy is the hierarchical architecture of web applications, site collections, sites, lists, libraries, and folders that would be deployed for the purposes of containing content items. It also includes metadata assets such as columns and content types. Taxonomy should always be architected as efficiently and logically as possible to ensure that end users can find content when necessary. In a Digital Asset Management solution, the taxonomy should also facilitate reuse of digital assets.

Consider a scenario in which SharePoint provides the repository solution for various digital assets used by a marketing enterprise. For this type of solution, it is important to locate common assets used by all departments in a “top-level” asset library for use by members of multiple departments.

Then department-specific assets can be stored and secured in asset libraries in each department site collection. This approach minimizes duplication (and increased storage expense) for common assets such as a corporate logo graphic.

For this you can take advantage of the *Content Organizer*. Often, content creators do not apply all required metadata to an asset. When the Content Organizer feature is enabled on a site, a *Drop Off Library* is created. Assets can be uploaded to the Drop Off Library even if they are missing required metadata. An asset manager can then apply any missing required properties, which enables SharePoint to send the asset to its final destination based on *routing rules*.

You use Content Organizer routing rules to separate digital assets based on content type. Again, the Drop Off Library can serve as a consistent location for uploading digital asset content. Once all required properties have been applied, SharePoint uses routing rules to determine the proper asset library for a given asset based on its content type. By separating rich media, audio, image, video, and possibly custom assets into individual asset libraries, asset consumers can more easily navigate to the asset's location. Also, using routing rules, it is possible to send large assets, such as video files, to an entirely different site collection that is stored in another content database for storage balancing. The Content Organizer is addressed in greater detail in Chapter 8 of this book.

STORAGE CONSIDERATIONS

When planning a digital asset management solution, it is important to consider the significant impact that media assets have on the size of a content database. Because graphic, audio, and video resource file sizes are typically very large, storage architecture should be carefully planned.

Managing Content Database Size

Consider the previous scenario in which an enterprise constructs a SharePoint portal that will provide training videos covering a wide variety of topics. If the average video is 60 minutes long and 120MB in size, 850 videos would reach the 200GB maximum supported size for collaboration content databases. It is important to architect the taxonomy in such a way that content database sizes are manageable. Divide topical areas into logical site collections that are stored in their own databases. For instance, if the organization provides technical training for Microsoft technologies, it would be useful to create separate training site collections for various disciplines such as the following:

- Operating system platforms
- IT management
- Application development
- Security
- Business productivity

In addition to the benefits of a reduced content database size, the increased storage granularity will improve backup and restore times. Also, if end-user load becomes unbalanced, the content database for one or more topical areas can be moved to a dedicated SQL Server.

Remote BLOB Storage

Remote binary large object (BLOB) storage (RBS) is a SQL Server feature that enables client applications to store binary data (files) in a remote BLOB store, rather than inline in the database. SharePoint 2010 and SQL Server 2008 R2 support the RBS feature framework. When the RBS feature pack is deployed and an RBS provider is installed and enabled on a SharePoint content database, binary document data will not be stored in the content database.

RBS becomes a very powerful feature when enabled for a content database that contains digital asset data. It provides a significant benefit, particularly when digital assets are very large files such as videos or large graphics. For this reason, RBS should be strongly considered when asset libraries are expected to contain hundreds or thousands of large documents. The “Remote BLOB Storage” section of Chapter 12 covers in detail the concepts — and pros and cons — of implementing RBS.

Maximum Upload Size

By default, 50MB is the maximum file size allowed during upload for a given SharePoint 2010 web application. If a user attempts to upload a document that is larger than 50MB, the upload will fail.

Digital assets, particularly video assets, are often larger than 50MB. When deploying a SharePoint site that will contain an asset library, it is important to consider the largest possible file that an end-user is likely to upload. Keep in mind that the maximum file size is set at the web-application level, which means that all users who are allowed to upload content will be able to upload files up to the maximum file size that is set. While the maximum file size often needs to be increased for digital asset management, it can negatively affect storage on other site collections and content databases associated with the same web application. Users may take advantage of the high limit and consume storage at a faster rate than anticipated.

In order to change the maximum upload file size for a given web application, follow these steps:

1. Launch SharePoint Central Administration.
2. On the Central Administration home page, select “Manage web applications” under the Application Management section.
3. On the Web Applications Management page, select the web application.
4. On the Web Applications tab, click General Settings ⇄ General Settings.
5. Scroll down to the bottom of the Web Application General Settings dialog and set Maximum Upload Size appropriately (see Figure 9-6).

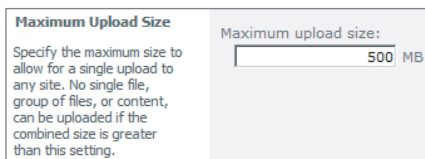


FIGURE 9-6

6. Click the OK button. The new maximum upload size is effective immediately.



Because SharePoint 2010 is an ASP.NET application, the configured file upload size for SharePoint 2010 is also subject to the `MaxRequestLength`, `ExecutionTimeout`, and `MaxAllowedContentLength` attribute values in the web `.config` file of the web application.



The maximum file upload size for SharePoint 2010 is 2GB. If media assets will be larger than 2GB, then they can't be uploaded to SharePoint. However, video assets that are larger than 2GB can still be stored outside of SharePoint and played using the media Web Part.

PERFORMANCE OPTIMIZATION

As always, it is important to consider performance tuning when architecting a DAM solution. In addition to the maximum upload size setting just mentioned, two other significant performance optimization controls should be managed when a DAM solution is deployed.

BLOB Caching

SharePoint web front-end (WFE) servers are capable of caching binary large objects (BLOBs) in a local directory during user requests. Then, during subsequent user requests, any BLOBs in the local cache can be served to end-users much faster than if the BLOB needed to be retrieved from its primary storage location. This results in a significant performance improvement for digital assets that are frequently consumed by end-users — either directly, such as when a video asset is accessed, or indirectly, such as when image assets are part of page loads.

Performance improvement is not the only benefit of serving assets out of local cache. When a video asset is served out of local cache, the end-user actually gains additional functionality. For example, a cached video asset enables an end-user to “skip ahead” to a later point in a video and begin playing from the new location. Also, when BLOB caching is enabled, SharePoint can begin streaming the file to the end-user before the file is completely retrieved, which significantly reduces *time to first byte (TTFB)*.

An administrator can control both the types of files that are cached as well as the size of the BLOB cache. While the size of the cache is really dependent on need and local storage availability, the following guidance is provided to determine the types of files that should be cached, as well as the location of the cache:

- Caching primarily benefits static assets such as infrequently modified files.
- If an asset type is expected to change or be frequently modified, caching may be counterproductive.

- Image, audio, and video file types, as well as .css and .js files, are excellent candidates for caching.
- The cache should be located on a volume other than the system and ULS logging volumes to ensure best server performance.

BLOB caching, which is disabled by default, is very beneficial and should be enabled for DAM solutions. The basic procedure for enabling BLOB caching follows:

1. Launch IIS Manager.
2. In the Connections pane, expand the web server and then the Sites node.
3. Right-click the applicable web application and select Explore.
4. Open the `web.config` file using Notepad.
5. Locate “<BlobCache>” in the `web.config` file.
6. Modify the “<BlobCache/>” element as necessary to enable and configure caching.
7. Save the `web.config` file.



Saving the `web.config` file will cause the web application to recycle. This action should only be performed during nonproduction hours or on a nonproduction system.

The “<BlobCache/>” element has several configuration options. Table 9-4 describes the various BlobCache element attributes.

TABLE 9-4: BlobCache Element Attributes

| ATTRIBUTE | DESCRIPTION |
|-----------|--|
| location | Defines the location of the BLOB cache. Ideally, the cache should be stored on a non-system volume that is also separate from the volume used to store ULS logs. |
| path | A regular expression that identifies the file extensions of the files that will be cached. |
| maxSize | Defines the maximum size (in GB) of the BLOB cache. Disk availability and projected cache utilization should be used to determine if the 10GB default size is sufficient. |
| max-age | Defines the maximum amount of time (in seconds) that the client browser caches BLOBs that are downloaded to the client computer. BLOBs that have not yet expired in the browser cache are not downloaded again during subsequent requests. By default, <code>max-age</code> is set to 86400 seconds (24 hours). This value can be increased if the solution is significantly static. |
| enabled | Indicates whether or not BLOB cache is enabled. |

Bit Rate Throttling

SharePoint is ultimately an IIS application. As such, it benefits from built-in capabilities of the IIS platform. One of those capabilities is *bit rate throttling*. Bit rate throttling is the process of dialing back the bandwidth that is allowed for any one user. Without it, IIS would serve content, such as video assets, using all available bandwidth, which can degrade network performance. IIS bit rate throttling ensures that as little bandwidth as possible is used while still allowing the proper downloading and streaming of media files. For a DAM solution that will stream media content, it is recommended that bit rate throttling be enabled.



Bit rate throttling is dependent on BLOB caching for proper operation. The BLOB cache must be enabled before bit rate throttling will properly operate.

Before bit rate throttling can be configured, IIS Media Services must be installed:

1. Download the x64 installer for IIS Media Services. At the time this book was written, IIS Media Services is currently version 4.0 and can be downloaded at <http://go.microsoft.com/?linkid=9751395>.
2. Execute the installer from the Download location.
3. Click the Next button on the welcome screen to initiate the setup wizard.
4. Agree to the End-User License Agreement and click the Next button to continue.
5. On the Custom Setup page, the default components selected for install are sufficient (see Figure 9-7). Click the Next button to continue.

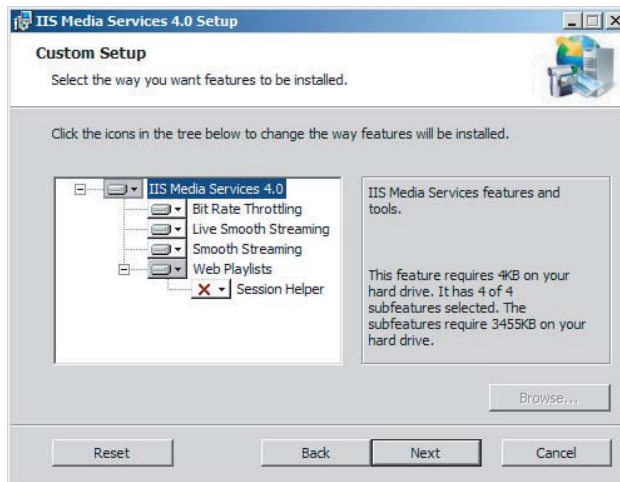


FIGURE 9-7

6. On the final installation wizard page, click the Install button to initiate the process.

7. After installation completes, click the Finish button to exit the setup wizard.

Once IIS Media Services is installed, the configuration options for bit rate throttling become available in IIS Manager. To configure bit rate throttling, follow these steps:



Bit rate throttling can be configured at either the server level and/or the website level. The following instructions outline the steps to configure bit rate throttling for a single website.

1. Launch IIS Manager.
2. Expand the web server and then the Sites node in the Connections pane.
3. Select the website that hosts the SharePoint site that will contain media assets. Notice the new Media Services configuration section at the bottom of the main content pane (see Figure 9-8).

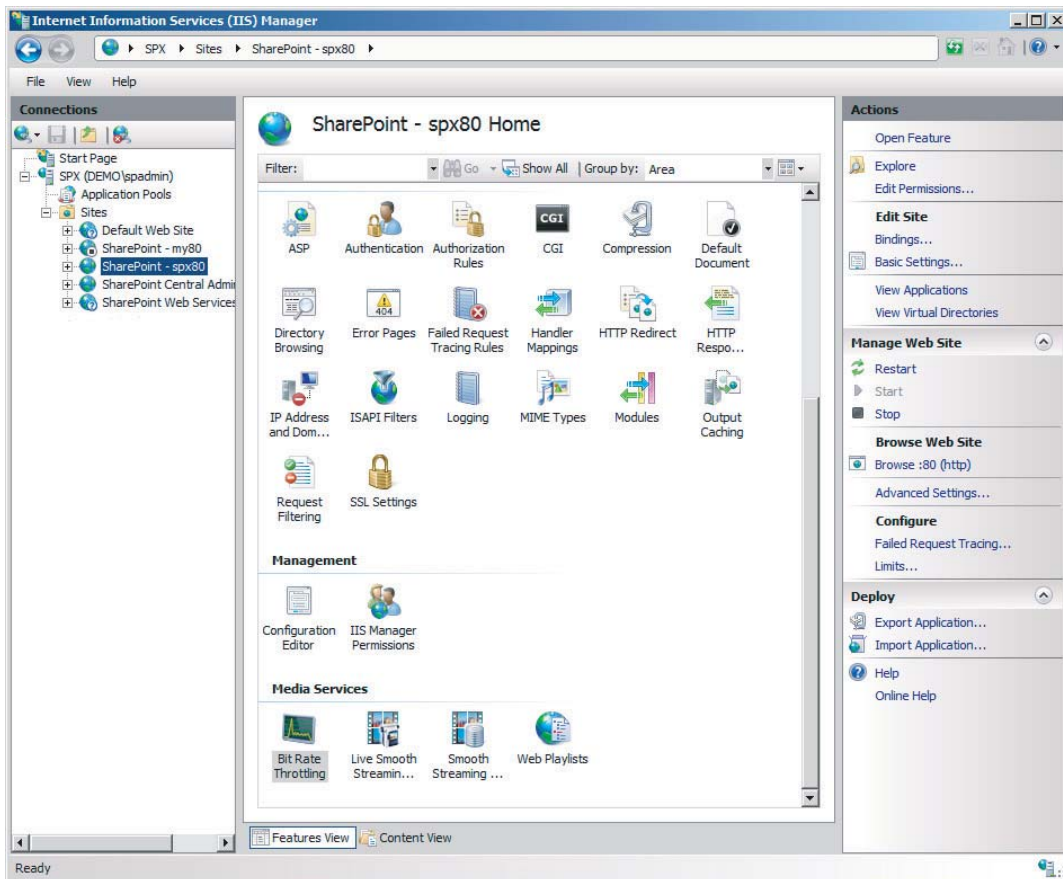


FIGURE 9-8

4. Double-click the Bit Rate Throttling configuration icon in the Media Services configuration section.
5. Notice that the Alerts pane on the right side of the window indicates that the bit rate throttling feature is disabled (see Figure 9-9).

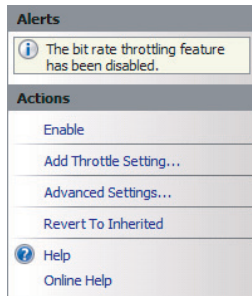


FIGURE 9-9

6. Click Enable in the Actions pane to enable.
7. Most of the popular streaming media file types are preconfigured with default values. In order to change the throttle settings for a specific file type, select the file type in the Bit Rate Throttling pane and click Edit in the Actions pane. This will launch the Edit Throttle Setting dialog shown in Figure 9-10.

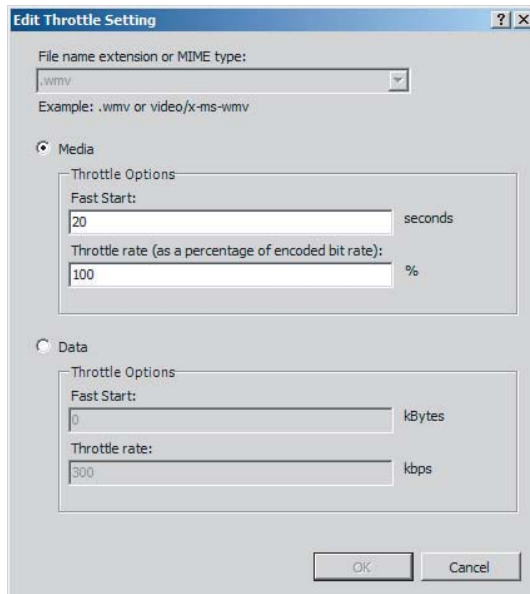


FIGURE 9-10

8. Update the Fast Start value as necessary. Fast Start is the number of seconds that media will be streamed to the client at full throttle. This allows the media client to rapidly fill the stream buffer and begin playing the media quickly.

9. Update the Throttle Rate value as necessary. Throttle rate is the percentage of the encoded bit rate that the media stream is throttled to after the Fast Start full throttle period is complete. The value should be at least 100% to ensure that media is streamed fast enough to stay ahead of playback speed. A throttle rate of 110% should allow the stream to stay ahead of playback while also mitigating temporary reductions of available bandwidth.
10. Click the OK button to apply the throttle settings to the file type.

With bit rate throttling enabled, bandwidth costs are minimized and the number of concurrent users is maximized. This is accomplished with no negative impact on the end-user experience. There are also several advanced configuration options for bit rate throttling but they are beyond the scope of the topic of enterprise content management.

SUMMARY

The new capabilities of the SharePoint 2010 asset library, along with related media Web Parts and controls, take SharePoint to the next level as a platform that is capable of hosting digital assets. With a careful taxonomy and storage architecture, a wide range of digital asset management solutions are now possible. BLOB caching and bit rate throttling for digital assets provide fine-tuning controls that ensure a positive end-user experience while minimizing bandwidth costs.

10

Document Imaging

WHAT'S IN THIS CHAPTER?

- Understanding document imaging and the Microsoft SharePoint as a document imaging platform
- Building a primitive document imaging system on top of Microsoft SharePoint
- Using Microsoft SharePoint APIs to create a document imaging system
- Leveraging the Microsoft .NET Framework to create a document imaging system

Most organizations are forced to deal with paper-intensive processes somewhere within their business. Some of these processes are unique to the industry with which the business is associated, such as the handling of loan applications in the financial industry or dealing with patient records within the healthcare industry. Other processes are common to many organizations, such as invoicing within the accounts payable department or applications within the human resources department.

Regardless of which paper-intensive processes are involved, organizations are plagued with large amounts of paper documents. Typically, these documents are stored in vast arrays of file cabinets that consume large amounts of physical space, and it is often difficult for users to easily find, let alone work with, these documents. Even companies that have begun to store documents electronically are often faced with large backlogs of paper documents.

As discussed in Chapter 3, storing these documents in paper format is both costly and inefficient. Table 10-1 highlights the drastic differences between storing documents electronically versus storing them in their paper-based format. Clearly, storing and managing paper-based documents is an inadequate system for most enterprises.

TABLE 10-1: Paper vs. Electronic Document Storage

| CONSIDERATIONS | FOUR-DRAWER FILE CABINET | 300GB HARD DRIVE |
|----------------|--|---|
| Price | \$199 | \$99 |
| Capacity | About 40,000 pages | About 6,000,000,000 pages (typical file size is 50KB for letter-size pages scanned at 300dpi) |
| Requirements | Office space, inflation, and location drive costs up | Computers, innovation, and increasing memory capacities drive costs down |
| File Retrieval | By manual labor, takes minutes to hours | By software, take milliseconds to seconds, faster if crawled by search engines |
| File Transfer | Travel at the speed of truck or cargo plane; depends on type of service (ground or air); \$40 might cover one overnight shipment | Travel at the speed of light or electric current; depends on type of network (fiber or copper); \$40 might cover one month of unlimited service |
| Consumables | File folders, labels, etc. | None |
| Backup | Difficult (photocopies and boxes) | Easy (tape, CD, or DVD) |

WHAT IS DOCUMENT IMAGING?

Document imaging is the component of enterprise content management that consists of the capture of paper documents into electronic format, the indexing and importation of these documents into a document repository, and the capability to later search and view these documents within the repository. Table 10-2 summarizes these document imaging concepts.

TABLE 10-2: Document Imaging Concepts

| CONCEPT | DESCRIPTION |
|---------|--|
| Capture | The act of converting paper documents into electronic documents as an image file format such as PDF, TIFF, JPG, BMP, PNG, GIF, etc. Capture is typically performed using a document scanner or digital camera. |
| Index | The act of associating metadata with an electronic document |
| Import | The act of transferring electronic documents and their index data into a document repository for storage |
| Search | The act of finding a document through the use of its associated index data or the textual content of the document itself |
| View | The act of actually viewing the document electronically within a document image viewing application |

Within the realm of ECM, document imaging is an interesting topic because it is really a cross-cutting component of ECM. In other words, besides the capture process, document imaging is closely related to various other ECM-related components. For example, when documents are imported into a document repository, the repository typically provides document management capabilities to manage security, versioning, and metadata association. The repository also provides the capability to easily recall these documents using metadata that was captured during indexing by using search functionality. Once these documents are found within the document imaging system, they are viewable with a document image viewer, which is capable of viewing most image-based document formats.

This cross-cutting aspect of document imaging within electronic content management is largely due to historical reasons. Document imaging systems existed long before the concept of enterprise content management. Because they existed when the “paperless office” concept was being popularized, many of the components that make up document imaging as we know it today have been rolled up under the ECM umbrella, as it consumes of a large portion of enterprise-related content.

SharePoint as a Document Imaging Platform

SharePoint’s rich functionality, extensibility, and application programming interfaces enable it to be used as a platform — that is, you can leverage its out-of-the-box components to create whatever solution you require. SharePoint excels at storing large numbers of documents, and it provides extensive document management capabilities, a robust search infrastructure, integrated security, and the capability to access most of its facilities remotely. Therefore, leveraging this platform to create a solid, scalable, and performant document imaging system is a perfect fit; and several third-party vendors have accomplished exactly that.

The best way to demonstrate how you can leverage SharePoint as a document imaging system is by building your own using SharePoint and the facilities it provides.

SETTING UP THE SCENARIO

In order to demonstrate the creation of a document imaging system on SharePoint, we are going to build a primitive document imaging system using SharePoint’s various APIs, extensibility points, and the Microsoft .NET Framework.

To begin, we must set up the scenario for the solution that we are going to build. Imagine a paper-intensive area of an organization. Typically, use of these paper documents would be better served by converting them to electronic format using a document scanner, then uploading them along with specific index (metadata) values to SharePoint. These index values can then be leveraged to perform exact relevance searching in order to find any needed document in a matter of seconds. Once the documents are located by executing a search, their contents can be viewed within a document image viewer. In this chapter, we will build an end-to-end solution to accomplish the tasks outlined in this scenario.

Solution Data Flow Diagram

Figure 10-1 provides an outline of the solution that is going to be built. We will be building a simple capture application that will be used to simulate the scanning of paper documents using a document scanner. This application will also be responsible for sending documents, along with index data, to SharePoint. We will then build a simple Microsoft Silverlight-based image viewer Web Part for

viewing documents. Finally, we will customize the SharePoint Search Web Part, which will enable searching for a document and opening it within the viewer Web Part.

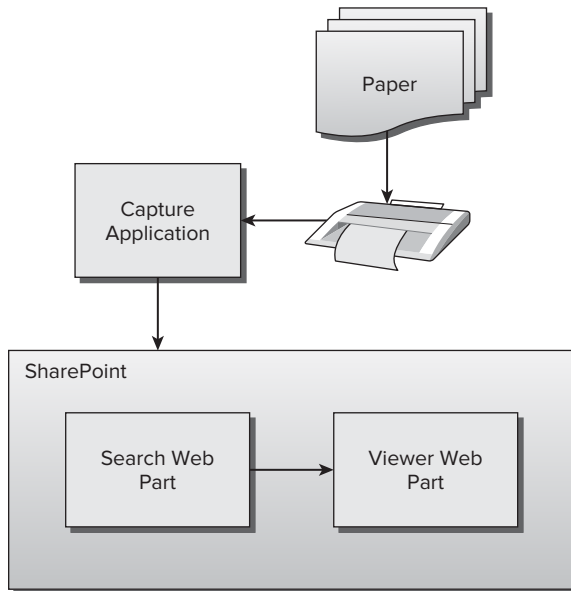


FIGURE 10-1

MODEL-VIEW-VIEWMODEL PRIMER

In this chapter, you will be building entire applications and application components in order to create an end-to-end solution. Many of these components are built using the Microsoft Windows Presentation Foundation (WPF) and Microsoft Silverlight programming models. When building applications using WPF and Silverlight, a common architectural design pattern that fits naturally into these paradigms is referred to as the Model-View-ViewModel pattern (in shortened form, MVVM). While a complete discussion of MVVM is beyond the scope of this book, it is important to grasp the basics of this pattern in order to understand the architecture of these code samples.



MVVM is a commonly used design pattern when developing applications using WPF and Silverlight. You can find a full explanation of this programming pattern in the MSDN Magazine article “WPF Apps with the Model-View-ViewModel Design Pattern,” located at <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>.

MVVM makes heavy use of WPF data binding. For those familiar with this concept in ASP.NET and Window Forms, data binding in WPF differs from the traditional sense of data binding. WPF

data binding refers to the ability of an application's UI to maintain a direct connection to business components through the use of dependency properties. As long as the business objects raise proper notification through the `INotifyPropertyChanged` event, the user interface components that are bound to these objects will automatically update their displayed data to reflect any change that is made to the underlying business data, and any changes made to the user interface components can conversely reflect their changes in the other direction as well.

By leveraging data binding, you can build an application that is designed to benefit from this capability using the MVVM pattern. As shown in Table 10-3, MVVM consists of three major components whose names reflect the pattern name.

TABLE 10-3: Components of MVVM

| COMPONENT | DESCRIPTION |
|-----------|--|
| View | The <code>View</code> is any user interface component that exposes a <code>DataContext</code> property. This is typically any class that inherits from <code>FrameworkElement</code> , such as <code>Control</code> , <code>UserControl</code> , or <code>Window</code> . |
| Model | The <code>Model</code> is any business component that serves to represent some piece of data loaded from an external or local service class. |
| ViewModel | The <code>ViewModel</code> is a class that sits between the <code>View</code> and the <code>Model</code> . It can essentially be interpreted as “the View’s Model.” The <code>ViewModel</code> is the class to which the <code>View</code> is bound. It serves to represent the data in a way that is suitable for display and interaction. Most application logic lives here. |

When creating an application using MVVM, the `View` will contain the user interface components such as buttons, text boxes, and layout panels. The `View` will be bound to an individual `ViewModel`. As the user interacts with the interface, these interactions are relayed to the `ViewModel`. The `ViewModel` handles these interactions to perform the actual business logic such as loading `Models` and updating their various properties. When changes are made to the underlying model, these changes are evented back to the `View`, which is subsequently, and automatically, updated to reflect these changes. Figure 10-2 provides an architectural overview of this pattern.

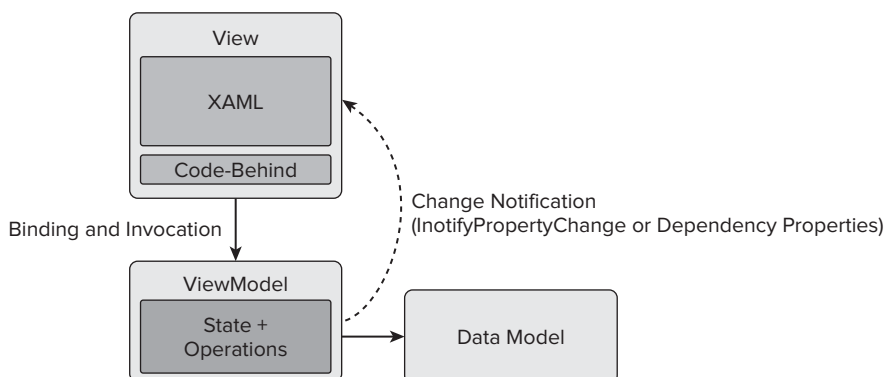


FIGURE 10-2

One tenet that should be followed when using this pattern is that while several different Views can utilize the same ViewModel, each individual View should require only a single ViewModel. In the case of composition, a ViewModel may contain one or many child ViewModel instances, but the View should only be bound to a single ViewModel.

CREATING A SIMPLE WPF CAPTURE APPLICATION

In this section, we will build a simple WPF-based capture application. This application will enable users to simulate a scan using the `OpenFileDialog`, which enables the selection of an individual TIFF file. This file will then be displayed in a paging-enabled viewer. The application will provide the capability to connect to SharePoint to retrieve the columns associated with a specified site, library, and content type. These columns will be displayed as labeled controls, which are used to accept data as index values. Once index values are provided, the application provides the option to release the document, along with the provided data, to SharePoint. Figure 10-3 provides a screenshot of the completed application.

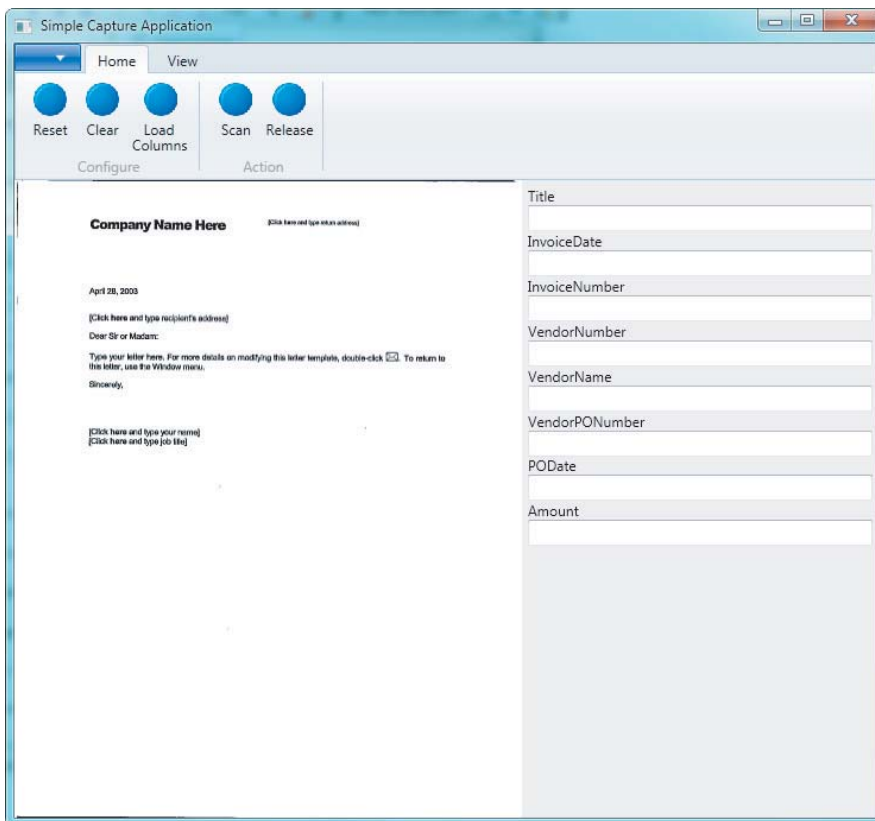


FIGURE 10-3

Architecture and Design

The simple capture application is built using the MVVM architectural design pattern. Figure 10-4 provides a structural diagram depicting the architecture for the application.

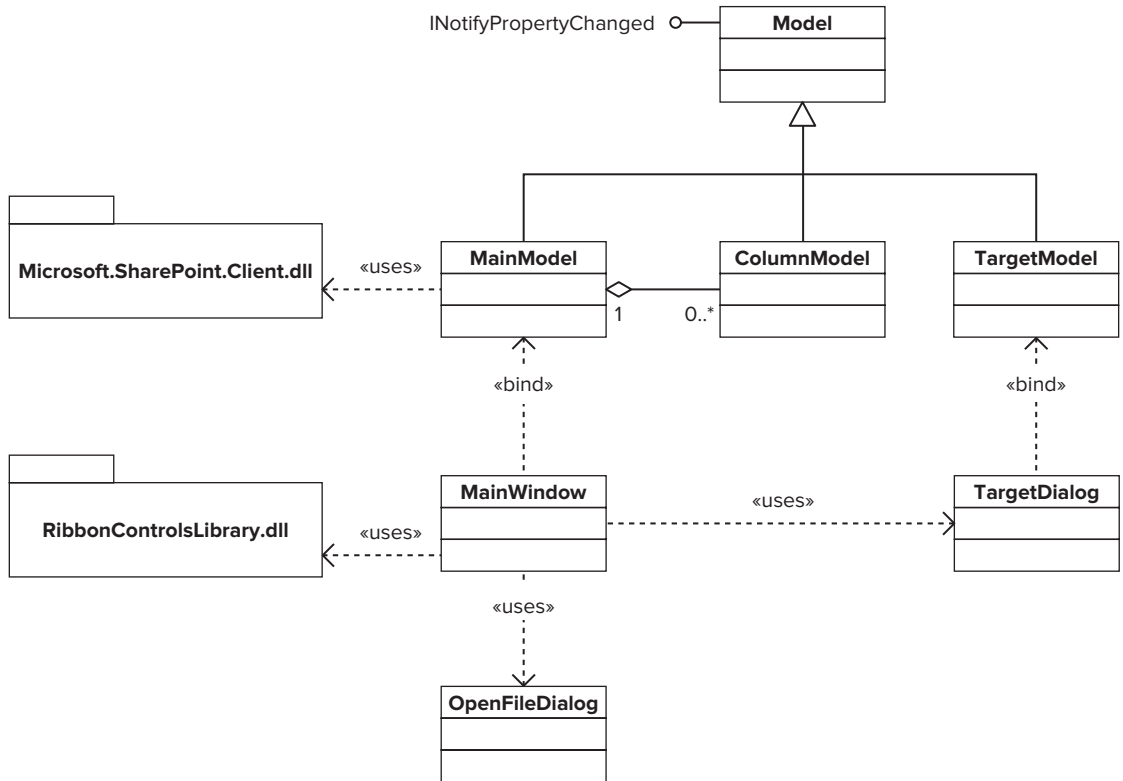


FIGURE 10-4

Implementation

This section will serve as a guide through the actual implementation of the simple capture application. Before beginning this project, you should first download and install the .NET Framework 4.0 Ribbon Control from the Microsoft website. This control will serve as the menu system for the application. Listing 10-1 provides a full listing of this application.

Building the MVVM Infrastructure

The first class that will be created is `Model`. This class provides the base class for the MVVM infrastructure by implementing the `INotifyPropertyChanged` event. This class will be the base for any `Model` or `ViewModel` that is created:

```
class Model : INotifyPropertyChanged
{
```

```

public event PropertyChangedEventHandler PropertyChanged;

public Model() { }

protected void OnPropertyChanged(string propertyName)
{
    if (PropertyChanged != null)
        this.PropertyChanged(this,
            new PropertyChangedEventArgs(propertyName));
}
}

```

By having this base class implement this event, all subclasses can be bound directly to a `View` in order to enable data binding.

Building the Target Dialog

The `TargetDialog` is the window that the application will use to prompt the user for a SharePoint target location by accepting a site URL, a document library name, and a content type name. It uses this information to retrieve the associated SharePoint columns, as well as the target location when the document is released to SharePoint.

This dialog consists of the `TargetDialog` class, which is the `View`, and `TargetModel` class, which is the corresponding `ViewModel`. The `TargetDialog` is bound to the `TargetModel` declaratively in XAML, as shown in the following snippet:

```

<Window ...>
...
    <Window.DataContext>
        <Listing1001.ViewModel:TargetModel />
    </Window.DataContext>
...
</Window>

```

Once this class has been bound, it is then possible to bind individual controls in `TargetDialog` to properties of `TargetModel`, as shown here:

```

<TextBlock Grid.Row="0" Grid.Column="0" Text="Site URL:" />
<TextBox Grid.Row="0" Grid.Column="1" Text="{Binding SiteUrl, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}" />

```

As these controls are edited, the values are automatically passed to the `Set` property of the corresponding `Model` property. In the previous snippet, the `TextBox` control was bound to the property call `SiteUrl` on the `TargetModel`. This property looks like the following:

```

public string SiteUrl
{
    get { return this.siteUrl; }
    set
    {
        this.siteUrl = value;
        this.OnPropertyChanged("SiteUrl");
    }
}

```


The binding pattern is used consistently throughout the application in order to pass control values to the model; and vice versa, to pass model properties to the controls.

Because the `TargetDialog` class is a dialog that will be used by another window, it exposes some properties that will be used by the calling window to get the user-provided values. The `TargetModel` actually holds the values, so they must be exposed through the `TargetDialog` in order for the calling window to gain access to them when the dialog closes. The following snippet shows how this is accomplished:

```
public string SiteUrl
{
    get { return this.model.SiteUrl; }
}
```

In order to use the `TargetDialog`, the calling class simply calls `ShowDialog` and then extracts the data collected from its exposed properties as follows:

```
TargetDialog dialog = new TargetDialog("SiteUrl", "LibraryTitle", "ContentType");
dialog.Owner = this;

if (dialog.ShowDialog() ?? false)
{
    this.model.SiteUrl = dialog.SiteUrl;
    this.model.LibraryTitle = dialog.LibraryTitle;
    this.model.ContentType = dialog.ContentType;
    ...
}
```

Building the Main Window

The `MainWindow` class is the entry point into the application and represents the main portion of the user interface. This class' `ViewModel` is the `MainModel` class. The `MainModel` class actually contains most of the logic for the application. The `MainWindow` class exposes the user interface components. As users interact with `MainWindow`, this window class relays these interactions through method invocation to the underlying `MainModel`, which in turn performs the indicated action and updates its internal properties. As these properties are updated, the `MainWindow` is updated to display these changes. Let's walk through a few of these interactions.

Loading SharePoint Columns

Interacting with SharePoint to return columns is accomplished by clicking the Load Columns Button in the Ribbon. Clicking this button invokes the Target Dialog, which collects user input and passes it to the `MainModel`:

```
private void LoadColumnsButton_Click(object sender, RoutedEventArgs e)
{
    TargetDialog dialog = new TargetDialog(this.model.SiteUrl,
                                           this.model.LibraryTitle,
                                           this.model.ContentType);

    dialog.Owner = this;

    if (dialog.ShowDialog() ?? false)
    {
```

```

        this.model.SiteUrl = dialog.SiteUrl;
        this.model.LibraryTitle = dialog.LibraryTitle;
        this.model.ContentType = dialog.ContentType;

        this.model.LoadColumns();
    }
}

```

After calling `LoadColumns` on the `MainModel`, the model uses the SharePoint client object model to get the list of fields for the target information that was provided, as shown in the following snippet:

```

public void LoadColumns()
{
    try
    {
        this.ClearColumns();

        if (this.IsTargetInformationPresent())
        {
            using (ClientContext context = this.GetClientContext())
            {
                List library = context.Web.Lists.
                    GetByTitle(this.LibraryTitle);

                context.Load(library, lib => lib.ContentTypes
                    .Include(ct => ct.Name,
                        ct => ct.Id,
                        ct => ct.Fields
                            .IncludeWithDefaultProperties()
                    .Where(fld => !fld.FromBaseType &&
                        !fld.Hidden &&
                        !fld.ReadOnlyField &&
                        fld.FieldTypeKind !=
                            FieldType.Computed))
                    .Where(
                        ct => ct.Name == this.contentType));

                context.ExecuteQuery();

                this.contentTypeId = library.ContentTypes[0].Id;

                foreach (Field field in library.ContentTypes[0].Fields)
                {
                    this.Columns.Add(new ColumnModel()
                    {
                        ColumnName = field.Title,
                        ColumnInternalName = field.InternalName
                    });
                }
            }
        }
        else
            this.Message = "Target Information Must Be Provided";

        this.OnPropertyChanged("IsReleaseValid");
    }
}

```

```

    }
    catch(Exception ex)
    {
        this.Message = ex.Message;
    }
}

```

As these fields are returned from SharePoint, they are converted to `ColumnModel` objects and added to the `MainModel`'s `Columns` collection. As they are added to this collection, the `MainWindow` is updated automatically to display a control for each field which is used to collect index data from the user. This occurs because the `MainWindow` has an `ItemControl` that is bound to the `Columns` property of the `MainModel`, as shown in the following snippet:

```

<ItemsControl ItemsSource="{Binding Columns}">
  <ItemsControl.ItemTemplate>
    <DataTemplate>
      <StackPanel Orientation="Vertical">
        <TextBlock Text="{Binding ColumnName}" />
        <TextBox Text="{Binding ColumnValue,
          Mode=TwoWay,
          UpdateSourceTrigger=PropertyChanged}" />
      </StackPanel>
    </DataTemplate>
  </ItemsControl.ItemTemplate>
</ItemsControl>

```

Scanning an Image

Scanning an image is simulated by using the `OpenFileDialog`. When the Scan button is clicked within the `MainWindow`, the `LoadImage` method is called on the `MainModel`, passing the `FileName` property from the `OpenFileDialog`:

```

OpenFileDialog dialog = new OpenFileDialog();

dialog.Multiselect = false;
dialog.CheckPathExists = true;
dialog.Filter = ".tif/.tif|*tif;*.tiff";

if (dialog.ShowDialog(this) ?? false)
{
    this.model.LoadImage(dialog.FileName);
}

```

The `LoadImage` method performs the actual loading of the image by setting the `ImagePath` property of the `MainModel`:

```

public void LoadImage(string path)
{
    try
    {
        this.ImagePath = path;
    }
    catch (Exception ex)
    {

```

```
        this.Message = ex.Message;
    }
}
```

When the `ImagePath` property is set, the image is loaded from the file system and the corresponding `Image` property is set as a `BitmapImage` object:

```
public string ImagePath
{
    get { return this.imagePath; }
    private set
    {
        this.imagePath = value;
        this.OnPropertyChanged("ImagePath");

        this.pageCount = 0;
        this.currentPage = 0;

        if (!string.IsNullOrEmpty(value))
        {
            using (Bitmap bitmap = new Bitmap(value))
            {
                this.ReadImageInfo(bitmap);
                this.Image = this.GetPageImage(bitmap, 0);
            }
        }
        else
            this.Image = null;
    }
}
```

The `MainWindow` is then able to display the image because it contains an `Image` control whose `Source` property is bound to the `MainModel`'s `Image` property, as shown in the following snippet:

```
<Image x:Name="ScannedImage" Grid.Row="1" Grid.Column="0"
        Source="{Binding Image}" />
```

Releasing to SharePoint

Releasing the document and its metadata to SharePoint is then accomplished by clicking the `Release` button in the Ribbon. After clicking this button, the `MainWindow` calls the `Release` method on the `MainModel`. This method uses the SharePoint client object model to send the image to SharePoint along with the associated index data that was provided:

```
public void Release()
{
    if (this.IsReleaseValid)
    {
        try
        {
            using (ClientContext context = this.GetClientContext())
            {
                List library = context.Web.Lists.
```

```

        GetByTitle(this.LibraryTitle);

        File file = library.RootFolder.Files
            .Add(new FileCreationInformation()
            {
                Url = Path.GetFileName(this.ImagePath),
                Content = File.ReadAllBytes(this.ImagePath),
                Overwrite = true
            });

        foreach (ColumnModel column in this.Columns)
        {
            file.ListItemAllFields[column.ColumnInternalName] =
                column.ColumnValue;
        }

        file.ListItemAllFields["ContentTypeId"] =
            this.contentTypeId.ToString();

        file.ListItemAllFields.Update();

        context.ExecuteQuery();

        this.Message = "Release Successful";
    }
}
catch (Exception ex)
{
    this.Message = ex.Message;
}
}
}

```

Now that the key implementation sections of the Simple WPF Capture Application have been reviewed, Listing 10-1 will provide the full code listing for the application.



LISTING 10-1: Full Listing for the Simple WPF Capture Application

Available for
download on
Wrox.com

```

//Listing1001.App.cs

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace Listing1001
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>

```

continues

LISTING 10-1 *(continued)*

```
        public partial class App : Application
        {
        }
    }

//Listing1001.App.xaml

<Application x:Class="Listing1001.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>

        </Application.Resources>
    </Application>

//Listing1001.MainWindow.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Listing1001.ViewModel;
using Microsoft.Win32;
using Microsoft.Windows.Controls.Ribbon;

namespace Listing1001
{
    public partial class MainWindow : RibbonWindow
    {
        private MainModel model = null;

        public MainWindow()
        {
            InitializeComponent();
        }

        private void Initialize()
        {
            this.model = this.DataContext as MainModel;
            this.model.PropertyChanged +=

```

```
        new System.ComponentModel.  
        PropertyChangedEventHandler(model_PropertyChanged);  
    }  
  
    private void Exit()  
    {  
        this.Close();  
    }  
  
    private void DisplayMessage(string message)  
    {  
        MessageBox.Show(message);  
        this.model.ClearMessage();  
    }  
  
    private void ScanButton_Click(object sender, RoutedEventArgs e)  
    {  
        OpenFileDialog dialog = new OpenFileDialog();  
  
        dialog.Multiselect = false;  
        dialog.CheckPathExists = true;  
        dialog.Filter = ".tif/.tif|*tif;*.tiff";  
  
        if (dialog.ShowDialog(this) ?? false)  
        {  
            this.model.LoadImage(dialog.FileName);  
        }  
    }  
  
    private void ClearButton_Click(object sender, RoutedEventArgs e)  
    {  
        this.model.Clear();  
    }  
  
    private void ResetButton_Click(object sender, RoutedEventArgs e)  
    {  
        this.model.Reset();  
    }  
  
    private void ReleaseButton_Click(object sender, RoutedEventArgs e)  
    {  
        this.model.Release();  
        this.model.Clear();  
    }  
  
    private void ExitButton_Click(object sender, RoutedEventArgs e)  
    {  
        this.Exit();  
    }  
  
    private void LoadColumnsButton_Click(object sender, RoutedEventArgs e)  
    {  
        TargetDialog dialog = new TargetDialog(this.model.SiteUrl,
```

continues

LISTING 10-1 (continued)

```

        this.model.LibraryTitle,
        this.model.ContentType);

    dialog.Owner = this;

    if (dialog.ShowDialog() ?? false)
    {
        this.model.SiteUrl = dialog.SiteUrl;
        this.model.LibraryTitle = dialog.LibraryTitle;
        this.model.ContentType = dialog.ContentType;

        this.model.LoadColumns();
    }
}

private void NextPageButton_Click(object sender, RoutedEventArgs e)
{
    this.model.NextPage();
}

private void PrevPageButton_Click(object sender, RoutedEventArgs e)
{
    this.model.PrevPage();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    this.Initialize();
}

private void model_PropertyChanged(object sender,
    System.ComponentModel.PropertyChangedEventArgs e)
{
    switch(e.PropertyName)
    {
        case "Message":
            if (this.model.Message != string.Empty)
                this.DisplayMessage(this.model.Message);
            break;
    }
}
}
}

//Listing1001.MainWindow.xaml

<ribbon:RibbonWindow x:Class="Listing1001.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ribbon=
    "http://schemas.microsoft.com/winfx/2006/xaml/presentation/ribbon"
    xmlns:Listing1001.ViewModel="clr-namespace:Listing1001.ViewModel"
    Title="Simple Capture Application" Height="600" Width="800"

```



```

    Loaded="Window_Loaded">
<Window.DataContext>
    <Listing1001.ViewModel:MainModel />
</Window.DataContext>
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="300" />
    </Grid.ColumnDefinitions>

    <ribbon:Ribbon Grid.ColumnSpan="2"
        HorizontalAlignment="Stretch" VerticalAlignment="Top">
    <ribbon:Ribbon.ApplicationMenu>
        <ribbon:RibbonApplicationMenu>
            <ribbon:RibbonApplicationMenuItem Header="Exit"
                Click="ExitButton_Click" />
        </ribbon:RibbonApplicationMenu>
    </ribbon:Ribbon.ApplicationMenu>
    <ribbon:RibbonTab Header="Home">
        <ribbon:RibbonGroup Header="Configure">
            <ribbon:RibbonButton x:Name="ResetButton" Label="Reset"
                LargeImageSource="Resources/menu_item.png"
                Click="ResetButton_Click" />
            <ribbon:RibbonButton x:Name="ClearButton" Label="Clear"
                LargeImageSource="Resources/menu_item.png"
                Click="ClearButton_Click" />
            <ribbon:RibbonButton x:Name="LoadColumnsButton"
                Label="Load Columns"
                LargeImageSource="Resources/menu_item.png"
                Click="LoadColumnsButton_Click" />
        </ribbon:RibbonGroup>
        <ribbon:RibbonGroup Header="Action">
            <ribbon:RibbonButton x:Name="ScanButton" Label="Scan"
                LargeImageSource="Resources/menu_item.png"
                Click="ScanButton_Click" />
            <ribbon:RibbonButton x:Name="ReleaseButton"
                Label="Release"
                LargeImageSource="Resources/menu_item.png"
                Click="ReleaseButton_Click"
                IsEnabled="{Binding IsReleaseValid}" />
        </ribbon:RibbonGroup>
    </ribbon:RibbonTab>
    <ribbon:RibbonTab Header="View">
        <ribbon:RibbonGroup Header="Paging">
            <ribbon:RibbonButton x:Name="PrevPageButton"
                Label="Previous"
                LargeImageSource="Resources/menu_item.png"
                Click="PrevPageButton_Click" />
            <ribbon:RibbonButton x:Name="NextPageButton" Label="Next"
                LargeImageSource="Resources/menu_item.png"

```

continues

LISTING 10-1 *(continued)*

```

        Click="NextPageButton_Click" />
    </ribbon:RibbonGroup>
</ribbon:RibbonTab>
</ribbon:Ribbon>

<Image x:Name="ScannedImage" Grid.Row="1" Grid.Column="0"
Source="{Binding Image}" />

<TextBlock Text="{Binding PageOfCount}" Grid.Row="1" Grid.Column="0"
FontWeight="Bold" FontSize="14"
HorizontalAlignment="Right" VerticalAlignment="Bottom"
Margin="0,0,10,10" />

<Border Grid.Row="1" Grid.Column="1"
Background="{DynamicResource {x:Static SystemColors.ControlBrushKey}}">
    <Grid Margin="5,5,5,5">
        <ScrollViewer Grid.Row="3" Grid.ColumnSpan="2"
            HorizontalScrollBarVisibility="Auto"
            VerticalScrollBarVisibility="Auto">
            <ItemsControl ItemsSource="{Binding Columns}">
                <ItemsControl.ItemTemplate>
                    <DataTemplate>
                        <StackPanel Orientation="Vertical">
                            <TextBlock Text="{Binding ColumnName}" />
                            <TextBox Text="{Binding ColumnValue,
                                Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
                        </StackPanel>
                    </DataTemplate>
                </ItemsControl.ItemTemplate>
            </ItemsControl>
        </ScrollViewer>
    </Grid>
</Border>
</Grid>
</ribbon:RibbonWindow>

//Listing1001.TargetDialog.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using Listing1001.ViewModel;

namespace Listing1001

```

```
{
public partial class TargetDialog : Window
{
    private TargetModel model = null;
    private Tuple<string, string, string> args = null;

    public TargetDialog() : this(string.Empty, string.Empty, string.Empty) { }

    public TargetDialog(string siteUrl,
                        string libraryTitle,
                        string contentType)
    {
        InitializeComponent();
        this.args = new Tuple<string, string, string>(siteUrl,
                                                    libraryTitle,
                                                    contentType);
    }

    public string SiteUrl
    {
        get { return this.model.SiteUrl; }
    }

    public string ContentType
    {
        get { return this.model.ContentType; }
    }

    public string LibraryTitle
    {
        get { return this.model.LibraryTitle; }
    }

    private void OkButton_Click(object sender, RoutedEventArgs e)
    {
        this.DialogResult = true;
        this.Close();
    }

    private void CancelButton_Click(object sender, RoutedEventArgs e)
    {
        this.DialogResult = false;
        this.Close();
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        this.model = this.DataContext as TargetModel;

        this.model.SiteUrl = this.args.Item1;
        this.model.LibraryTitle = this.args.Item2;
    }
}
```

continues

LISTING 10-1 (continued)

```

        this.model.ContentType = this.args.Item3;
    }
}

//Listing1001.TargetDialog.xaml

<Window x:Class="Listing1001.TargetDialog"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:Listing1001.ViewModel="clr-namespace:Listing1001.ViewModel"
    Title="Load Columns" ResizeMode="NoResize"
    WindowStartupLocation="CenterOwner"
    WindowStyle="ToolWindow" SizeToContent="Height" Width="400"
    Loaded="Window_Loaded">
<Window.DataContext>
    <Listing1001.ViewModel:TargetModel />
</Window.DataContext>
<Grid>
    <Border Grid.Row="1" Grid.Column="1"
        Background="{DynamicResource {x:Static SystemColors.ControlBrushKey}}">
        <Grid Margin="5,5,5,5">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>

            <TextBlock Grid.Row="0" Grid.Column="0" Text="Site URL:" />
            <TextBox Grid.Row="0" Grid.Column="1"
                Text="{Binding SiteUrl, Mode=TwoWay,
                    UpdateSourceTrigger=PropertyChanged}" />

            <TextBlock Grid.Row="1" Grid.Column="0" Text="Library:" />
            <TextBox Grid.Row="1" Grid.Column="1"
                Text="{Binding LibraryTitle, Mode=TwoWay,
                    UpdateSourceTrigger=PropertyChanged}"
                />

            <TextBlock Grid.Row="2" Grid.Column="0" Text="Content Type:" />
            <TextBox Grid.Row="2" Grid.Column="1"
                Text="{Binding ContentType, Mode=TwoWay,
                    UpdateSourceTrigger=PropertyChanged}"
                />

            <StackPanel Grid.Row="3" Grid.ColumnSpan="2"
                Orientation="Horizontal"

```

```

        HorizontalAlignment="Right">
        <Button x:Name="OkButton" Content="OK" Margin="5,5,2,5"
            IsDefault="True"
            Height="20" Width="50" Click="OkButton_Click" />
        <Button x:Name="CancelButton" Content="Cancel"
            Margin="2,5,5,5" IsCancel="True"
            Height="20" Width="50" Click="CancelButton_Click" />
    </StackPanel>
</Grid>
</Border>
</Grid>
</Window>

```

```
//Listing1001.ViewModel.Model.cs
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;

namespace Listing1001.ViewModel
{
    class Model : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        public Model() { }

        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
                this.PropertyChanged(this,
                    new PropertyChangedEventArgs(propertyName));
        }
    }
}

```

```
//Listing1001.ViewModel.MainModel.cs
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Collections.ObjectModel;
using System.Windows.Media.Imaging;
using Microsoft.SharePoint.Client;
using System.Drawing;
using System.Drawing.Imaging;

```

```

namespace Listing1001.ViewModel
{

```

continues

LISTING 10-1 *(continued)*

```
class MainModel : Model
{
    private string siteUrl = string.Empty;
    private string libraryTitle = string.Empty;
    private string contentType = string.Empty;
    private ContentTypeId contentTypeId = null;
    private ObservableCollection<ColumnModel> columns = null;
    private BitmapImage image = null;
    private string message = string.Empty;
    private string imagePath = string.Empty;
    private int currentPage = 0;
    private int pageCount = 0;

    public MainModel()
    {

    }

    public string SiteUrl
    {
        get { return this.siteUrl; }
        set
        {
            this.siteUrl = value;
            this.ClearColumns();
            this.OnPropertyChanged("SiteUrl");
        }
    }

    public string LibraryTitle
    {
        get { return this.libraryTitle; }
        set
        {
            this.libraryTitle = value;
            this.ClearColumns();
            this.OnPropertyChanged("LibraryTitle");
        }
    }

    public string ContentType
    {
        get { return this.contentType; }
        set
        {
            this.contentType = value;

            this.ClearColumns();
            this.contentTypeId = null;

            this.OnPropertyChanged("ContentType");
        }
    }
}
```

```
    }
}

public ObservableCollection<ColumnModel> Columns
{
    get
    {
        if (this.columns == null)
            this.columns = new ObservableCollection<ColumnModel>();

        return this.columns;
    }
}

public BitmapImage Image
{
    get { return this.image; }
    private set
    {
        this.image = value;
        this.OnPropertyChanged("Image");
        this.OnPropertyChanged("PageOfCount");
        this.OnPropertyChanged("IsReleaseValid");
    }
}

public string ImagePath
{
    get { return this.imagePath; }
    private set
    {
        this.imagePath = value;
        this.OnPropertyChanged("ImagePath");

        this.pageCount = 0;
        this.currentPage = 0;

        if (!string.IsNullOrEmpty(value))
        {
            using (Bitmap bitmap = new Bitmap(value))
            {
                this.ReadImageInfo(bitmap);
                this.Image = this.GetPageImage(bitmap, 0);
            }
        }
        else
            this.Image = null;
    }
}

public string PageOfCount
{
    get
    {
```

continues

LISTING 10-1 *(continued)*

```

        return string.Format("{0} of {1}",
            this.pageCount > 0 ?
                this.currentPage + 1 : 0,
                this.pageCount);
    }
}

public bool IsReleaseValid
{
    get
    {
        return string.IsNullOrEmpty(this.Message) &&
            !string.IsNullOrEmpty(this.SiteUrl) &&
            !string.IsNullOrEmpty(this.ContentType) &&
            !string.IsNullOrEmpty(this.LibraryTitle) &&
            !string.IsNullOrEmpty(this.ImagePath) &&
            this.Columns.Count > 0;
    }
}

public string Message
{
    get { return this.message; }
    private set
    {
        this.message = value;
        this.OnPropertyChanged("Message");
    }
}

public void ClearMessage()
{
    this.Message = string.Empty;
}

public void LoadColumns()
{
    try
    {
        this.ClearColumns();

        if (this.IsTargetInformationPresent())
        {
            using (ClientContext context = this.GetClientContext())
            {
                List library =
                    context.Web.Lists.GetByTitle(this.LibraryTitle);

                context.Load(library, lib =>

```



```

        lib.ContentTypes.Include(ct => ct.Name,
                                ct => ct.Id,
                                ct => ct.Fields.IncludeWithDefaultProperties().Where(
                                    fld => !fld.FromBaseType &&
                                    !fld.Hidden &&
                                    !fld.ReadOnlyField &&
                                    fld.FieldTypeKind != FieldType.Computed)).Where(
                                ct => ct.Name == this.contentType);

        context.ExecuteQuery();

        this.contentTypeId = library.ContentTypes[0].Id;

        foreach (Field field in library.ContentTypes[0].Fields)
        {
            this.Columns.Add(new ColumnModel()
            {
                ColumnName = field.Title,
                ColumnInternalName = field.InternalName
            });
        }
    }
    else
        this.Message = "Target Information Must Be Provided";

        this.OnPropertyChanged("IsReleaseValid");
    }
    catch (Exception ex)
    {
        this.Message = ex.Message;
    }
}

public void Release()
{
    if (this.IsReleaseValid)
    {
        try
        {
            using (ClientContext context = this.GetClientContext())
            {
                List library =
                    context.Web.Lists.GetByTitle(this.LibraryTitle);

                File file = library.RootFolder.Files.Add(
                    new FileCreationInformation()
                    {
                        Url =

```

continues

LISTING 10-1 *(continued)*

```
        System.IO.Path.GetFileName(this.ImagePath),
        Content =
        System.IO.File.ReadAllBytes(this.ImagePath),
        Overwrite = true
    });

    foreach (ColumnModel column in this.Columns)
    {
        file.ListItemAllFields[column.ColumnInternalName]
            = column.ColumnValue;
    }

    file.ListItemAllFields["ContentTypeId"]
        = this.contentType.ToString();

    file.ListItemAllFields.Update();

    context.ExecuteQuery();

    this.Message = "Release Successful";
}
}
catch (Exception ex)
{
    this.Message = ex.Message;
}
}

public void Clear()
{
    foreach (ColumnModel column in this.Columns)
        column.ColumnValue = string.Empty;

    this.ImagePath = string.Empty;
}

public void Reset()
{
    this.SiteUrl = string.Empty;
    this.LibraryTitle = string.Empty;
    this.ContentType = string.Empty;
    this.ImagePath = string.Empty;
}

public void LoadImage(string path)
{
    try
    {
        this.ImagePath = path;
    }
    catch (Exception ex)
```

```
        {
            this.Message = ex.Message;
        }
    }

    public void NextPage()
    {
        if (!string.IsNullOrEmpty(this.ImagePath))
        {
            using (Bitmap bitmap = new Bitmap(this.ImagePath))
            {
                if (this.currentPage < this.pageCount - 1)
                    this.Image = this.GetPageImage(bitmap, ++this.currentPage);
            }
        }
    }

    public void PrevPage()
    {
        if (!string.IsNullOrEmpty(this.ImagePath))
        {
            using (Bitmap bitmap = new Bitmap(this.ImagePath))
            {
                if (this.currentPage > 0)
                    this.Image = this.GetPageImage(bitmap, --this.currentPage);
            }
        }
    }

    private bool IsTargetInformationPresent()
    {
        return !string.IsNullOrEmpty(this.SiteUrl) &&
            !string.IsNullOrEmpty(this.ContentType) &&
            !string.IsNullOrEmpty(this.LibraryTitle);
    }

    private void ClearColumns()
    {
        this.Columns.Clear();
        this.OnPropertyChanged("Columns");
        this.OnPropertyChanged("IsReleaseValid");
    }

    private BitmapImage GetPageImage(Bitmap bitmap, int page)
    {
        System.IO.MemoryStream ms = new System.IO.MemoryStream();

        bitmap.SelectActiveFrame(FrameDimension.Page, page);
        bitmap.Save(ms, ImageFormat.Tiff);

        ms.Position = 0;

        BitmapImage image = new BitmapImage();
    }
}
```

continues

LISTING 10-1 *(continued)*

```
        image.BeginInit();
        image.StreamSource = ms;
        image.EndInit();

        return image;
    }

    private void ReadImageInfo(Bitmap bitmap)
    {
        this.pageCount = bitmap.GetFrameCount(FrameDimension.Page);
    }

    private ClientContext GetClientContext()
    {
        ClientContext context = new ClientContext(this.SiteUrl);

        //Default credentials are assumed. Set credentials here if necessary.
        context.Credentials = null;

        return context;
    }
}

//Listing1001.ViewModel.ColumnModel.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Listing1001.ViewModel
{
    class ColumnModel : Model
    {
        private string columnName = string.Empty;
        private string columnValue = string.Empty;
        private string columnInternalName = string.Empty;

        public ColumnModel() { }

        public string ColumnName
        {
            get { return this.columnName; }
            set
            {
                this.columnName = value;
                this.OnPropertyChanged("ColumnName");
            }
        }
    }
}
```

```
    }

    public string ColumnInternalName
    {
        get { return this.columnInternalName; }
        set
        {
            this.columnInternalName = value;
            this.OnPropertyChanged("ColumnInternalName");
        }
    }

    public string ColumnValue
    {
        get { return this.columnValue; }
        set
        {
            this.columnValue = value;
            this.OnPropertyChanged("ColumnValue");
        }
    }
}
}
```

```
//Listing1001.ViewModel.TargetModel.cs
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Collections.ObjectModel;
using System.Windows.Media.Imaging;
using Microsoft.SharePoint.Client;
using System.Drawing;
using System.Drawing.Imaging;

namespace Listing1001.ViewModel
{
    class TargetModel : Model
    {
        private string siteUrl = string.Empty;
        private string libraryTitle = string.Empty;
        private string contentType = string.Empty;

        public TargetModel()
        {
        }

        public string SiteUrl
        {
```

continues

LISTING 10-1 *(continued)*

```

        get { return this.siteUrl; }
        set
        {
            this.siteUrl = value;
            this.OnPropertyChanged("SiteUrl");
        }
    }

    public string LibraryTitle
    {
        get { return this.libraryTitle; }
        set
        {
            this.libraryTitle = value;
            this.OnPropertyChanged("LibraryTitle");
        }
    }

    public string ContentType
    {
        get { return this.contentType; }
        set
        {
            this.contentType = value;
            this.OnPropertyChanged("ContentType");
        }
    }
}
}
}

```

Deployment

This application can be copied and executed on any machine running .NET Framework 4. Because this application uses the SharePoint client object model to interact with SharePoint, it does not need to run on the SharePoint server; and in a real-world scenario, it would interact on a user's desktop with a locally attached scanner in order to convert documents into electronic format and subsequently send them, along with their index data, to SharePoint.

CREATING A SIMPLE SILVERLIGHT VIEWER WEB PART

In this section, we will be building a simple Silverlight-based viewer Web Part. This Web Part provides the capability to view TIFF images that exist within Microsoft SharePoint. The Web Part supports the capability to open a file from a parameter in the query string or from a JavaScript method that it outputs to the page.

Architecture and Design

The simple viewer Web Part is built using the MVVM architectural design pattern. Figure 10-5 shows a structural diagram depicting the architecture for the application. Note that the Web Part is dependent on a couple of custom SharePoint services. One of these services is an HTTP handler that is responsible for using the SharePoint Server object model to open the file and stream out individual page binaries. The other service is a web service that also uses the SharePoint Server object model to open the file and return page count information to support paging through the document.

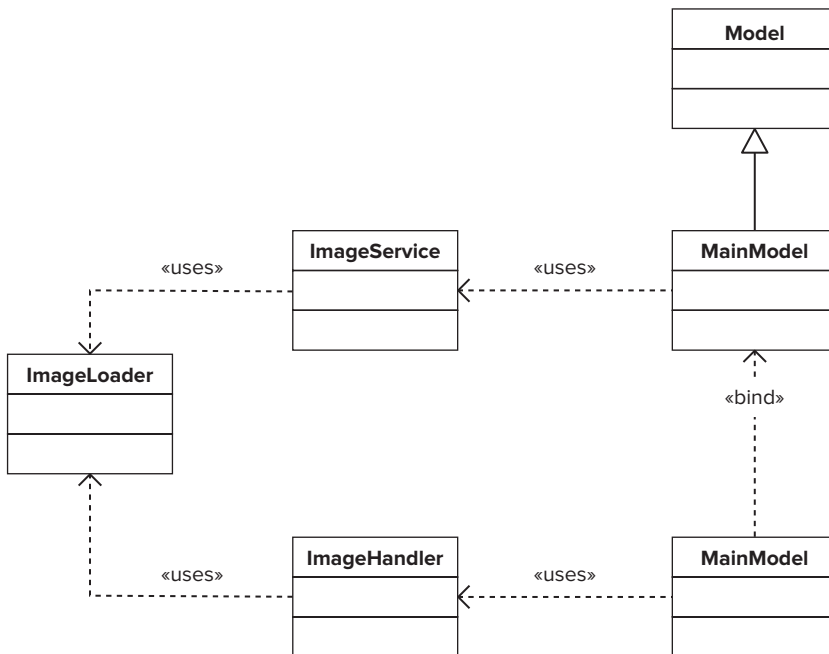


FIGURE 10-5

Implementation

This section serves as a guide to constructing the custom imaging service and the viewer Web Part. Listing 10-2 provides a full listing for the services, whereas Listing 10-3 provides a full listing for the viewer Web Part.

Building the Image Loader

The first component to build is the `ImageLoader` class. This class is used by both services to load files from SharePoint and read image information. This class exposes two public methods: `GetPageCount` and `GetPage`.

Both of these methods call a private method, `LoadImage`, to load the requested file from SharePoint using a URL which is passed as a parameter:

```
private byte[] LoadImage(string url)
{
    using (SPSite site = new SPSite(url))
    using (SPWeb web = site.OpenWeb())
    {
        SPFile file = web.GetFile(url);
        return file.OpenBinary();
    }
}
```

`GetPageCount` calls `LoadImage` and then returns the number of pages contained within the image:

```
public int GetPageCount(string url)
{
    using (MemoryStream inStream = new MemoryStream(this.LoadImage(url)))
    using (Bitmap bitmap = new Bitmap(inStream))
    {
        return bitmap.GetFrameCount(FrameDimension.Page);
    }
}
```

`GetPage` calls `LoadImage` and then returns the binary for the requested page number:

```
public byte[] GetPage(string url, int page)
{
    using (MemoryStream inStream = new MemoryStream(this.LoadImage(url)))
    using (Bitmap bitmap = new Bitmap(inStream))
    {
        bitmap.SelectActiveFrame(FrameDimension.Page, page);

        using (MemoryStream outStream = new MemoryStream())
        {
            bitmap.Save(outStream, ImageFormat.Png);
            return outStream.ToArray();
        }
    }
}
```

Building the Imaging Web Service

The next component to build is the imaging web service. This web service will be a custom SharePoint web service that follows the guidelines set forth by Microsoft for developing custom web services for SharePoint.

The web service class is called `ImageService`, which exposes a single web method called `GetPageCount`. `GetPageCount` accepts a URL that it passes to the `ImageLoader` class to return the number of pages contained within the image:

```
[WebMethod]
public int GetPageCount(string url)
{
    return new ImageLoader().GetPageCount(url);
}
```


Building the Imaging HTTP Handler

The final component of the imaging services that we are building is the HTTP handler that is responsible for returning the actual binary of a requested page. We use an HTTP handler for this functionality so that the page binary can be referenced directly from a URL and returned as a stream to be properly displayed by a Silverlight Image control.

The HTTP handler consists of a class called `ImageHandler`. This class uses the `ProcessRequest` method of the `IHttpHandler` interface to extract some query string parameters to obtain URL and page number information; sets the return stream to a Silverlight-supported image format; and then uses the URL and page number to return the requested image page to the `OutputStream` of the HTTP response:

```
public void ProcessRequest(HttpContext context)
{
    HttpContext.Current.Response.Clear();
    HttpContext.Current.Response.ContentType = "image/png";

    string url = this.GetUrl(context.Request);

    if (!string.IsNullOrEmpty(url))
    {
        int page = this.GetPage(context.Request);

        byte[] binary = new ImageLoader().GetPage(url, page);

        context.Response.OutputStream.Write(binary, 0, binary.Length);
    }
}
```

Building the Viewer Web Part

Now that we have our services in place, the last component to build is the Silverlight viewer application that will be hosted in SharePoint's Silverlight Web Part. This application uses the MVVM design pattern, so understanding this pattern is important for understanding how this application functions. You should review the section "Model-View-ViewModel Primer" from earlier in this chapter for more information on this design pattern.

The viewer application consists of two main classes: `MainModel` and `MainPage`. `MainModel` serves as the `ViewModel` for the user interface; therefore, it inherits from `Model`. `MainPage` represents the user interface that is bound to `MainModel`.

When the application loads, `MainPage` is instantiated and everything is kicked into motion within the `UserControl_Loaded` event handler. The first thing this handler does is check for a "url" query string parameter. If there is one, it calls `LoadImage` of the `MainModel`, passing the provided URL:

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    this.model = this.DataContext as MainModel;

    string url = this.GetUrlFromQueryString();

    if (!string.IsNullOrEmpty(url))
        this.model.LoadImage(url);
}
```

```

    HtmlPage.RegisterScriptableObject("MainPage", this);
    HtmlPage.Window.Eval(string.Format("var slHost =
        document.getElementById('{0}')" , HtmlPage.Plugin.Id));
    HtmlPage.Window.Eval("function LoadImage(url){
        slHost.Content.MainPage.LoadImage(url); return false; }");
}

```

Within the call to `LoadImage`, the `MainModel` sets its `Url` and `Page` properties from the passed parameters. The method then makes a call to the imaging web service to get the number of pages for the requested image:

```

public void LoadImage(string url, int page)
{
    this.Url = url;
    this.Page = page;

    this.imageService.GetPageCountAsync(this.url);
}

```

When this service call returns, the `PageCount` property is set and the `INotifyPropertyChanged` event is raised for the `TransferUrl` to trigger the loading of the current page image:

```

private void LoadImageReady(int pageCount)
{
    this.PageCount = pageCount;
    this.OnPropertyChanged("TransferUrl");
}

```

When the `INotifyPropertyChanged` event is raised for the `TransferUrl`, the Silverlight `Image` control in `MainPage` makes a request to the URL exposed from the `TransferUrl` property of `MainModel` to which it is bound:

```
<Image x:Name="Image" Grid.Row="0" Source="{Binding TransferUrl}" />
```

The `TransferUrl` property consists of the URL to the image handler along with query string parameters for the SharePoint image URL and page number:

```

public string TransferUrl
{
    get
    {
        return string.Format("{0}?url={1}&page={2}", this.imageHandlerUrl,
            this.Url, this.Page);
    }
}

```

Making the Application Accessible from JavaScript

The final implementation detail to point out is how you make the application callable from JavaScript. This is accomplished using the Silverlight HTML Bridge. During the `UserControl_Loaded` handler, you registered the `MainPage` as a `ScriptableObject` and injected a JavaScript method into the current web page:

```

HtmlPage.RegisterScriptableObject("MainPage", this);
HtmlPage.Window.Eval(string.Format("var slHost =

```

```
document.getElementById('{0}'), HtmlPage.Plugin.Id));
HtmlPage.Window.Eval("function LoadImage(url){
    slHost.Content.MainPage.LoadImage(url); return false; }");
```

The JavaScript method that was injected into the page accepts a URL as a parameter and then passes it to the `MainPage` using the `ScriptableMember` of `LoadImage`. This scriptable method calls `LoadImage` on the `MainModel`, which causes the image to load in the `Image` control as discussed earlier:

```
[ScriptableMember]
public void LoadImage(string url)
{
    this.model.LoadImage(url);
}
```

Deployment

This section walks you through the deployment of the imaging services and the viewer application as a Web Part.

Deploying the Imaging Services

In order to deploy the imaging services, the project components should be copied to their required locations on the SharePoint web front end server as described in Table 10-4.

TABLE 10-4: Imaging Services Deployment

| COMPONENT | DEPLOYMENT |
|-------------------------|-----------------------|
| Listing1002.dll | Global Assembly Cache |
| ImageHandler.ashx | %14 Hive%\ISAPI |
| ImageService.asmx | %14 Hive%\ISAPI |
| ImageServicewsdl.aspx | %14 Hive%\ISAPI |
| ImageService disco.aspx | %14 Hive%\ISAPI |

Deploying the Viewer Application as a Web Part

In order to deploy the Silverlight viewer application as a Web Part, the compiled XAP file should be copied to %14 Hive%\TEMPLATE\LAYOUTS\ClientBin.

Once the XAP has been copied to this location, the built-in SharePoint Silverlight Web Part can be added to any Web Part page and then directed to load the deployed XAP, as shown in Figure 10-6.

Now that the key sections of the imaging services and viewer web part have been introduced, Listing 10-2 and Listing 10-3 will provide the full listing for these components.

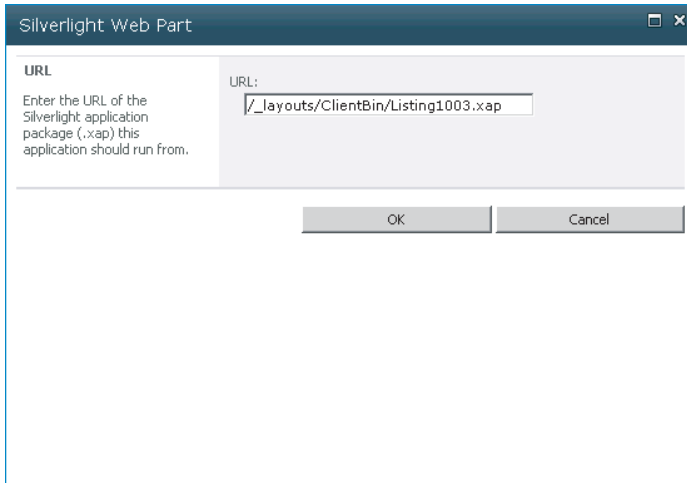


FIGURE 10-6

LISTING 10-2: Imaging Web Services

```
//Listing1002.ImageLoader.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.SharePoint;
using System.Drawing;
using System.IO;
using System.Drawing.Imaging;

namespace Listing1002
{
    class ImageLoader
    {
        public int GetPageCount(string url)
        {
            using (MemoryStream inStream = new MemoryStream(this.LoadImage(url)))
            using (Bitmap bitmap = new Bitmap(inStream))
            {
                return bitmap.GetFrameCount(FrameDimension.Page);
            }
        }

        public byte[] GetPage(string url, int page)
        {
            using (MemoryStream inStream = new MemoryStream(this.LoadImage(url)))
            using (Bitmap bitmap = new Bitmap(inStream))
            {
                bitmap.SelectActiveFrame(FrameDimension.Page, page);

                using (MemoryStream outStream = new MemoryStream())
```

```

        {
            bitmap.Save(outStream, ImageFormat.Png);
            return outStream.ToArray();
        }
    }

private byte[] LoadImage(string url)
{
    using (SPSite site = new SPSite(url))
    using (SPWeb web = site.OpenWeb())
    {
        SPFile file = web.GetFile(url);
        return file.OpenBinary();
    }
}
}
}

//Listing1002.ImageHandler.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.Services;
using System.Web;
using System.Web.SessionState;

namespace Listing1002
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class ImageHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            HttpContext.Current.Response.Clear();
            HttpContext.Current.Response.ContentType = "image/png";

            string url = this.GetUrl(context.Request);

            if (!string.IsNullOrEmpty(url))
            {
                int page = this.GetPage(context.Request);

                byte[] binary = new ImageLoader().GetPage(url, page);

                context.Response.OutputStream.Write(binary, 0, binary.Length);
            }
        }

        public bool IsReusable
        {

```

continues

LISTING 10-2 *(continued)*

```

        get
        {
            return false;
        }
    }

    private string GetUrl(HttpRequest request)
    {
        if (!string.IsNullOrEmpty(request["url"]))
            return request["url"];

        return string.Empty;
    }

    private int GetPage(HttpRequest request)
    {
        int page;

        if (!string.IsNullOrEmpty(request["page"]) &&
            int.TryParse(request["page"], out page) &&
            page > 0)
        {
            return page;
        }

        return 0;
    }
}

//Listing1002.ImageHandler.ashx

<%@ WebHandler Language="C#" Class="Listing1002.ImageHandler, Listing1002,
version=1.0.0.0, Culture=neutral, PublicKeyToken=9696ac8aeb53b872" %>

//Listing1002.ImageService.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Services;

namespace Listing1002
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    public class ImageService : System.Web.Services.WebService
    {
        [WebMethod]

```

```

        public int GetPageCount(string url)
        {
            return new ImageLoader().GetPageCount(url);
        }
    }
}

//Listing1002.ImageService.asmx

<%@ WebService Language="C#" Class="Listing1002.ImageService, Listing1002,
version=1.0.0.0, Culture=neutral, PublicKeyToken=9696ac8aeb53b872" %>

```

LISTING 10-3: Full Listing for the Simple Silverlight Viewer Web Part

```

//Listing1003.App.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace Listing1003
{
    public partial class App : Application
    {
        public App()
        {
            this.Startup += this.Application_Startup;
            this.Exit += this.Application_Exit;
            this.UnhandledException += this.Application_UnhandledException;

            InitializeComponent();
        }

        private void Application_Startup(object sender, StartupEventArgs e)
        {
            this.RootVisual = new MainPage();
        }

        private void Application_Exit(object sender, EventArgs e)
        {
        }

        private void Application_UnhandledException(object sender,

```

continues

LISTING 10-3 (continued)

```

        ApplicationUnhandledExceptionEventArgs e)
    {
        // If the app is running outside of the debugger
        // then report the exception using
        // the browser's exception mechanism.
        // On IE this will display it a yellow alert
        // icon in the status bar and Firefox will display a script error.
        if (!System.Diagnostics.Debugger.IsAttached)
        {
            // NOTE: This will allow the application to continue running
            // after an exception has been thrown
            // but not handled.
            // For production applications this error handling should be
            // replaced with something that will
            // report the error to the website and stop the application.
            e.Handled = true;
            Deployment.Current.Dispatcher.BeginInvoke(delegate {
                ReportErrorToDOM(e); });
        }
    }

    private void ReportErrorToDOM(ApplicationUnhandledExceptionEventArgs e)
    {
        try
        {
            string errorMsg = e.ExceptionObject.Message +
                e.ExceptionObject.StackTrace;
            errorMsg = errorMsg.Replace('"', '\\').Replace("\r\n", @"\n");

            System.Windows.Browser.HtmlPage.Window.Eval("throw
                new Error(\"Unhandled Error in Silverlight Application \" +
                    errorMsg + "\");");
        }
        catch (Exception)
        {
        }
    }
}

//Listing1003.App.xaml

<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="Listing1003.App"
    >
    <Application.Resources>

    </Application.Resources>
</Application>

```



```
//Listing1003.MainPage.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Windows.Browser;
using Listing1003.ViewModel;

namespace Listing1003
{
    public partial class MainPage : UserControl
    {
        private MainModel model = null;

        public MainPage()
        {
            InitializeComponent();
        }

        [ScriptableMember]
        public void LoadImage(string url)
        {
            this.model.LoadImage(url);
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
            this.model = this.DataContext as MainModel;

            string url = this.GetUrlFromQueryString();

            if (!string.IsNullOrEmpty(url))
                this.model.LoadImage(url);

            HtmlPage.RegisterScriptableObject("MainPage", this);
            HtmlPage.Window.Eval(string.Format("var slHost = document.getElementById('{0}');", HtmlPage.Plugin.Id));
            HtmlPage.Window.Eval("function LoadImage(url){ slHost.Content.MainPage.LoadImage(url); return false; }");
        }

        private string GetUrlFromQueryString()
        {
            if (HtmlPage.Document.QueryString.ContainsKey("url"))

```

continues

LISTING 10-3 *(continued)*

```

        return HtmlPage.Document.QueryString["url"];

        return string.Empty;
    }

    private void PreviousPageButton_Click(object sender, RoutedEventArgs e)
    {
        this.model.PreviousPage();
    }

    private void NextPageButton_Click(object sender, RoutedEventArgs e)
    {
        this.model.NextPage();
    }
}

//Listing1003.MainPage.xaml

<UserControl x:Class="Listing1003.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:Listing1003_ViewModel="clr-namespace:Listing1003.ViewModel"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400"
    Loaded="UserControl_Loaded">
    <UserControl.DataContext>
        <Listing1003_ViewModel:MainModel />
    </UserControl.DataContext>
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <Image x:Name="Image" Grid.Row="0" Source="{Binding TransferUrl}" />
        <StackPanel Orientation="Horizontal" Grid.Row="1"
            HorizontalAlignment="Center" Margin="5">
            <Button x:Name="PreviousPageButton" Content="&lt;" Width="30"
                Height="20" Margin="0,0,2,0"
                Click="PreviousPageButton_Click"/>
            <Button x:Name="NextPageButton" Content="&gt;" Width="30"
                Height="20" Margin="2,0,0,0"
                Click="NextPageButton_Click"/>
        </StackPanel>
        <TextBlock Text="{Binding PageOfCount}" Grid.Row="1" Margin="5"
            HorizontalAlignment="Right" VerticalAlignment="Bottom" />
    </Grid>
</UserControl>

//Listing1003.ViewModel.Model.cs

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;

namespace Listing1003.ViewModel
{
    public class Model : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        public Model() { }

        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
                this.PropertyChanged(this,
                    new PropertyChangedEventArgs(propertyName));
        }
    }
}

//Listing1003.ViewModel.MainModel.cs

using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Windows.Browser;
using Listing1003.ImageClient;

namespace Listing1003.ViewModel
{
    public class MainModel : Model
    {
        private const string IMAGESERVICE = "_vti_bin/ImageService.asmx";
        private const string IMAGEHANDLER = "_vti_bin/ImageHandler.ashx";

        private ImageServiceSoapClient imageService = null;
        private string imageHandlerUrl = string.Empty;
        private string url = string.Empty;
        private string message = string.Empty;
        private int page = 0;
        private int pageCount = 0;

        public MainModel()
```

continues

LISTING 10-3 *(continued)*

```

    {
        string schemeAndServer = HtmlPage.Document.DocumentUri.GetComponents(
UriComponents.SchemeAndServer, UriFormat.Unescaped);

        this.imageHandlerUrl = string.Format("{0}/{1}",
            schemeAndServer, IMAGEHANDLER);

        this.imageService = new ImageServiceSoapClient("ImageServiceSoap",
            string.Format("{0}/{1}",
                schemeAndServer, IMAGESERVICE));
        this.imageService.GetPageCountCompleted +=
            new EventHandler<GetPageCountCompletedEventArgs>(
                imageService_GetPageCountCompleted);
    }

public string Url
{
    get{ return this.url; }
    private set
    {
        this.url = value;
        this.OnPropertyChanged("Url");
    }
}

public int Page
{
    get { return this.page; }
    private set
    {
        this.page = value;
        this.OnPropertyChanged("Page");
        this.OnPropertyChanged("PageOfCount");
    }
}

public int PageCount
{
    get { return this.pageCount; }
    private set
    {
        this.pageCount = value;
        this.OnPropertyChanged("PageCount");
        this.OnPropertyChanged("PageOfCount");
    }
}

public string PageOfCount
{
    get { return string.Format("{0} of {1}" ,

```

```
        this.Page + 1, this.PageCount); }
    }

    public string TransferUrl
    {
        get
        {
            return string.Format("{0}?url={1}&page={2}",
                this.imageHandlerUrl,
                this.Url, this.Page);
        }
    }

    public string Message
    {
        get { return this.message; }
        private set
        {
            this.message = value;
            this.OnPropertyChanged("Message");
        }
    }

    public void LoadImage(string url)
    {
        this.LoadImage(url, 0);
    }

    public void LoadImage(string url, int page)
    {
        this.Url = url;
        this.Page = page;

        this.imageService.GetPageCountAsync(this.url);
    }

    public void PreviousPage()
    {
        if (this.Page > 0)
        {
            this.Page--;
            this.OnPropertyChanged("TransferUrl");
        }
    }

    public void NextPage()
    {
        if (this.Page < this.PageCount - 1)
        {
            this.Page++;
            this.OnPropertyChanged("TransferUrl");
        }
    }
}
```

continues

LISTING 10-3 *(continued)*

```

    }

    private void LoadImageReady(int pageCount)
    {
        this.PageCount = pageCount;
        this.OnPropertyChanged("TransferUrl");
    }

    private void imageService_GetPageCountCompleted(
        object sender, GetPageCountCompletedEventArgs e)
    {
        if (e.Error != null)
        {
            this.Message = "Error Loading Page Count";
        }
        else
            this.LoadImageReady(e.Result);
    }
}
}
}

```

SETTING UP THE SHAREPOINT INFRASTRUCTURE

Now that all the components are built and deployed, the final step to complete the solution is configuring SharePoint. This section will guide you through that process.

Configuring SharePoint Search

The first step is configuration of the Microsoft SharePoint search service application to automatically generate managed properties when it discovers properties during the crawl process. This will ensure that as new columns are created in SharePoint, a corresponding managed property is automatically created by SharePoint Search during the crawl. In order to configure this, the following steps must be performed:

1. Open Central Administration and click Manage Service Applications under the Application Management section.
2. Select Search Service Application. Then, within the Ribbon menu, click Manage.
3. Within Search Service Administration, click Metadata Properties.
4. Click Categories, then select the SharePoint from the list of categories, and then click Edit Category.
5. Check “Automatically generate a new managed property for each crawled property discovered in this category” (see Figure 10-7).
6. Click OK.

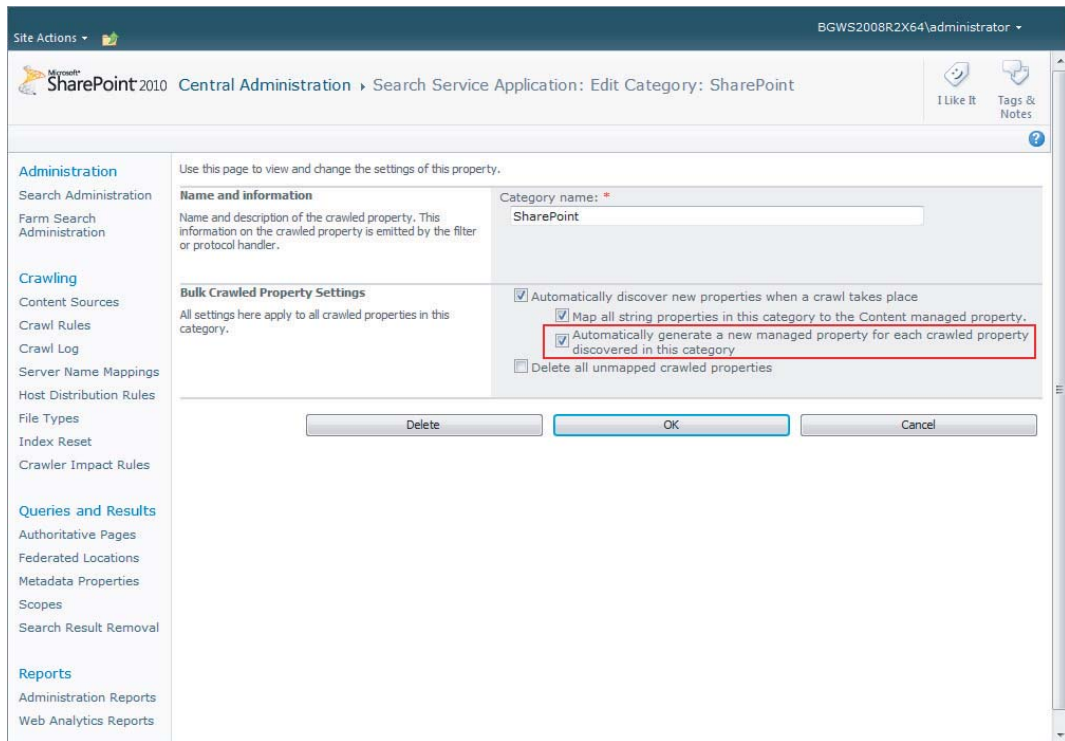


FIGURE 10-7

Creating the SharePoint Content Type

The next configuration step is creating the SharePoint content type that will be used in the solution. Navigate to the site content type gallery and create a content type called “Chapter10” that inherits from Document with the columns listed in Table 10-5.

TABLE 10-5: Chapter 10 Content Type Columns

| COLUMN | COLUMN TYPE |
|----------------|---------------------|
| InvoiceDate | Date and Time |
| InvoiceNumber | Single line of text |
| VendorNumber | Single line of text |
| VendorName | Single line of text |
| VendorPONumber | Single line of text |
| PODate | Date and Time |
| Amount | Number |

Creating the SharePoint Document Library

The next configuration step is creating the SharePoint document library that will be used in the solution. To do this, create a new document library called **Chapter10**. Then enable Content Type Management from Library Settings and add the Chapter10 content type to this library, as shown in Figure 10-8.

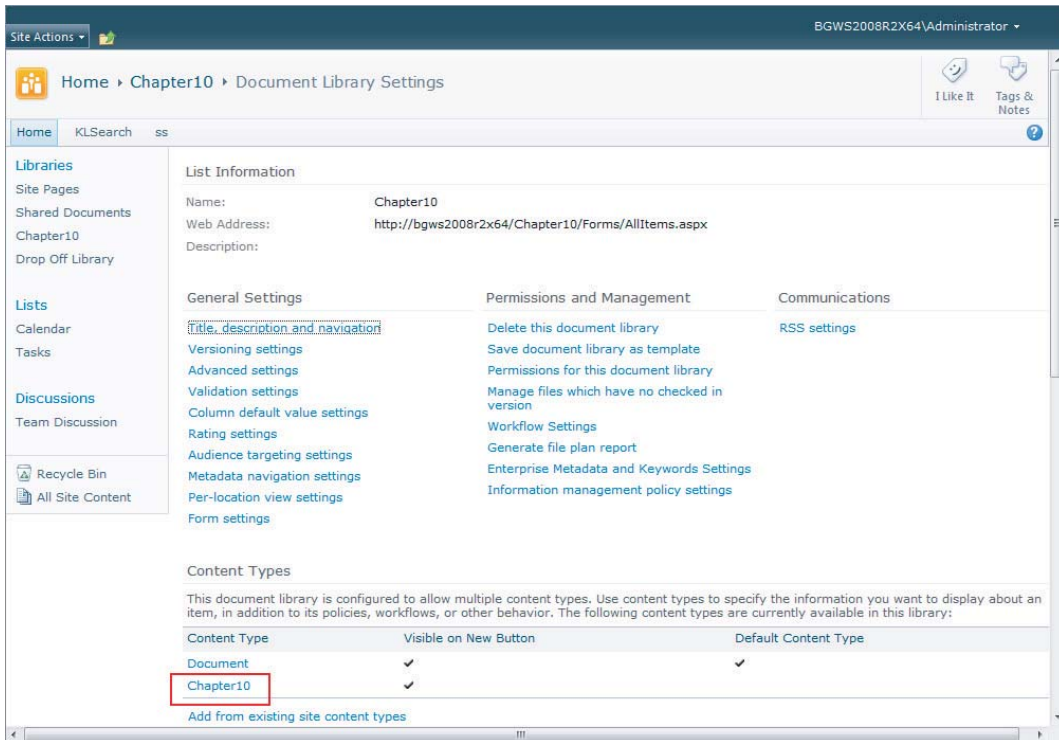


FIGURE 10-8

Creating the SharePoint Web Part Page

The next configuration step is to create a SharePoint Web Part page that will be used to host your solution. To do this, the following actions should be performed:

1. From Site Settings, click More Options.
2. Select Web Part Page and click Create.
3. Enter **Chapter10** for the page name.
4. For the layout template, select Header, Footer, 2Columns, 4 Rows.
5. Select Site Assets for the document library save location (see Figure 10-9).
6. Click Create.

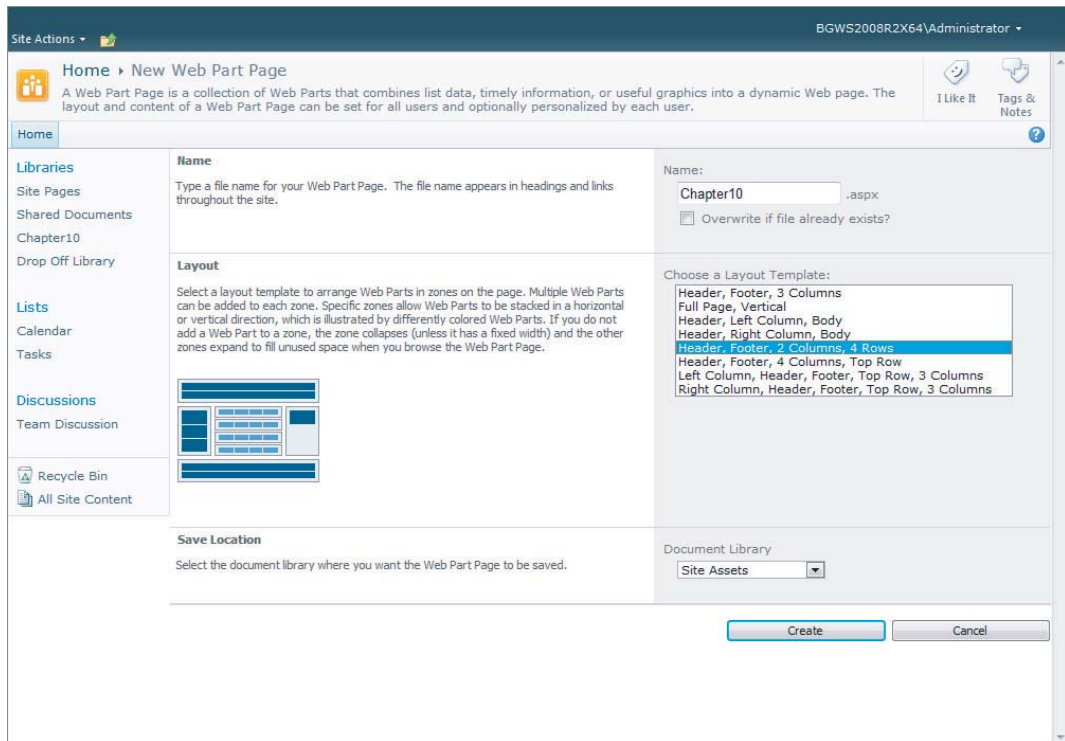


FIGURE 10-9

Setting Up the SharePoint Web Part Page

The final solution configuration step is to set up the newly created Web Part page. Navigate to the Site Assets document library and perform the following actions:

1. Click Edit Page.
2. Add the Search Core Results Web Part to the Left Column section.
3. Add the Advanced Search Box Web Part to the Row 1 section.
4. Add the Silverlight Web Part to the Right Column section and point it to the Silverlight Viewer Application you deployed earlier in this chapter.
5. Now customize the Advanced Search Box Web Part and the Search Core Results Web Part as outlined in the following sections.

Customizing the Advanced Search Box Web Part

To configure the Advanced Search Box, the following actions must be performed:

1. Click Edit Page.
2. From the drop-down menu in the upper right corner of the web part, select Edit Web Part.

3. In the Scopes section of the Editor Part, uncheck Show Scope Picker and Show Languages Picker.
4. Under the Property section in the Editor Part, select the ellipsis next to the Properties text box.
5. Paste Listing 10-4 into the dialog that opens. Listing 10-4 provides XML that is used to customize the properties that are displayed in the Property Restrictions section of the Advanced Search Web Part. The XML configures the Web Part to display the properties for the Chapter10 content type created earlier in the chapter.
6. Click OK in the Editor Part to save the Web Part configuration.

LISTING 10-4: Custom Advanced Search Property XML

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <LangDefs>
    <LangDef DisplayName="English" LangID="en"/>
  </LangDefs>
  <Languages>
    <Language LangRef="en"/>
  </Languages>
  <PropertyDefs>
    <PropertyDef Name="Path" DataType="text" DisplayName="URL"/>
    <PropertyDef Name="Size" DataType="integer" DisplayName="Size (bytes)"/>
    <PropertyDef Name="Write" DataType="datetime"
      DisplayName="Last Modified Date"/>
    <PropertyDef Name="FileName" DataType="text" DisplayName="Name"/>
    <PropertyDef Name="Title" DataType="text" DisplayName="Title"/>
    <PropertyDef Name="Author" DataType="text" DisplayName="Author"/>
    <PropertyDef Name="owsAmount" DataType="decimal" DisplayName="Amount"/>
    <PropertyDef Name="owsInvoiceDate" DataType="datetime"
      DisplayName="Invoice Date"/>
    <PropertyDef Name="owsInvoiceNumber" DataType="text"
      DisplayName="Invoice Number"/>
    <PropertyDef Name="owsVendorName" DataType="text" DisplayName="Vendor Name"/>
    <PropertyDef Name="owsVendorNumber" DataType="text"
      DisplayName="Vendor Number"/>
    <PropertyDef Name="owsVendorPONumber" DataType="text"
      DisplayName="Vendor PO Number"/>
    <PropertyDef Name="owsPODate" DataType="datetime" DisplayName="PO Date"/>
  </PropertyDefs>
  <ResultTypes>
    <ResultType DisplayName="TIFFs" Name="default">
      <KeywordQuery>FileExtension="tif" OR FileExtension="tiff"</KeywordQuery>
      <PropertyRef Name="Title" />
      <PropertyRef Name="Author" />
      <PropertyRef Name="FileName" />
      <PropertyRef Name="Size" />
      <PropertyRef Name="Path" />
      <PropertyRef Name="Write" />
      <PropertyRef Name="owsAmount" />
    </ResultType>
  </ResultTypes>
</root>
```

```

    <PropertyRef Name="owsInvoiceDate" />
    <PropertyRef Name="owsInvoiceNumber" />
    <PropertyRef Name="owsVendorName" />
    <PropertyRef Name="owsVendorNumber" />
    <PropertyRef Name="owsVendorPONumber" />
    <PropertyRef Name="owsPODate" />
    <PropertyRef Name="CreatedBy" />
    <PropertyRef Name="ModifiedBy" />
  </ResultType>
</ResultTypes>
</root>

```

Customizing the Search Core Results Web Part

To configure the Search Results Core Web Part, click Edit Web Part. In the Editor Part, under the Miscellaneous section, uncheck Enable Data View Caching. Next, under the Display Properties section, click XSL Editor. When the dialog opens, paste Listing 10-5 into the dialog and click Save. Now click OK in the Editor Part to save the Web Part configuration. Listing 10-5 provides custom XSL that is used to format the search results that are returned from SharePoint. This listing is nearly equivalent to the XSL that is used by default. The main difference is that the link that is generated for each item is modified to call the JavaScript method exposed by the Silverlight viewer application, as shown in the following snippet from Listing 10-5:

```

<!-- Chapter 10: Add New HREF to call Silverlight Viewer Web Part -->
<a id="{concat($currentId, '_Title')}">
  <xsl:attribute name="href">
    <xsl:text>javascript:LoadImage('</xsl:text>
    <xsl:value-of select="$url" />
    <xsl:text>'); void(0);</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="title">
    <xsl:value-of select="title"/>
  </xsl:attribute>
  <xsl:choose>
    <xsl:when test="hithighlightedproperties/HHTitle[. != '']">
      <xsl:call-template name="HitHighlighting">
        <xsl:with-param name="hh" select="hithighlightedproperties/HHTitle" />
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise><xsl:value-of select="title"/></xsl:otherwise>
  </xsl:choose>
</a>

```

LISTING 10-5: Custom Core Results XSLT

```

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:srwrt="http://schemas.microsoft.com/WebParts/v3/searchresults/runtime"
  xmlns:ddwrt="http://schemas.microsoft.com/WebParts/v2/DataView/runtime">
  <xsl:output method="xml" indent="no"/>
  <xsl:param name="Keyword" />

```

continues

LISTING 10-5 (continued)

```
<xsl:param name="ResultsBy" />
<xsl:param name="ViewByUrl" />
<xsl:param name="ShowDropDown" />
<xsl:param name="ViewByValue" />
<xsl:param name="SortBy" />
<xsl:param name="SortOptions" />
<xsl:param name="Relevancy" />
<xsl:param name="ModifiedDate" />
<xsl:param name="DropDownOption" />
<xsl:param name="Multiply" />
<xsl:param name="PictureTaken" />
<xsl:param name="IsNoKeyword" />
<xsl:param name="IsFixedQuery" />
<xsl:param name="ShowActionLinks" />
<xsl:param name="MoreResultsText" />
<xsl:param name="MoreResultsLink" />
<xsl:param name="CollapsingStatusLink" />
<xsl:param name="CollapseDuplicatesText" />
<xsl:param name="AlertMeLink" />
<xsl:param name="AlertMeText" />
<xsl:param name="SrchRSSText" />
<xsl:param name="SrchRSSLink" />
<xsl:param name="SearchProviderText" />
<xsl:param name="SearchProviderLink" />
<xsl:param name="SearchProviderAlt"/>
<xsl:param name="ShowMessage" />
<xsl:param name="IsThisListScope" />
<xsl:param name="DisplayDiscoveredDefinition" select="True" />
<xsl:param name="NoFixedQuery" />
<xsl:param name="NoKeyword" />
<xsl:param name="ResultsNotFound" />
<xsl:param name="NoResultsSuggestion" />
<xsl:param name="NoResultsSuggestion1" />
<xsl:param name="NoResultsSuggestion2" />
<xsl:param name="NoResultsSuggestion3" />
<xsl:param name="NoResultsSuggestion4" />
<xsl:param name="NoResultsSuggestion5" />
<xsl:param name="AdditionalResources" />
<xsl:param name="AdditionalResources1" />
<xsl:param name="AdditionalResources2" />
<xsl:param name="IsSearchServer" />
<xsl:param name="Period" />
<xsl:param name="SearchHelp" />
<xsl:param name="Tags" />
<xsl:param name="Authors" />
<xsl:param name="Date" />
<xsl:param name="Size" />
<xsl:param name="ViewInBrowser" />
<xsl:param name="DefinitionIntro" />
<xsl:param name="IdPrefix" />
<xsl:param name="LangPickerHeading" />
<xsl:param name="LangPickerNodeSet" />
```

```

<xsl:param name="IsDesignMode">True</xsl:param>

<!-- When there is keyword to issue the search -->
<xsl:template name="dvt_1.noKeyword">
  <span class="srch-description2">
    <xsl:choose>
      <xsl:when test="$IsFixedQuery">
        <xsl:value-of select="$NoFixedQuery" />
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$NoKeyword" />
      </xsl:otherwise>
    </xsl:choose>
  </span>
</xsl:template>

<!-- When empty result set is returned from search -->
<xsl:template name="dvt_1.empty">
  <div class="srch-results">
    <xsl:if test="$AlertMeLink and $ShowActionLinks">
      <span class="srch-alertme" > <a href ="{$AlertMeLink}" id="CSR_AM1"
        title="{ $AlertMeText}" >
    
    <xsl:text disable-output-escaping="yes">&nbsp;&nbsp;&nbsp;</xsl:text>
    <xsl:value-of select="$AlertMeText" /></a>
  </span>
</xsl:if>

  <xsl:if test="string-length($SrchRSSLink) &gt; 0 and $ShowActionLinks">
    <xsl:if test="$AlertMeLink">
      |
    </xsl:if>
    <a type="application/rss+xml" href ="{$SrchRSSLink}" title="{ $SrchRSSText}"
    id="SRCHRSSL"><xsl:text disable-output-
    escaping="yes">&nbsp;&nbsp;&nbsp;</xsl:text><xsl:value-of select="$SrchRSSText" /></a>
    <xsl:if test="string-length($SearchProviderLink) &gt; 0">
      |
    <a href ="{$SearchProviderLink}" title="{ $SearchProviderText}" >
    
    <xsl:text disable-output-escaping="yes">&nbsp;&nbsp;&nbsp;</xsl:text>
    <xsl:value-of select="$SearchProviderText" /></a>
    </xsl:if>
  </xsl:if>
</div>

<div class="srch-results" accesskey="W">
<span class="srch-description2" id="CSR_NO_RESULTS">
  <p>
    <xsl:value-of select="$ResultsNotFound" />
    <xsl:text disable-output-escaping="yes">&nbsp;&nbsp;&nbsp;</xsl:text>

```

continues

LISTING 10-5 (continued)

```

        <strong><xsl:value-of select="$Keyword" /></strong>
        <xsl:value-of select="$Period" />
    </p>
    <h3>
        <xsl:value-of select="$NoResultsSuggestion" />
    </h3>
    <ul>
        <li><xsl:value-of select="$NoResultsSuggestion1" /></li>
        <li><xsl:value-of select="$NoResultsSuggestion2" /></li>
        <li><xsl:value-of select="$NoResultsSuggestion3" /></li>
        <xsl:if test="string-length($NoResultsSuggestion4) > 0">
            <li><xsl:value-of select="$NoResultsSuggestion4" /></li>
        </xsl:if>
        <xsl:if test="string-length($NoResultsSuggestion5) > 0">
            <li><xsl:value-of select="$NoResultsSuggestion5" /></li>
        </xsl:if>
    </ul>
    <h3>
        <xsl:value-of select="$AdditionalResources" />
    </h3>
    <ul>
        <li><xsl:value-of select="$AdditionalResources1" />
        <xsl:text disable-output-escaping="yes">&nbsp;&nbsp;&nbsp;</xsl:text>
        <xsl:choose>
            <xsl:when test="string-length($IsSearchServer) > 0">
                <a href="javascript:HelpWindowKey('MSEndUser_FindContent')"
                label="$SearchHelp">
            <xsl:value-of select="$SearchHelp" /></a>
            </xsl:when>
            <xsl:otherwise>
                <a href="javascript:HelpWindowKey('WSEndUser_FindContent')"
                label="$SearchHelp">
            <xsl:value-of select="$SearchHelp" /></a>
            </xsl:otherwise>
        </xsl:choose>
        </li>
        <li><xsl:value-of select="$AdditionalResources2" /></li>
    </ul>

</span>
</div>
</xsl:template>

<!-- Main body template. Sets the Results view (Relevance or date) options -->
<xsl:template name="dvt_1.body">
    <xsl:if test="$ShowActionLinks">
        <div class="srch-sort-right2" accesskey="W">
            <xsl:if test="$LangPickerNodeSet and count($LangPickerNodeSet) > 0">
                <xsl:value-of select="$LangPickerHeading" />
                <select class="srch-dropdown" onchange="window.location.href=this.value"

```

```

id="langpickerdd">
  <xsl:for-each select="$LangPickerNodeSet">
    <xsl:element name="option">
      <xsl:attribute name="value"><xsl:value-of select="@url"/>
    </xsl:attribute>
    <xsl:if test="@selected = 'true'">
      <xsl:attribute name="selected">selected</xsl:attribute>
    </xsl:if>
    <xsl:value-of select="@title"/>
  </xsl:element>
</xsl:for-each>
</select>
<xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
</xsl:if>
<xsl:if test="$ShowDropDown = 'true'">
  <xsl:value-of select="$SortBy" />
<select id="dropdown" title="{ $SortOptions}"
onchange="PostToUrl(this.value)" class="srch-dropdown">
  <xsl:if test="$DropDownOption = '0' or $ViewByUrl != ''">
    <xsl:element name="option">
      <xsl:attribute name="value"><xsl:value-of select="$ViewByUrl"/>
    </xsl:attribute>
    <xsl:if test="$DropDownOption = '0'"><xsl:attribute
name="selected">selected</xsl:attribute></xsl:if>
    <xsl:value-of select="$Relevancy"/>
  </xsl:element>
</xsl:if>
  <xsl:if test="$DropDownOption = '1' or $ViewByUrl != ''">
    <xsl:element name="option">
      <xsl:attribute name="value"><xsl:value-of select="$ViewByUrl"/>
    </xsl:attribute>
    <xsl:if test="$DropDownOption = '1'"><xsl:attribute
name="selected">selected</xsl:attribute></xsl:if>
    <xsl:value-of select="$ModifiedDate"/>
  </xsl:element>
</xsl:if>
</select>
</xsl:if>
  <xsl:if test="$AlertMeLink">
    <xsl:if test="$ShowDropDown = 'true'">
      </xsl:if>
      <span class="srch-alertme" > <a href="{ $AlertMeLink}" id="CSR_AM2"
title="{ $AlertMeText}">
</a>
<xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
</span>
</xsl:if>
    <xsl:if test="string-length($SrchRSSLink) &gt; 0">
      <a type="application/rss+xml" href="{ $SrchRSSLink}"
title="{ $SrchRSSText}" id="SRCHRSSL">
</a>

```

continues

LISTING 10-5 (continued)

```

<xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
  </xsl:if>
  <xsl:if test="string-length($SearchProviderLink) > 0">
    <a href="{ $SearchProviderLink}" title="{ $SearchProviderAlt}" >
</a>
    </xsl:if>
  </div>
</xsl:if>
<div class="srch-results" accesskey="W">
<xsl:apply-templates />
</div>
<xsl:call-template name="DisplayMoreResultsAnchor" />
</xsl:template>
<!-- This template is called for each result -->
<xsl:template match="TotalResults">
</xsl:template>
<xsl:template match="NumberOfResults">
</xsl:template>

<xsl:template match="Result">
  <xsl:variable name="id" select="id"/>
  <xsl:variable name="currentId" select="concat($IdPrefix,$id)"/>
  <xsl:variable name="url" select="url"/>

  <div class="srch-Icon" id="{concat($currentId,'_Icon')}">
    
  </div>
  <div class="srch-Title2">
  <div class="srch-Title3">
    <!-- links with the file scheme only work in ie if they are unescaped. For
        this reason here we will render the link using
disable-output-escaping if the url begins with file.-->
    <xsl:choose>
      <xsl:when test="substring($url,1,5) = 'file:' and $IsDesignMode = 'False'">
        <xsl:text disable-output-escaping="yes">&lt;a href=" />
        <xsl:value-of disable-output-escaping="yes"
select="srwt:HtmlAttributeEncode($url)" />
        <xsl:text disable-output-escaping="yes">" id=" />
        <xsl:value-of disable-output-escaping="yes"
select="srwt:HtmlAttributeEncode(concat($currentId,'_Title'))" />
        <xsl:text disable-output-escaping="yes">" title=" />
        <xsl:value-of disable-output-escaping="yes"
select="srwt:HtmlAttributeEncode(title)" />
        <xsl:text disable-output-escaping="yes">"&gt;</xsl:text>
      <xsl:choose>
        <xsl:when test="hithighlightedproperties/HHTitle[. != '']">
          <xsl:call-template name="HitHighlighting">
            <xsl:with-param name="hh" select="hithighlightedproperties/HHTitle" />
          </xsl:call-template>
        </xsl:when>

```



```

        <xsl:otherwise><xsl:value-of select="srwr:HtmlEncode(title)"/>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text disable-output-escaping="yes">&lt;/a&gt;</xsl:text>
</xsl:when>
<xsl:otherwise>
    <!-- Chapter 10: Add New HREF to call Silverlight Viewer Web Part -->
    <a id="{concat($currentId, '_Title')}">
        <xsl:attribute name="href">
            <xsl:text>javascript:LoadImage('</xsl:text>
            <xsl:value-of select="$url" />
            <xsl:text>'); void(0);</xsl:text>
        </xsl:attribute>
        <xsl:attribute name="title">
            <xsl:value-of select="title"/>
        </xsl:attribute>
    <xsl:choose>
        <xsl:when test="hithighlightedproperties/HHTitle[. != '']">
            <xsl:call-template name="HitHighlighting">
                <xsl:with-param name="hh" select="hithighlightedproperties/HHTitle" />
            </xsl:call-template>
        </xsl:when>
        <xsl:otherwise><xsl:value-of select="title"/></xsl:otherwise>
    </xsl:choose>
    </a>
</xsl:otherwise>
</xsl:choose>
</div>
</div>

<div class="srch-Description2">

<xsl:choose>
<xsl:when test="hithighlightedsummary[. != '']">
    <xsl:call-template name="HitHighlighting">
        <xsl:with-param name="hh" select="hithighlightedsummary" />
    </xsl:call-template>
</xsl:when>
<xsl:when test="description[. != '']">
    <xsl:value-of select="description"/>
</xsl:when>
<xsl:otherwise>
    
</xsl:otherwise>
</xsl:choose>
</div >

<div class="srch-Metadata2">
    <xsl:call-template name="DisplayAuthors">
        <xsl:with-param name="author" select="author" />
    </xsl:call-template>
    <xsl:call-template name="DisplayDate">
        <xsl:with-param name="write" select="write" />
    </xsl:call-template>

```

continues

LISTING 10-5 (continued)

```

<xsl:if test="string-length(popularsocialtag0) &gt; 0">
  <xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
  <xsl:value-of select="$Tags" />
  <xsl:value-of select="popularsocialtag0"/>
<xsl:if test="string-length(popularsocialtag1) &gt; 0">
  ::
  <xsl:value-of select="popularsocialtag1"/>
</xsl:if>
<xsl:if test="string-length(popularsocialtag2) &gt; 0">
  ::
  <xsl:value-of select="popularsocialtag2"/>
</xsl:if>
</xsl:if>
<xsl:call-template name="DisplaySize">
  <xsl:with-param name="size" select="size" />
</xsl:call-template>

</div>

<p class="srch-Metadatal">
<span><span class="srch-URL2" id="{concat($currentId, '_Url')}">

  <xsl:choose>
    <xsl:when test="hithighlightedproperties/HHUrl[. != '']">
      <xsl:call-template name="HitHighlighting">
        <xsl:with-param name="hh" select="hithighlightedproperties/HHUrl" />
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="url"/>
    </xsl:otherwise>
  </xsl:choose>
</span>
<xsl:call-template name="DisplayCollapsingStatusLink">
  <xsl:with-param name="status" select="collapsingstatus"/>
  <xsl:with-param name="workid" select="workid"/>
  <xsl:with-param name="id" select="concat($currentId, '_CS')"/>
</xsl:call-template>
<xsl:call-template name="ViewInBrowser">
  <xsl:with-param name="browserlink" select="serverredirectedurl" />
  <xsl:with-param name="currentId" select="$currentId" />
</xsl:call-template>
</span>
</p>
</xsl:template>

<xsl:template name="HitHighlighting">
  <xsl:param name="hh" />
  <xsl:apply-templates select="$hh"/>
</xsl:template>

<xsl:template match="ddd">

```

```

    &#8230;
</xsl:template>
<xsl:template match="c0">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c1">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c2">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c3">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c4">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c5">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c6">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c7">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c8">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c9">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>

<xsl:template name="DisplayAuthors">
    <xsl:param name="author" />
    <xsl:if test="string-length($author) > 0">
        <xsl:value-of select="$Authors" />
        <xsl:choose>
            <xsl:when test="string-length(author_multival) > 0">
                <xsl:for-each select="author_multival">
                    <xsl:variable name="p" select="position()"/>
                    <xsl:if test="$p > 1">
                        <xsl:text disable-output-escaping="yes">&#44;</xsl:text>
                        <xsl:text disable-output-escaping="yes">&#32;</xsl:text>
                    </xsl:if>
                    <xsl:value-of select="."/>
                </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="author"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
</xsl:template>

```

continues

LISTING 10-5 (continued)

```

<xsl:template name="DisplayDate">
  <xsl:param name="write" />
  <xsl:if test="string-length($write) > 0">
    <xsl:if test="string-length(author) > 0">
      <xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
    </xsl:if>
    <xsl:value-of select="$Date" />
    <xsl:value-of select="write"/>
  </xsl:if>
</xsl:template>

<!-- The size attribute for each result is prepared here -->
<xsl:template name="DisplaySize">
  <xsl:param name="size" />
  <xsl:if test="string-length($size) > 0">
    <xsl:if test="number($size) > 0">
      <xsl:if test="string-length(write) > 0 or string-length(author) > 0">
        <xsl:text disable-output-escaping="yes">&#8195;</xsl:text>
      </xsl:if>
      <xsl:value-of select="$Size" />
      <xsl:choose>
        <xsl:when test="round($size div 1024) < 1">
<xsl:value-of select="$size" /> Bytes</xsl:when>
        <xsl:when test="round($size div (1024 * 1024)) < 1">
<xsl:value-of select="round($size div 1024)" />KB</xsl:when>
        <xsl:otherwise>
<xsl:value-of select="round($size div (1024 * 1024))"/>MB</xsl:otherwise>
      </xsl:choose>
    </xsl:if>
  </xsl:if>
</xsl:template>

<xsl:template name="ViewInBrowser">
  <xsl:param name="browserlink" />
  <xsl:param name="currentId" />
  <xsl:if test="string-length($browserlink) > 0">
    <span class="srch-urllink"><a href="{ $browserlink}"
id="{concat($currentId, '_Vblink')}">
      <xsl:value-of select="$ViewInBrowser" />
    </a></span>
  </xsl:if>
</xsl:template>

<!-- A generic template to display string with non 0 string length (used for
author and lastmodified time -->
<xsl:template name="DisplayString">
  <xsl:param name="str" />
  <xsl:if test="string-length($str) > 0">
    -
    <xsl:value-of select="$str" />
  </xsl:if>

```

```

</xsl:template>

<!-- document collapsing link setup -->
<xsl:template name="DisplayCollapsingStatusLink">
  <xsl:param name="status" />
  <xsl:param name="workid" />
  <xsl:param name="id" />
  <xsl:if test="$CollapsingStatusLink">
    <xsl:choose>
      <xsl:when test="$status=1">
        <xsl:variable name="CollapsingStatusHref"
select="concat(substring-before($CollapsingStatusLink,
'$COLLAPSE_PARAM$'), 'duplicates:&quot;', $workid,
'&quot;', substring-after($CollapsingStatusLink, '$COLLAPSE_PARAM$'))"/>
        <span class="srch-urllink"><a href="{ $CollapsingStatusHref}"
id="$id" title="{ $CollapseDuplicatesText}">
          <xsl:value-of select="$CollapseDuplicatesText" />
        </a></span>
      </xsl:when>
    </xsl:choose>
  </xsl:if>
</xsl:template>

<!-- The "view more results" for fixed query -->
<xsl:template name="DisplayMoreResultsAnchor">
  <xsl:if test="$MoreResultsLink">
    <a href="{ $MoreResultsLink}" id="{concat($IdPrefix, '_MRL')} ">
      <xsl:value-of select="$MoreResultsText" />
    </a>
  </xsl:if>
</xsl:template>

<xsl:template match="All_Results/DiscoveredDefinitions">
  <xsl:variable name="FoundIn" select="DDFoundIn" />
  <xsl:variable name="DDSearchTerm" select="DDSearchTerm" />
  <xsl:if test="$DisplayDiscoveredDefinition = 'True' and
string-length($DDSearchTerm) > 0">
    <script language="javascript">
      function ToggleDefinitionSelection()
      {
        var selection = document.getElementById("definitionSelection");
        if (selection.style.display == "none")
        {
          selection.style.display = "inline";
        }
        else
        {
          selection.style.display = "none";
        }
      }
    </script>
    <div class="srch-Description2 srch-definition2">
      <a href="javascript:ToggleDefinitionSelection();"

```

continues

LISTING 10-5 (continued)

```

id="{concat($IdPrefix,'1_DEF')}" mss_definition="true">
  <xsl:value-of select="$DefinitionIntro" /><strong><xsl:value-of
select="$DDSearchTerm"/></strong></a>
  <div id="definitionSelection" class="srch-Description2"
    style="display:none;">
    <xsl:for-each select="DDefinitions/DDefinition">
      <br/>
      <xsl:variable name="DDUrl" select="DDUrl" />
      
      <xsl:value-of select="DDStart"/>
      <strong>
        <xsl:value-of select="DDBold"/>
      </strong>
      <xsl:value-of select="DDEnd"/>
      <br/>
      <span class="srch-definition">
        <xsl:value-of select="$FoundIn"/>
        <xsl:text disable-output-escaping="yes">&#160;</xsl:text>
        <a href="{ $DDUrl }">
          <xsl:value-of select="DDTitle"/>
        </a>
      </span>
    </xsl:for-each>
  </div>
</div>
</xsl:if>
</xsl:template>

<!-- XSL transformation starts here -->
<xsl:template match="/">
  <xsl:if test="$AlertMeLink">
    <input type="hidden" name="P_Query" />
    <input type="hidden" name="P_LastNotificationTime" />
  </xsl:if>
  <xsl:choose>
    <xsl:when test="$IsNoKeyword = 'True'" >
      <xsl:call-template name="dvt_1.noKeyword" />
    </xsl:when>
    <xsl:when test="$ShowMessage = 'True'">
      <xsl:call-template name="dvt_1.empty" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="dvt_1.body"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<!-- End of Stylesheet -->
</xsl:stylesheet>

```

THE SOLUTION FROM END TO END

With all the components in place, the solution can now be leveraged to perform document imaging using SharePoint. Returning to the beginning of the chapter, recall the existing paper-intensive process. We are going to replace this manual, paper-intensive process with an electronic content enabled process.

First, open the simple capture application created earlier. Click Load Columns from the Ribbon menu and enter the URL to your SharePoint site, along with the Chapter10 library name, and the Chapter10 content type name. The columns created earlier will load into the index panel in the capture application.

Now, imagine that you are scanning documents: Click Scan in the Ribbon menu. This will open a dialog that enables you to select an image file that will be loaded into the capture application.

The current columns being displayed are common columns used during the capture of Invoice documents. Next, pretend to enter the invoice data from the “scanned” document into their corresponding columns. Finally, click the Release button in the Ribbon to send the document to SharePoint.

Once the document is in SharePoint, navigate to the Chapter10 Web Part Page we created in SharePoint. Within the Advanced Search Box Web Part, add a property restriction for a column that was used to index the documents and click Search. The Search Core Results Web Part now returns the document that you sent to SharePoint with the capture application. By clicking the title of the document that is returned, the URL to this document is passed to the Silverlight Viewer Web Part using JavaScript that was injected into the page, causing the image to be displayed within the viewer.

There you have it, an end-to-end solution that includes scanning a document, indexing the document, sending the document to SharePoint, then finding the document using the document’s index data and loading the document into a viewer for display.

SUMMARY

Most organizations are faced with inefficient paper-intensive processes somewhere within their business. Document imaging provides a solution to this problem by providing the means to convert these documents into electronic format and send them to the document repository, where users can easily find and view them for use in various business operations.

By leveraging SharePoint’s extensibility, you were able to create an end-to-end solution in this chapter — one that provides the key components of a document imaging system: capture, index, import, search, and view. Document imaging enables an enterprise to reduce the vast amount of paper typically involved with most business processes, and greatly improves worker efficiency in terms of finding documents and using them within these processes. The solution provided in this chapter demonstrates how an organization can leverage SharePoint as a document imaging platform to reduce the overhead associated with storing and using paper in its daily business functions.

11

Electronic Forms with InfoPath

WHAT'S IN THIS CHAPTER?

- An overview of electronic forms, why they exist, and when to use them
- Working with Microsoft's forms
- Understanding of how InfoPath integrates with SharePoint
- Architecting and managing electronic forms solutions on the Microsoft platform

ELECTRONIC FORMS OVERVIEW

Forms have been an integral part of business processes since well before the ENIAC ushered in the era of general-purpose computing in 1946. Tax forms, job applications, surveys, and financial aid request forms are a few examples of forms you are likely familiar with, and you might have dealt with the paper versions of these many times in your life, perhaps too many times.

Most forms (paper or electronic) exist simply to drive a business process. A university admissions application, for example, provides all the relevant information the admissions staff needs to process your request for admittance, ultimately returning a yes or no answer. Therefore, generally, the form itself is a by-product of the process and likely has little or no use after that process is complete, except perhaps as a historical record.

Paper forms have been manually passed around offices for generations in the service of various business processes. Perhaps a paper purchase order form requires managerial approval for purchases exceeding \$500. Perhaps after Susie fills out the original request, it needs to be inter-office mailed to Bob for his signature. From there, it needs to be forwarded to someone in charge of purchasing if approved, or sent back to Susie if rejected. Unfortunately, because

we're dealing with people and paper, this process is prone to error. Forms are lost, steps are skipped, lives are lost (well, hopefully that last one doesn't happen). Sounds horrific, right?

Of course, computers excel at automating this kind of process, and the earliest computer programs have provided electronic forms for doing just that. Many technologies are used to gather information from users and store it in a usable fashion. There are the obvious general-purpose programming languages combined with a user interface, which can take the form of desktop or web applications. For full-featured applications this approach can make sense, but it can be overkill for a simple forms solution.

Other technologies used for forms include PDF or simple word processors such as Microsoft Word. Tools in this category have the attribute of carrying the user data along with the form definition itself, which can be nice for portability purposes. However, these tools aren't necessarily focused 100% on electronic forms processing, so they might not be adequate in many situations.

Then there are tools whose sole purpose in life is working with electronic forms. Tools in this category need to be able to support rich authoring of forms, including the capability to implement simple to complex business rules, handle moderately complex layout, as well as manage user submission. These tools must provide users with an intuitive method for filling in form data, as well as saving and submitting it. Finally, this data needs to be able to be easily retrieved and analyzed, which, again, is why most forms exist to begin with: to support a business process. InfoPath falls into this category.

Is It a Form or an Application?

There is sometimes a very fine line between a pure forms solution and a full-fledged application, making it difficult to choose the proper technology to solve a given problem. Making the correct determination during the architecture phase of a project can be critical because electronic forms software doesn't necessarily scale in the same way that a purely custom solution will — both in terms of maintainable deliverables as well as functionality itself.

In many cases this distinction can be obvious, especially when the problem at hand is directly replacing a manual business process that completely revolves around an existing and well-defined paper form or set of forms. In cases like these, an electronic forms + workflow solution might be the perfect fit all around. However, depending on the flexibility of an electronic forms package and the complexity of the requirements, more demanding projects might fall like a house of cards if the right technologies were not chosen up front.

Because of these points, it is crucial to evaluate up front whether you are building a forms solution or a full-bore application. As part of this evaluation, investigate the complexity of the business rules, user interface requirements, data sources (and destinations), reporting requirements, security requirements, and how you expect the solution to evolve over time. Obviously, each of these points needs to be measured against the actual capabilities of a given electronic forms software package. This chapter describes Microsoft's answer to electronic forms: InfoPath.

INFOPATH OVERVIEW

Microsoft InfoPath was introduced to the world in Office 2003. It promised to be the tool of choice for those who needed to capture and store information from users without having to resort to (gasp) writing code. Therefore, one of InfoPath's strongest suits is its design capabilities. For simple to

moderately complex forms, someone without any coding skills can define a form's data, layout, and rules in a fairly intuitive user interface. A form's definition can then be published — to SharePoint, for example — so that users are able to view and fill out forms in a special desktop application or even directly in a web browser. Figure 11-1 shows a view of the InfoPath Designer 2010 application, which is used to create form definitions.

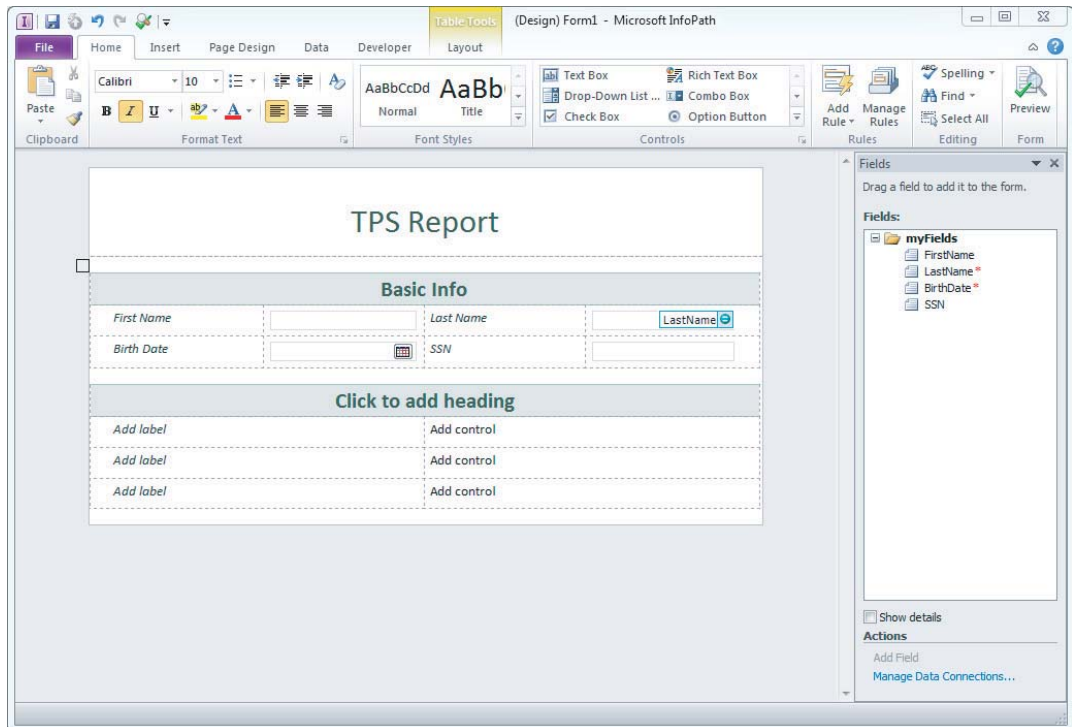


FIGURE 11-1

This screenshot reveals several important things about InfoPath. First, InfoPath Designer, like the rest of the Office suite 2010, sports the Ribbon (also known as the Fluent UI) for quick access to relevant features based on what you are currently doing in the application. Second, layout gets the spotlight in the main document area. Form layout in InfoPath is a breeze; anyone who can format a Word document can lay out an InfoPath form with ease. The Format Text section of the Ribbon contains familiar formatting features such as bold, italic, bullets, and so on. This example also contains a Table Tools Ribbon tab, which is currently highlighted because the cursor is inside a table. Again, all of this is very familiar to users who have used Microsoft Word or other Office applications.

The next aspect of this screenshot to note is the right-hand docked pane titled Fields. This is a very important pane when designing a form because it is here that the data elements and associated properties are defined. Each field has properties such as data type, default value, and whether or not a field is required or repeating. In addition, form fields can be grouped together. Groups are discussed later in the chapter. In this example, there are four fields, two of which are required. A field that is required is annotated with an asterisk in the Fields pane.

Finally, notice that there are four controls in the design area that correspond to the fields in the task pane. On the surface there is nothing too exciting here. However, by using simple drag-and-drop functionality, InfoPath Designer will try to help you when designing a user interface to populate these fields. The controls shown in the design area were created by simply dragging a field from the task pane onto the canvas. By default, controls that make the most sense for a given data type are used. The most obvious example here is the `BirthDate` field. Notice that the control for this field appears as a date picker because the `BirthDate` field was defined as a `date`.

These features just scratch the surface of InfoPath's capabilities, but they represent some very fundamental and important aspects of designing InfoPath forms. More features are showcased in a bit, but a quick look at some new features in InfoPath 2010 will help put InfoPath's capabilities in context.

What's New in 2010

One of the main changes in InfoPath 2010 is the introduction of InfoPath Filler. What used to be the InfoPath desktop application is now split into two applications: InfoPath Designer and InfoPath Filler. As you can probably guess, InfoPath Designer is the tool that enables you to create and manage form definitions, whereas InfoPath Filler is geared toward the end users who are consuming forms solutions. In addition, as mentioned earlier, the InfoPath desktop applications now take full advantage of the now ubiquitous Ribbon UI. For authoring forms, which can be a tedious and complex task, this change is more than welcome.

Another great new feature enables users to complete forms that have been deployed to a SharePoint forms library online or offline by using the new SharePoint Workspace application that is discussed in Chapter 5. Users who are mobile and disconnected can now fill out data in a form using Workspace. When network connectivity is available, a synchronization process pushes the data filled out locally to SharePoint.

InfoPath 2010 has also raised the bar in terms of the control types available out of the box. Some of the controls correspond to new features in SharePoint 2010, and some simply represent a maturation of the product. In the former category, the Managed Metadata picker and the Business Connectivity Services (BCS) picker enable you to interact with data in or surfaced in SharePoint. The BCS Picker is a boon for those integrating with preexisting line-of-business systems. Imagine trying to enable users to pick from a list of customers on a paper form! Other new or enhanced control types include a people picker control, a date/time picker, a picture button, a signature line control for digital signatures, and a hyperlink editor.

More InfoPath Fundamentals

InfoPath includes several major features that enable you to easily design and manage forms. In the following sections, you will learn about forms services, deploying forms, templates, rules, data connections, custom code, and publishing.

Forms Services

One of InfoPath's best features is the capability to publish form templates to SharePoint. InfoPath Forms Services was introduced in SharePoint 2007 as Forms Server, and it is new and improved in 2010. The biggest advantage of Forms Services is the fact that users can fill out forms in a browser without needing any special software installed or browser plug-ins. It's all HTML and client-side

script. In addition, deploying forms in Forms Services enables them to be centrally managed, which obviously allows for easier governance of data.

Aside from the document-driven view of InfoPath focused on in this chapter, InfoPath actually enables quite a few other nice features in SharePoint. While these features aren't necessarily related to forms from an ECM perspective, they are still worth mentioning here. For example, InfoPath forms can be used as custom list create and edit forms. This enables users to utilize a custom UI when editing SharePoint lists; doing something like this in previous versions of SharePoint was only possible using very technically oriented tasks. In addition, InfoPath forms can be created as task forms in a SharePoint workflow. This is true for both SharePoint Designer workflows as well as workflows developed in Visual Studio.

Deploying Forms

InfoPath offers several options for publishing, or deploying, completed form templates for end-user consumption. Figure 11-2 shows the Office Backstage view when publishing from within InfoPath Designer.

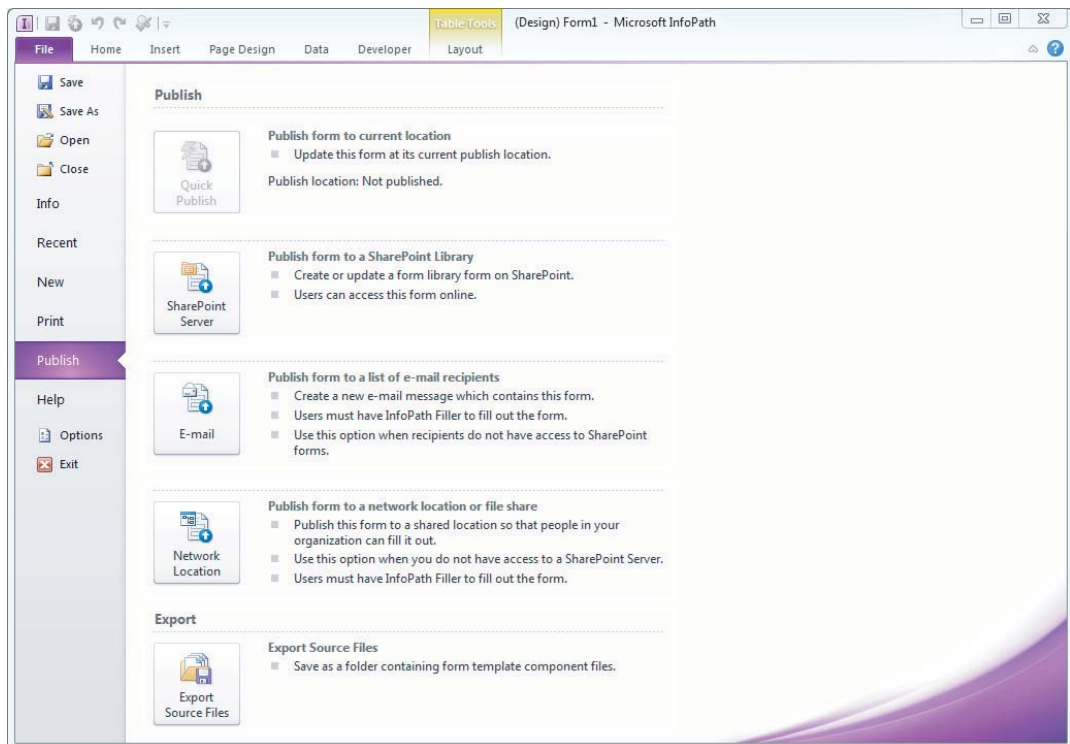


FIGURE 11-2

Note that InfoPath offers three major options for publishing:

- SharePoint form libraries
- E-mail
- Shared network locations

In organizations where SharePoint is deployed, publishing to SharePoint is in all likelihood the most appropriate option. That's because SharePoint includes a feature called Forms Services that enables InfoPath forms to be rendered and filled out within the browser — no browser add-ons or plug-ins are required. In addition, forms filled out by users automatically have a storage location in a Forms Library, and security is managed by the robust SharePoint infrastructure. Forms Services is discussed in more detail later in this chapter.

If deploying to SharePoint is not an option, then a form can be deployed to users by using either e-mail or a shared network location. Both of these options can be a nice fallback, but both also require the InfoPath Filler 2010 product to be installed on a user's machine. As shown by its prominence in Figure 11-2, Microsoft assumes users will utilize SharePoint as the deployment vehicle by default, and only use the other mechanisms if SharePoint is not available. This was not the case in earlier versions of InfoPath, which were released when SharePoint was not nearly the success that it is today.

Templates and Form Data

InfoPath is based on the life cycle of “design and then publish.” Form templates have an extension of .xsn and essentially represent a package of other files. If necessary, the source files can actually be extracted from the XSN file by using the same Publish section of the Office Backstage view shown previously in Figure 11-2. The types of files included in an extracted XSN are as follows:

- One or more XSD files that define the data to be represented in a form
- XSLT, which is used to generate the form layout
- Sample data XML
- Another XML file containing important metadata about the form. This is the `manifest.xsf` file that is registered with Windows as a “Microsoft InfoPath Form Definition File.”
- If applicable, script files and/or .NET assemblies containing custom code for the form
- Other resources such as images, HTML, etc.

Figure 11-3 shows a list of the exported files from a very simple form. It's fairly obvious already that InfoPath is very much based around XML. This fact can come in handy when inspecting or modifying form data programmatically; you'll learn more about that later in this chapter.






| Name | Date modified | Type | Size |
|--|------------------|---|-------|
|  manifest.xsf | 5/9/2011 7:56 AM | Microsoft InfoPath Form Definition File | 5 KB |
|  sampledata.xml | 5/9/2011 7:52 AM | XML Document | 1 KB |
|  template.xml | 5/9/2011 7:56 AM | XML Document | 1 KB |
|  myschema.xsd | 5/9/2011 7:52 AM | XML Schema File | 2 KB |
|  view1.xsl | 5/9/2011 7:52 AM | XSLT Stylesheet | 22 KB |

FIGURE 11-3

The preceding list only outlines what is contained within a form template; the following list describes the various aspects of a form that are defined by a form template:

- The data to be contained within the form. This includes the fields gathered from users, and data that might be collected from external systems such as web services or BCS.

- The layout of the form, or the user interface. This can include more than one view. Views are different representations of the data included within the form. Different views may display the same data in a different way or distinct data from other views shown together. Note that views are only for representing data to users in various ways; they are not a security mechanism. You should assume that anyone who can access a form template will be able to access all the data in that form.
- The behavior and logic of a form. This includes business rules like “set field A to some value when field X is entered.” This also includes default field values and field validation. All the items in this category are defined via the UI without using a lick of code. Rules are discussed in more detail in the next section.
- Following from the previous bulleted item, a form template can also define custom code that can be used to inspect and manipulate form data. This code can be written in C# or Visual Basic .NET and developed in a first-class development environment. Custom code in forms is also discussed in more detail later in this chapter.
- Form templates also define how a form is submitted. As mentioned earlier, InfoPath forms can be displayed and stored in SharePoint form libraries. In addition, forms can be submitted to databases, web services, or via e-mail. Aside from submitting, users can save a form to an XML file on their local computer prior to submitting. There are several differences between saving and submitting, which are discussed later in this chapter.

The preceding items describe the makeup of an InfoPath form template. After a user fills out a form, the data is saved to an XML format. The following code shows a slightly abbreviated version of the XML that is generated after saving the simple form shown earlier in Figure 11-1:

```
<?xml version="1.0" encoding="UTF-8"?>
<?mso-infoPathSolution solutionVersion="1.0.0.7"
  productVersion="14.0.0"
  PIVersion="1.0.0.0" href="{path to manifest.xsf}" ?>
<?mso-application progid="InfoPath.Document"
  versionProgid="InfoPath.Document.3"?>

<my:myFields xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:my="{namespace of this form}"
  xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
  xml:lang="en-us">
  <my:FirstName>Todd</my:FirstName>
  <my:LastName>Kitta</my:LastName>
  <my:BirthDate>1900-01-01</my:BirthDate>
  <my:SSN>111-11-1111</my:SSN>
</my:myFields>
```

As you can see, the preceding XML only represents data; it does not indicate what the form it came from looked like or how it behaved. There is a pointer back to the form’s originating template in the `xmlns:my` namespace attribute of the `myFields` node. Several other instructions that assist in linking back to the source template are included in the `?mso-*` nodes at the top of the XML. Because InfoPath is based so extensively on XML, forms solutions built on InfoPath are very portable, transparent, and extensible.

Rules

Rules are at the heart of what makes InfoPath forms so powerful, especially considering rules do not require custom coding. Microsoft defines three main categories into which rules fall:

- Validation rules
- Formatting rules
- Action rules

The following sections discuss each of these rule types. While each type has its own distinct purpose and capabilities, they do share some common characteristics. First and foremost, each rule type has a name and a condition. The name is simply for identification purposes during the designing of a form template; a condition dictates whether or not a rule is applied to a form.

Conditions are simply a set of statements that evaluate to true or false as a whole. They work just like an `if` statement in C#. Figure 11-4 shows a simple condition that ensures that a field called `DesiredDeliveryDate` is not today or in the past. Note that the value on the right, the one that is being compared to `DesiredDeliveryDate`, is a date function. InfoPath supports many different types of functions dealing with dates, text, numbers, and so on.

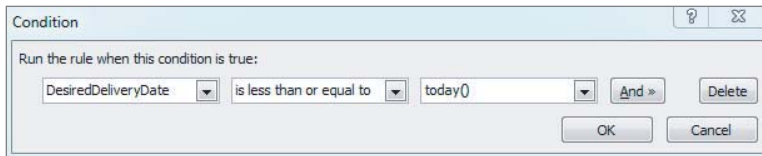


FIGURE 11-4

Validation Rules

Validation rules define the logic that specifies whether the values in a form are allowed or not. Conditions are used to define the rules that indicate invalid values. Feedback is provided to the user in the form of visually altered controls, and may optionally include a dialog with an error message.



Dialogs can only be displayed when filling out a form in the Microsoft InfoPath Filler 2010 application. Dialogs do not apply to forms rendered in Forms Services.

Figure 11-5 shows a form with a date field containing an invalid value.

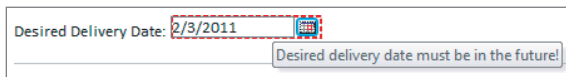


FIGURE 11-5

There are several different ways to add validation and other types of rules. First, right-clicking on a control and selecting Rules ⇨ Manage Rules from the context menu will open the Rules task pane for the selected control. This task pane can also be accessed via the Ribbon by clicking the Manage Rules button in the Home Ribbon tab. For this task pane to be useful, a control must be selected. Finally, the Home Ribbon tab also has an Add Rule button that offers some great shortcuts for creating all kinds of rules.

When a control is selected and this menu button is clicked, a context-specific list of condition types is displayed. Each condition type has another submenu that contains a list of actions that can be performed if the condition is met. Figure 11-6 shows this clearly organized menu structure. Note that the available conditions are specific to dates (e.g., “is in the future”; “is in the past”). Using this menu system, you can easily create the condition that was created earlier manually (refer to Figure 11-4) with a few clicks and no typing! Another nice feature of this menu is that you don’t need to think about whether you are adding a validation rule or a formatting rule, etc. Rather, the rule type is inferred based on the options selected.

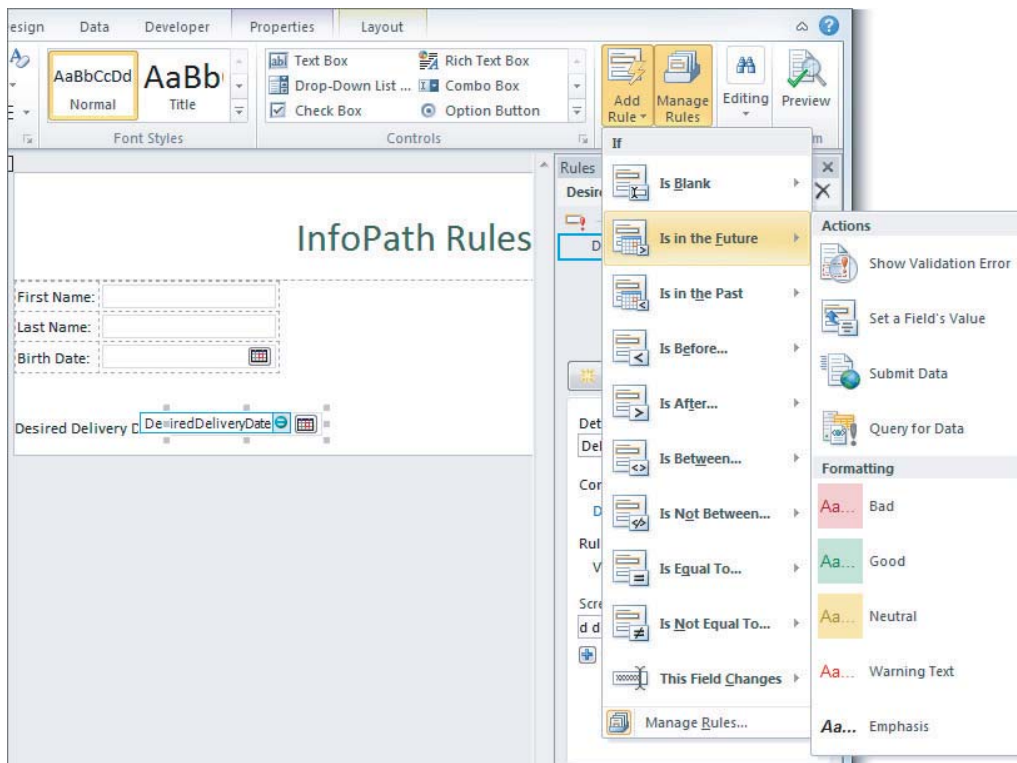


FIGURE 11-6

Figure 11-7 shows the outcome of the rule that was added in Figure 11-6. Note that the configuration shown is exactly what was generated by InfoPath, and no manual manipulation was performed. Of course, if the generated rule does not meet the exact validation requirements at hand, it can be easily modified by hand.

Note that the information in this section about adding and managing rules is not specific to validation rules. Adding and manipulating rules is, of course, essential behavior for all types of rules.

Formatting Rules

Formatting rules enable the form designer to alter the form's look and feel, as well as layout, based on conditions. This can be used for obvious things like making a textbox red if a numeric value is less than a certain value, or highlighting rows in a table that represent items requiring a user's attention. These types of rules allow manipulation of text formatting, as well as the color of objects.

In addition to changing text format or color, controls can be hidden or disabled if a given condition is met. This type of formatting rule can be very useful to display only information that is relevant to what a user is currently doing. For example, perhaps an entire section of the form should be hidden if a checkbox is unchecked. Figure 11-8 shows an example of the Rules task pane during the editing of a formatting rule.

In this example a rule called “HideUrgentDescription” hides the selected control if the `IsUrgent` field is false. At runtime, InfoPath takes care of monitoring when this field's value is changed, and the rule is reevaluated. Figure 11-9 shows this form in action. Notice that the Urgent Request checkbox is enabled, and the textbox below it is visible. Also notice that the textbox in question has some text above it, which would only apply if the textbox itself were also visible. Showing and hiding multiple elements like this requires that the rule be created on a container, rather than a singular control like a textbox. Therefore, to achieve behavior like this it is sometimes necessary to alter the form's schema. Figure 11-10 shows this form's configured fields and the fact that the `UrgentDescription` field is in a group by itself. The field is grouped like this only so that it could be added within a group in the designer.

Action Rules

Action rules specify behavior that can occur if a certain condition is met. Table 11-1 describes the actions that are available in InfoPath 2010, and their compatibility across the web and InfoPath Filler.

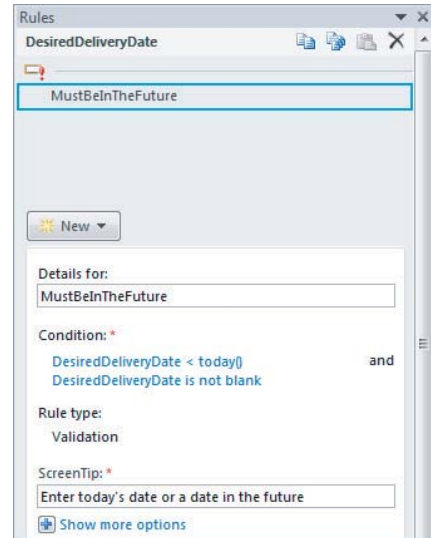


FIGURE 11-7

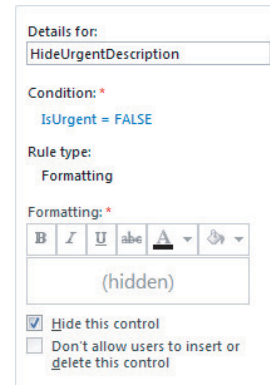


FIGURE 11-8

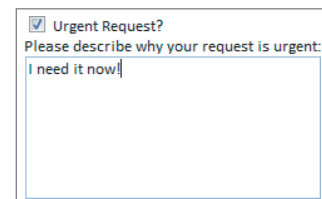


FIGURE 11-9

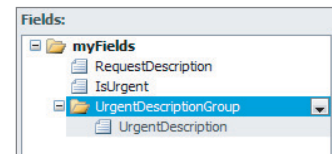


FIGURE 11-10

TABLE 11-1: Actions in Action Rules

| ACTION NAME | DESCRIPTION | WEB OR FILLER COMPATIBILITY |
|--------------------------------------|---|-----------------------------|
| Show a message | Shows a message of hardcoded text in a dialog | Filler only |
| Show the value of a field or formula | Similar to “show a message” except that the dialog shows an evaluated expression | Filler only |
| Switch views | Sets the current view to the one specified. This is only usable as a rule on a button or on form load. | Both |
| Set a field’s value | Sets a form field’s value to the value of an evaluated expression | Both |
| Query for data | Uses a form’s data source to query for data, perhaps from an external system. Can be useful for populating fields based on some key value. | Both |
| Submit data | Submits the current form to the configured destination | Both |
| Open a new form to fill out | Opens another form to fill out. Can be useful if certain information is required from a user, given a value in the current form | Filler only |
| Close the form | Closes the current form | Both |
| Sign signature line | Opens the configured signature line control to obtain a digital signature from the user | Filler only |
| Send data to web part | Sends the fields that have been promoted as Web Part connection parameters to any connected Web Parts. Obviously, this action only works in web forms as indicated. | Web only |

The interesting thing about action rules is that they can be evaluated based on several different events. In addition to these rules being kicked off upon a control’s value changing, action rules can also be executed when a button is clicked or even when the form loads. These additional methods of activating action rules makes them especially versatile when adding capabilities to InfoPath forms — again, without writing any code.

The Rule Inspector

While the declarative rules system in InfoPath is extremely easy to use and arguably powerful, all the logic in a form is isolated in each control or in the form load process. In addition to that, a form’s logic consists of default values and code. InfoPath provides an extremely useful overview of all this information in one handy screen called the Rule Inspector. Figure 11-11 shows an example that has several rules configured. Without a comprehensive list of form logic in a UI like this, it would be very difficult to get a big picture view of what is happening in a particular form.



FIGURE 11-11

External Data

InfoPath isn't limited to dealing with data that is self-contained within a single form. There are entities referred to as *data connections* that allow other data to flow into and out of an InfoPath form. By mashing up form data, SharePoint data, and perhaps data from other line-of-business systems, forms become inherently more useful to an organization.

The data connections that allow data into a form are referred to as *receive* data connections. Out of the box, the following receive data connections are available:

- Web services (SOAP or REST)
- SharePoint lists
- Microsoft SQL Server
- XML files

These data connections are often used to allow users to choose a value or values from a predefined and governed list of data. For example, in a system that processes purchase orders, a user might need to select a vendor from a dropdown list. Rather than simply type in the vendor name, which could lead to typos or using unapproved vendors, a call to a SOAP web service could be made, which would return a list of company-approved vendors. SharePoint lists also enable form data to be maintained in a central location, where it can be managed with the built-in SharePoint security mechanisms.

When adding a receive data connection to a form, a secondary data source is added to the form's list of data sources. The primary data source generally represents your form's local data. You will need to keep this in mind when interacting with data retrieved externally to ensure that the correct data source is selected as appropriate. This can be a little confusing when you are first starting to work with InfoPath.

Data that is retrieved from an external data source can optionally be cached within a form, which enables this data to be used in disconnected scenarios. For example, perhaps a user is frequently traveling and using SharePoint Workspace to complete InfoPath forms when a connection is not available. Using this feature enables the user to work with business data when offline.

Similarly, there is a category of data connections called *submit* data connections. These connections are able to send data from the form to the following destinations:

- Web services
- SharePoint libraries
- E-mail
- A hosting application (e.g., a web application hosting the InfoPath control)

As previous alluded to, InfoPath natively allows the saving of forms, in addition to submitting. In any “serious” forms solution, however, submitting using the preceding options is probably the best approach. Following are a few differences between saving and submitting:

- Submission enables form data to be sent to a variety of destinations, including SharePoint forms libraries, e-mail, and external data systems. Saving only enables you to save an XML file to the file system.
- Saving does not enforce validation, meaning users can save forms that are not yet complete. This is an advantage if filling out a form might take a long time and a user might want to save a draft.
- When using submission as opposed to saving, a form is made aware of this fact upon submission and can perform certain behaviors as a result. For example, custom code can handle the submission event and do basically whatever needs to be done, or the form can be closed or a new form opened. These options are simply configured via the form's submit options, as shown in Figure 11-12. None of these capabilities are available for forms that are simply saved.

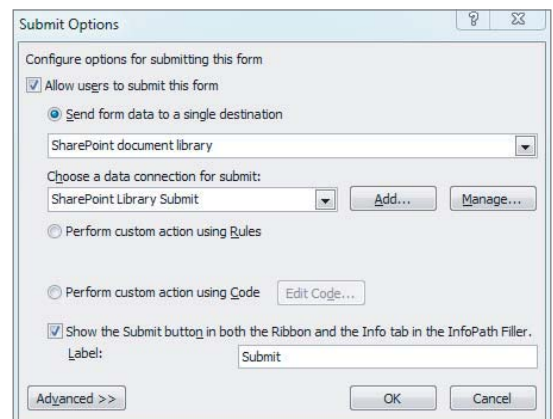


FIGURE 11-12

Custom Code

As great as InfoPath Designer might be, with its rules, calculations, layout, and so on, nothing is more flexible than good old-fashioned code. Therefore, InfoPath allows developers to include .NET code in form templates to augment the out-of-the-box features. You need to do two things before code can be written in an InfoPath form:

- Install Visual Studio Tools for Applications (VSTA). You can find this in the Office 2010 installation under the InfoPath node.
- Choose a language. By default, if a project is created in a form template, Visual Basic .NET will be chosen for the language. You can choose a different language in the form options (in the Backstage view).

To actually access the development environment, click the Code Editor button on the Developer Ribbon tab in InfoPath Designer. If a code project has not yet been created for this template, a new one will be created at this point in the default language.

Because InfoPath is so heavily based on XML, in order to affect any form data at all, programmatic XML manipulation must take place. To get an idea of what InfoPath code might look like, take a look at the following code. Figure 11-13 shows the form's schema. Notice that the "Line" group is repeating.

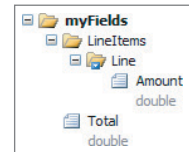


FIGURE 11-13



Available for
download on
Wrox.com

```
// NOTE: The following procedure is required by Microsoft
// InfoPath.
// It can be modified using Microsoft InfoPath.
public void InternalStartup()
{
    ((ButtonEvent)EventManager.ControlEvents["CTRL3_5"]).Clicked +=
        new ClickedEventHandler(CTRL3_5_Clicked);
}

public void CTRL3_5_Clicked(object sender, ClickedEventArgs e)
{
    // select all of the line item nodes
    XPathNodeIterator nodes =
        this.MainDataSource.CreateNavigator().Select(
            "/my:myFields/my:LineItems/my:Line/my:Amount",
            this.NamespaceManager);

    // loop through each line item value and keep track of the sum
    decimal total = 0;
    foreach (XPathNavigator n in nodes)
    {
        total += Convert.ToDecimal(n.Value);
    }
}
```

```

// find the total node
XPathNavigator totalNav =
    this.MainDataSource.CreateNavigator().SelectSingleNode(
        "/my:myFields/my:Total", this.NamespaceManager);

// delete the nil attribute if it exists
if (totalNav.MoveToAttribute("nil",
    "http://www.w3.org/2001/XMLSchema-instance"))
    totalNav.DeleteSelf();

// set the total node's value
totalNav.SetValue(total.ToString());
}

```

Code snippet FormCode.cs

This code is simply looping through each line item and summing the values into a local variable. It then takes the variable's value and sets it to the "Total" field. However, notice the funky code that is deleting a `nil` attribute prior to setting the total value. This is very common behavior in InfoPath development because fields of certain data types will have a `nil` attribute whenever no value is present. If you actually want to set the field's value, it is not `nil`. Therefore, this attribute needs to programmatically be removed before setting the value. Yes, it's kind of a pain, but it's something that comes with the territory when dealing with InfoPath XML.

Publishing

Earlier, a distinction was made between submitting a form and saving a form. There is a similar distinction when considering form templates that is often a point of confusion for new InfoPath developers. When you create a new template in InfoPath Designer and save it to your local machine, you can think of this as saving a draft — like you would save an InfoPath form, rather than submit it. Not until you *publish* a form template is it truly deployed and ready for consumption by users.

Publishing a form is actually quite easy. The following options are available for publishing from within InfoPath Designer:

- SharePoint form library
- SharePoint site content type
- E-mail (requires InfoPath Filler)
- Network location (requires InfoPath Filler)

Figure 11-14 shows the publishing options available from the Backstage view of InfoPath Designer. Notice the Quick Publish option at the top, which simply refreshes the form to the previously published location. Obviously, this option is only available after an initial publish has taken place.

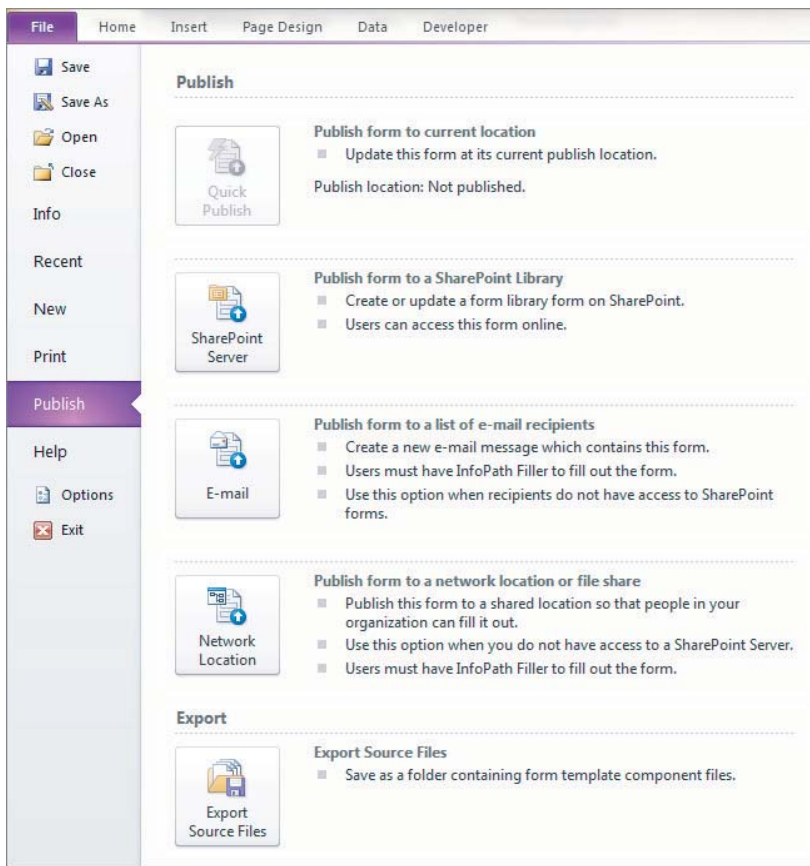


FIGURE 11-14

DETERMINING A FORMS STRATEGY

Commonly, projects involving InfoPath are those replacing manual, paper-based processes. These types of projects are generally the low-hanging fruit of forms-based projects. This section provides some tips and guidance that can be useful when pursuing this type of work.

One of the most common pitfalls in approaching a project of this type is taking for granted the current business process. More often than not, manual business processes are not well documented and only suffice because certain members of the organization have been doing them for so long. It cannot be stressed enough that a thorough review of a business process should be done and reviewed with all stakeholders of a project. Never assume that the process is “easy” or “well known.” Keep in mind that computers cannot handle undefined areas of a process; they can only do what you explicitly tell them.

The previous point is especially important with regard to exceptions to the process. This is another area which is commonly overlooked in these types of projects. Following are some examples:

- What should happen when the normal approver is out on vacation for six weeks and a decision is needed immediately?
- How would a high-ranking employee override the current process and move things to a more advanced point in the process?
- What happens when an invoice is received with no matching purchase order?

Exceptions such as these are common to most business processes, and they should be considered from the outset of a project.

Another suggestion when pursuing a forms project is to take a fresh look at the form (and its data) as well as the process in light of the technology that will be used for automation. When you are using InfoPath Forms Services along with SharePoint workflow to automate a process that is currently completely manual, you have many candidates for optimization in terms of both the forms and the workflow.

Paper-based forms were designed to look good and be efficient on paper. It might sound obvious, but trying to create an exact copy of a paper form in InfoPath can be tedious, unnecessary, and perhaps even a hindrance for the form's users. Taking a fresh look at both the data needed to facilitate a business process and how users will interact with a form can greatly increase efficiency and user satisfaction.

This same advice to “take a fresh look” also applies to the business process. Just because something is done “the way it has always been done” doesn't mean it should be done like that, especially in the context of SharePoint. Doing things the SharePoint way can create new ways to interact with processes and drive new efficiencies.

InfoPath forms are editable documents. While a business process is “in flight,” this is typically appropriate. However, certain business processes might dictate that once a process has completed its life cycle, the form itself should become read-only, and an official record of some kind. This aspect of forms should be carefully considered in scenarios that have legal requirements related to storage and archiving. One approach is to move the form to a different site or library where permissions are set appropriately, or even to use item-level permissions to ensure that a completed form cannot be changed. Another possibility might be to render the form into a natively read-only document format such as PDF. This enables form data portability while meeting the read-only requirement. However, and unfortunately, SharePoint and InfoPath do not offer such functionality out of the box; but there are third-party libraries available that can assist in performing this InfoPath to PDF conversion.

The preceding paragraphs describe form considerations from a process or business perspective. When dealing with InfoPath, as we are, important technological points must also be considered, including integration with other systems, degree of form customization during development, and planning for the future.

When planning InfoPath projects, a common consideration is integration with other line-of-business systems. As mentioned previously, InfoPath has the native ability to connect with SQL Server or web services. If it makes sense to integrate with an external system in a forms solution, the programmatic interface must be well defined and understood. Authentication is an important consideration here, especially if forms will be filled out in a web browser, rather than InfoPath Filler. For example, will users filling out the form need to access an external system as themselves or a specific user

account? (The SharePoint Secure Store is an option for storing credentials, but an in-depth discussion of that feature is beyond the scope of this chapter.) Forms can make use of data connection files (.udcx), which can be stored in data connection libraries in SharePoint. UDCX files define various data sources, where they live, and how to connect to them. Using data connection files can be a better alternative to storing hardcoded information about a data connection in the form itself.

Another important consideration when creating a forms solution in InfoPath is whether or not to leverage custom code. While using .NET code can create a wide array of new possibilities with regard to customization, it also increases the complexity of any solution. InfoPath can accomplish quite a bit with the standard, UI-driven rules engine, discussed earlier in this chapter, and that should always be a form developer's first line of defense when designing logic. That said, using custom code in InfoPath is not necessarily a bad thing. Keep in mind, however, that because InfoPath is XML-based, something as simple as setting field values in a form will be done by manipulating an in-memory representation of the form's data. Deciding to use custom code should also take your versioning and source control strategy to the next level, as code assets need some TLC in order to prevent problems down the road.

CREATING A CUSTOM FORM: AN EXERCISE

In this exercise, we will consider a solution that enables a form user to create an online ad listing. This type of solution might be useful in a company that wants to allow employees to list items or services for sale or for free. Because this solution should be available to a wide audience, we will create a browser-compatible form.

Form Data and Layout

Start out by launching InfoPath Designer 2010 and selecting SharePoint Form Library from the New section of the Backstage view. Selecting this as a starting point will provide us with a form template that is web browser-compatible out of the gate.

Before defining the fields that this form will capture, use the Page Design Ribbon to apply a layout to the template that is a little more aesthetically pleasing than the default layout. Start by selecting all the default content and deleting it. Next, select one of the predefined layouts in the Page Layout Templates menu button. This example will use the Color Bar layout and the SharePoint - Standard theme, which can be chosen from the Themes selection box, also found in the Page Design Ribbon tab.

The next, and very important, step is to define the data form. Use Table 11-2 as a guide to adding fields to the form.

TABLE 11-2: Classified Ad Form Fields

| FIELD NAME | TYPE/CONFIGURATION | PARENT |
|---------------|--------------------|---------------|
| ListingTitle | String field | Root |
| ListerDetails | Group | Root |
| ListerName | String field | ListerDetails |
| ListerEmail | String field | ListerDetails |

| FIELD NAME | TYPE/CONFIGURATION | PARENT |
|-------------------------|------------------------------|---------------|
| ListerPhoneNumber | String field | ListerDetails |
| ItemDetails | Group | Root |
| DetailedItemDescription | XHTML field | ItemDetails |
| Attachments | Group | ItemDetails |
| Attachment | Repeating base64binary field | Attachments |
| HasCost | Boolean field | ItemDetails |
| AssociatedCode | Double field | ItemDetails |
| ListingStartDate | Date | Root |
| ListingEndDate | Date | Root |

After defining the form’s data, the next step is to build the layout. Experiment with the different layout options, including layout tables. Layout tables can be made very easily by right-click dragging a group from the Fields task pane to the design surface. Once you release the right mouse button, a context menu will appear, and one of the options is to add controls in a layout table. After laying out all the fields on the design surface, your form should look something like Figure 11-15.

The screenshot shows a web form titled "Classified Ad Listing Form". It features several input fields and sections:

- A text input field for "Enter a catchy title for your listing:".
- A group of three input fields for "Lister's Name:", "Lister's Email:", and "Lister's Phone Number:".
- A large text area for "Detailed Item Description:".
- A section titled "Add photos or other attachments which would be helpful for other viewers" containing a button "Click here to insert a picture".
- A dropdown menu with options "Repeating Table" and "Optional Section".
- A checkbox labeled "Does your listing have an associated cost?" followed by a text input field.
- Two date pickers for "Listing Start Date:" and "Listing End Date:".
- A "Submit" button at the bottom.

FIGURE 11-15

Form Rules

A few areas of the form require validation and other rules. Three main rules stand out as necessary. First, make sure the lister's e-mail textbox is a valid e-mail address. Use the Add Rule button from the Home Ribbon tab to select Is Not an E-mail Address → Show Validation Error.

Second, the “has cost” checkbox should drive the visibility of the cost textbox itself. In addition, the cost textbox should be required if there is an associated cost. Figure 11-16 shows what the configured rule should look like in the Rules task pane. Note that this rule should be configured on the cost textbox itself.

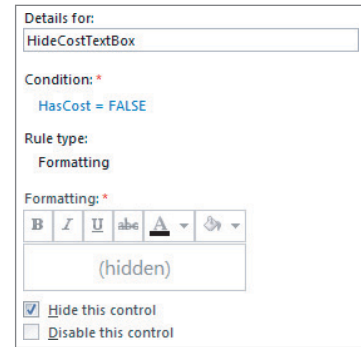


FIGURE 11-16

The final rules this form needs are related to ensuring that the start and end dates are valid. Neither of these fields should be required, and the user can enter one or both dates. However, the form should ensure that if both fields are entered, then the end date is later than the beginning date. To enforce this logic, only one rule needs to be added to the end date field. However, you should also add a rule to each field to ensure that if there is a value, it is not in the past. The “not in the past” rules are easily added via the same Ribbon button that was used to add the e-mail validation earlier. You may even want to tweak the auto-created rule so that users cannot add dates that are less than or equal to the current date. Because this rule is exactly the same for both the start and end date, it can be copied from the first control to the second control with the “copy rule” feature in the Rules pane. When the rule is pasted on to the second control, InfoPath actually alters the rule so that it is applicable to the control onto which it was just pasted. Therefore, in this case, the pasted rule needs no further modification.

The last date rule should simply ensure that the end date is later than the start date if the start date has a value. Figure 11-17 indicates what this rule looks like in the Condition dialog.

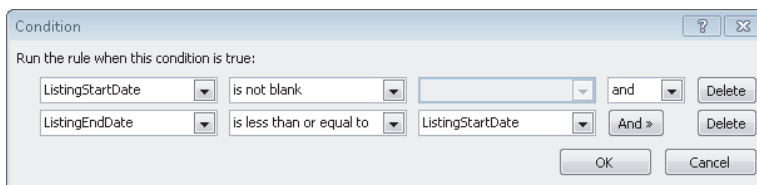


FIGURE 11-17

Form Submission

Next we are going to configure the form's submission to a SharePoint forms library; this is a multi-step process. First, you need to create a forms library to which forms can be submitted. Navigate to the site in which you will be submitting your form and select Site Actions → More Options. In the resulting dialog, type **form** in the search textbox at the top right; this will display the Form Library item in the main area. Finally, give the new library a name like “Classified Ads” and click the Create button. Before hopping back over to InfoPath, copy the URL of the newly created library to your clipboard.

The next step to configuring form submission within the InfoPath form template is to add the connection to the form library just created. This can be done from the Data Ribbon tab by clicking the

Data Connections button. From the resulting dialog, click the Add button to display the dialog shown in Figure 11-18.

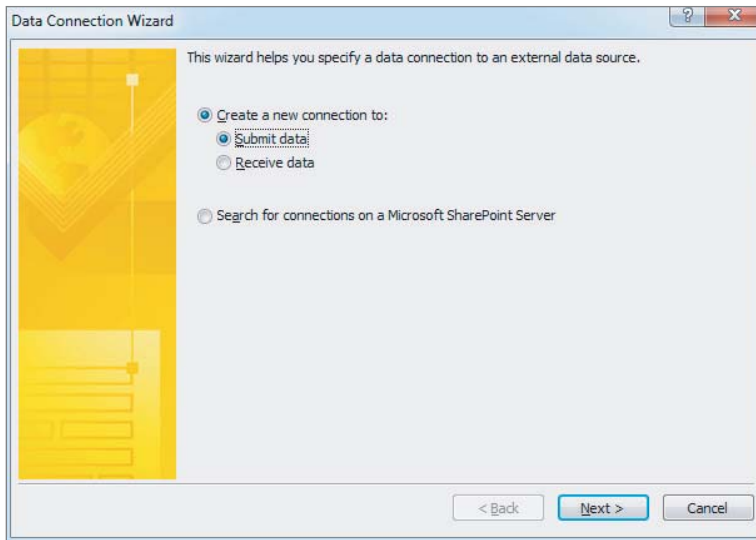


FIGURE 11-18

In this case, we need a data connection that submits data; select this option and click Next. On the next screen, select the option to submit to a SharePoint library and click Next. The subsequent screen requires the URL to the form library that was created earlier, as well as an expression that dictates the filename that ultimately ends up in the form library. This expression can be a hardcoded string, but a better choice is a more dynamic formula based on form data. In this example, simply name the form the same as the `ListingTitle` field. After configuring these two fields, click Next and wait for InfoPath to communicate with SharePoint. When the communication has completed, a summary screen is displayed and you can enter a custom name for this data connection. The default name should work fine; click Finish to add the connection.

Next, we want to ensure that only the appropriate controls are available once the form is rendered in the web browser. To do this, navigate to File ⇨ Form Options ⇨ Web Browser. The dialog shown in Figure 11-19 will appear.

In the next and final step of configuring submission, we will use a button control to submit the form. Therefore, we certainly do not want to display the save-related functions; nor do we even need to show the toolbar at all. Hide the toolbar by unchecking the “show InfoPath commands...” checkbox. This will constrain the form such that only the button we are about to add can cause a submission to occur.

Finally, add the button to the bottom of the form. Right-click on the button to edit its properties. Give it a meaningful label, select Submit from the action dropdown, and then click the Submit Options button. The resulting dialog will look like the one shown in Figure 11-20. Configure the options to match those shown in the figure. Notice that the data connection created earlier is selected in the second dropdown box. Once you click the OK button, submission has been fully configured for this form.

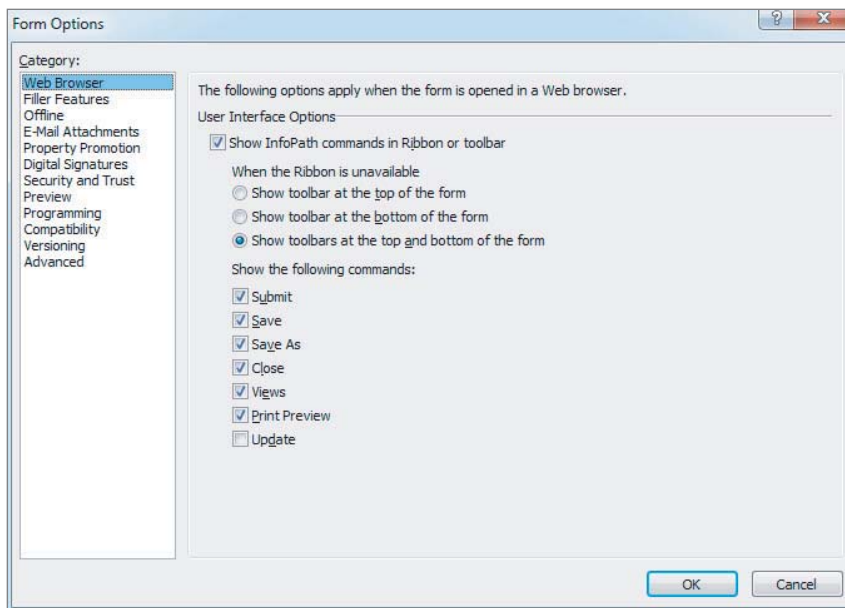


FIGURE 11-19

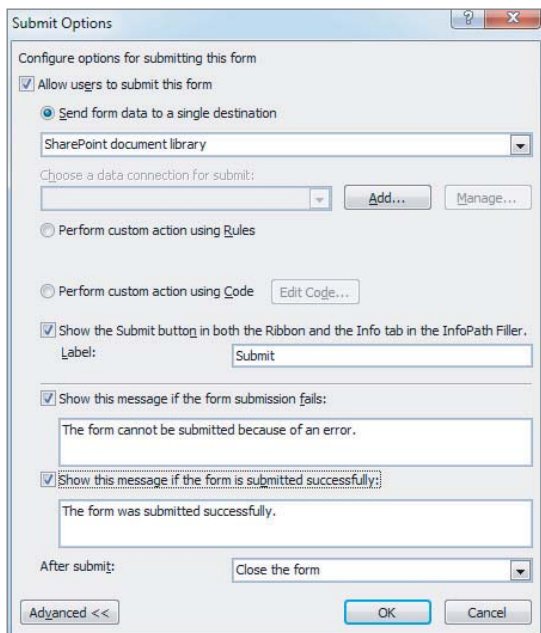


FIGURE 11-20

Publishing the Form

At this point, we are just about ready to publish the form to SharePoint so that users can consume it via their browsers. There is just one more task that needs to be completed before doing so. A nice feature of InfoPath forms that are submitted to SharePoint libraries is that the form's metadata can be exposed to the library UI, which allows users to browse the internal form data without actually opening the form. This also allows for sorting and filtering. This process is called *property promotion*. The property promotion settings can be accessed from the Form Options dialog in the Backstage view, as shown in Figure 11-21.

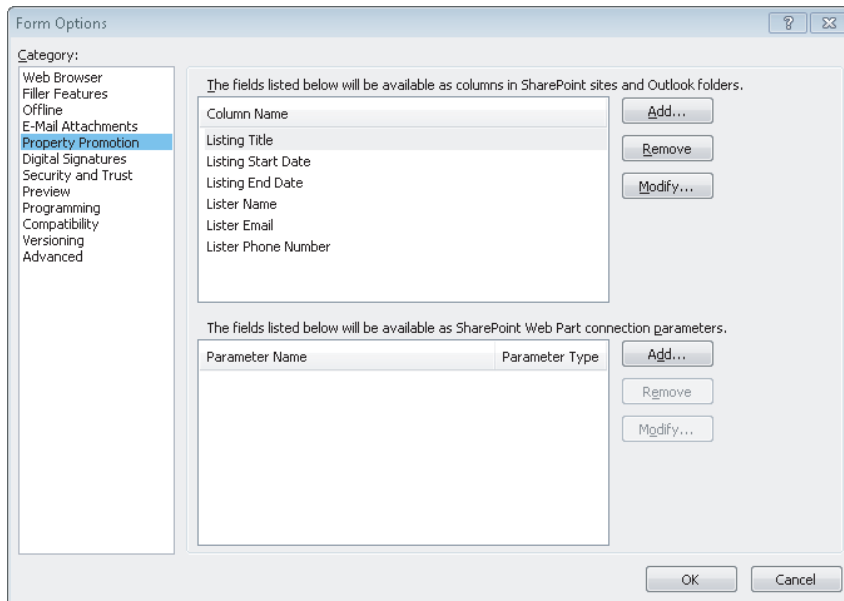


FIGURE 11-21

By promoting properties and deploying to SharePoint, not only can users browse, filter, and sort these fields, they can access the field values via the SharePoint object model just as if they were standard column values.

Before deploying, take a look at Figure 11-22, which shows the final form design, to ensure that everything is in its place. In addition, if you refer back to Figure 11-11, you will see this form's Rule Inspector so that you can compare your rules against the ones that were configured in this exercise.

To publish the form, use the Backstage view and go to Publish ⇄ SharePoint Server; this will launch the publishing wizard. The wizard is fairly straightforward and requires some basic information such as the library you are publishing to, the type of deployment (choose Form Library over Site Content Type for this example), and then the form library, which was created in a previous step. The wizard also shows you the properties that were configured for promotion. When you complete the

wizard and click the Publish button, InfoPath will push the form template to SharePoint, and it is now ready for use!

FIGURE 11-22

SUMMARY

Electronic forms play an important role in enterprise content management, as organizations typically hold a good deal of crucial data in forms that users have generated. InfoPath is the tool of choice when an electronic forms solution is needed on the Microsoft platform. SharePoint is a natural partner for InfoPath because of SharePoint's storage, security, and user interface mechanisms. In addition, SharePoint is able to render forms in the browser, so users don't need to have special software installed.

InfoPath provides many capabilities for form designers to create rich and complex forms, including integration with external systems, no-code rules, advanced layout, many options for publish, and even .NET development capabilities if needed.

It is important to consider all the requirements of a data capture solution when evaluating an electronic forms package like InfoPath as a potential answer. InfoPath is a tool that has many strengths and many weaknesses; it needs to be applied appropriately to a given problem and these strengths and weaknesses need to be taken into consideration up front.

12

Scalable ECM Architecture

WHAT'S IN THIS CHAPTER?

- Building a storage architecture
- Scaling the topology
- Architecting a scalable taxonomy
- Tuning SharePoint performance

As the Microsoft SharePoint platform has grown in functionality and popularity in recent years, organizations are often caught off guard when they discover that a rogue SharePoint installation, running on a workstation-class computer under the desk of a random employee, has become a mission-critical system. While this is a testament to the usability of SharePoint, an implementation running in this environment has little chance of performing reliably for any reasonable length of time.

This chapter provides guidance for architecting a SharePoint farm that will meet an enterprise's needs for years to come. Architectural considerations are presented from the ground up, starting with storage. Then farm topology and taxonomy concepts are related to various sizing requirements. Finally, techniques for managing and tuning farm resources are presented.

STORAGE ARCHITECTURE, THE KEY TO PERFORMANCE

A poorly performing database can undermine the performance of any application. Microsoft has gone to great lengths to optimize database architecture for best possible performance while maintaining the necessary flexibility to support the feature-rich SharePoint platform. However, even the best possible database design can be undermined by poor disk performance.

SharePoint is unique in that it serves a broad range of use cases. This single platform can function as a relatively static content management system with modest storage requirements, a collaboration system with significant storage requirements, or even a large-scale document repository with extreme storage requirements, all in the same farm implementation! It only takes one of these functional requirements to get the SharePoint movement started in an organization. Problems can arise, however, when storage requirements are not properly planned for and the business begins to expand the original purpose of the SharePoint implementation.

Storage performance is particularly important in the SharePoint ECM solution due to document archive requirements. It is often necessary to store documents for seven years or more. This can be a challenge for organizations that have high document volumes resulting from business processes with partners, vendors, and customers.

Performance Pitfalls

Before diving into the prescriptive guidance for proper storage architecture, it's important to understand the most common pitfalls that impact disk I/O performance. Armed with this information, it will be easier to understand the recommendations in the remainder of the chapter.

Consider a scenario in which a document management system developed in-house has ballooned to a 1.5TB repository containing 16 million documents. The legacy system is no longer able to handle the performance requirements of the 10,000 system users. SharePoint is an excellent fit for the future of this system. It is decided that a migration is in order to import the repository documents along with the metadata from the legacy solution. Immediately, IT budgets for the new storage, assuming that there is a one-to-one storage size relationship between the legacy system and the new SharePoint storage requirements. Therefore, before any further architecture is performed, IT allocates five 15,000 RPM SAS drives, each 600GB in size, to build a 2.4TB RAID 5 disk array in the shared storage area network (SAN).

Too Few Disks in the Array

The first problem with this scenario is that there are simply too few disks. While it is possible to contain the 1.5TB of content in the 2.4TB RAID 5 array, the IOs per second (IOPS) will be limited in this scenario. Determining the required IOPS for a given implementation is not an exact science. Microsoft recommends a multi-phased approach consisting of modeling, designing, testing, optimizing, deploying, and monitoring. During these phases, the following variables must be considered:

- Storage size requirements to accommodate all content
- Requests per second (RPS) generated by users and other system processes
- Minimum acceptable end user request latency

While it is beyond the scope of this chapter to dive into the details of a SharePoint Server 2010 capacity planning model or how to test and validate the design, a good rule of thumb for large content repositories that have a high RPS requirement of several hundred or more is to have 2 IOPS per GB of content. In this scenario, the IOPS requirement would be approximately 3,072. If the

content storage requirement were lower or the RPS requirement were lower, then the rule of thumb or “starting point” IOPS guidance could be adjusted downward.

For the purposes of this scenario, perhaps the five 15,000 RPM SAS drives provide between 700 and 1,000 IOPS depending on the caching capabilities of the SAN. In this case, while the content storage requirements might be met, performance requirements might not be.

A better solution would be to purchase sixteen 15,000 RPM SAS drives that are 146GB in size. The IT group could then build two, eight-drive RAID 5 arrays and divide the content databases between them. This would theoretically yield two 1TB arrays with something like 2,500 to 2,800 combined IOPS capability. This would help ensure that future storage requirements can be met.

Of course, this is a scenario with IOPS estimates that are somewhat contrived, but it illustrates the point of purchasing too few disks to serve a given content storage requirement. The appropriate solution recommendation of having the additional disks is significantly more expensive than using fewer, larger disks, but it will be more expensive in the long run if end users are waiting for content retrieval or, worse, they tire of waiting and stop using the system entirely.

The principle to remember here is that given the same size disk array, the array with more disks of a smaller size will significantly outperform an array with fewer disks of a larger size.

Shared SAN vs. DAS vs. NAS

The second problem with this scenario is the shared SAN. In this scenario, IT procures new drives, but that is usually not the case. More commonly, a given SAN solution is set up with a large storage base consisting of numerous disks intended for use as shared storage. Often, Microsoft Exchange, Microsoft SharePoint, other Microsoft SQL Servers, and even file shares will share the same SAN. Some SANs are smart enough to allocate the proper number of disks to achieve a requested IOPS for a given storage volume, but many are not.

SharePoint 2010 is a very broad-featured platform but its significant disk requirements go far beyond the content database to include application service databases like those used by the Enterprise Search service applications, which require extensive IOPS support. Shared SAN storage is often difficult to properly allocate because each of the systems that use the shared storage often have peak usage times that coincide.

Network area storage (NAS) can be even further limited in performance by the nature of network-based transfer protocols. Even with a 10-gigabit Ethernet backbone, latency is often an issue. At the time of this writing, Microsoft only supports NAS for use with content databases that are configured to use remote BLOB storage (RBS), which is discussed later in this chapter. Further, the NAS device must respond to a ping within 1ms and return the first byte of data within 20ms upon request. While there are always exceptions to the rule, for the purposes of most highly scalable SharePoint 2010 solutions, NAS storage should be avoided.

As its name suggests, direct-attached storage (DAS) is any storage that is directly attached to the server, typically by SCSI connection or Fibre Channel. The biggest benefit of DAS is the guarantee of *unshared* storage and direct control over configuration of the disk array. This usually results in

a much more predictable performance pattern in terms of supplying SharePoint with all the IOPS it needs to efficiently serve users.

The allure of the SAN is cost, which can be shared across cost centers. It is typically easier to justify the purchase of a single large SAN that would service multiple departments than it is to justify a quality DAS system that serves only SharePoint. However, the upfront performance benefits and long-term performance guarantee of DAS storage is hard to deny. While DAS vs. SAN pros and cons can always be debated, in most cases DAS is the preferred storage technology for SharePoint — regardless of whether it is installed on bare metal servers or in a virtual machine environment.

The principal to remember here is that if SharePoint is to become a mission-critical system, it can't be treated as a second-class citizen. In most cases, it will be less expensive in the long run to provide SharePoint with the best and most reliable storage subsystem that the budget will allow.

Content Storage Size Factors

The last big pitfall that our scenario illustrates is that there is more to SharePoint storage planning than just the one-to-one transfer of the document binary data to SharePoint. In SharePoint 2010, even for just the content database, Microsoft provides a 10KB overhead calculation per document for metadata. The following calculation can be used to estimate content database size:

- Estimate the number of documents (D). In the preceding scenario, the number of documents (D) is 16,100,000 (16.1 million).
- Estimate the average size of the documents (S). In this case, there are 16.1 million documents consuming 1.5TB. This works out to an average document size (S) of about 100KB.
- Estimate the number of list items (L) in the SharePoint repository. For collaboration systems, this might be as high as three list items per document; but for most document archive repositories, the number of list items (L) is usually just one list item per document.
- Determine the average number of versions (V) per document. For the purposes of this calculation, assume that the number of versions (V) per document is one.
- Finally, calculate the estimated size of the content database(s), using this formula:

$$((D \times V) \times S) + (10KB \times (L + (V \times D)))$$
- As a result, the estimated storage requirement for all content databases would be 1.65TB. In other words, there is an additional 150GB of just metadata.

In addition to the storage requirements of the content databases, you must also consider storage requirements for other components in the farm, such as the crawl database, the property database, index partitions (for each query server), and the user profile database.

- To calculate the estimated size of the crawl database, take the sum of all content databases and multiply by 0.046.
 - In this scenario, the crawl database would be approximately 78GB.

- To calculate the estimated size of the property database, take the sum of all content databases and multiply by 0.015.
 - In this scenario, the property database would be approximately 25GB.
- To calculate the estimated size of the index partitions, take the sum of all content databases and multiply by 0.035.
 - In this scenario, the index partitions would consume approximately 60GB, not even accounting for partition mirroring or index master merge storage requirements.
- The search administration database is relatively small; allocate 10GB.
- The size of the user profile database is estimated to be approximately 1MB per user.
 - In this scenario, with 10,000 users, the user profile database would be approximately 10GB.

Other service application database storage requirements would also need to be estimated. For this exercise, they will be left out because they are relatively small. However, you need to make a few more important allocations:

- Allocate 25% of the largest database for Temp DB. For this scenario, Temp DB would be a maximum size of 25GB.
- Allocate a 15% to 30% factor for all databases, depending on backup frequency, for log files. To be conservative, in this scenario 20% will be used for an approximate LOG storage requirement of 377GB.
- Allocate a 25% storage overhead factor for all storage arrays for drive defragmentation. In this scenario, approximately 566GB needs to be allocated for defragmentation padding.

The purpose of this exercise is to demonstrate that even though the storage requirement for the documents is 1.5TB in the legacy system, SharePoint will require, at minimum, an additional 1.1TB in storage to handle the other ancillary storage requirements! Taking this even further, this estimate is just for the migration of the existing system to SharePoint. It's important to calculate estimated annual load rates and to project storage estimates for the next several years.

In summary, determining storage requirements for SharePoint is not just a simple summation of the projected document bytes. The formula is a bit more complex than that. If properly followed, a far more accurate estimate of required storage space will emerge. Ultimately, however, no projection will ever be as accurate as direct monitoring. For example, recycle bin usage and auditing requirements are often moving targets that can affect estimates.

Database Storage and Capacity Planning

Now that you know how to avoid the pitfalls, it's important to go even further and plan for the proper deployment of the SharePoint databases on the storage subsystem. For a highly scalable SharePoint deployment, database management goes far beyond just allowing SQL Server to create all the database files in the default folder locations.

There are several “it depends” variables when it comes to proper database deployment technique. Remember that the goal of this chapter is to identify the *best* techniques for a highly scalable SharePoint solution, not necessarily the cheapest. However, an effort has been made to provide viable alternative options that cover additional scenarios. For example, a document archive solution constructed to contain 5 or 10 million documents will not need extreme performance capabilities if only 10 users are accessing the system.

SQL Server Supporting Concepts

Before diving into storage best practices for various SharePoint database resources, several concepts need to be explained, such as optimizing storage array utilization, instant file initialization, pre-sizing database files, database autogrowth, and adding additional data files to databases. These concepts will be referenced frequently.

Optimizing Storage Array Utilization

The “why” behind this concept was partially covered earlier in the performance pitfalls discussion points; but for the sake of completeness, a complete description is provided.

Theoretically, the only way to completely eliminate disk I/O contention and ensure the best possible performance is to place each and every SharePoint database data or log file on a unique disk array such that no other database file can compete for disk I/O. Practically, however, this isn’t going to happen unless cost is of no concern.

Therefore, the best way to mitigate disk I/O contention is to group together data and log files that serve different purposes and then monitor performance metrics to ensure that disk queuing is not limiting SharePoint performance. This technique minimizes the possibility that multiple data files on the same disk array will require simultaneous access.

If possible, a few very important database resources such as TempDB data files, Crawl database data and log files, and content database log files should be placed on segregated storage. The storage priority for each of these database resources will be described later in this chapter.

Instant File Initialization

Typically, when a SQL database data file is resized or “autogrown,” SQL Server zeros out the pages as it adds space. The reason for this is beyond the scope of this chapter, but the performance implications are significant.

Fortunately, it is possible to implement a configuration that allows for *instant file initialization*. With instant file initialization enabled, SQL Server initializes space in a data file without zeroing out pages. Enabling instant file initialization allows for rapid large data file pre-sizing, faster autogrow operations, and significantly faster database restore operations when necessary. For example, when enabled, it might take just a few seconds, rather than 15 minutes, to pre-allocate a 25GB data file.

Enabling instant file initialization is a two-step process. First, the SQL Server service needs to be running as an Active Directory (AD) user (service) account. To verify that SQL Server is running as an AD service account, launch the Services Management Console.

1. Click Start ⇨ Run.
2. Enter `services.msc` into the Open field of the Run dialog and click the OK button. The Services Management console will be launched.
3. Scroll down to view the SQL Server service. The Log On As field contains the name of the service account (see Figure 12-1).

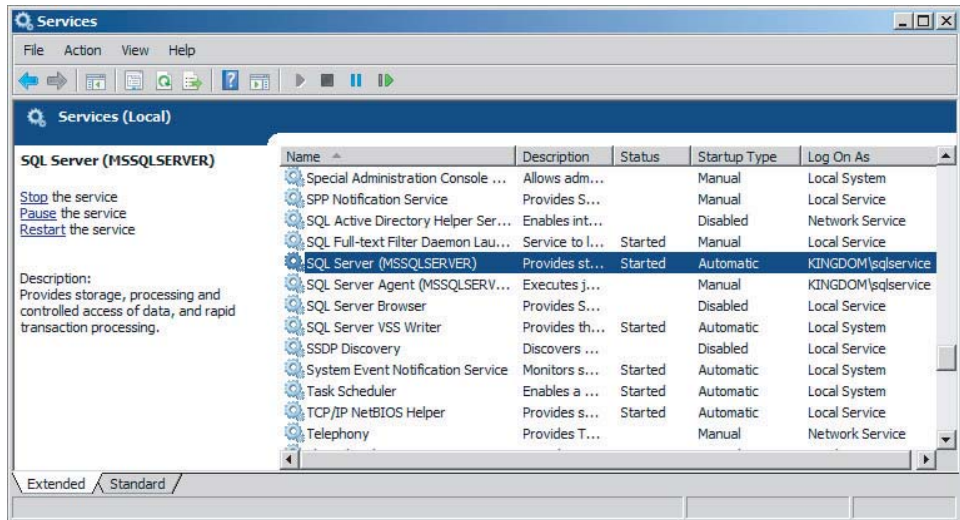


FIGURE 12-1

If SQL Server is not running as an AD service account, the SQL Server administrator needs to determine if this can be accomplished and when the configuration change should take place with respect to business practices and policies.

Second, the SQL Server AD service account must be granted the Perform Volume Maintenance Tasks privilege using the Local Security Policy Management Console.

1. Click Start ⇨ Run.
2. Enter `secpol.msc` into the Open field of the Run dialog and click the OK button. The Local Security Policy Management Console will be launched.
3. Under Security Settings, expand Local Policies and select User Rights Assignment.
4. In the Policy navigation pane, scroll down to view the “Perform volume maintenance tasks” policy (see Figure 12-2).
5. Right-click the “Perform volume maintenance tasks” policy and click Properties.
6. Add the SQL Server AD service account to the Local Security Setting and click the OK button.

In order to activate instant file initialization, the SQL Server service must be restarted.

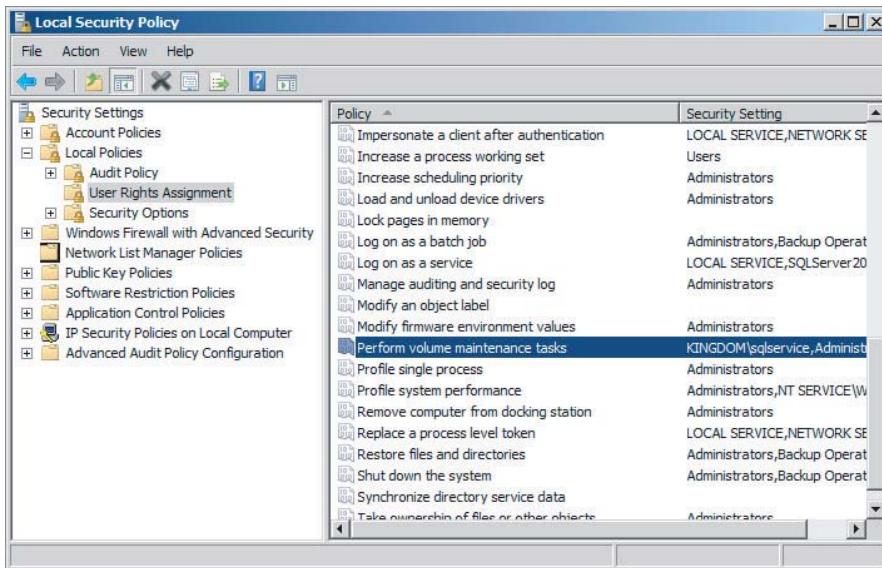


FIGURE 12-2

Pre-sizing Database Files and Adjusting Autogrowth

Pre-sizing database data files can have a significant impact on SQL Server and SharePoint performance in three distinct ways.

First, by pre-sizing a data or log file to a predetermined size, SQL Server won't need to autogrow the file when the database runs out of space. Growing data files by several small increments causes significant file fragmentation. Over time, this increases the number of disk reads and writes that are necessary for storing and retrieving data. Unfortunately, the default setting for new, nonsystem databases is to autogrow in 1MB increments. Imagine how many autogrow operations have to take place for a content database to go from 1MB to 100GB!

Second, by pre-sizing log files in large increments, the number of virtual log files (VLFs) is reduced. The physical log file for a given database consists of multiple VLFs. A large number of small VLFs leads to more log file fragmentation. Conversely, a small number of large VLFs leads to less log file fragmentation. A good increment of growth would be 8GB chunks. For example, if the starting log size were 1MB and the target log size were 16GB, you could consider applying an increase of 8GB first and then applying a second increase of 8GB. By following this technique, internally SQL Server will allocate a total of 32 new VLFs that are 512MB each. This is a much better configuration than the default alternative; with the 1MB default initial size and a 10% default autogrowth, by the time the log file reached 16GB, there would be thousands of very small VLFs and a significantly fragmented log file. Note that instant file initialization does not apply to log files. Increasing a log file by 8GB will likely take several minutes.

Finally, by taking the time to predict and implement reasonable data and log file sizes, SQL Server doesn't have to take the time to extend the size of the files in the middle of a database transaction. Although the effect is small, it is also cumulative — particularly during large data loads such as those that occur during a data migration to SharePoint.

The easiest way to pre-size a database file, add additional database files, or change autogrowth settings for a database file is to use Microsoft SQL Server Management Studio. The following instructions are for an SQL Server 2008 R2 implementation but they generally apply to SQL Server 2008 and SQL Server 2005 as well.

1. Click Start ⇨ All Programs ⇨ Microsoft SQL Server 2008 R2 ⇨ SQL Server Management Studio.
2. Right-click the database name and select Properties.
3. In the “Select a page” pane, select Files.
4. Type the desired size of the data or log file, in megabytes (see Figure 12-3).
5. Click the Autogrowth ellipsis for the relevant data file.
6. In the Change Autogrowth for [Database File Name] dialog, set the values appropriately and click the OK button (see Figure 12-4).
7. Click the Add button to add an additional data file.
8. Enter a Logical Name, set the initial file size, configure Autogrowth, and ensure that the data file is stored in the proper location.
9. Click the OK button to apply the changes.

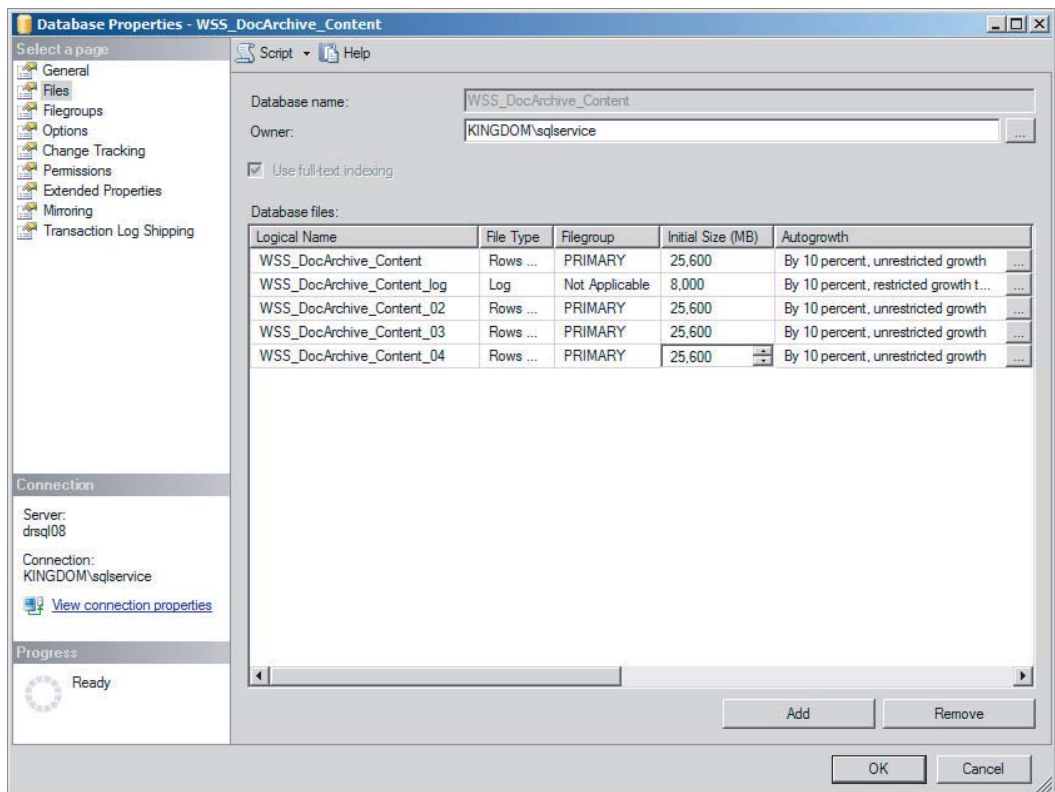


FIGURE 12-3

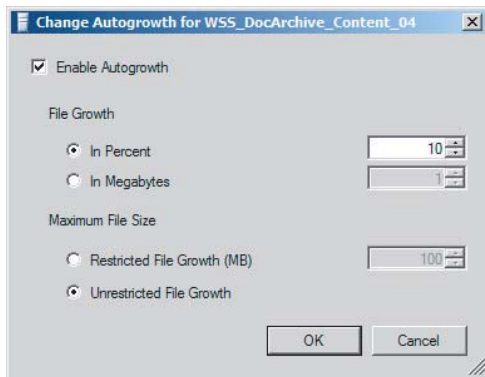


FIGURE 12-4

If instant file initialization is enabled, changes to data files should only take a few seconds to complete. Remember that instant file initialization does not apply to increases in the log file size. Note also that it is counterproductive to add additional log files. Each database needs only one log file. Adding additional log files will not improve performance — and could potentially decrease it.

TempDB

SharePoint takes advantage of the TempDB database in SQL Server by storing intermediate results during query and sort operations when necessary. Poor performance can have a significant impact on SharePoint performance and, more specifically, the end user experience.

Note that there is only one TempDB database per instance of SQL Server. However, for very large SharePoint farms, more than one SQL Server will be required. One SQL Server would be dedicated to content storage and light service application databases, while an additional one or more SQL Servers would be deployed to support large-scale Enterprise Search crawl and property databases.

Number of Data Files

TempDB is a special database. The rules that apply to “standard” databases don’t necessarily always apply. That said, the recommended number of TempDB data files will end up being similar to the recommendation for many of the other databases used by SharePoint:

Before the recommendations are presented, it’s important to understand the difference between a physical processor, physical processor cores, and logical processor cores. A physical processor is the actual single physical computer chip package that is inserted into the CPU socket on the motherboard. A physical processor core is a collection of transistors and supporting pathways that operate as a single physical unit inside of a physical processor package. There may be one or many physical processor cores in a physical processor package. A logical processor core is a logical processor unit that is presented to the operating system as a processing unit such that the processor instance appears in the Task Manager utility. This can be confusing because certain processor technology, such as Hyper-Threading, allows a single physical processor core to appear as 2 processors. Hyper-Threading is an additional hardware interface to the processor that facilitates multi-threaded operations to improve CPU performance. This is an important distinction because

there are not actually 2 physical processor cores. So if a server has 2 processors that have 4 physical processor cores each, there are actually 8 physical processor cores that can do work. However, if Hyper-Threading were enabled, Task Manager would show 16 logical processors. In this example, the relevant number is the 8 physical processor cores.

1/4 to 1/2 the number of physical processor cores in the SQL Server

The reason why you add additional data files to TempDB is to minimize the chances of data file I/O contention during read and write operations to TempDB. As with many configuration options in SQL Server, it's easy to err in either direction: Not adding additional data files could at some point reduce the performance of SQL Server due to I/O contention, but adding too many data files to TempDB could also reduce performance.

Some Microsoft documentation suggests that the formula be a one-to-one relationship of TempDB data files to CPU cores. Real-world observations often show this to be overkill. When in doubt, follow the advice of a qualified SQL Server DBA. They have techniques for monitoring specific metrics that can drive more accurate guidance for a specific running instance of SQL Server.

Adding additional data files to TempDB is just like adding data files to any database. Follow the instructions previously described to add additional files to TempDB.

Pre-Sizing TempDB

By default, each time the SQL Server service is restarted, TempDB is recreated based on the original configured size of 8MB. This value is significantly undersized for a SharePoint farm that is expected to hold potentially millions of documents.

There are multiple schools of thought with respect to the configured sizing of TempDB. The most consistent prescriptive guidance available suggests that the sum of the TempDB data files should be equal to approximately 25% of the size of the largest predicted SharePoint database in a given SQL Server instance. For example, if the largest content databases is expected to reach 750GB in size, then the sum of the TempDB data files should be approximately 75GB. Therefore, the formula for pre-sizing the data files in TempDB is as follows:

$$[\text{Max TempDB Size (MB)}] * .25 / [\text{Number of TempDB data files}] = \text{Data File Size (MB)}$$

Note that TempDB data files should be equal in size. Follow the instructions previously defined for pre-sizing the TempDB data files.

Instant file initialization is again important in this situation. Once the SQL Server instance is restarted, it will recreate the TempDB using these configured settings. If instant file initialization is not enabled, it can take SQL Server several minutes to initialize the new TempDB before becoming available.

Be sure to monitor the size of TempDB after SharePoint and SQL Server have been running for an extended period of time. If TempDB routinely exceeds the estimated pre-sizing size, then the pre-sizing calculation and configuration should be repeated using a new estimate.

Storage Volume Performance and Reliability

TempDB data files must be stored on a fast storage volume. Ideally, the storage should be a disk array, consisting of many disks, optimized for both read and write operations such as RAID 10.

If at all possible, TempDB should be placed on a unique disk array that is not shared by content databases or other SharePoint databases.

The storage location of the TempDB log file is less important. This is because TempDB, by default, uses the “Simple” recovery model. Because TempDB is recreated each time SQL Server is restarted and the data in TempDB is, by definition, temporary, there is no need for backup and restore. Therefore, log file writes are minimized to improve the performance of insert and delete operations.

In some circumstances, such as when the SharePoint farm is supporting an extremely high volume of content and a large number of users, TempDB may become a bottleneck. In this case, additional performance benefit may be realized by moving each of the individual data files to their own unique storage array.

Log Files

Behind TempDB, log files should get the priority for available high-performance storage. Remember that any transaction that affects a SharePoint database can't complete until a record of the transaction is written to the log file. Therefore, there will always be a dependency on the performance of the database log file.

Number of Log Files Per Database

There should be only one log file per database. SQL Server writes to the transaction log file in a sequential and circular pattern. As backups occur, the log is cleared and SQL Server is able to sequentially write to those locations again. SharePoint would never use more than one log file to improve performance, so there is never a need to have more than one log file.

Pre-sizing Log Files

Log files should be pre-sized. Of course, there is no magical way to reliably define this value for any given database, for various reasons. The most important factor that determines how large a log file will grow is the frequency of transactions with respect to the frequency of transaction log backups. Here are a few scenarios:

- If data trickles into SharePoint at a fairly slow pace and rarely changes and backups are defined to run weekly, then log files should remain fairly small.
- If data is rapidly being added, updated, or removed from SharePoint and crawls are frequently running but backups are also very frequent, then log files will also likely remain fairly small.
- If data is rapidly being added, updated, or removed from SharePoint and crawls are frequently running but backups are fairly infrequent, then the log file will be much larger.

Proper sizing of transaction logs for any given SharePoint database should be an exercise in analyzing empirical data. Watch the logs closely. If they are frequently “autogrowing,” then it is probably time to extend them manually or possibly review backup frequencies.

That said, you need a starting point. There are many schools of thought, but many agree on a starting point of somewhere around 15% to 30% of the predicted database size, depending on backup frequency. Follow the procedures previously defined for pre-sizing log files.

Storage Volume Performance and Reliability

For transaction logs, reliability is more important than performance, although performance is important as well. The transaction log is the lifeline to recovering data that has been modified since the last backup of the database. The storage array should be fault tolerant if at all possible.

Transaction log file I/O is write-heavy, so a storage array, such as RAID 10, that has been optimized for write operations is recommended.

The storage array used for transaction logs should be isolated to just the transaction logs if at all possible. Keeping file I/O contention to a minimum with respect to transaction logs is an important key to SharePoint performance.

In extreme cases, when the SharePoint farm contains a high volume of content, you may need to further isolate the transaction log storage array for major SharePoint subsystems. Consider a SharePoint farm that contains 40 million documents, with an additional 22,000 new documents being added every day. Content database logs will be heavily loaded with add transactions, while crawl database logs will be heavily loaded with add, modify, and delete transactions. In this case, the crawl subsystem is working hard to keep up with the new documents being added, so it's important that end users who are accessing the new content not feel the pain of delayed logging operations. Again, empirical evidence dictates the final word. Modifications to data and log storage locations can be made based on tuning principals, which are presented in the Storage Tuning and Optimization section later in this chapter.

Crawl Databases

Initial users of SharePoint 2007 faced a problem with respect to database size and crawl performance. For content databases, the general maximum size recommendation was 100; but if there were many 100GB content databases containing tens of millions of documents, then the search database could easily be 200GB to 400GB in size! Further limiting scalability was the fact that the property catalog was stored in the same database, and (by default), in the same file group as the tables that managed the crawling of all those millions of documents. Eventually, Microsoft made a script available that separated the property catalog and the crawl processing tables, but scalability was still limited.

For SharePoint 2010, Microsoft wisely redesigned major components of the Enterprise Search architecture. One of the most important changes is the fact that the property catalog now resides in one or more property databases, and the crawl processing tables reside in one or more crawl databases.

As you consider the storage architecture for the crawl databases, keep in mind that it doesn't make any sense to build a farm that can store 60 million documents if database I/O contention causes a full crawl to take three weeks.

Number of Data Files

Similar to other high-performance databases that underpin SharePoint 2010, adding additional data files to crawl databases is beneficial. Microsoft recommends the following guideline:

1/4 to 1/2 the number of physical processor cores in the SQL Server

The reason you add additional data files to crawl databases is to minimize the chances of data file I/O contention during read and write operations to the crawl database. The crawl database is one of the hardest-hit databases in the entire SharePoint farm, so you should utilize all available techniques for maximizing throughput to and from the crawl database.

Follow the instructions previously defined to add additional crawl database data files.

Pre-Sizing the Crawl Database

Pre-sizing crawl databases can reduce data file contention and help maintain crawl database performance as the corpus grows. (The corpus is the size, type, and number of documents in the farm.) To begin the pre-sizing calculation, first identify the total storage size of all content databases. For existing systems, simply add up the total *in-use* size of the individual content databases. For green-field installations, whereby only new content is loaded or some type of legacy migration will be performed, identify the best estimated total size for legacy content with respect to the best estimated load rate. To estimate the total size of all crawl databases, use the following guideline:

Multiply the *total used content database size* by .046.

The recommended maximum number of items managed by a single crawl database is 25 million. With that in mind, consider an example in which a highly scaled farm containing 75 million documents consumes a total content database size of 7.75TB. In this case, the predicted crawl database size would be as follows:

$$7939\text{GB} * .046 = 365.2\text{GB}$$

However, because there are 75 million documents in the corpus, there would be at least three crawl databases, each with an estimated total size of 121GB. In this example, each crawl database should be pre-sized to approximately 121GB.

Note that crawl databases are not likely to be distributed perfectly evenly, so it is important to monitor the size of the crawl databases as they grow to ensure that they have been properly pre-sized. Follow the instructions previously defined for pre-sizing the crawl database data files for each crawl database.

Storage Volume Performance and Reliability

In order for a highly scaled SharePoint 2010 Enterprise Search solution to perform properly, there are very specific and important recommendations with respect to storage of crawl databases. Consider these best practices:

- Multiple crawl databases should each have their own unique disk arrays.
- A crawl database should not share a disk array with any other SharePoint 2010 database resources such as property databases, content databases, TempDB, or log files.
- The storage array for a fully utilized crawl database of up to 25 million items should be able to facilitate at least 3,500 IOPS for crawl processing. As much as 6,000 IOPS can be consumed if available.
- The storage array for crawl databases needs to provide good performance for both read and write operations, so it should be constructed in a RAID 10 or similarly performing design.

As usual, mileage may vary depending on the type of content being crawled and the actual size of the corpus. Smaller farms can get by with some storage sharing, but expectations should be realistic. If crawl speed is a business-critical metric, then proper storage architecture is paramount.

If the crawl database is lost, a full crawl will be needed to restore full search capabilities, so a fault-tolerant storage solution is recommended.

Content Databases

The content database is the focal point for storage of all SharePoint list and document library items. For SharePoint ECM purposes, it is one of the most important databases because it contains the metadata and, in most cases, the binary data for all those millions of documents that comprise the digital assets of any company.

Number of Data Files

Adding additional data files to content databases is a given. Content databases are consistently among the largest databases in the farm. The standard calculation for the number of data files continues to be the one shown previously:

1/4 to 1/2 the number of physical processor cores in the SQL Server

Consider adding additional data files to any content database that is estimated to contain 20GB of content or more. Follow the instructions previously defined to add additional property database files.

Pre-Sizing the Content Database

Pre-sizing the content database helps minimize file fragmentation and improves I/O performance for both read and write operations. If legacy content will be migrated into content databases after implementation, estimate the size of the content storage requirements by using the formula described earlier in this chapter, based on the taxonomy that will be deployed.

After the estimated size of a given content database has been determined, follow the instructions previously defined for pre-sizing the content database data files for each content database.

Storage Volume Performance and Reliability

Obviously, the content database is the focal point of nearly all processing in SharePoint. Actions that occur every day, such as adding, editing, and removing content, retrieving documents, and crawling content, all create load on the content databases. For this reason, the highest performing storage available, such as a RAID 10 storage array, can improve the performance of all subsystems.

However, for document archive solutions, for which content is rarely modified or even accessed, it is possible to get by with a high-quality RAID 5 storage array. Another option is to enable Remote BLOB Storage (RBS). This enables metadata to be stored on high-performing RAID-10-type storage arrays while the actual binary content can be stored in a RAID 5, lower-performance array to save costs. RBS is covered later in this chapter.

Regardless of the performance level chosen for the content database storage array, it is important to use a highly available solution. The content database is the single most important database in

any SharePoint implementation. All other databases can be rebuilt or recreated in some way; but if a content database is lost and can't be restored from backup, nothing short of manual re-entry can create the lost content.

Property Databases

A property database contains a superset of managed property values, called the *property catalog*, for every document that is crawled. This results in data tables that grow very rapidly. While a property database is likely to be smaller than a crawl database, performance is still important.

A *fielded query* is one that makes use of one or more managed properties, typically custom metadata fields, to search for documents that exactly match the provided query values. The property catalog is used by these fielded queries to locate the appropriate documents with 100% relevance, or *exact relevance*.

Number of Data Files

Similar to other SharePoint 2010 databases, adding additional data files to property databases is beneficial. Microsoft recommends the following guideline:

1/4 to 1/2 the number of physical processor cores in the SQL Server

While not absolutely required, adding additional data files to property databases will minimize database file contention and facilitate optimal property-based query performance. Property-based queries are very important for large, document-archive-type SharePoint ECM solutions because custom document metadata can be used to provide exact relevance searches.

Follow the instructions previously defined to add additional property database data files.

Pre-Sizing the Property Database

Pre-sizing property databases increases fielded-query performance as the corpus grows. In order to pre-size a property database it is important to estimate the projected storage size. Again, this is not an exact science because the number of managed metadata columns can significantly affect the size of the property database. Therefore, as before, the following guidelines provide calculations for an initial target size. Real-world monitoring as the system is brought online will always provide more accurate predictions.

To begin the pre-sizing calculation, first identify the total storage size of all content databases. For existing systems, simply add up the total in-use size of the individual content databases. For green-field installations, whereby only new content is loaded or some type of legacy migration will be performed, identify the best estimated total size for legacy content with respect to the best estimated load rate. Use the following to estimate the total size of all property databases before partitioning:

Multiply the *total used content database size* by .015.

The recommended maximum number of items managed by a single property database is 50 million. However, the number of managed properties affects property database performance. In some cases, it may be necessary to add one or more additional property databases to maintain performance targets. Note that additional property databases may need to be placed on an additional database server to realize the performance gain.

Once the number of property databases is determined, divide the total size of the property database by the number of property databases to identify an initial estimated property database size. Follow the instructions previously defined for pre-sizing the property database data files for each property database.

Storage Volume Performance and Reliability

In order for a highly scaled SharePoint 2010 Enterprise Search solution to perform properly, there are very specific and important recommendations with respect to storage of property databases. Consider these best practices:

- Multiple property databases should each have their own unique disk arrays when possible.
- A property database should not share a disk array with any other SharePoint 2010 database resources such as crawl databases, content databases, TempDB, or log files when possible.
- The storage array for a fully utilized property database of up to 50 million items should be able to facilitate at least 2,000 IOPS for property catalog processing.
- The storage array for property databases needs to provide good performance for both read and write operations. Therefore, the array should be constructed in a RAID 10 or similarly performing design.

As usual, mileage may vary depending on the number of managed properties present and the actual size of the corpus. Smaller farms and, to some degree, some medium-size farms can get by with some storage sharing, but be realistic with expectations. In addition to slow response times for end user fielded queries, crawl processing performance can also be affected by poorly performing property databases.

If the property database is lost, a full crawl will be needed to restore full search capabilities. Therefore, a fault-tolerant storage solution is recommended.

Property Database Memory Requirements

For property databases, there is an additional recommendation that the database server have enough RAM to keep 33% of the primary tables in memory. The primary tables are as follows:

- MSSDocSdids
- MSSDocProps
- MSSDocResults

If the database server does not have enough RAM to keep 33% of the property database primary tables in memory and fielded query performance is suffering, consider adding additional RAM to the SQL Server that services the property database. If more than one property database is deployed in the topology, another option is to move one or more property databases to an additional SQL Server.

Service Application Databases

Each individual service application database has its own storage size and performance requirements but most are minimal. Up to this point, the database storage recommendations have encouraged

separation of the storage arrays. For most of the service application databases, shared storage will suffice.

Usage and Health Data Collection

This service application database is one of the few that can grow quickly and require high-performance storage. This is particularly true for collaborative environments. According to the publication “Capacity Planning for Microsoft SharePoint Server 2010” by Microsoft, for collaborative environments that use out-of-the-box settings, 1 million HTTP requests requires 2GB of storage. However, for document repository-type solutions in which relatively few users access a large quantity of data, there will be significantly fewer requests. To estimate size, use one of the following calculations:

- $115 \times \text{page hits/second}$
- $5 \times \text{HTTP requests}$

If the projected database size will be larger than 20GB, then pre-sizing the database and adding additional data files can help minimize file contention.

For a large, typically collaborative farm solution servicing a high number of requests, it is recommended that the usage database be placed in its own storage array.

Business Data Connectivity Service

The Business Data Connectivity services database will typically consume only a small amount of space (50MB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

Application Registry Service

The Application Registry services database will typically consume only a small amount of space (1GB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

Subscription Settings

Subscription settings typically consume only a small amount of space (1GB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

User Profile Databases

The storage requirements for the user profile databases are relative to the number of users associated with the farm. For the profile database, allocate approximately 1MB per user profile. For the synchronization database, allocate approximately 630KB per user profile. For the social tagging database, allocate approximately .009MB per tag, comment, or rating.

If the profile, synchronization, or social tagging databases are projected to exceed 20GB, then pre-sizing the database(s) and adding additional data files can help minimize file contention.

Managed Metadata Service

The managed metadata service database typically consumes only a small amount of space (1GB or less) and storage performance is not of great concern. In most cases, shared storage will be sufficient and there is no need to pre-size the database or add additional data files.

For extremely large farms, or services farms with a very large file plan or term store, monitor the size and performance of the managed metadata service database as it grows. If it exceeds 20GB, then adding additional data files can help minimize file contention.

Web Analytics Service

The Web Analytics service databases can consume a significant amount of storage depending on retention period, daily volume of data being tracked, and the number of site collections, sites, and subsites in the web application being analyzed. In other words, the primary function of the solution dramatically affects storage size and performance requirements, as each click, search, or rating added becomes a record in the database.

Microsoft provides sample performance metrics on what they call a *midsize* dataset. This sample dataset includes 30,000 SharePoint components, 117,000 unique users, 68,000 unique queries, 500,000 unique assets, and a reporting database size of 200GB. In this scenario, it is important to properly manage storage of the staging and reporting databases. Additional data files should be added to these databases. Pre-sizing is very difficult for databases of this type, so it's important to ensure that instant file initialization is enabled and the databases are configured to autogrow in large enough chunks to minimize the number of autogrow operations required. Given the analytical nature of the required processing, disk I/O is at a premium and the storage array needs to be dedicated to only the staging and reporting databases when possible.

Continuing with the ECM theme, consider a SharePoint solution that is designed as a large-scale document archive system with some collaboration. In the majority of solutions like this, SharePoint will contain a lot of documents but they will be handled by relatively few users. Storage requirements for the midsize dataset defined above will likely be significantly greater than what is necessary for most large-scale document archive-type SharePoint solutions. As in many cases, "it depends" applies here. The size and usage pattern of any given SharePoint implementation will ultimately drive the disk storage requirements for web analytics. Regular monitoring is highly encouraged to prevent underpowered storage or conflicts with other storage requirements.

Secure Store Service

The Secure Store service database typically consumes only a small amount of space, and storage performance is not of great concern. Allocate 5MB for each 1,000 credentials stored. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

State Service

The State service database typically consumes only a small amount of space (1GB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

Word Automation Service

The Word Automation service database typically consumes only a small amount of space (1GB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

PerformancePoint Service

The PerformancePoint service database typically consumes only a small amount of space (1GB or less) and storage performance is not of great concern. Shared storage will be sufficient. There is no need to pre-size the database or add additional data files.

Management Databases

Requirements for the configuration database and the Central Administration database are minimal. Allocate 2GB for the configuration database and 1GB for Central Administration. Shared storage will be sufficient. There is no need to pre-size these databases or add additional data files.

Unless SQL Server database mirroring will be used to ensure configuration database availability, the recovery model for the configuration database should be set to Simple. Otherwise, the configuration database transaction log will grow significantly.

Prioritizing Disk I/O

This chapter has emphasized the importance of understanding database storage architecture because of the significant impact that it can have on farm performance. The following points summarize the priority of disk allocation:

1. TempDB and transaction logs are the most critical components. When possible, segregate them onto high-performance read/write storage.
2. Database transaction log files should also be segregated onto high-performance read/write storage when possible.
3. Search subsystem crawl and property databases should be separated when possible. Crawl databases can consume more IOPS than any database in the farm.
4. Content databases are the most important databases in the farm with respect to reliability. Above all else, ensure that they are stored on redundant storage.
5. Be careful with Usage and Health databases, User Profile databases, and Web Analytics databases. Their growth rates can surprise you. Allocate storage properly for them from the start.
6. Most other application service databases and management databases have minimal storage requirements from both a size and performance perspective, but always use redundant storage options.
7. Always remember that estimates are just a starting point. It is imperative that regular monitoring be performed to ensure that storage resources are not being consumed unexpectedly.

Index Partition Storage

In SharePoint 2010, Microsoft has made several changes to the Search Service Application architecture. One key change is that crawlers are now stateless. They maintain only a small portion of the composite index at any given time, enough to work through the indexing of the files that are currently being processed. The actual index is stored in one or more index partitions, which are maintained on one or more query servers. As content is indexed by the crawler, the index partitions on the query servers are updated.

Calculating the total index size is a difficult task because all SharePoint implementations have a different *corpus*, which is the size, type, and number of documents in the farm. However, there is a formula for calculating the estimated total index size:

1. Multiply the *total used content database size* by .035 to get the *total index size*.
2. Divide the *total index size* by the number of index partitions to get the *query component index size*.
3. Multiply the *query component index size* by 4 to calculate the *disk storage requirement* for a single query component.

Although it may seem like a lot of extra storage is being factored in for a given query component, additional storage must be allocated to allow for additional processing such as master merge and index repartitioning.

Note that the query component index size also factors into the amount of RAM required on the query server: 33% of the index for an active query component must fit into RAM.

4. Multiply the *query component index size* by .33 to determine the RAM requirement for a given query server.

If an index server will host more than one active query component, sum the RAM requirement for each query component to determine server RAM requirements.

Index files stored on the query server are regularly updated by crawlers, so write performance is very important. Also, the index files are frequently accessed as end users execute queries. Therefore, read performance is also important. As a result, it is important to implement a high-performance storage array such as a RAID 10 configuration to store the index files for all query components on a query server.

Storage Tuning and Optimization

Your shiny new SharePoint 2010 farm has been deployed and maybe even has some legacy data all loaded up. Is it time to call it a weekend and hope for the best when all the users show up on Monday morning? The answer to that is categorically “no.” Hopefully, the deployment schedule allotted time for testing, monitoring, and tuning.

Storage Performance Monitoring

One of the many advantages of implementing a wide storage subsystem that consists of many storage volumes is the capability to monitor physical disk metrics by disk volume. When all or most of

the SharePoint databases are located on one volume, it can be difficult to monitor and then adjust for disk over-utilization. Table 12-1 describes a few key performance disk counters that can be tremendously valuable when monitoring SharePoint storage volumes.

TABLE 12-1: Disk Counters for Monitoring Performance

| COUNTER | DESCRIPTION |
|------------------------------|---|
| Avg. Disk Queue Length | The average number of read and write requests that were queued for the selected disk during the sample interval. A higher disk queue length is acceptable only if the number is stable and reads/writes are not suffering. Values in the decimal range are optimal. Values in the single-digit range are acceptable. Rising double-digit values are cause for concern. Triple-digit or higher values almost always indicate farm performance degradation. |
| Avg. Disk Read Queue Length | The average number of read requests that are queued |
| Avg. Disk Write Queue Length | The average number of write requests that are queued |
| Disk Reads/Sec | The number of reads to disk per second |
| Disk Writes/Sec | The number of writes to disk per second |

The most important metric in Table 12-1 is Avg. Disk Queue Length, as it provides an overall representation of whether or not the selected storage volume is able to keep up with the demands that SQL Server and thus SharePoint are asking of it.

Database Data File Management

This section looks at two common scenarios that need to be addressed. First, what should be done if storage performance monitoring identifies that one or more databases are causing enough I/O to warrant storage array separation? In this case, obviously it is time to spin up a new array and move a database or two. However, before any database movement is performed, it is extremely important to validate the integrity of the database and then ensure that a full backup is successfully performed on it.

Execute the following steps to perform a database integrity check and then move the database:

1. Execute a full DBCC integrity check on the database.
2. Perform a full back up of the database and validate that it was successful.
3. Detach the database from SQL Server.
4. Move the individual data files and possibly the log file to the new destination locations on one or more different disk volumes.

5. Reattach the database. Using SQL Management Studio, select the database “mdf” file from its new location. If any of the related database files display a message status of Not Found, simply locate that file in its new location.
6. Verify that the database is accessible.

Second, what if an existing SharePoint farm contains a data file that is extremely large in size? In this case, it is possible to add additional pre-sized data files to the database and redistribute the content inside the data files. The easiest way to execute this process is to follow these steps:

1. Execute a full DBCC integrity check on the database.
2. Perform a full backup of the database and validate that it was successful.
3. Add the calculated number of data files to the primary file group of the database using the procedures defined earlier in this chapter. For example, if there should be four data files but there is currently only one, add *four new* data files. After this procedure, the initial data file will not be used but it can't be removed, so there would be a total of five data files: four with data and a fifth that is essentially empty.
4. Pre-size the new data files equally such that they can contain at least all of the data in the existing data file using the procedures defined earlier in this chapter.
5. Execute the following script in a SQL Management Studio query window:

```
USE [Database_Name]
GO

DBCC SHRINKFILE (N'Database_Name', EMPTYFILE)
GO
```

This procedure will empty the contents of the original MDF file by distributing the data equally into the new NDF data files. Depending on the size of the original data file, this can take several hours to complete. Once the process is complete, the new NDF files should contain roughly equal amounts of data, leaving the MDF file essentially empty. It is then possible to release unused space from the MDF file. You can then either pre-size the data file to share future load with the other files, or you can disable autogrowth to prevent any future data from being loaded into the file.

This is a rare case in which performing a database shrink operation can actually improve a database's performance. Typically, shrink operations dramatically increase database fragmentation and should be avoided; but in this case, the shrink is just reallocating the data to the new data files. That said, it is still important to check database fragmentation after the reallocation process, and execute a defragmentation operation if necessary.

Remote BLOB Storage

Remote Binary Large Object (BLOB) Storage (RBS) is a very important concept when it comes to using SharePoint as an ECM solution. The ECM industry has known for years that a relational database is not always the most efficient place to store binary file data. Virtually all other document

management solutions available today store the document binaries either somewhere on the file system or in *content addressable storage*, also known as CAS devices. However, like SharePoint, they all store the structured metadata in a relational database.

The capability to externalize binary content in SharePoint was not possible until SharePoint 2007 SP1, when Microsoft enabled *external binary storage (EBS)*. However EBS had shortcomings that prevented rapid adoption. Some of those early concerns with SharePoint 2007 EBS included the following:

- BLOBs were orphaned during update operations.
- Orphan cleanup could be very resource intensive.
- Future support of the EBS API was in doubt.

SharePoint 2010 has eliminated these issues. RBS takes advantage of the transactional nature of SQL Server and provides a well-architected approach to orphan maintenance, solving the shortcomings of EBS. Here are a few features of RBS:

- The RBS Provider contract ensures that binary data is persisted to the BLOB store during processing, minimizing the possibility of creating orphaned documents.
- Orphan cleanup takes advantage of SQL indexes and is much less resource intensive.
- RBS is transparent to the SharePoint API so custom solutions are not affected by externalization.
- RBS will be supported by Microsoft in future versions of SharePoint.

Now that RBS is here to stay, it's important to discuss why binary externalization is so important to SharePoint. The most obvious benefit with RBS enabled is that content databases that contain very large files will no longer consist of primarily binary data. But it is important to understand that even with RBS enabled, the architect must include external binary storage requirements as part of the content database storage size. In other words, if a content database contains collaboration content consisting of 40GB of metadata and the related RBS BLOB store contains 260GB of binary data, then the content database has effectively reached a collective size of 300GB, from a capacity planning viewpoint. The latest guidance from Microsoft is that any content database, including collaboration databases, may be allowed to reach 4TB in size as long as disk subsystem performance meets recommended IOPS targets and high availability, disaster recovery, future capacity, and performance testing have been observed. There is no longer a size limit for content databases that contain primarily archive content based on a Document Center or Records Center site template, as long as Microsoft TechNet guidance is followed. These limits will be addressed in further detail in the "Content Database Size Supported Limits" section later in this chapter. There has been a misconception that when binary content is externalized to a BLOB store, the storage size of the metadata in the content database was the only concern with respect to capacity planning. This is unfortunately not the case. However, there are many other benefits to implementing RBS:

- Large files can be "stored" in SharePoint without concern about content database inflation. This is particularly important for digital asset management (DAM) files such as large graphics, audio, or video files. Also, sites that might contain large Microsoft Word, Excel, or PowerPoint or PDF documents can benefit.

- RBS Provider solutions can implement advanced BLOB storage solutions that enable encryption for high-security implementations, or compression and de-duplication techniques that save disk space.
- Advanced RBS Provider solutions can implement performance tiers to offer the most efficient use of storage technologies. For example, frequently accessed content can remain on high-performance storage while rarely accessed content can be moved to lower-performing storage. In the long term, documents could even be archived to cloud storage.
- Less expensive storage can be used to store binary content, while more expensive, higher-performing storage can be used for the structured metadata in the content database.
- BLOBs can be stored on WORM devices; facilitating compliance regulations restrict document deletion policies. Some industry regulations do not allow documents to be deleted. So Write Once Read Many (WORM) CAS devices are deployed to prevent binary content from being deleted. RBS supports these devices.

When to Implement an RBS Solution

Clearly, RBS has several benefits — but as is often the case, there are trade-offs that must be considered:

- Disaster recovery procedures are more complicated and must be carefully planned.
- Additional administrative tasks are associated with maintaining BLOB stores and ensuring that orphan management is handled. In general, farm administration is significantly more complex.
- With RBS enabled, upgrade to the next version of SharePoint will be more complicated.
- RBS Provider feature processing can add latency when saving or retrieving documents to or from the BLOB store. For example, it takes longer to encrypt and compress a binary stream as it is being saved to a BLOB store than it would to just send the stream straight to the BLOB store. Encryption, compression, and other synchronous processing features can increase storage and retrieval latency.
- Implementing an RBS solution can be expensive. RBS Provider licensing and possibly SQL Enterprise licensing can add to the total cost of the project. SQL Enterprise licensing considerations will be explained further in the “SQL Server Licensing Considerations” section in this chapter.

With these potential concerns in mind, when does it make sense to implement RBS? RBS is most efficient in systems that store large documents or documents that are infrequently accessed, such as large-scale document archives or digital asset management (DAM) sites that contain very large documents. As for general guidelines, you can expect a performance benefit from implementing RBS in the following cases:

- One or more content databases are expected to be larger than 500GB.
- BLOBs are, on average, larger than 256KB.
- BLOBs are at least 80KB and the database server is a performance bottleneck. RBS reduces both the I/O and processing load on the database server.

There are additional scenarios that might also lend themselves to an RBS solution. Consider implementing RBS when the following applies:

- The reduced cost of lower-performing storage for BLOB data is more important than the minimal storage and retrieval latency added by enabling RBS.
- Regulatory compliance requires a WORM solution.
- The benefits of BLOB compression and/or de-duplication outweigh the administrative overhead of BLOB store management and disaster recovery planning.
- The 4GB (SQL Server 2008 Express) or 10GB (SQL Server 2008 R2 Express) database size limitation needs to be mitigated by externalizing BLOB data. It is possible to store nearly 1.5 million documents per content database in a Microsoft SharePoint Foundation 2010 solution when used in combination with SQL Server 2008 R2 Express and the FILESTREAM RBS Provider. This is particularly important when upgrading from SharePoint 2007 where the Windows Internal Database (WID) was used. WID is not available to SharePoint 2010 deployments, so larger content databases must be mitigated.
- The administrative overhead of creating multiple site collections and content databases for the containment of similar content is greater than the administrative overhead of BLOB store management and disaster recovery planning.

RBS Provider Options

Several third-party vendors offer RBS Providers either free or at a reasonable cost. Typically, the free providers offer basic functionality without cost or support but require additional licensing for integration with other “suite” components or advanced BLOB storage devices. The providers that are not free typically offer more features out of the box.

Microsoft has also created an RBS Provider called the FILESTREAM RBS provider. The FILESTREAM provider is a free solution but it has important licensing restrictions that are addressed later in this chapter. The FILESTREAM provider does not provide any of the more advanced encryption, compression, or de-duplication features that some of the third-party providers have, but it is a heavily tested solution and performs well.

The API for creating a custom RBS provider is freely available from Microsoft, but it is generally not recommended that organizations attempt to build their own custom provider.

While it is beyond the scope of this book to dive into the pros and cons of each third-party RBS provider, some important considerations must be addressed when you are evaluating any RBS provider:

- Does the provider properly participate in backup and restore operations?
- How well has the vendor tested the provider in disaster recovery and business continuity scenarios?
- What is the installation and administration experience like?

- How does the provider handle the externalization or migration of existing BLOB data?
- What is the latency impact of implementing the provider and each of its features?
- Does the provider implement any techniques that will threaten upgrading to future versions of SharePoint?
- What are the long-term administrative and maintenance costs of the provider?

Backup and Restore Considerations

Regardless of the provider chosen, a key principal must be followed with respect to backup and restore. During backup operations it is imperative that backups start on SQL Server content databases before they are started on the related BLOB store. Conversely, during restore, the BLOB store must be restored before the related SQL Server database is restored.

Following this procedure guarantees that end users can retrieve a document that appears to be available in SharePoint. If this procedure is not followed, then it would be possible for the metadata of a document to be added to a content database *after* the BLOB store backup has started. Then upon restore, the document would not be present in the BLOB store. This would result in a retrieval error when the end user attempted to open the document.

For some third-party vendors, the RBS provider may be just one part of a full backup and recovery suite of products. In this case, the order of backup and restore operations can be managed automatically. A simplified disaster recovery solution can be another reason to choose one RBS provider vendor over another.

SQL Server Licensing Considerations

Two key licensing requirements must be followed when considering the cost and architecture of implementing an RBS solution.

First, all third-party RBS providers are considered to be *remote* storage solutions. All remote storage solutions require all RBS-enabled content databases to be stored on a SQL Server 2008 R2 *Enterprise Edition* database server. This single licensing requirement can have a significant impact on solution deployment cost.

Second, SharePoint 2010 supports the implementation of what is known as the *local* FILESTREAM provider. The FILESTREAM provider is considered *local* when the FILESTREAM provider database resources and FILESTREAM file group are deployed in the content database. The FILESTREAM provider is considered *remote* when the FILESTREAM provider database resources and FILESTREAM file group are deployed in a database other than the content database, possibly on another SQL server. Implementing the remote FILESTREAM provider requires SQL Server 2008 R2 Enterprise Edition.

Both of these licensing requirements are very important and must be considered in light of cost and SQL Server storage architecture.

SHAREPOINT 2010 SCALABLE TOPOLOGY DESIGN

Now that storage architecture has been discussed in significant detail, it is time to move on to server topology. The number of users serviced, the number of documents or items contained, and internal or external processes that add load to SharePoint are all factors that determine when a small, medium, or large farm should be implemented.

Knowing the Users, the Corpus, and the Processes

Before deciding on a small, medium, or large server topology, it is important to understand the primary factors that affect performance:

- **Expected peak number of active concurrent users** — Knowing the peak number of active concurrent users that may touch the farm in some way at any given time helps to determine the number of web front-end servers needed to serve all users.
- **Expected number of content items to be stored in the farm** — The number of content items expected helps to determine the number of crawl servers needed to crawl all content.
- **Type of content stored in the farm** — Collaborative content tends to be Microsoft Office–type documents that can be crawled in order to build a robust content index. This type of solution requires the full array of query servers to support content index storage and query processing. Conversely, some document archives that consist of TIFF or other image-only PDF documents will only be crawled for their metadata. In the latter scenario, the content index will be very small and fewer query servers will be required for content index storage and query processing. However, in this scenario, a heavier load may be placed on property databases.
- **Purpose of the farm** — Is the farm serving in a collaborative capacity or is it being designed as a document archive system? The search subsystem is used very differently in a farm that is primarily used for document archive versus a farm that is primarily used for collaboration. In a document archive farm, fielded searches are usually just as important as free-text searches. This may necessitate separating the property database onto a separate SQL Server.

In a collaboration-type farm, free-text search is more important, so performance emphasis is on the query servers. This scenario typically includes numerous service applications, which create more load on application servers in a collaborative-type farm than in a document-archive type farm. Additional application servers may be needed to handle the service application processing load in a collaboration-type farm.

- **Availability requirements for the farm** — Is redundancy at all levels required? Extensive redundancy requirements are often part of the service-level agreement (SLA), causing a need for many additional servers.
- **External processes that will access or submit content to SharePoint** — Users aren't the only entities creating load in SharePoint. In many highly scaled document archive systems, there is often some form of high-volume content loading process. Just to provide an example of what SharePoint is capable of, if properly architected, SharePoint can consume literally millions of documents per day during migration and bulk load–type operations; but in order to do this, many web servers and database servers are necessary, as well as an extremely powerful storage subsystem.

Many other usage patterns and performance variables can affect the topology needs of a given SharePoint farm. In the end, no amount of planning can result in a perfectly architected topology the first time. The benefits of testing, regular monitoring, and frequent adjustments can't be underestimated. Fortunately, SharePoint 2010 provides extremely flexible and adjustable topology options to meet the requirements of virtually any solution.

Farm Size Definitions

While every farm will have different performance requirements, there are established farm sizing recommendations that can help provide the best starting point based on common factors. The fundamental factors of farm performance are as follows:

- **Latency** — The duration of time from when a consumer initiates an action until the last byte is transmitted to the client application.
- **Throughput** — The number of concurrent requests that a server or server farm is able to process.
- **Data scale** — The content size of the corpus that the system can host. The structure and distribution of content databases has a significant impact on the time it takes the system to process requests (latency) and the number of concurrent requests it can serve (throughput).
- **Reliability** — The ability of the system to meet the targets set for latency and throughput over time.

In order to achieve the established requirements with respect to these four fundamental factors, a physical architecture consisting of a small, medium, or large farm is required.

A *small farm* can usually be served by a *two-tiered* approach consisting of at least two web servers and a database server. One of the web servers will host the Central Administration site and perform application service processing, while the other serves content to end users. A small farm could be expected to support a system with either 10,000 to 20,000 users but with limited content capacity or a document archive system with relatively few users and from 5 to 10 million content items (see Figure 12-5).

A small farm can be scaled out to *three tiers* using a dedicated application server depending on the number of users, content items, and application services required. Adding the additional application server to a small farm allows for moderate service usage in a collaborative-type farm. However, it will not necessarily increase the number of documents that can be stored in a document archive-type farm (see Figure 12-6).

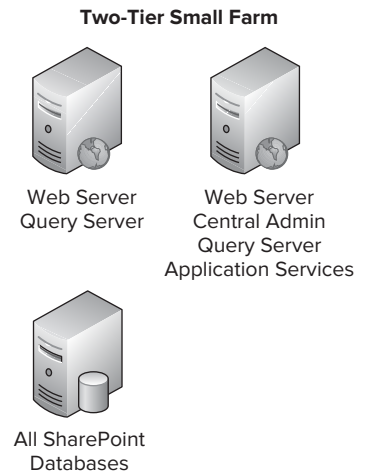


FIGURE 12-5

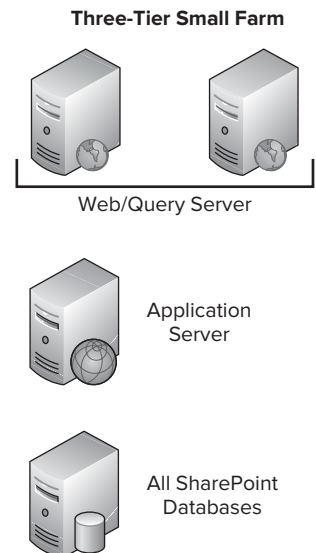


FIGURE 12-6

By segregating the search databases to a second database server in a small farm, the topology could be expected to support up to 10 million items, as well as 10,000 to 20,000 users (see Figure 12-7).

A *medium farm* can be served by a three-tiered approach that separates query and crawl servers from other application servers, while adding additional database servers to further segregate search databases and additional content database storage. A medium-size document archive farm could handle approximately 20 to 40 million documents. Additional web servers and application servers could also be added to support a larger user community and additional application services (see Figure 12-8).

A *large farm* consisting of 80 to 100 million documents supporting a collaborative solution is scaled out using the concept of server groups. The relevant server groups are as follows:

- Web servers for incoming requests (end users)
- Web servers dedicated to crawl processing and administration
- Crawl servers
- Query servers
- Application Services and Central Administration
- Sandbox solution code execution servers
- Database servers for search databases
- Database servers for content databases
- Database server for all other SharePoint databases

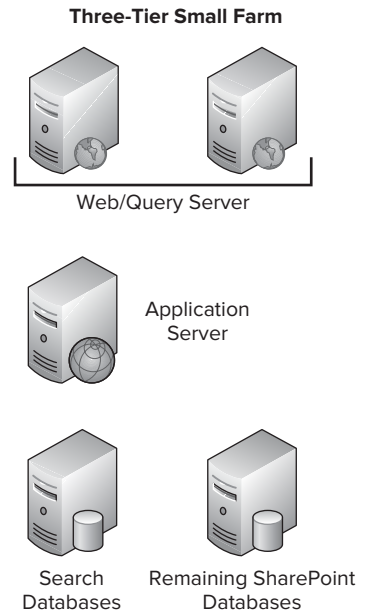


FIGURE 12-7

Microsoft recommends that any SharePoint solution containing more than 40 million documents should be serviced by a SharePoint search services farm. This enables the content farm servers to be dedicated to serving content and application services processing. However, when SharePoint 2010 is used as the platform for a large-scale document repository, many of the application services that might be found in a collaboration environment are often not necessary. For this reason, a composite large farm topology can be provided. The farm described in Figure 12-9 could be expected to support approximately 80 to 100 million documents.

SharePoint 2010 can also be extended to create what is known as an *extreme farm*. An extreme-scale farm is initially designed like a large farm; multiple content database servers and an extremely robust storage subsystem are implemented to support a repository of approximately 500 million documents. The difference is that the extreme-scale farm has only a few crawl/query servers that are used only for crawling profile data and/or relatively small quantities of collaborative documents. The remaining crawl and query services are provided by a dedicated FAST for SharePoint 2010 search services farm, which can be scaled out to support the vast repository.

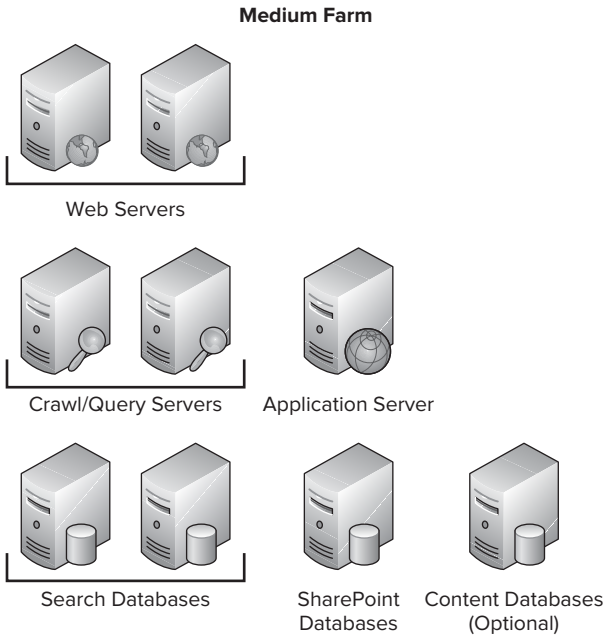


FIGURE 12-8

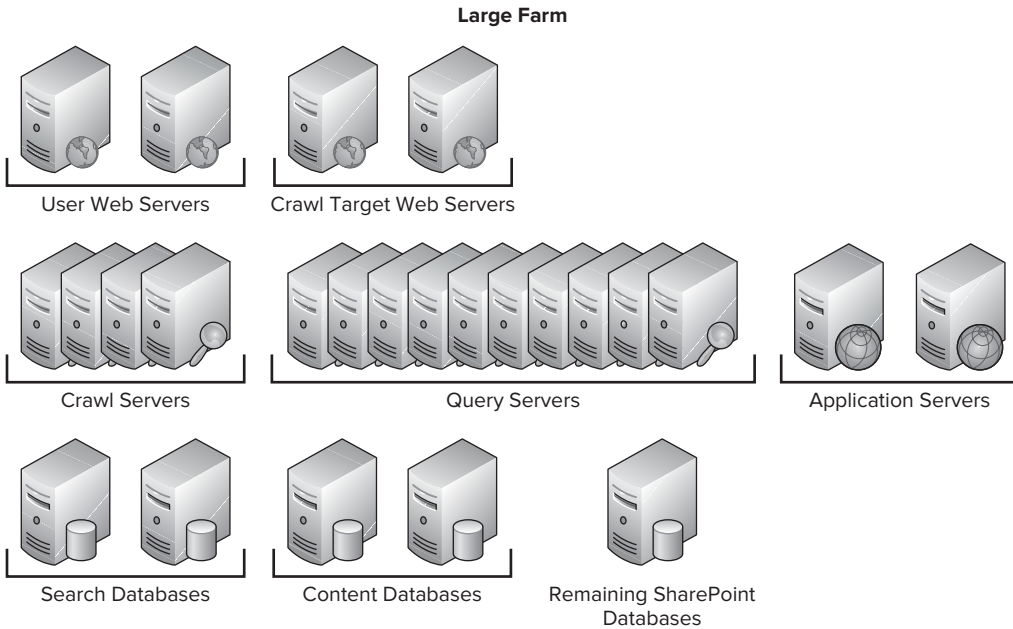


FIGURE 12-9

The Case for Additional Web Servers

Adding additional web servers obviously adds request capacity and redundancy to a SharePoint farm. Therefore, adding additional web servers may be necessary in the following circumstances:

- An additional web server is needed for redundancy requirements.
- The existing web server is overtaxed with user requests, causing additional latency. Before adding another web server for this reason, ensure that the latency is not due to database server resource constraints.
- Crawl operations are dragging down web server performance, causing additional latency for end user requests. In this scenario, the new web server should be configured as a dedicated crawl target.
- The existing web server is also hosting other application service roles and is being overtaxed. Adding an additional web server in this situation enables the separation of application service processing from web server end-user request handling.

The Case for Additional Application Servers

Adding additional application servers is commonly necessary. Because application servers perform such a wide variety of service application processing, any one of the service applications can bog down the performance of the application server. Here are a few examples:

- Excel calculations or other Office web application services may need a dedicated application server for processing, depending on usage patterns.
- Crawl processing is underperforming, resulting in crawls that run longer than expected. Adding additional crawl servers and crawlers may be the solution.
- Query servers may need to be added to handle additional index partitions or so that existing index partitions can be mirrored to facilitate fault tolerance.
- A high number of active workflows can drag down application server processing. Adding additional application servers increases workflow capacity.

The Case for Additional SQL Servers

Scaling out a farm to support more users, more processes, or more content often requires adding one or more database servers. It won't help to add additional web servers if there are resource

constraints in the database server(s). In many situations, adding additional database servers is the only way to reduce latency or add additional throughput or capacity. Here are a few examples:

- Crawl or query processing is interfering with end-user web requests. In this situation, it may be necessary to add an additional database server that is dedicated to the search service application databases.
- Crawl processing is reducing the latency of query execution. In this situation, it may be necessary to add an additional database server that allows for the separation of crawl databases and property databases.
- The property database server does not have enough resources to maintain 33% of the property catalog in memory, resulting in underperforming property-based queries. In this situation, the solution is to either add additional RAM to the database server or add an additional database server that is dedicated to the property database.
- Web Analytics service application processing is dragging down database resources and affecting farm responsiveness. In this situation, it may be necessary to add an additional database server that is dedicated to web analytics.
- The corpus is extremely large and the amount of data stored in the content databases is beginning to degrade document retrieval or crawl processing performance. In this situation, it may be necessary to add an additional database server that is dedicated to storage of new content databases.

SCALABLE TAXONOMY FACTORS

When designing a scalable taxonomy, always keep in mind the software boundaries of SharePoint 2010. All other taxonomy components and content processing techniques are implemented with respect to the boundaries, thresholds, and supported limits.

Boundaries are built-in limits that can't be exceeded due to design restrictions. An example of this is the 2GB document size limit. Under no circumstances is it possible to load a document into SharePoint that is larger than 2GB.

Thresholds are default values that can't be exceeded unless the value is modified. An example of this is the default document size limit. By default, the document size threshold is 50MB, but it can be raised to the boundary of 2GB.

Supported limits reflect tested values for a given parameter, and they represent known limitations of SharePoint. Exceeding supported limits can cause erratic or degraded behavior; and in some cases,

exceeding supported limits can be harmful. An example of a supported limit is the number of application pools per web server. The supported limit is 10. While it is technically possible to have more than 10, in many cases having more than 10 will degrade web server performance.

Microsoft uses a *graphic equalizer* as a metaphor for describing the relationship of thresholds and supported limits. By increasing the value of one limit, the effective value of another limit may be decreased. Consider the scenario in which one slider on the graphic equalizer represents the supported limit of 30 million items in a document library. This value is affected by another slider that represents the maximum size of documents in a farm, with a threshold default value of 50MB. If the maximum file size in the farm is set to 1GB to support very large files, then the number of documents that can be stored in a document library is significantly reduced.

Although it is beyond the scope of this book to cover all the software boundaries, thresholds, and supported limits of SharePoint 2010, the following factors greatly affect SharePoint's ability to act as a large-scale document management system or archive:

- **Content database size** — The supported limit is 4TB for collaborative-type data as long as related Microsoft TechNet guidance is followed. This particular supported limit can be raised or lowered depending on additional factors that are described in the “Content Database Size Supported Limits” section later in this chapter.
- **List view threshold** — The default threshold is 5,000. While this can be adjusted, it is not recommended. This threshold ultimately drives the maximum number of documents that should be stored in a parent container such as a document library folder.
- **Documents in a library** — The maximum supported limit is 30 million, but the actual limit may be dictated by the crawl database items maximum supported limit of 25 million items.
- **Indexed Items** — The maximum supported limit of items that can be indexed by a search service application is 100 million, with a related limit of 10 million items per index partition.

Content Organization and Scalable Taxonomy

The best way to scale up the number of documents that can be contained in a document library is to organize content into folders. For high-volume loading, a date-and-time-based folder structure can be used to ensure that fewer than 5,000 documents are always loaded into a given parent container. However, caution must be taken with this approach. Nesting the folders too deeply, such that there are only a few documents per folder, can have a negative impact on crawl performance.

Microsoft provides an additional solution for building a scalable folder structure. The *Content Organizer* is a powerful tool that can facilitate a scalable taxonomy. The Content Organizer is an evolution of the record-routing capabilities of the Records Center, first introduced in SharePoint 2007. The Content Organizer can route documents to a library in the current site or to another *Send To Connection* site. Two other new features in SharePoint 2010 also assist in content organization:

- **Content Type Syndication** — This is a feature provided by the Managed Metadata application service. It enables one site collection to be configured as a *content type hub* such that

other site collections can subscribe to that hub. Content types are then synchronized from the content type hub site to the subscribing sites.

- **Send To Connection** — Any site that has the Content Organizer enabled can be configured (in Central Administration) as a Send To Connection. Send To Connections can serve as destination locations for other Content Organizers.



The Content Organizer is discussed further in Chapter 8.

An Exercise in Scalable Taxonomy Design

Consider this scenario: A large product distribution company has a legacy document management system that contains approximately 40 million documents. Due to degrading performance and high maintenance costs, the company is in need of a new document management solution. The company already has an enterprise agreement with Microsoft that allows them to use SharePoint 2010 as the document management platform.

Because of the high volume of transactions and the large product line, the solution needs to handle 35,000 new documents per day, which will be loaded into SharePoint via high-volume scanning processes. The solution requirements are as follows:

- SharePoint 2010 will serve as the solution platform.
- Long-term administrative overhead needs to be minimized.
- Documents need to be maintained in the system for a minimum of seven years.
- End users should be affected as little as possible with respect to taxonomy-driven scalability concerns or taxonomy changes.

Given these requirements, the company chooses a hybrid medium-to-large SharePoint farm topology and purchases appropriate servers and storage subsystems. The following architecture could be implemented.

A primary site collection is created with all of the site columns and content types necessary to support the wide array of document types that will be loaded into SharePoint. The Content Organizer feature is enabled, which automatically creates a Drop Off Library where all users and processes will send documents. It is also configured to be a content type hub.

Multiple data storage site collections are created. One of the document types, an accounts payable invoice, accounts for 70% of all documents. It is projected that there will be nearly 10 million accounts payable invoices every 18 months. It is decided that five content databases, each with a single library in a single site, are created for the sole purpose of containing all the accounts payable invoices for a seven-year period. Two additional content databases are created for the site collections; these will contain the documents that will be added in the seven-year period.

Each data storage site collection is configured to subscribe to the content type hub, which synchronizes all the site columns and content types from the primary site collection. In each data storage site collection, libraries are created to contain each of the different document content types.

The Content Organizer feature is also enabled in each data storage site collection. Routing rules are created and used to shuffle documents of a given content type to their related document library. The Content Organizer is configured to automatically create folders such that a maximum of 5,000 documents are allowed to exist in each parent container. Each data storage site collection is configured as a Send To Connection in Central Administration, enabling it to serve as a destination site collection.

In the primary site collection, routing rules are created and used to route documents to appropriate Send To Connection site collections based on content type. This enables all end users and document upload processes to send content to a single Drop Off Library for routing purposes. Initially, the Accounts Payable Invoice content type is routed to the first of five Send To Connection site collections.

After 18 months, the first Accounts Payable Invoice data storage site collection is full. An administrator simply edits the Accounts Payable Invoice routing rule in the primary site collection and points the routing to the second data storage site collection. End users are not affected and document loading processes don't have to be reconfigured because the primary site collection Drop Off Library path has not changed. This pattern continues for seven years until all Accounts Payable Invoice data storage site collections are full.

This solution enables more than 65 million documents to be loaded into SharePoint in seven years — with virtually no impact on end users or document upload processes. Administrative overhead is minimal with respect to any maintenance necessary to support the scalable taxonomy implementation.

This solution can be further enhanced in several ways. The Content Organizer timer job can be configured to run more frequently so that document routing is closer to real time. Workflows can be assigned to individual document libraries to facilitate additional processing on documents after they have been loaded. Information management policies can be implemented to delete content or launch expiration workflows in order to limit retained document liability. A migration effort could be undertaken to move all the legacy documents into the new repository.

This is just one demonstration of the many ways in which SharePoint 2010 can be configured to manage a very large number of documents.

Content Database Size Supported Limits

In the days of SharePoint 2007, the 100GB maximum database size “limit” was often misunderstood by SharePoint administrators because of confusing guidance issued by Microsoft. Some of this confusion has carried over to SharePoint 2010 guidance. Even though the software boundaries described on TechNet have been more consistent, in some cases there is still room for subjective interpretation.

The crux of the newest recommendation is two-fold:

- The maximum size of the content database is limited by the speed with which the database can be restored upon failure, based on SLA requirements.
- The maximum size of the content database is also limited by the acceptable latency allowed for content retrieval. Directly affecting latency are the number of concurrent user requests combined with the amount of content stored in a content database.

For example, collaborative data is likely to be frequently accessed by end users. Therefore, for content databases containing one or more site collections that primarily consist of collaborative data, Microsoft recommends a maximum content database size of 4TB. This value is based on performance testing, but of course it is subject to empirical evidence from actual testing on the real-world farm deployment. However, Microsoft only supports collaboration content databases larger than 200GB (up to 4TB) if disk subsystem performance supports at least .25 IOPS per GB and the organization has developed plans for high availability, disaster recovery, future capacity, and performance testing. Unless a specific support waiver is granted, Microsoft will not support collaboration type content databases that are larger than 4TB until the size of the content database is reduced to 4TB or less by either deleting content or moving sites to another content database.

In another scenario, a content database containing only a single document archive–type site collection that is rarely accessed will experience very little end user pressure. This site collection will typically be a Document Center or a Records Center. In this scenario, Microsoft does not place a specific limit on the size of the content database as long as TechNet guidance for content database size in a document archive scenario is followed. In addition to the guidance content databases up to 4TB, TechNet specifies guidance that supports document archive type content databases to scale beyond the 4TB limit. Again, this recommendation is subject to empirical evidence from actual testing on the real-world farm deployment. Keep in mind that it takes a very long time to back up and restore even a 4TB content database. If backup and restore operations cannot be completed fast enough to comply with SLA requirements, then the actual maximum content database size would be smaller.

PERFORMANCE AND RESOURCE THROTTLING

SharePoint 2010 provides both new and existing capabilities for ensuring that end users are not affected by performance bottlenecks. While BLOB and object caching, as well as site collection quotas, continue to facilitate low latency and maintain farm scalability, SharePoint 2010 introduces resource throttling capabilities based on specific thresholds to ensure that end user actions are not allowed to degrade farm performance.

Resource throttling thresholds are managed by web application in Central Administration. In the Central Administration home page, click “Manage web applications” under the Application Management section. A list of web applications will be displayed. Select the appropriate web application. Then, on the Ribbon’s Web Applications tab, select General Settings ⇄ Resource Throttling. A series of thresholds and configuration options for the selected web application are presented. The following thresholds are relevant to maintaining farm responsiveness:

- **List View Threshold** — Represents the maximum number of items that a database operation can include at one time. If the operation exceeds this value it will not be allowed.
- **Object Model Override** — Determines whether or not object model calls are allowed to programmatically override the List View Threshold for a given query.
- **List View Threshold for Auditors and Administrators** — Represents the maximum number of items that a database operation can include when the user is an Auditor or Administrator. The concept here is that Auditors and Administrators understand that large list queries can affect other end users and additional care will be taken.

- **List View Lookup Threshold** — Represents the maximum number of Lookup, Person/Group, or workflow status fields that a database query can include at one time. These operations involve additional database joins or make additional use of TempDB and can thus affect farm responsiveness.
- **Daily Time Window for Large Queries** — Specifies a time when large queries may be executed. This allows external processes or regular users to perform unrestricted queries during off hours when other users are not likely to be affected.
- **List Unique Permissions Threshold** — Specifies the maximum number of unique permissions that a list can have at one time
- **HTTP Request Monitoring and Throttling** — Enables a job that monitors web server performance. When web servers become overloaded, request throttling can block new requests to minimize overload, which allows existing requests to complete.

Resource throttling is an important mechanism for short-circuiting end user requests that can have a significant impact on other end users.

SUMMARY

In this chapter, you learned the important architectural concepts needed to lay a foundation for a highly scalable SharePoint 2010 ECM farm deployment. The key takeaways are as follows:

- Storage architecture is the key to SharePoint scalability. A high-quality storage subsystem helps to ensure that request latency is minimized and RPS throughput is maximized.
- Remote BLOB Storage can have a tremendous positive impact on SharePoint ECM scalability but it can add administrative overhead.
- SharePoint 2010 provides for an extremely flexible topology that enables ECM solutions ranging from just a couple hundred thousand documents to hundreds of millions of documents.
- Thanks to significant new features in SharePoint 2010 such as the Content Organizer, Content Type Syndication, and Send To Connections, a self-managing and highly scalable ECM solution can be architected.
- SharePoint 2010 exposes new resource throttling techniques that can short-circuit user requests that would otherwise have a negative impact on farm performance.

PART III

SharePoint ECM Support Concepts

- ▶ **CHAPTER 13:** ECM File Formats
- ▶ **CHAPTER 14:** The SharePoint ECM Ecosystem
- ▶ **CHAPTER 15:** Guidance for Successful ECM Projects

13

ECM File Formats

WHAT'S IN THIS CHAPTER?

- Working with living documents
- Installing and configuring Office Web Apps
- Archiving living documents
- Searching archived documents

You've made it past the main course and into dessert. Our hope is that you'll enjoy this part of the book as much as you do apple pie or whatever your favorite binge happens to be. Although you've made it past some of the specific ECM feature-focused categories of the book, this chapter should prove valuable as we dive into the ECM file formats.

ECM documents can typically be grouped in one of two categories: either *living* or *archived*. The differences between the two, knowing when to move from living to archived, and knowing how to convert documents for long-term preservation (remember the ECM components from Chapter 1?) are important aspects of ECM processes. By the end of this chapter, you should be comfortable with these concepts.

Another component of ECM is delivery, and this chapter discusses searching and viewing possibilities for the different document formats. Understanding iFilters for archived documents is also important so that users can find these documents. The new Office Web Apps are part of the viewing strategy, so this chapter briefly covers them too, discussing both installation and configuration options.

IT'S ALIVE — YOUR DOCUMENT, THAT IS

Okay, it may not be literally living, but those documents that move throughout your ECM system, being edited, getting approved, and sometimes even (paperless office forbid) being converted to paper and ink, are alive to your organization. Just like living organisms, they can grow and are continuously changing, just like a life form.

This section takes a look at the most well-known formats being used in ECM systems today, focusing on SharePoint, of course, and provides some ideas about how to use them to make your SharePoint ECM system an even more productive platform for your user base.

Microsoft Office Formats

Since SharePoint evolved from the Microsoft Office product line, it seems logical that the most common documents stored in SharePoint are the familiar Microsoft Office formats. These Microsoft Office formats come in two varieties, the Microsoft Office Binary formats (doc, xls, ppt) and the newer Open XML formats (recognized by the “x” on the end of the file extensions). Each format has its distinct advantages, although it would be hard to argue against the Open XML formats with today’s available technology.

SharePoint 2010 also has new features that take advantage of the Microsoft Office Formats, and the following sections examine these file types and some of these new features that make them easier to work with.

Microsoft Office Binary

The Microsoft Office Binary formats are normally considered the DOC, XLS, and PPT documents that most of us have worked with for the past 15 years or so. This format has evolved considerably over the years, normally being updated along with the version of Microsoft Office that includes it.

In early 2008, in an apparent attempt to be seen as a good community player, Microsoft published the file formats for binary files, letting everyone in the world know how to work with these files. However, by 2008 the Open XML formats were already published, so there wasn’t really any reason to keep the binary formats confidential any more.

It is outside the scope for this book to cover details about the specification or the updates being made to it. Microsoft, however, does publish the specification on its website.

Because many users are still using older Windows versions and older Office versions, backward compatibility drives the use of the Microsoft Office Binary formats today. Although Microsoft released the “Microsoft Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats” to enable Office 2003 users to open and save Open XML formats, Microsoft Office documents in the older binary formats are still common in the world today.

One may wonder why those hard to understand binary formats were the norm for so long and why they are still popular for older machines. It's because the greatest asset those formats had was that they were built to be manipulated quickly on older computers. Accessing binary files is many times faster than bouncing through XML nodes with .NET streams. Imagine parsing through a Word document of several hundred pages in XML format on a pre-Pentium PC with less than a megabyte of RAM.

In other words, even though we have binary format specifications, compatibility with most of the world's Office documents, faster operations, and guaranteed backward compatibility, using the new Open XML formats, described next, is preferable for any solutions development you may need to do.

Office Open XML

After a somewhat negative look at the binary formats for Office, this section turns to its replacement: the Office Open XML (also known as OOXML and just Open XML) format introduced by Microsoft with the Office 2007 and SharePoint 2007 releases. The previous section wasn't really meant to bash the old binaries, but rather to set up the freshness of the newer XML-based files.

As Microsoft realized a need to standardize their file formats for Office, they recognized an entirely new format was necessary. The new file format needed to be easier to understand, to edit, and of course to be available for the standards organizations such as ECMA and ISO. The by-product of the need for being easier to understand is a format that is easily interrogated. What we got was several new formats that are easily recognized by the 'x' on the end their filename extension.

These newer DOCX, XLSX and PPTX files have been very cool and fun since they hit our desktops. One of the first tricks we learned when beta testing Office 2007 years ago was renaming the files with .ZIP extensions and opening them for viewing like any other ZIP file. Give it a try. As shown in Figure 13-1, you'll find some XML representing the markup, fonts, and styles — as well as any images or other resources that may be used.

Developing Open XML

What makes the Open XML format especially appealing is the developer experience. Rather than dig through those binary specifications mentioned earlier, developers can access a complete SDK built around Open XML. The SDK is currently in version 2.0 and can be downloaded from the Microsoft public website. Using the classes and tools provided in the SDK, developers can create client or server solutions for scenarios of all kinds — from basic document creation to complex security and information protection.



The Open XML SDK 2.0 for Microsoft Office can be downloaded from
www.microsoft.com/downloads/en/details.aspx?FamilyID=c6e744e5-36e9-45f5-8d8c-331df206e0d0.

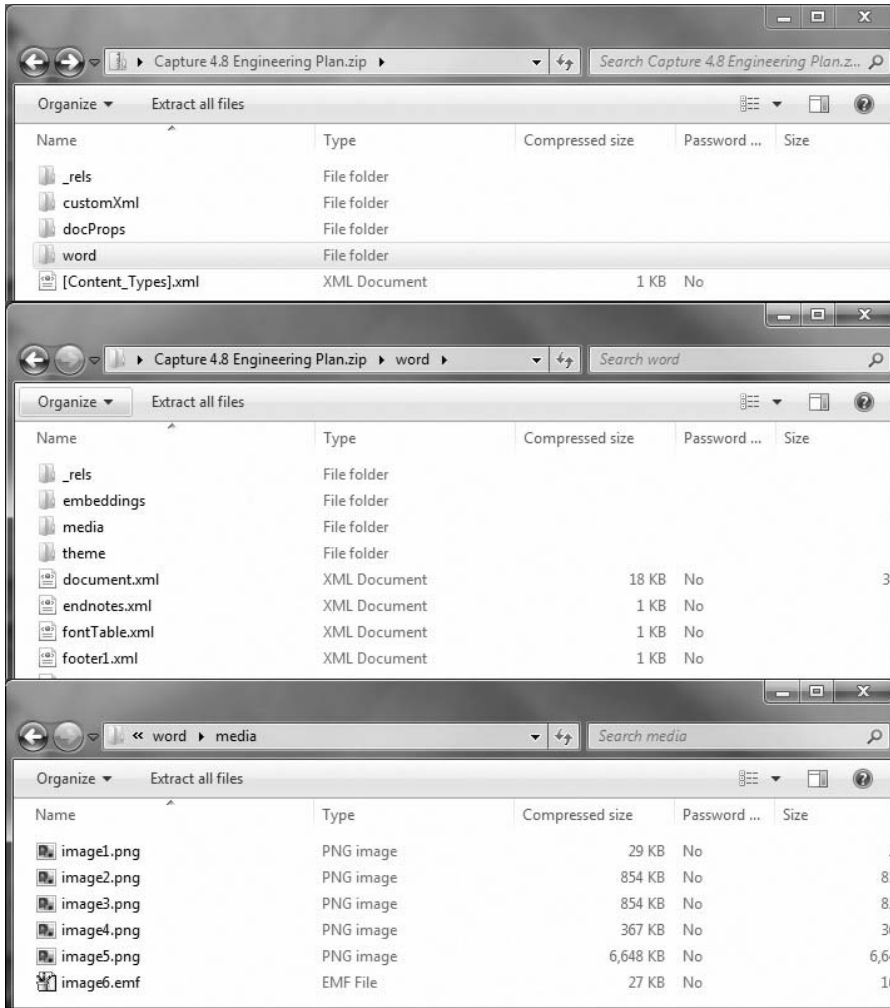


FIGURE 13-1

For development on Microsoft Word specifically, the Open XML SDK also has a good companion in the Word Automation Services you will read about later in this chapter. Combining the two APIs, you can accomplish almost any type of manipulation on Open XML–based Word documents.

Although the MSDN site is a great place for all things Microsoft, many other developer resources are available. One of the best resources for developing with Open XML is openxmldeveloper.org, a free, public community for sharing information and code about the format.

Standardization

Open XML was standardized first by ECMA (European Computer Manufacturers Association), and this standardized version is the format implemented in both Office 2007 and 2010. Although

Open XML has also formally achieved international standardization by the ISO and IEC, no current Microsoft Office product currently supports these versions.

Viewing and Editing Microsoft Office Formats with Office Web Apps

A major addition to SharePoint 2010 is the new Office Web Apps. These web applications allow Word, Excel, PowerPoint, and OneNote documents to be opened and edited directly inside the browser. Office Web Apps work with the newer Open XML formats as well as the older binary formats.

If your organization participates in the volume license program, you can download the Office Web Apps from the Volume License Service Center at www.microsoft.com/licensing/servicecenter/. A license key from the same site will also be necessary.



Only organizations with volume licensing for Microsoft Office 2010 can run Office Web Apps on-premise. Only users licensed through this model can legally use these applications when installed on either Microsoft SharePoint Foundation 2010 or Microsoft SharePoint Server 2010.

Installation

Office Web Apps are supported on both Windows SharePoint Foundation 2010 and SharePoint Server 2010. Unfortunately, they are not supported on Windows 7 (although “hacks” may exist). The most important thing to understand about installing Office Web Apps on SharePoint server farms is that the Office Web Apps must be individually installed on each server in the farm, although they do not need to be activated on each server. Standalone installations work either with or without a domain controller.



If you are installing Windows SharePoint Foundation and you would like to use Search Server Express, install the search software first, as you cannot do so after the Office Web Apps are installed.

Running through the setup of Office Web Apps is straightforward. After launching the setup, entering your product key, and accepting the license terms, you will be asked to choose file locations. The second option on this page, and normally the only one enabled, is the only decision you have to make during setup, and one you should carefully consider. As shown in Figure 13-2, here you can set the location of search index files. Because these files can be very large, you need to ensure that they are in a location with enough hard disk space.

Configuration

After the installation is completed you are prompted to run through the SharePoint 2010 Products Configuration Wizard. If this completes successfully, the services should start automatically (see Figure 13-3) and the service applications should be created (see Figure 13-4).

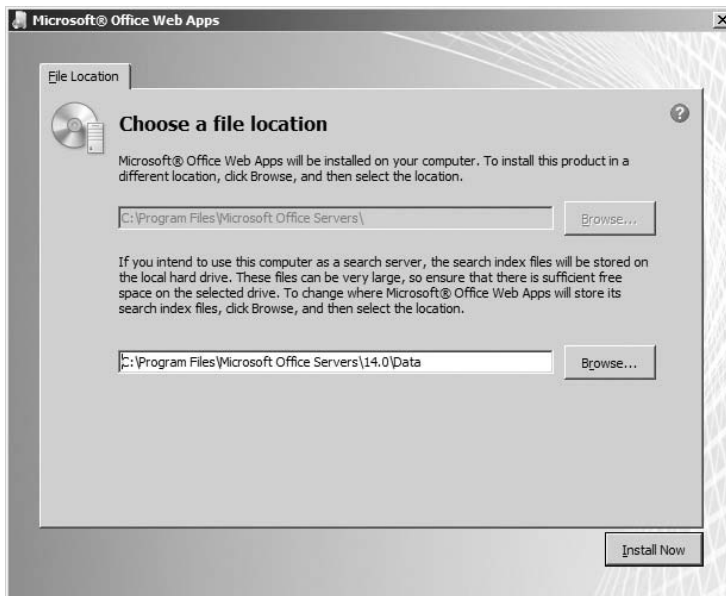


FIGURE 13-2

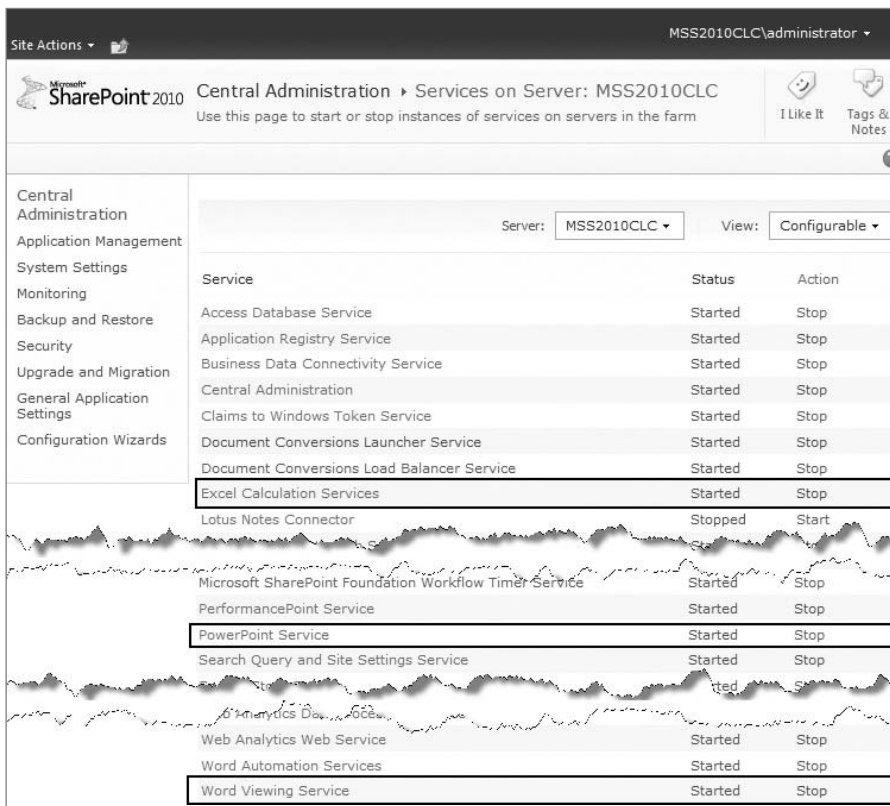


FIGURE 13-3

| Central Administration | Name | Type | Status |
|------------------------------|--|---|---------|
| Application Management | Access Services | Access Services Web Service Application | Started |
| System Settings | Access Services | Access Services Web Service Application Proxy | Started |
| Monitoring | Application Discovery and Load Balancer Service Application | Application Discovery and Load Balancer Service Application | Started |
| Backup and Restore | Application Discovery and Load Balancer Service Application Proxy_c5a8bfab-b84b-4cc4-b847-69976ae5fe4a | Application Discovery and Load Balancer Service Application Proxy | Started |
| Security | Application Registry Service | Application Registry Service | Started |
| Upgrade and Migration | Application Registry Service | Application Registry Proxy | Started |
| General Application Settings | Business Data Connectivity Service | Business Data Connectivity Service Application | Started |
| Configuration Wizards | Business Data Connectivity Service | Business Data Connectivity Service Application Proxy | Started |
| | Excel Services Application | Excel Services Application Web Service Application | Started |
| | Excel Services Application | Excel Services Application Web Service Application Proxy | Started |
| | Managed Metadata Service | Managed Metadata Service | Started |
| | Managed Metadata Service | Managed Metadata Service Connection | Started |
| | PerformancePoint Service Application | PerformancePoint Service Application | Started |
| | PerformancePoint Service Application | PerformancePoint Service Application Proxy | Started |
| | PowerPoint Service Application | PowerPoint Service Application | Started |
| | PowerPoint Service Application | PowerPoint Service Application Proxy | Started |
| | Search Administration Web Service for Search Service | Search Administration Web Service | Started |
| | Word Automation Services | Word Automation Services Proxy | Started |
| | Word Viewing Service | Word Viewing Service Application | Started |
| | Word Viewing Service | Word Viewing Service Application Proxy | Started |

FIGURE 13-4

The PowerPoint service application and the Word Viewing Service both have specific settings for the Office Web Apps, while OneNote does not have a specific service application and does not have any configuration options. Excel Services was available in previous version of SharePoint, and the settings are specific for those tasks; they are not related to the Office Web Apps.

As show in Figure 13-5, you can activate two site collection features to control how the Office Web Apps interact with users.

If both features are deactivated, documents are opened in the web applications only when the browsing device does not have Microsoft Office installed. If the Open Documents in Client Applications by Default option is activated, all Word, Excel, PowerPoint, and OneNote files will be opened in the web applications by default. They can still be downloaded and opened in Microsoft Office applications locally.

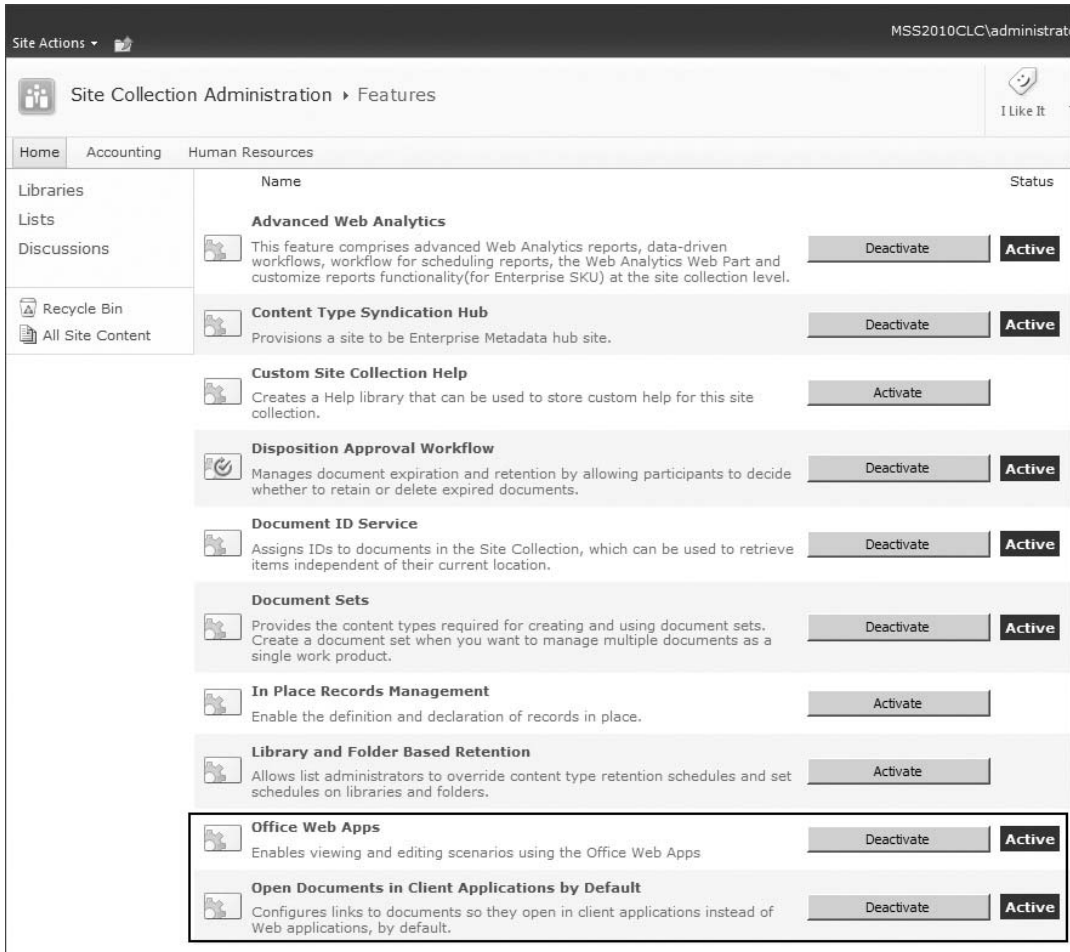


FIGURE 13-5

When the Office Web Apps option is activated, the default action on links will not change, but new menu items (see Figure 13-6) will be available to specify whether to open or edit in the web applications or in Microsoft Office on the requesting machine.

Word Automation Services

Microsoft introduced a new conversion technology for SharePoint Server 2010 called Word Automation Services. This service runs unattended on SharePoint Servers and is meant for rendering Word documents into different formats. The service itself is a new service application that can be turned on and off through Central Administration. As shown in Figure 13-7, the configuration dialog for the service application contains several options that administrators can change to tweak the service.

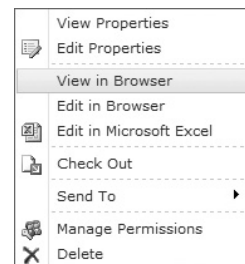


FIGURE 13-6

| | |
|---|---|
| <p>Supported File Formats</p> <p>Specify which file formats can be opened by the service application. If a file type is unchecked, users of this service will receive an error when attempting to open files of that type.</p> | <p>Supported File Formats:</p> <p><input checked="" type="checkbox"/> Open XML Document (.docx, .docm, .docx, .dotm)</p> <p><input checked="" type="checkbox"/> Word 97-2003 Document (.doc, .dot)</p> <p><input checked="" type="checkbox"/> Rich Text Format (.rtf)</p> <p><input checked="" type="checkbox"/> Web Page (.htm, .html, .mht, .mhtml)</p> <p><input checked="" type="checkbox"/> Word 2003 XML Document (.xml)</p> |
| <p>Embedded Font Support</p> <p>To preserve visual fidelity across different machines, a user can choose to embed fonts within the document. Use this setting to determine whether or not embedded fonts are used when converting documents.</p> | <p>Disable embedded fonts?</p> <p><input type="radio"/> Yes</p> <p><input checked="" type="radio"/> No</p> |
| <p>Maximum Memory Usage</p> <p>Specify the maximum percentage of system memory made available to the service application, as a percentage of overall system memory.</p> | <p>Percentage of memory:</p> <p><input type="text" value="100"/></p> |
| <p>Recycle Threshold</p> <p>Specify the number of documents converted by a conversion process before it is restarted.</p> | <p>Recycle threshold:</p> <p><input type="text" value="100"/></p> |
| <p>Word 97-2003 Document Scanning</p> <p>To provide added security when loading Word 97-2003 documents, extra checks are performed before those documents are opened. These checks can have an impact on overall service performance.</p> <p>Only disable this setting if you trust ALL documents loaded by the service application.</p> | <p>Disable Word 97-2003 Document scanning?</p> <p><input type="radio"/> Yes</p> <p><input checked="" type="radio"/> No</p> |
| <p>Conversion Processes</p> <p>Specify the number of conversion processes created on each server available to the service application. The number of conversion processes is equivalent to the number of conversions that can be performed simultaneously.</p> | <p>Conversion processes:</p> <p><input type="text" value="1"/></p> |
| <p>Conversion Throughput</p> <p>Specify the frequency with which groups of conversions are started, and the number of conversions within each group. Setting these values too high or too low can affect performance.</p> | <p>Frequency with which to start conversions (minutes):</p> <p><input type="text" value="15"/></p> <p>Number of conversions to start (per conversion process):</p> <p><input type="text" value="300"/></p> |
| <p>Job Monitoring</p> <p>Specify the length of time before conversions are monitored and, if necessary, restarted.</p> | <p>Length of time before conversion status is monitored (minutes):</p> <p><input type="text" value="5"/></p> |
| <p>Maximum Conversion Attempts</p> <p>Specify the maximum number of times a conversion is attempted before its status is set to Failed.</p> | <p>Maximum conversion attempts:</p> <p><input type="text" value="2"/></p> |

FIGURE 13-7



Word Automation Services uses SharePoint 2010 Standard or Enterprise; it will not work with SharePoint Foundation.

The service can read from several different Word formats, including, obviously, Open XML and the binary format, but also RTF and MHTML. The output format can be any of the following:

- PDF
- XPS
- Office Open XML (DOCX, DOCM)
- Binary Word (DOC)
- RTF
- MHTML
- Word XML

The only downside for administrators is that development is required to use this service, as no out-of-the-box functionality is provided. Of course, this is an upside for developers, who now have new APIs in their box of toys. Unfortunately for both parties, SharePoint only includes a service for Word documents, not the other Microsoft Office file formats — not really a complaint, but hopefully a bug in the ear of some product managers in Redmond.

Developing with Word Automation Services

The Word Automation Services programming model was created mainly for the conversion of Word documents to and from other formats; and it is meant to work in conjunction with the Open XML SDK, not to replace it. Think of the Open XML SDK as way to manipulate documents, and Word Automation as a way to convert them.

As a developer, the most interesting thing to know about the service is that even though it resides within SharePoint, it requires very little knowledge of SharePoint development in order to use it. The service itself is running inside SharePoint, but you don't necessarily have to be in SharePoint (Web Parts, web page, workflow, etc.) to use it. Applications completely unrelated to SharePoint can use the service for document conversion, with one caveat: the files to convert from and the files to convert to must reside within the same SharePoint farm as the service itself. Also, because it uses assemblies located on the SharePoint server, the code needs to be executed there.

One of the most common scenarios for the conversion service is the need to convert older binary-formatted word document (DOC) files to Open XML (DOCX). The following steps will take you through the creation of a new C# application in Visual Studio 2010 that converts binary Word documents in one SharePoint document library to Open XML documents in another:

1. Create a Windows Forms application in Visual Studio by creating a new Windows Forms Application project (see Figure 13-8).



Because SharePoint 2010 is built on .NET Framework 3.5, make sure you target 3.5 when starting your Visual Studio 2010 project. In addition, SharePoint Server applications must target Any CPU, rather than the default of x86.

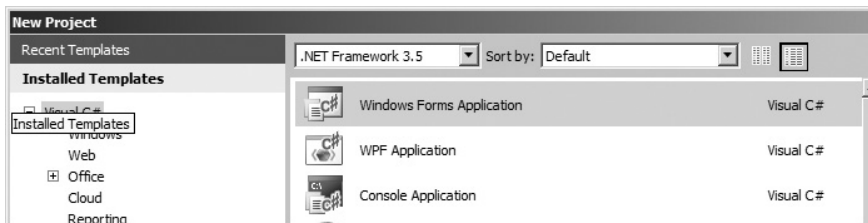


FIGURE 13-8

2. In order to use any of the Word Automation Services functionality, you need to reference the `Microsoft.Office.Word.Server.dll` in your Visual Studio project. The assembly

is located in the ISAPI directory of the 14 hive (C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\ISAPI), as shown in Figure 13-9.

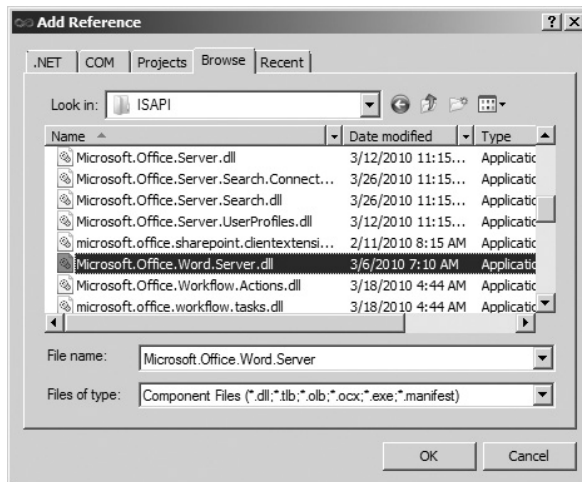


FIGURE 13-9

3. This scenario will also use a reference to SharePoint document libraries as the source and destination, so you need to add a reference to `Microsoft.SharePoint.dll` from the same ISAPI described in step 2.
4. Create a simple form with one button that will start the conversion (see Figure 13-10).

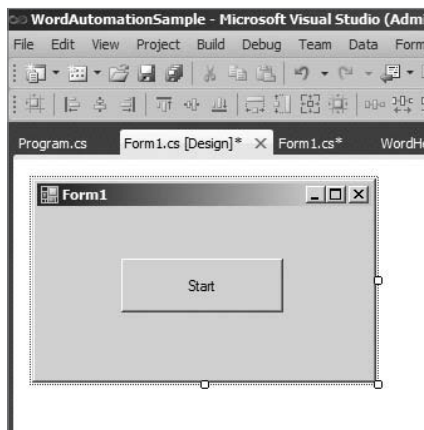


FIGURE 13-10

5. In order to check the conversion status later, create a class called `WordHelper.cs` as shown in Listing 13-1. This contains the code necessary to create a conversion job, specify job settings, tell the job where to find the files to convert, and start the job.



Available for
download on
Wrox.com

LISTING 13-1: Word Automation Sample Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Office.Word.Server.Conversions;
using Microsoft.SharePoint;

namespace WordAutomationSample
{
    class WordHelper
    {
        ConversionJob myJob;

        public ConversionJob MyJob
        {
            get { return myJob; }
            set { myJob = value; }
        }

        public void Convert()
        {
            //Create a instance using your Word Automation Serviced name
            myJob = new ConversionJob("Word Automation Services");

            //Set the conversion properties
            myJob.Settings.OutputFormat = SaveFormat.Document; //DOCX
            myJob.Settings.OutputSaveBehavior = SaveBehavior.AlwaysOverwrite;

            //Establish the SharePoint Server, Site
            //and Document Libraries to use
            SPSite mySite = new SPSite("http://CLCWS2008Dev");
            SPList myInputList = mySite.RootWeb.Lists["Input"];
            SPList myOutputList = mySite.RootWeb.Lists["Output"];

            //Give the Job the same (default) credentials as the site
            myJob.UserToken = mySite.UserToken;

            //Add the Document Library to the Conversion Job
            myJob.AddLibrary(myInputList, myOutputList);

            //Start the job
            myJob.Start();
        }
    }
}
```

6. Open the form created in step 4, double-click the button, and add code to the click event on the button to run the conversion:

```
private void button1_Click(object sender, EventArgs e)
{
    WordHelper helper = new WordHelper();
    helper.Convert();
}
```

- That's all the code needed, but you need to prepare a couple of document libraries, as referenced by the `Convert` method. As the code in Listing 13-1 shows, it is assumed that the root site is being used, so create libraries called `Input` and `Output` as shown in Figure 13-11.

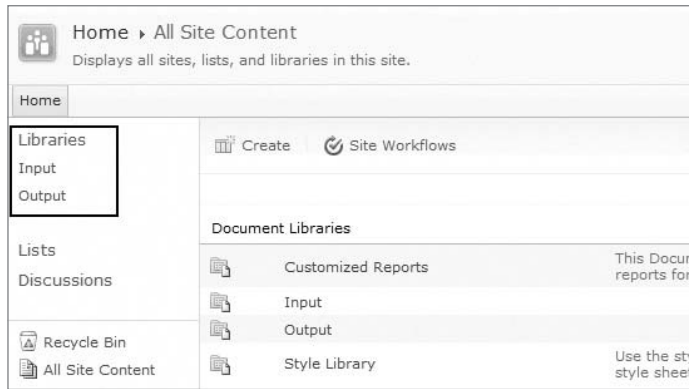


FIGURE 13-11

- Add at least one Word document saved in the binary format (DOC) to the `Input` library (see Figure 13-12).

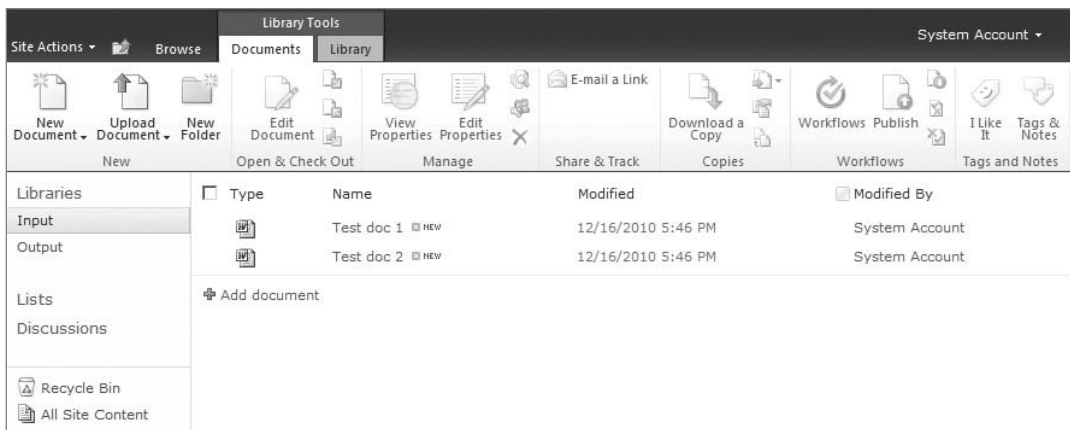


FIGURE 13-12

- Launch your application. Because conversion is happening unattended on the server, it may not happen immediately upon completion of code execution.

At this point you are possibly thinking how much better this would be if you had some idea of when (or if) the conversion was actually complete, as shown in Figure 13-13. The next set of steps add the capability to monitor the job by simply clicking a button to query for the status.

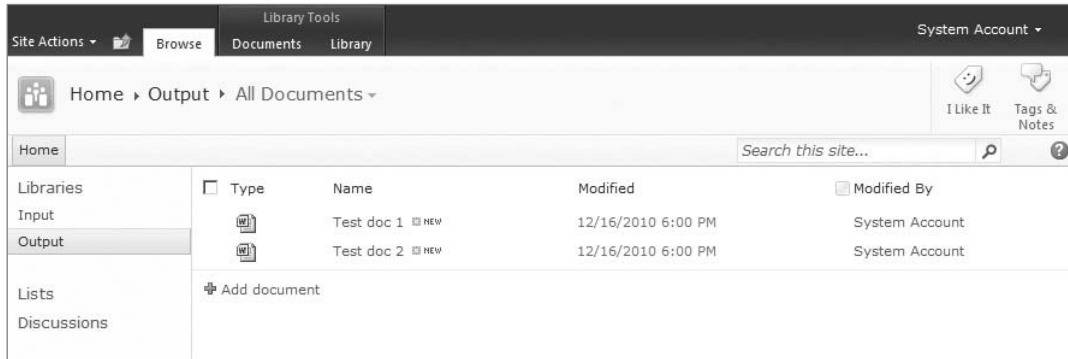


FIGURE 13-13

Recall in Listing 13-1 that you created a class variable called `myJob`. You used a class in the example so this variable could be converted to a property and used to monitor the state of the conversion process.

1. Encapsulate this variable by adding the following code to the `WordHelper` class:

```
public ConversionJob MyJob
{
    get { return myJob; }
    set { myJob = value; }
}
```

2. While still in the edit mode of the `WordHelper` class, add another method called `IsComplete` to check whether or not the status is complete. The new method, along with the previous property addition, is shown in Listing 13-2.



LISTING 13-2: Modified Word Automation class for monitoring

Available for
download on
Wrox.com

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Office.Word.Server.Conversions;
using Microsoft.SharePoint;

namespace WordAutomationSample
{
    class WordHelper
    {
        ConversionJob myJob;

        public ConversionJob MyJob
```

```

    {
        get { return myJob; }
        set { myJob = value; }
    }

public void Convert()
{
    //Create a instance using your Word Automation Serviced name
    myJob = new ConversionJob("Word Automation Services");

    //Set the conversion properties
    myJob.Settings.OutputFormat = SaveFormat.Document; //DOCX
    myJob.Settings.OutputSaveBehavior = SaveBehavior.AlwaysOverwrite;

    //Establish the SharePoint Server, Site
    //and Document Libraries to use
    SPSite mySite = new SPSite("http://CLCWS2008Dev");
    SPList myInputList = mySite.RootWeb.Lists["Input"];
    SPList myOutputList = mySite.RootWeb.Lists["Output"];

    //Give the Job the same (default) credentials as the site
    myJob.UserToken = mySite.UserToken;

    //Add the Document Library to the Conversion Job
    myJob.AddLibrary(myInputList, myOutputList);

    //Start the job
    myJob.Start();
}

public bool IsComplete()
{
    //Get an instance of the ConversionJobStatus class
    ConversionJobStatus status = new ConversionJobStatus("Word
Automation Services", myJob.JobId, null);

    //Set each type of count
    int count = status.Count;
    int failCount = status.Failed;
    int successCount = status.Succeeded;
    int cancelCount = status.Canceled;

    //Add all of the counts up to determine if conversion is complete
    if (failCount + successCount + cancelCount == count)
    {
        return true;
    }
    else
    {
        return false;
    }
}
}
}

```

Although the new `Convert` method simply returns `true` or `false` depending on whether or not the conversion is completed, the variables `failCount`, `successCount`, and `cancelCount` could also be used to determine failure and success rates. In this case, you had to add those all up to determine whether they matched the total number of documents represented by the count variable.

3. Add a second button to the form that will check the conversion status, as shown in Figure 13-14. For this example, just reply with a simple message stating whether or not the conversion is complete, as shown in the following code:

```
private void button2_Click(object sender, EventArgs e)
{
    string msg;

    if (helper.IsComplete())
        msg = "Conversion Complete";
    else
        msg = "Conversion in Process";

    MessageBox.Show(msg);
}
```

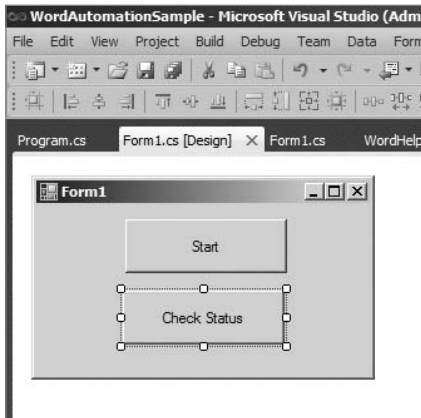


FIGURE 13-14

4. Before you execute the application a second time, you can delete the documents in the Output library or leave them alone. In the code, we set the `OutputSaveBehavior` property on the job to overwrite, so the code will work either way.
5. Execute the application a second time and click the Start button. Now use the Check Status button to determine whether the conversion is completed, as shown in Figure 13-15.

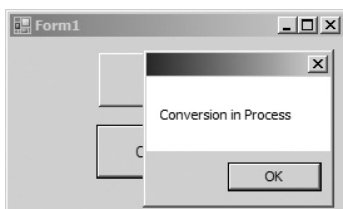


FIGURE 13-15

Open Document Format

Although you are unlikely to use documents of this format within SharePoint, it is at least worth mentioning the Open Document Format. Also known as OpenDoc or ODF, the format specifications were originally written by Sun Microsystems and later adopted by OASIS (Organization for the Advancement of Structured Information Standards). The format is also published as an international standard: ISO/IEC 26300:2006.

Like Open XML, files in the Open Document Format take the format of a .ZIP file when their file extension is changed. The document itself is an XML format, supporting schemas for the following type of documents:

- Word processing
- Spreadsheets
- Presentations
- Databases
- Graphics
- Formulae (for mathematical equations)

Although there is no real support for any ODF files in SharePoint, Microsoft Office applications can open and save to many of the different ODF formats. For example, Microsoft Word 2010 is capable of opening ODT (Open Document Text) files, as shown in Figure 13-16. The Office Web Apps do not have any ODF compatibility.

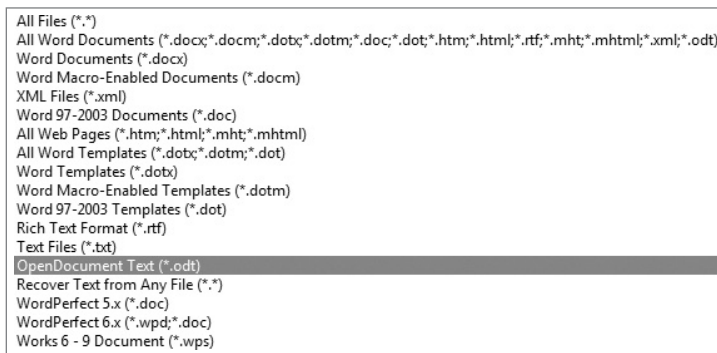


FIGURE 13-16

You can find more information about the Open Document Format on the OASIS website:

www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

ARCHIVE FORMATS

Archive formats are file types that lend themselves to permanent and final renditions of documents. Recall from Chapter 1 that one of the ECM components discussed was “preserve.” In order to preserve a document, you need to save it in a format that ensures it will not be edited and will be readable for the foreseeable future.

In ECM systems, archived documents either started out as living and needed to be preserved in a specific state, or started as document images that were once paper documents and entered the system in an archived state.

TIFF

The TIFF (Tagged Image File Format) was first published in the 1980s primarily as a file format for graphic artists, photographers, desktop publishing, and image scanners. The last update to the specification (which is version 6.0) was in 1992 by Adobe, which currently holds the rights. The specification can be downloaded from the Adobe public website at no charge.

TIFF is primarily a container for images and tagged content. The tagged content can be anything from markup to OCR data. TIFF itself does not specify a compression format or any other type of graphics format. It does, however, allow many different types of formats to be used within it. TIFF files can contain a single one-page image or multiple pages of images of different formats.

Although it is still used, TIFF's popularity has dwindled recently, except for heavy usage in document imaging and fax servers. Although TIFF supports many color formats, bi-tonal has always predominated in these two industries. Bi-tonal is popular for several reasons. First, by employing only two possible colors (black or white), it enables the smallest possible compression — an important consideration when document imaging systems were first conceived, as storage space was very expensive. Even with the much reduced cost of storage today, when storing millions of files, file size can still be an issue.

The second reason bi-tonal works well is that OCR (optical character recognition) engines have a much greater success rate when executed on images with black text and white background. Many OCR engines will even convert a color image to bi-tonal in order for the OCR to process faster and more efficiently. Before text can be extracted from documents with OCR, they are often cleaned up to make the document more legible to the engines. Bi-tonal images are generally easier to clean up than color.

A final reason is that most organizations accept black-and-white documents as the norm. A large majority of paper documents scanned today are black and white; and even when they have color elements, bi-tonal scanning is generally accepted.

The TIFF specification has never been standardized by a major organization such as the ISO or ANSI, but without a major update in almost 20 years, the Adobe-owned specification has become a de-facto standard on its own. With its open tagged format, it is very easy to create new uses for TIFF files without breaking backward compatibility with most viewers, a fact that has encouraged organizations to find proprietary uses for TIFF files, specifically in two areas: OCR and markup (also known as annotations).

OCR and iFilters

Although third-party software may all employ different tagging schemes in TIFF documents for OCR data, there seems to be a commonly used tagging scheme that just happens to be used by the iFilter that comes with Windows Server 2008. How convenient, right? This iFilter provides both

SharePoint and FAST Search Servers with the capability to search text data in TIFF documents. This iFilter can also perform OCR on TIFF documents in cases where it has not been accomplished prior to uploading the documents to SharePoint. To install the iFilter, follow these instructions on the SharePoint Index Server(s) or FAST Search Server(s):

1. Open Server Manager on the Windows Server.
2. Select Features on the tree.
3. In Features Summary, select Add Features.
4. Select Windows TIFF IFilter ⇨ Next ⇨ Install.
5. Select Close, and then select Roles on the tree.
6. In Roles Summary, select Add Roles.
7. Select File Services and then click Next twice.
8. Select Indexing Service under Windows Server 2003 File Services.
9. Click Next, and then click Install.



These steps simply turn the iFilter on without configuration. If you only need to add previously recognized text on TIFF documents, you can skip the steps for configuring the iFilter.



Turning on OCR for TIFF documents may have a negative impact on your Index Server(s), as text recognition is often a processor-intensive operation.

Once installed, the TIFF iFilter should be configured in order for OCR to be performed:

1. Open the Local Group Policy Editor by selecting Start and typing **gpedit.msc**.
2. On the tree, select Computer Configuration ⇨ Administrative Templates ⇨ OCR.
3. Double-click the first option, “Force TIFF IFilter to perform OCR...”.
4. Select Enabled, as shown in Figure 13-17.
5. Click OK.
6. Double-click the second option, “Select OCR languages from code page.”
7. Select Enabled.
8. Select the languages that you want the OCR process to recognize.
9. Click OK.



When using the Windows TIFF iFilter to create and search full-text data in SharePoint, the OCR data is not added to the TIFF file itself; it resides only inside the SharePoint and FAST search indexes.

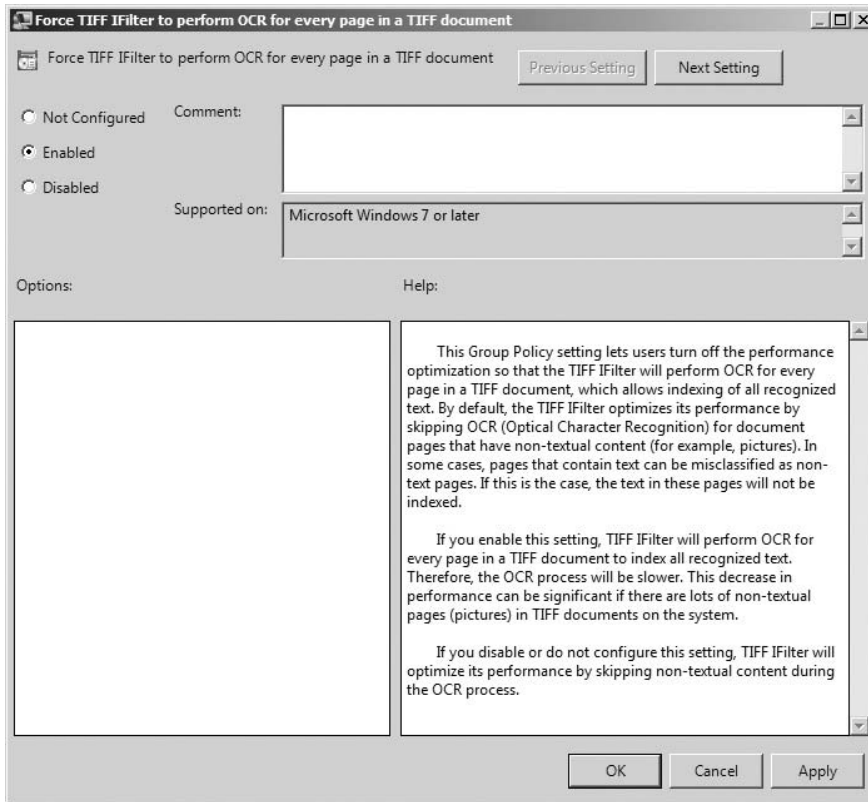


FIGURE 13-17

SharePoint Search

The iFilter is now installed and configured. However, a few additional steps are required for SharePoint Search:

1. Reboot the Index Server(s). Restarting SharePoint Services may also work.
2. Perform a full crawl on necessary content sources. After the first full crawl, incremental crawls will work on new content.

FAST Search

For FAST, follow these additional steps:

1. Find the `user_converter_rules.xml` file in the `formatdetector` directory on the FAST server (see Figure 13-18).

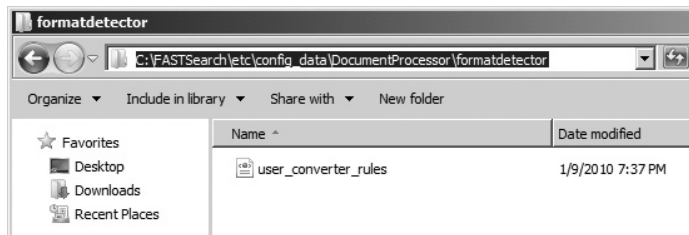


FIGURE 13-18

2. Modify the ConverterRules section of this XML file:

```
<ConverterRules>

  <IFilter>
    <trust>
      <ext name=".tif" mimetype="image/tiff" />
      <ext name=".tiff" mimetype="image/tiff" />
    </trust>
  </IFilter>

  <MimeMapping>
    <mime type="image/tiff">Tiff File</mime>
  </MimeMapping>

</Converter Rules>
```



Once changes are made to the user_converter_rules.xml file, ensure that the file is backed up, as any FAST Server updates or service packs are likely to overwrite this file.

3. Run the psctrl PowerShell command (see Figure 13-19).



FIGURE 13-19

4. Execute a second crawl on the FAST Server.



OCR does not seem to be executed by the crawler(s) but by another service running on a timer. Even though a crawl may be executed after a TIFF document is uploaded, it will not show up in the Search Index until after the OCR process is finished and a subsequent crawl is performed.

Markup

Also known as *annotations*, markup is often used in TIFF files to point out specific details in documents. Markup can be just about anything that can help with these details, such as highlights, lines, or freehand text. Markup can be stored externally to a TIFF file, which is typically specific to software vendors, but markup can also be stored internally. Although many different formats have been developed to markup TIFF files, in the world of ECM only two specifications have found mainstream usage. The first we probably shouldn't even call a specification because it is not published and it is only used by the now deprecated Microsoft Office Document Imaging (MODI) product. Many users however still use older Microsoft Office products, even with the latest version of SharePoint, and may be using TIFF documents with this markup format.

The second and more popular specification was used in older Microsoft products such as the Imaging and Fax viewer in Windows XP and Imaging for Windows in previous versions. The format still is available today in Imaging for Windows, but it is sold by Global 360 and is not included with Windows. In SharePoint, this markup format is very popular and is used by many companies, providing TIFF image viewers on the server.

Development

Building applications to create and edit TIFF files has become mainstream, and almost any graphics application or API can manipulate either single- or multi-page TIFF documents. Within Windows, GDI+ enables creating and editing TIFF documents (along with many other graphics capabilities). Although this API is based on Win32 and was originally designed for C/C++ programmers, the `System.Drawing` namespace in the .NET Framework gives you access to the same API through managed code.

PDF

The Portable Document Format (PDF) is a file format invented by Adobe Systems in 1993. The format is essentially a document container for formatted text, images, fonts, and vector graphics, and many types of multimedia, among other possibilities. With no-cost viewers available for just about any device or operating platform in existence, PDF could easily be considered the best multi-platform format in existence for the archiving of just about any media. Many of the vendors offering free viewers also offer editing applications. However, these applications are typically platform dependent.

OCR and iFilters

Many PDF files contain text data; for example, when Microsoft Word documents are converted to PDF, the text data and formats are typically retained. These files are typically known as *text only*, or sometimes *text on image* if the original document contained graphics. In this case, the converted document looks almost identical to its original format when viewed.

Image files are also often converted to PDF for archiving, and the image data must be converted to text in order for it to be searchable in SharePoint. These PDF files are known as Image+Text or Image on Text, meaning the image is all that is visible but the OCR text data is stored in the background.

Both formats are searchable within SharePoint with any of the iFilters available on the market. Most iFilters will OCR image-only PDF files and put the text in the background to create the Image+Text format, while others will not only OCR the file, but also attempt to recreate the file as text only or text on image.

As all iFilters are third-party products, installing and configuring them is outside the scope of this book. Chapter 14, however, discusses the SharePoint EcoSystem, as well as some partners with quality iFilters.

Markup

As with most documents being archived, the document itself needs to stay in its original form; markup is the best way to make notes without modifying that format. Markup with PDF files comes in two forms, markup compatible with Adobe Acrobat and markup not compatible with it. Of course, Adobe is the standard for viewing and editing PDF files. For example, the authors of this book work for a software company that builds PDF viewing/editing capabilities into SharePoint, and we also use Adobe as our standard PDF viewer.

This being said, there is no way to achieve any type of markup on PDF documents natively in SharePoint or Windows. You need to either adopt one of many third-party products or build the code yourself. Note, though, that attempting PDF manipulation on your own will likely end in frustration and some incompatibilities.

Development

Regardless of whether you need to programmatically add markup, create new PDF files, or modify existing documents, you have many options for accomplishing the task. As stated earlier, trying to manipulate PDF files on your own will rarely prove to be cost effective, and it is not covered here; but if you are determined to do so, Table 13-1 shows some of the more popular libraries and SDKs.

TABLE 13-1: Popular PDF Libraries and SDKs

| LIBRARY/SDK | WEBSITE | LICENSE |
|-----------------|--|-------------------------------|
| PDF Library SDK | www.datalogics.com/ | Commercial |
| Acrobat SDK | www.adobe.com | Commercial (requires Acrobat) |
| iTextSharp | http://itextpdf.com/ | Open Source/Commercial |

Viewing and Editing

No current version of SharePoint has native PDF viewing built in. Many free viewers on the market today (such as Adobe Acrobat Reader) require PDF documents to be downloaded locally and viewed. Many vendors also build viewers that install on SharePoint servers and enable viewing in the browser without downloading files.

Editing PDF by downloading typically presents a problem, as few (if any) vendors allow PDF files to be edited and saved directly back to SharePoint — that is, without having to rebrowse for the save locations. KnowledgeLake’s Connect for SharePoint product does offer a solution to this by

intercepting browser downloads from SharePoint and allowing them to be saved back by simply using the native save capability in the application.

Living Document Conversion

Because PDF is not a format conducive to living documents, but rather one more suitable for archived documents, it was great to see SharePoint 2010 add the Word Automation Services feature described earlier in this chapter. A typical scenario for this conversion service is an organization that has a life-cycle or workflow process that uses Microsoft Word documents, and at the end of the process the documents need to be converted into a permanent record suitable for long-term archiving (see the next section, “PDF/A”) and universal viewing.

The small application you built earlier in the chapter can also be used to convert Microsoft Word documents to PDF/A. As shown in the following example, all you need to do is modify three lines of code in `WordHelper.cs` to change the output format:

```
//myJob.Settings.OutputFormat = SaveFormat.Document; //DOCX
myJob.Settings.OutputFormat = SaveFormat.PDF; //PDF
myJob.Settings.FixedFormatSettings.UsePDFA = true; //PDF/A
```

When the application is executed, the documents in the Input library will again be converted into their new format in the Output library. Figure 13-20 shows the result of this change: the Output library now contains two PDF/A files in addition to the DOCX files generated earlier.

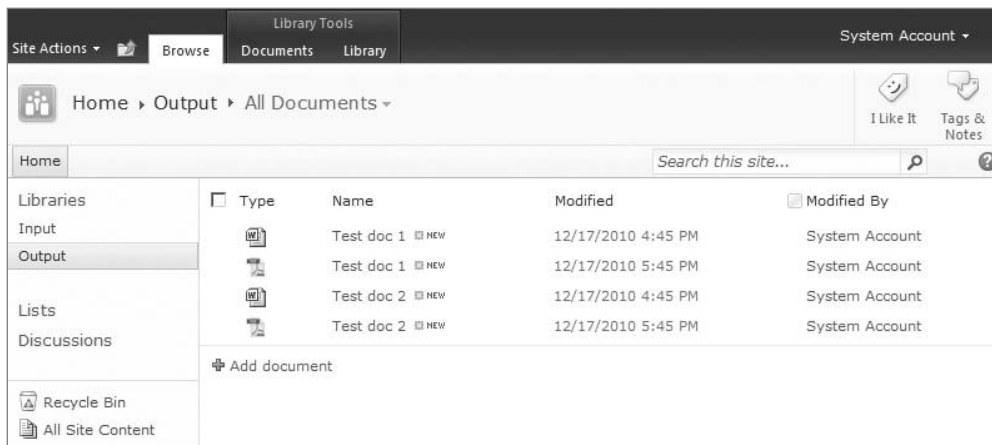


FIGURE 13-20

PDF/A

The PDF/Archival (PDF/A) format is a subset of the PDF file format. It has become an international standard for the long-term archiving of documents. PDF/A is only a subset because many of the features of the current PDF format (1.4) needed to be removed, as they were not suitable for long-term preservation. PDF/A ensures that a document can be reproduced in its expected visual state in the future, when viewer technology may be quite different from now.

Although the format of PDF and PDF/A files are the same, PDF/A files must contain XMP (Extensible Metadata Platform) data, which is a labeling technology that embeds both standardized and proprietary data into PDF documents. These PDF/A files have several restrictions, however, that ultimately provide a format that should be completely self-contained and device-independent. The restrictions are as follows:

- Audio and video are not allowed.
- JavaScript or any other type of code is not allowed.
- Any external references are now allowed.
- Embedded files are not allowed.
- Fonts must be embedded, which helps guarantee that the file can be viewed in the future when the fonts used may not be readily available.
- Colors defined must be device-independent.
- LZW Compression is not allowed.
- Encryption is not allowed (although DocEncrypt by PFU Systems will still meet PDF/A regulations).
- Transparent images are not allowed (a restriction that may be removed in a future revision).

PDF/A is becoming the most widely accepted format for storing archived documents and is being adopted by many agencies worldwide. Organizations such as the National Archives (www.archives.gov) are accepting only PDF/A for documents submitted in electronic form. This is helping to drive most federal agencies to PDF/A, as many of them submit their permanent records to the NARA (National Archives and Records Administration).

Standardization

PDF/A is really just a standardization of the PDF format, which is deemed superior for long-term archiving. The standard is what created the format, not Adobe itself. Like many standards in the ECM industry, although it was influenced by several different organizations (especially document/imaging management), PDF/A was primarily driven by the AIIM (www.aiim.org). Today, the PDF/A committee is part of the AIIM organization and is primarily responsible for defining the restrictions and metadata tags required.

Driven by an industry need to ensure long-term viewing compatibility, PDF/A became an international standard in 2005 and was labeled ISO 19005-1. The standard is based on PDF 1.4 and is also known as PDF/A-1. Another standard currently in process will be known as PDF/A-2 or ISO 19005-2. The standard can be downloaded from the ISO website:

www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38920

OCR and iFilters

Most current OCR engines that export to PDF also have the capability to create PDF/A documents during export. Because the format of searchable text in PDF and PDF/A is essentially the same, iFilters in SharePoint should work on either one.

Creating, Viewing, and Editing

Creating PDF/A files has become a standard option for any software that creates PDF documents. Adobe added the capability to generate the archived format starting with Adobe Acrobat 8. Microsoft added the capability to Office applications beginning with Office 2007. Follow these steps in most Office 2010 applications to save your work as a PDF/A document:

1. From the File tab on the Ribbon, select Save & Send.
2. Select Create PDF/XPS Document ⇨ Create/XPS, as shown in Figure 13-21.

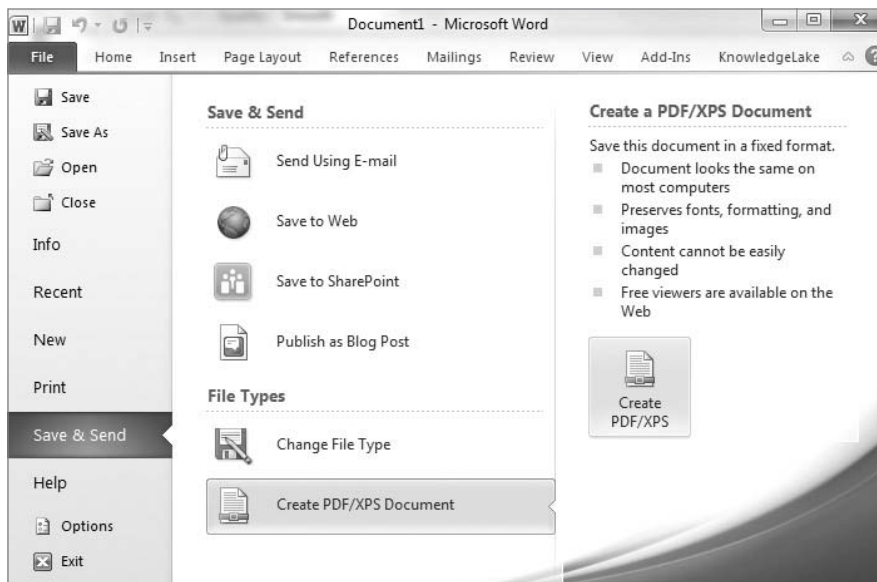


FIGURE 13-21

3. From the file dialog, select the Options button in the bottom-right corner.
4. In the Options dialog that appears, check the ISO 19005-1 checkbox under PDF options, as shown in Figure 13-22.

The terms “editing” and “PDF/A file” are sort of mutually exclusive. Archived documents should not be modified, so the capability to modify and resave is not necessary. However, many PDF editing applications do enable opening a PDF/A file, as it is essentially just a PDF. However, care needs to be taken if modification is necessary; instead of allowing the document to be saved, a new PDF/A document should be created, leaving the original unmodified.

XPS (Open XML Paper Specification)

The Open XML Paper Specification is a format very similar in structure to the Open Office XML formats, but it is much more flexible. Although many people believe that XPS was created as a “PDF killer,” that now seems not to be the case, or Microsoft changed its plans after inception.

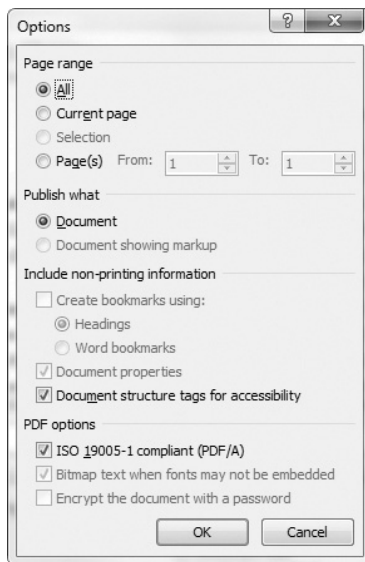


FIGURE 13-22

XPS has two general purposes. The first is for scanning paper documents into a format that can move directly to a print stream. The second is for printing. XPS, much like PDF, was designed to show exactly what a document would look like in print format when viewed — thus the name “Paper Specification.” Any format that can represent itself well in print format is suitable for archiving, and XPS does fit that description.

XPS documents are very easy to create, as current versions of Windows include a print driver that will render any document that can be printed into an XPS document. When printing a document, simply select Microsoft XPS Document Writer, and an XPS file will be created and launched in the XPS Viewer.

XPS documents can also be generated with both Microsoft Office 2007 and Microsoft Office 2010, as shown in Figure 13-21 in the previous section. You can also accomplish this same task with the Office Web Applications in SharePoint 2010.

OCR and iFilters

Like PDF, some OCR engines are capable of creating XPS documents from image files and adding the extracted text to the files in order to make them searchable. XPS is perhaps the easiest file format to contain the extracted text, as Microsoft publishes and maintains an API to manipulate XPS documents. However, only a handful of products currently support the capability to export XPS documents.

Unlike the TIFF iFilter built into Windows Server 2008, the XPS iFilter does not have the capability to perform OCR on XPS documents, but it will search full-text data on previously recognized documents. To install and configure the XPS iFilter, follow these instructions on the SharePoint Index Server(s):

1. Open Server Manager on the Windows Server.
2. Select Features on the tree.
3. In Features Summary, select Add Features.
4. Select XPS Viewer ⇌ Next ⇌ Install.
5. Click Close (and close the Server Manager).
6. Open SharePoint Central Administration.
7. In Application Management, select Manage Service Applications.
8. Click Search Service Application.
9. In Search Administration, select File Types, and then New File Type, as shown in Figure 13-23.

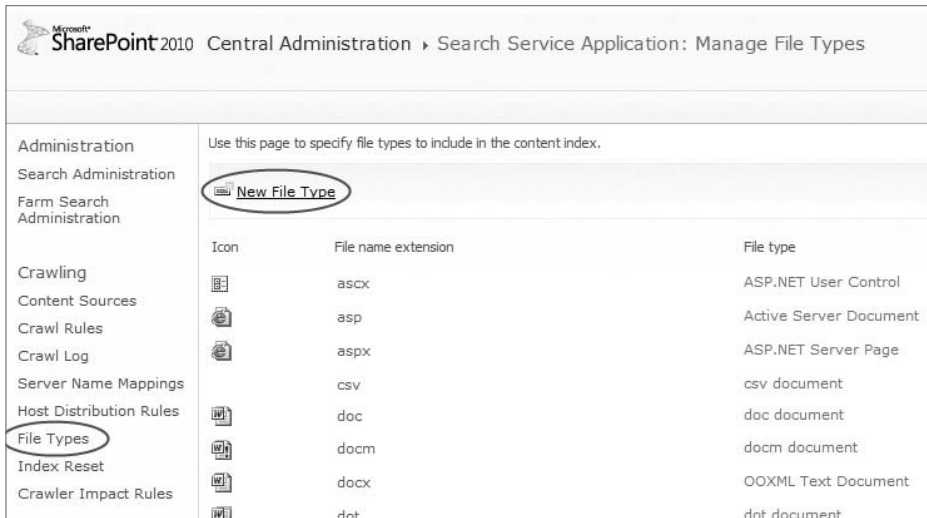


FIGURE 13-23

10. On the Add File Type screen, enter `xps` as shown in Figure 13-24.

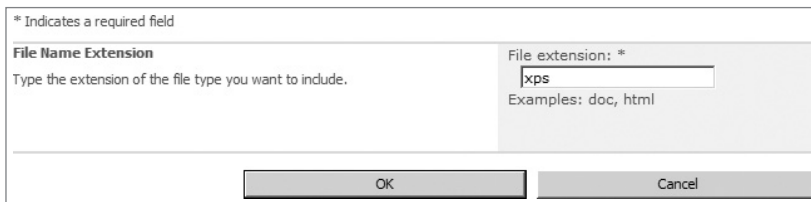


FIGURE 13-24

11. Verify that the XPS file type has been added by checking the Manage File Types screen, as shown in Figure 13-25.

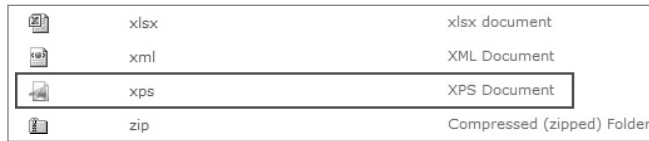


FIGURE 13-25

- 12.** You now need to modify the registry on the SharePoint Index Server(s). Using the registry editor, add the following key and its corresponding values:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\14.0\Search\Setup\Filters\
.xps]
  Default = (value not set)
  Extension = xps
  FileTypeBucket REG_DWORD = 0x00000001 (1)
  MimeTypes = application/xps
```

- 13.** A second registry key needs to be created on the SharePoint Index Server(s):

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\14.0\Search\Setup\
ContentIndexCommon\Filters\Extension\.xps]
  Default = {1E4CEC13-76BD-4ce2-8372-711CB6F10FD1}
```



This GUID comes from the Registry Entry for the XML Paper Specification Filter registration for XPSFILF.DLL. This GUID should typically be the same as shown in the [Default] key, but a quick search of the registry to make sure is a good idea.

- 14.** Restart the SharePoint Search Server or reboot the Index Server.
- 15.** If XPS documents are already stored in SharePoint, a full crawl will be required; otherwise, the next incremental crawl will only index newly uploaded documents, as shown in Figure 13-26.

Markup

Unlike TIFF and PDF, there are no specifications for XPS files or any common way to add markup to them. Many vendors are shying away from any type of support for markup in XPS for fear that whatever format is used may become obsolete if Microsoft ever publishes its own standard.

Development

Like Open XML, the .NET Framework includes the necessary components for manipulating XPS documents. The `System.Windows.XPS` namespace can be used to read and write XPS documents.

For C++ programmers and those who need to write native Windows applications, the XPS Document API contains lower-level capabilities.

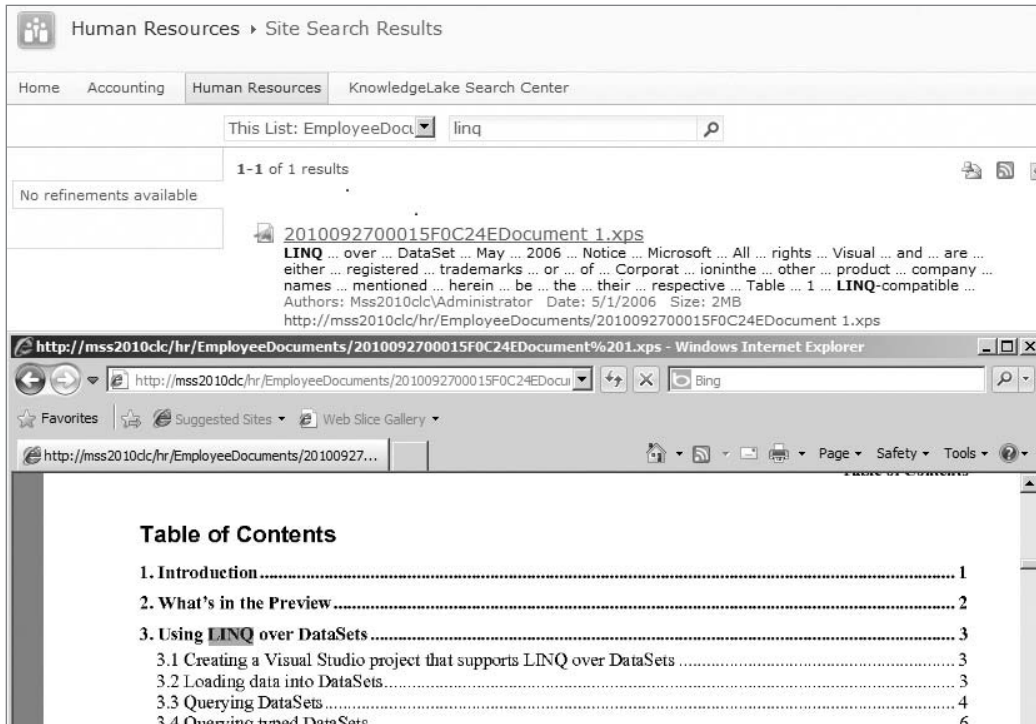


FIGURE 13-26

SUMMARY

Enterprise content management systems need to handle a variety of documents (content) for a wide range of purposes (context). As shown in this chapter, these documents fall into one of two categories: living (or dynamic) and archived. Living documents begin their life as dynamic components developed by the organization, whether it is content from a word processor, spreadsheets, presentations, or something else. These documents are often part of a business process that, when completed, requires the documents to be archived and stored in a format that can be easily reproduced in the future.

Microsoft Office is typically the primary source of living documents. This chapter discussed both the legacy binary formats and the new Open XML formats for documents generated by Office. Microsoft introduced a new technology in SharePoint 2010 called Office Web Apps, which enables the viewing of Word, Excel, PowerPoint, and OneNote documents in the browser with no additional plug-ins.

As living documents become records, or when paper documents are scanned into SharePoint, they need to be converted to another format suitable for longer-term storage. In addition, because these documents may become permanent records, and perhaps be used for auditing purposes, they should be stored in a format that is not conducive to editing. This chapter discussed three formats for archive documents — TIFF, PDF, and XPS — and covered their advantages and disadvantages. PDF also has an international standard called PDF/A (the A is for “archive”) that enables documents to be put into a state that, it is hoped, will be viewable for generations to come.

14

The SharePoint ECM Ecosystem

WHAT'S IN THIS CHAPTER?

- Getting to know the ecosystem
- Becoming a partner in the ecosystem
- Participating in the community
- Discovering ISV solutions

Few can argue the success of the Microsoft Partner Network. It is one of the most successful channel organizations ever compiled in the history of information technology. At the heart of this channel are independent software vendors (ISVs). These ISVs are a focus of this chapter, with specific attention on ISVs in the SharePoint ECM Ecosystem.

Before moving directly into the discussion of the SharePoint ECM Ecosystem, this chapter provides an overview of the general Microsoft Partner Ecosystem and Microsoft Partnerships in a nutshell. The goal is for you to understand the importance of the ecosystem and why it is crucial to understand and embrace it not only as an ISV, but also as a reseller or implementer.

THE MICROSOFT PARTNER ECOSYSTEM

Although the term *ecosystem* is an obvious play on words, Microsoft and its partners are essential to the success of one another, much in the same way that organisms and biological environments rely on each other. This reliance between Microsoft and its partners creates what is known as the *Microsoft Partner Ecosystem*.

Biological ecosystems are mutually beneficial to all the participants, and the Microsoft ecosystem is no different — providing benefits to all parties. Microsoft benefits when partners sell licenses of its software and when they deploy applications onto Microsoft operating systems, which in turn promotes Microsoft to customers and other potential partners.

Partners also benefit nicely from the ecosystem, making participation just plain good business. For example, in 2010, according to International Data Corporation (IDC), it was estimated that the Microsoft ecosystem made \$8.70 in software and services for each dollar (or other unit of currency) that Microsoft made. This is an increase from 2009, when this was estimated at \$4.43. In addition, ecosystem revenues were at \$580 billion in 2010, up from \$537 billion in 2009, and \$475 billion in 2007. This trend demonstrates that the ecosystem is growing continuously and unlikely to slow down in the years to come.

In addition to the notable financial benefits, partners also benefit by gaining valuable contacts within Microsoft and with other partners in the ecosystem. It is well known that partners like doing business with other partners, as they have the mutual goal of gaining and retaining customers on the Microsoft platform.

Becoming a Partner

If the preceding discussion about the reasons to become part of the Microsoft Partner Network have piqued your interest, you are probably wondering how an organization goes about becoming a part of this ecosystem. First you need to understand the types of partnerships that are available.

Microsoft provides several different types of partnerships, and there are two levels of competency: Silver and Gold. Potential partners start by filling out a simple online form to become part of the network. As you implement customers on the Microsoft platform, as well as obtain a workforce trained on the same platform, your opportunities for partnership benefits increase. As your benefits increase, your commitment to the network will also likely increase. This increased commitment can earn your company recognition as a Silver- or Gold-Certified Partner.

Detailed information about partnership is available on the Microsoft Partner Network website at <https://partner.microsoft.com>. This chapter focuses on the ISV/Software Competency. Let's look at what it takes to become a Gold-Certified Partner in the Microsoft Network as an ISV.

ISV/Software Competency

Organizations worldwide are looking into how Microsoft technologies can help their operations run more efficiently and provide a rapid return on investment, not just by improving the handling of enterprise content but throughout all aspects of the organization affected by technology.

In order to accomplish these goals, organizations are looking at solutions from independent software vendors (ISVs). In turn, these ISVs are working with Microsoft — not necessarily to help them build the solution, but to ensure that it addresses the corporate audience, integrates into both existing business solutions and other Microsoft technologies, and promotes the solution globally. Of course, in turn Microsoft relies on ISVs to complete the vision. SharePoint ISVs are a good example of this, adding the necessary components to create a complete ECM platform.

After an ISV has joined the Microsoft Partner Network as a basic subscriber, there is a clear path to Gold Certified Partner competency. Silver competency is first gained by employing or contracting individuals who have earned the Microsoft Certified Professional (MCP) title and passed the necessary licensing and marketing assessments. In addition you are required to provide customer references in accordance with the competency being sought. Of course, there is a yearly fee, but it includes more than enough software to offset the cost.

Gold Certified Partner competency is gained by achieving Silver status plus providing more customer references, participating in the Customer Satisfaction Index, and of course paying a higher yearly fee.

For ISVs, additional steps need to be taken depending on the specific type of software provided. The network's goal is to assure customers that dealing with a network partner means dealing with a vendor proven in the ecosystem, a vendor that can meet or exceed expectations.

THE SHAREPOINT ECOSYSTEM

The SharePoint user base, according to Microsoft, is adding about 7.3 million new users per year. In addition, Microsoft says that SharePoint is the fastest-growing server product they have ever produced; in the twenty-first century, it has completely dominated the ECM headlines. The number one reason for this is, you guessed it, the ecosystem.

The SharePoint ecosystem is primarily made up of user communities and ISV partners, but systems integrators and others play major roles also. The communities are the best way for individuals to get involved — whether they are independent or working for an organization building SharePoint solutions.

Technical Community

The SharePoint Ecosystem has taken on a life of its own, creating a passionate culture for the growing community of users, administrators, and developers. There are endless opportunities for you to be involved in the culture, even if you aren't involved with a SharePoint ISV. Since you are reading this book, perhaps you are already participating in the Microsoft community at large and want to delve deeper. The following sections take a look at some of the opportunities available for participation in the community.

MVP Program

The MVP (Most Valuable Professional) Program gives hard-won peer recognition to individuals who have excelled in their particular community. SharePoint MVPs have proven their excellence in SharePoint and their willingness to serve as leaders in the community. The program is for developers, architects, and administrators alike. Any driven individual in the community can earn this title and be recognized as a leader on forums and at conferences. The coolest thing about this award is that MVPs are nominated by people in the industry, not by Microsoft.

As none of the authors of this book are MVPs, it's safe for us to give proper credit to these individuals. Keep in mind that you don't have to hold the MVP title to be a SharePoint "rock star"; but if you meet someone with this title, chances are good that they already are.

Social Networks and Forums

An obvious way to easily and quickly participate in the community is to follow and get involved in social networks. It may seem more productive to stick with business-oriented networks such as LinkedIn, massive amounts of information can be found on Facebook pages; and of course Twitter has many industry-related tags.

Although social networks may be the rage, good old-fashioned forums are still one of the best places to ask questions and submit answers. The MSDN forum at <http://social.msdn.microsoft.com/Forums/en/category/sharepoint> is one of the most popular.

User Groups

You should be able to find a SharePoint user group in most large U.S. cities, as well as many other countries around the world. A simple Internet search of “SharePoint User Group” will quickly return enough results for you to find a nearby location.

ISV Solutions

As mentioned earlier, SharePoint’s success has been largely due to the ecosystem created around it. For those living in the ECM industry but outside Microsoft technologies, it may have seemed like a sudden (and very abrupt) jolt in the system, as SharePoint has likely made the single largest impact of any one piece of technology. However, this however was not the case for the earliest adopters.

As early as the 1990s, people and organizations working with Microsoft technologies knew that something was brewing with Microsoft Site Server, which offered a new concept for collaboration and document management. Microsoft Site Server eventually split into a couple different technologies; one was the ill-fated Commerce Server, but the other was SharePoint Portal Server.

Although it was not initially built on a platform destined to become an ECM powerhouse, Microsoft SQL Server found its way into the storage system by the time version 2 was released in 2003. This version, known as Microsoft SharePoint Portal Server 2003 and Windows SharePoint Services 2.0, was the key that opened Pandora’s Box, so to speak. Since this release, ISVs have been building just about anything imaginable using the SDK, and a handful of vendors started turning SharePoint into a complete ECM platform.

One of the most common scenarios organizations run through when making a decision to expand their SharePoint system into an ECM Platform is whether to build custom solutions themselves as opposed to buying solutions from ISVs. Some of the key factors in determining the correct path are return on investment (ROI), timeframe, conversion, available expertise, and real needs of the organization.

Before embarking on any project, regardless of whether the decision is to build or buy, it is important to identify the ROI that is expected. Automation always sounds ideal, but sometimes the time savings doesn’t equal the real cost of the implementation. If you are planning on buying, the cost is easier to determine, although there may still be some customization. If you are planning on building, be sure to factor in enough project management, architecture, development, and quality assurance. User acceptance testing should be considered regardless of whether you build or buy.

The needed timeframe often drives an organization to buy solutions. Even if a solution will only take you 75% of the way towards a happy user base, using the right ISV should drastically reduce the timeframe to implementation. This can also shorten the time to achieve ROI.

Conversion (or migration) is something that is very important to consider not only when making a decision to build or buy but anytime a project may involve moving from one system to another. Moving to well-known packaged software can often ease conversion as the software may have

internal features to aid in the process and a larger community. If the solution is to replace a manual process, this decision becomes somewhat moot.

Something heard about a lot in the SharePoint user community is the lack of available expertise. Even though you may have determined that it might be less expensive and within the timeframe to custom build a solution, there might be a lack of available talent to do the actual build. If your organization is lucky enough to have the right system architects and developers then you are in the minority.

Understanding the real needs of the organization is also a key factor in deciding whether you should build or buy. Sometimes the software needed may not even exist in the ISV community, which can force you to start an internal development project. Other times a combination of both ISV and custom code may be the best alternative. The users ultimately use the software and they are ones who need to become more productive, so be sure their needs can be met regardless of whether you build it yourself or make a purchase.

Politics, extreme technology opinions, too conservative/too liberal approaches, and similar negative factors can unfortunately end up influencing the decision of whether to build or buy, but if your goal is to provide the correct solution, try to minimize these factors as much as possible. Perhaps one of the most important things to remember is to make a good decision, not necessarily the best decision. Don't shortcut the process of making a good decision and don't waste too much time looking for perfection.

The remainder of this section focuses on some of the vendors — not all early adopters, but all fully vested in SharePoint Products and Technologies — and the components they provide to complete the SharePoint ECM platform. They are not only leaders in the SharePoint community, but each provides tools and solutions that extend the capabilities of SharePoint, rather than replace them.

ABBYY

Founded in 1989, ABBYY is a leading provider of document conversion, data capture, and linguistic software — with an installed base of more than 30 million users. ABBYY has been a certified Microsoft ISV since 1999, and in 2003 was named a Microsoft Gold-Certified Partner with an ISV competency. Small, medium, and large enterprises, as well as local, state, and federal governments use ABBYY's products.

ABBYY is headquartered in Moscow, with major offices in the United States, Germany, United Kingdom, Ukraine, Taiwan, Japan, and Australia. ABBYY not only specializes in solutions, but also makes its software available as an SDK that can be used in many types of ECM scenarios.

Capture

ABBYY enhances SharePoint with data capture products that address the needs of people and organizations handling both electronic and paper documents in myriad formats and languages. With ABBYY products, users can access information that would otherwise be locked away in complex documents, enabling them to acquire critical business intelligence quickly and efficiently without the resource drain associated with data entry.

Full-Text Searching

Scanned or faxed documents are not useful to users until the information provided in these documents can be easily found. Full-text searching is the capability to search documents not for specific

metadata or tags, but by the text that exists within the document. For electronic documents, such as those in Microsoft Office formats, this searchability is built into SharePoint. Documents that start as paper require specialized software for full-text searches.

ABBYY Recognition Server resolves these issues by providing the following:

- A server-based solution for automating document processing in enterprise and service-based environments
- Software that is easy to operate and does not require specific training
- Running on multiple servers so that even extremely large workloads can be managed, either by scheduled or continuous document processing
- Intelligence to recognize a myriad of document types and convert them into a format that can be read by SharePoint with the highest degree of accuracy
- A SharePoint iFilter to routinely crawl stored data for new or altered documents to update document indexes
- Automatic routing of images with text that cannot be understood by SharePoint to the Recognition Server, where any text can be processed and returned to SharePoint
- Document recognition in 195 different languages
- Accurate recognition of textual data contained in even the lowest quality of images
- Saving queries for re-use so results can be organized using an intuitive interface
- Allowing users to create an inbox for SharePoint documents and a framework for workflow
- Tracking document changes without the need for additional software or complicated customization

Unstructured Searching

As stated in Chapter 1, ECM tools enable an organization to manage its unstructured information. Unstructured content represents approximately 80% of the information in the average corporate enterprise. Getting this unstructured information off of paper and into SharePoint presents many challenges.

ABBYY FlexiCapture technology resolves these issues by:

- Enabling SharePoint integrators to leverage information from scanned images, image-based e-mail attachments, and existing digital archives in more powerful ways
- “Teaching” the system how to extract classification and indexing information from scanned images imported into SharePoint
- Enabling system administrators to design rules for documents and the information needed to properly process that document based on the unique business needs of the organization
- Providing separation and classification of paper documents prior to being imported into SharePoint
- Enabling documents to be imported directly into SharePoint document libraries, as shown in Figure 14-1

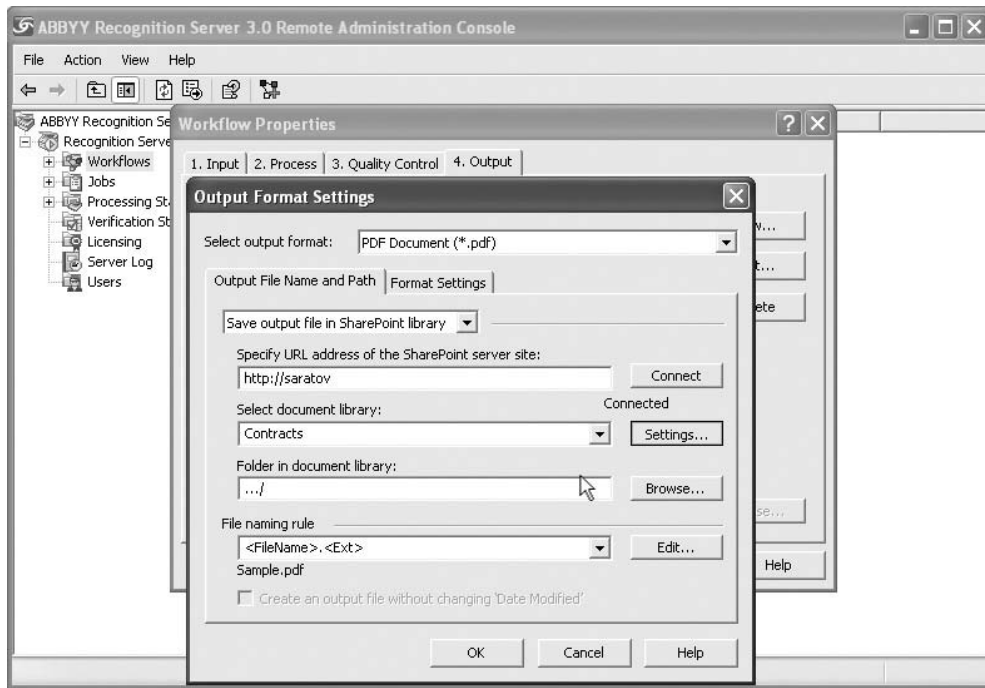


FIGURE 14-1



Perhaps the largest unstructured content problem exists in accounts payable departments. FlexiCapture can be used to identify different kinds of invoices and automate the extraction of appropriate metadata for a particular form or document. Once the correct information is extracted, the metadata can be used to automate proper processing, and even automate the triggering of e-mails for approvals.

The ability to accurately capture, store, and index documents is critical to building an organizational knowledge base. The barrier to achieving this has been the difficulty of handling documents in myriad formats and the sheer amount of paper still being used by many organizations. Transforming unstructured data into a form that can be handled by traditional databases has been, until recently, a laborious process. ABBYY's technology enables the use of this data without a drain on organizational resources.



For more information about ABBYY, you can browse to their website at www.abby.com.

AvePoint

AvePoint is a global technology company headquartered in Jersey City, New Jersey. Since its founding in 2001, it has become one of the largest providers of infrastructure management software solutions for Microsoft SharePoint products and technologies. AvePoint employs a SharePoint-exclusive research team so that they can work with small, medium, enterprise, and government organizations to develop powerful and flexible infrastructure management solutions for SharePoint environments.

AvePoint is a Microsoft Gold-Certified Partner and GSA provider. AvePoint's DocAve software platform is a solution for comprehensive, scalable, and truly integrated SharePoint backup and recovery, administration, replication, migration, archiving, deployment management, reporting, storage optimization, and content life-cycle management.

Capture

Capturing content to SharePoint is often assumed to mean scanning paper or uploading electronic documents; however, content from other digital legacy data stores such as Lotus Notes, EMC Documentum, and local file shares must be migrated into SharePoint. This must be accomplished while maintaining all associated metadata. Migrating legacy data and its associated metadata consolidates enterprise content and reduces legacy licensing costs.

ECM systems should have the capability to connect with this legacy content and utilize SharePoint's management and presentation features without directly importing into the legacy databases.

AvePoint addresses these other capture scenarios with the following features:

- Delivering and presenting data from legacy stores directly through SharePoint, providing a single point of access for end-users to interact with enterprise content
- Migrating legacy data stores, along with the content metadata, including file shares, Exchange public folders, Lotus Notes, Documentum eRoom, and others, granularly or in bulk, according to business demands and schedules

Store

Storage in SharePoint means efficiently managing native SQL Server content database storage to unify enterprise content residing on disparate legacy systems. As mentioned in Chapter 1, the vast majority of content uploaded by users into SharePoint is unstructured data such as digitized paper, Office documents, PDFs, pictures, or audio and video files, all of which get stored as binary large objects (BLOBs) in SQL, potentially degrading platform performance. Organizations must manage the growing amount of this unstructured data in order to maximize its potential for scalability and performance, while efficiently archiving content in a manner that ensures all compliance objectives are met.

AvePoint addresses this challenge with the following features:

- Offloading to tiered storage all discrete enterprise content exceeding a custom-set rule, utilizing full support of Microsoft's EBS and RBS APIs
- Performing enterprise-wide archiving operations with a unified business-rule engine to optimize storage

- Replicating all ECM content, configurations, and securities to all regional SharePoint farms in real time, with customizable throttle controls, compression, and two-way synchronization
- Centrally managing all user permissions, settings, and configurations, granularly or in bulk, across the entire multi-farm environment

Preserve

All enterprise content must be protected in a manner that conserves storage resources while enabling fast, full-fidelity restoration of all prior metadata and securities. The ECM platform itself must also be fully protected, including system configurations, customizations, and workflows, through customizable system backups. Organizations should ensure a seamless transition to a warm stand-by system should the main production system fail, as well as swift platform restoration following a disaster event.

AvePoint addresses preservation with the following:

- Fast, full-fidelity recovery of any lost or corrupted ECM documents, objects, sites, and site collections directly to the production environment or to any other SharePoint location with no need for staging.
- Granular, intelligent backup capabilities to optimize backup strategies by prescribing variable backup routines/frequencies to content sets in order to aggressively protect critical ECM data.
- A warm, stand-by ECM system environment for one-switch disaster recovery.
- An intelligent, automated end-of-life strategy for old and stale content according to business-specific requirements, while optimizing storage of SharePoint's SQL Server.
- Unified, automated eDiscovery across all active and archived enterprise content, which allows search and identification using keyword, user, group, object, or metadata field. Output eDiscovery reports and records in all industry-accepted formats for third-party review.
- Regulatory compliance obligations with full auditing of all ECM system and user activities, immutable content capture, and comprehensive, customizable report generation.

Delivery

Delivery ensures that all the proper stakeholders have access to the right content, at the right time. Organizations must ensure they have a unified presentation platform for all legacy enterprise content — regardless of source and type — subject to enterprise-content search. Content and customizations must be propagated automatically from staging/testing through to production, in order to limit human error and promote a culture of governance. The ECM system must also ensure timely delivery of comprehensive reporting for management and compliance purposes.

AvePoint aids in delivery by:

- Utilizing SharePoint's web-based interface to provide secure access to the ECM system from any computer with Internet connectivity
- Scaling the ECM system with relative ease due to its distributed architecture, cost-effective licensing model, and automated deployment and application life-cycle management tools

- Exposing all networked file shares and legacy database content via SharePoint
- Analyzing and reporting upon all platform activity, content access/modification histories, and end-user activities for management analysis and compliance satisfaction
- Synchronizing SharePoint content among multiple farms to ensure delivery of the most up-to-date content to all end users



For more information about AvePoint, see www.avepoint.com.

GimmelSoft

Headquartered in Houston, Texas, GimmelSoft is the software product division of Gimmel, a Microsoft Gold-Certified Partner focusing on consulting, technology services, and SharePoint-compliant content and records management solutions. The company serves both corporate and government organizations. Founded in 2002, Gimmel was well known for completing Department of Defense (DoD) Directive 5015.2 for Records Management on SharePoint, and used this offering to launch GimmelSoft.

Records Management

Organizations historically have struggled to make enterprise-level electronic content and records management successful due to low user adoption. The increased complexity of traditional ECM packages combined with the records management training required to use them has been a barrier to enterprise adoption of these solutions. Moreover, even though the SharePoint platform provides the basic tools for records management, the sites are not centrally managed and governed, posing legal, efficiency, and infrastructure challenges.

GimmelSoft builds on SharePoint's ease of use and increasing end user adoption to provide central management and control of the information life cycle and the platform's records management functionality. Unlike the traditional ECM integrations to SharePoint, which call for duplication of content repositories and movement of content out of the SharePoint repository, GimmelSoft is a pure SharePoint solution, built using Microsoft technologies and leveraging all the power and versatility of the SharePoint platform.

Compliance

Department of Defense (DoD) standards have become widely adopted for records management applications in both government and commercial organizations. The DoD Directive 5015.2 is the specific directive that records managements systems strive to achieve. The GimmelSoft Compliance Suite addresses this directive by:

- Extending the records management capabilities of SharePoint 2010 without requiring a separate platform for content management of records.
- Enabling users, record managers, and administrations to create, move, and copy records. These features directly relate to the specific processes required to perform actions requested by the user within the constraints of the application in terms of integrity and requirements.

- Providing a single user interface, as shown in Figure 14-2, for defining the specific attributes and levels of the file plan structure, as well as the capability to push the structure into a set of SharePoint constructs such as libraries, folders, and information management policies.
- Providing the capability to classify e-mails and attachments as a single record in a single category or as separate records in potentially many categories or folders.
- Providing the capability to define and manage the events associated with a record's life cycle. These events may be defined as single events (e.g., sale of an asset, termination of an employee) or recurring (e.g., contract review).
- Including capabilities for cut-off and disposition processing, period management (periods of time to perform regular events), closing folders, and “as-of reporting.”
- Providing the capability to export selected records into a specified output format or import records into the system. The import and export functionality provided through transfers also includes the capability to extend the input/output formats and to map specific record attributes to this format.
- Providing functionality that handles records that have been enumerated as critical or “vital” to the organization. These capabilities include identifying, tracking, and systematically reviewing these records and their designation.
- Providing the ability to create bi-directional, parent-child and peer-to-peer relationships between two or more records.

The screenshot displays the 'File Plan Structure' management interface. On the left is a navigation pane with options: File Plan, Periods, Events, Disposition Instructions, Organization-Defined Fields, Legal Justifications, Reports, Export Tool, and Administration. The main area is titled 'File Plan Structure' and contains a table of nodes. Below the table are 'Add Item' and 'Edit Item' buttons. The 'Edit Item' dialog is open, showing fields for Name, ID, and Description.

| Name | ID | Description |
|-----------------------------|---------|-------------------------------------|
| Administration | ADM | Administration File Plan |
| Departmental Administration | ADM-010 | Departmental Administrative Records |
| General Correspondence | ADM-020 | General Correspondence Records |
| Internal Services | ADM-030 | Internal Services Records |
| Physical Security | ADM-040 | Physical Security Records |
| Project Management | ADM-050 | Project Management Records |
| Audit | AUD | Audit Records |

| General | |
|--------------|-------------------------------------|
| Name* | Departmental Administration |
| ID* | ADM-010 |
| Description* | Departmental Administrative Records |

FIGURE 14-2

Governance

SharePoint has a well-known reputation as a governance nightmare, although it's not necessarily the platform causing the problem but more likely its fast, widespread adoption without proper enterprise oversight. In a properly implemented records management system, governance plays an important role in providing consistent rules and policies, as well as a unified user experience. The GimmelSoft Governance Suite consists of the following capabilities to ensure information governance consistency and adoption within the SharePoint environment:

- A drop zone classification Web Part that extends the standard SharePoint drag and drop functionality. The drop zone classification Web Part enables users to assign a specific set of properties and a target location for those files that are dropped onto the Web Part location. When a user drags a file or files to the Web Part, specific rules, defined for the Web Part, are applied to the content. This process enforces property assignment consistency, and minimizes user input and governance directly through the user experience.
- An enhanced advanced search Web Part to leverage content types, site columns, and managed metadata directly from the point-and-click interface. Users can configure the displayed attributes for the results of the search and the available search scopes. This Web Part provides users with significant functionality for the displayed results through an integrated action menu and a source document library link for each item returned as part of a search.
- Enhanced templates to extend the definition of file format templates from a one-template-per-content type to supporting an unlimited number of file format templates per content type. This capability ensures that users are not limited by a specific content type format definition and have the capability to create a number of possible formats for any content type. In addition, the Enhanced Templates feature provides an architecture whereby templates can be managed within a central location and published to individual sites.
- Provisioning that enables farm administrators to work with their governance team to pre-define a set of provisioning definitions for new sites in the SharePoint farm.
- Metadata rules and inheritance to define global and local rules for how properties are defaulted and defined for a given content type or location. This functionality can extend the default value capability of SharePoint through the capability to define default value rules for specific content types deployed throughout the SharePoint farm. In addition, local rules may be defined to provide specific rules at the level of a site, a subsite, a document library, or a folder.



For more information about GimmelSoft, see www.gimmelsoft.com.

KnowledgeLake

Headquartered in St. Louis, Missouri, KnowledgeLake specializes in ECM products and solutions for the SharePoint platform. It focuses on delivering solutions for paper document problems in small, medium, and enterprise companies, as well as state, local, and federal government organizations. It accomplishes this by building easy-to-use, quality products to work with SharePoint. KnowledgeLake is able to exceed customer expectations with a dedicated services organization

that understands these types of problems and how to solve them, using technology from both KnowledgeLake and other ISVs.

KnowledgeLake was founded in 1999 by a group of ECM-focused entrepreneurs driven to provide lower cost, but not lower value, document imaging solutions. This idea propelled KnowledgeLake to be one of the first companies to the SharePoint ECM space, and the first company to market a document imaging solution on the SharePoint platform, in 2003.

Capture

KnowledgeLake Capture was the first paper-to-digital capture software for SharePoint Portal Server 2003 and Windows SharePoint Services 2.0. Since then, a complete capture platform has been built using in-house technologies, as well as best-of-breed software specializing in specific capture components. Today, KnowledgeLake Capture is fully integrated with the 2007/3.0 and 2010/4.0 SharePoint platforms. By focusing exclusively on SharePoint, KnowledgeLake has been able to integrate into the Microsoft platform much deeper than legacy capture software without an ECM repository focus.

KnowledgeLake facilitates the capturing of paper content with the following features:

- Integration with any TWAIN or ISIS-compliant capture device
- Automatic classification of documents using barcodes, checklists, and OMR/OCR/ICR technologies
- Automatic indexing (also known as tagging) via integration of external data through SharePoint Business Connectivity Services (BCS), as well as reading information off paper using barcodes and OMR/OCR/ICR technologies
- Using line-of-business (LOB) systems to drive the matching of information from paper to searchable document data

KnowledgeLake also facilitates electronic content capture with the following:

- Integration with most desktop applications for saving and retrieving documents directly to and from SharePoint
- Integration with Microsoft Outlook to quickly move data from Outlook or Exchange into SharePoint
- Integration with Microsoft Office applications (Word, Excel, PowerPoint) to enable content to be quickly indexed and saved to SharePoint

Viewing

Like paper capture, KnowledgeLake was also first to bring to market a web-based viewing application specifically for image data stored in SharePoint. Initially released using AJAX, today it uses Silverlight to bring a desktop-like experience to the web browser (see Figure 14-3). The KnowledgeLake SharePoint Viewer delivers documents to end users by offering all of the following capabilities:

- Views not only image data but PDF and Office document formats
- Enables users to edit PDF documents without purchasing and installing desktop software

- Encrypts portions of image documents that should be secured from specific groups of users
- Enables the editing of document properties within the viewer
- Adds markup capabilities to documents in order to assist with workflow tasks
- Enables users to bookmark specific pages to quickly find them in extremely large documents
- Provides an easy-to-use interface for documents throughout the workflow process
- Enforces the security rights applied to documents in SharePoint
- Eliminates the need to download documents to client machines, which presents both security and latency concerns

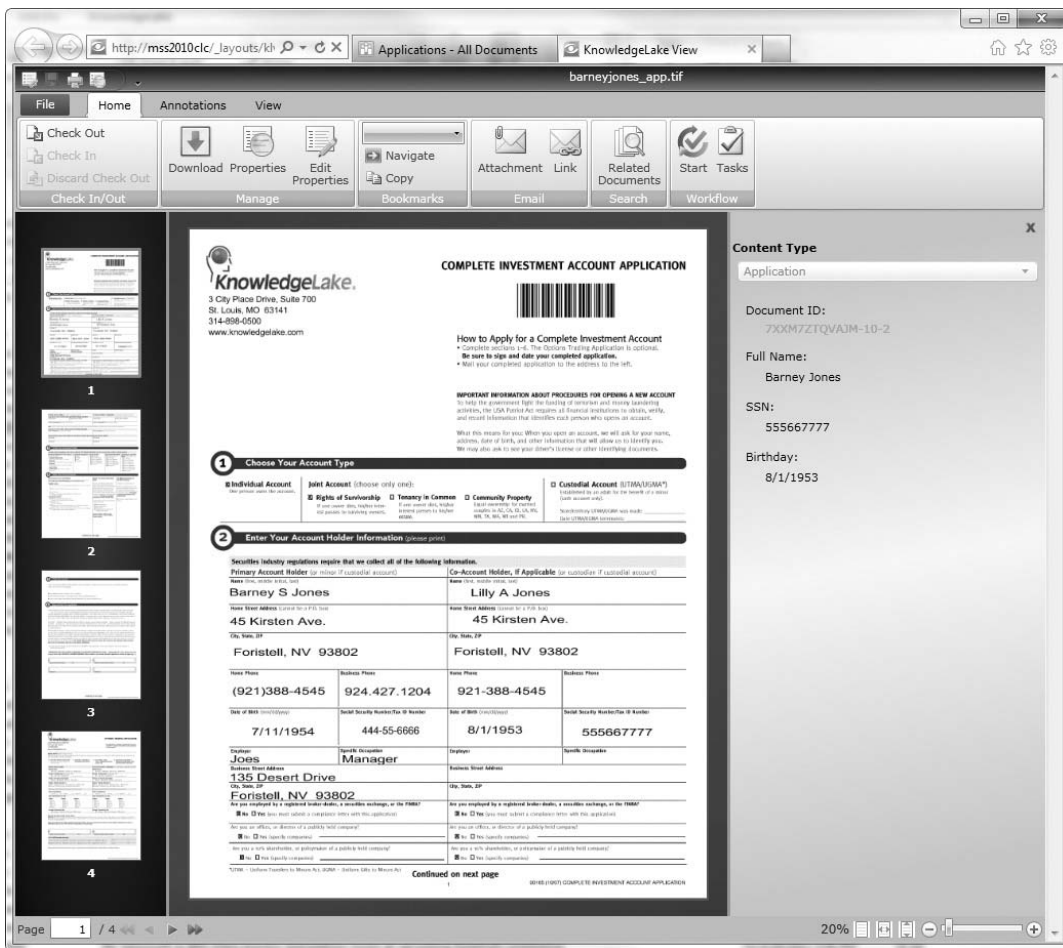


FIGURE 14-3

Searching

Although searching has always been a significant part of the SharePoint experience, finding an exact document instead of many relevant documents is essential to knowledge workers. KnowledgeLake,

again, was the first company to build an interface that enables users to search on column-level properties without the need for complicated and resource-intensive filters. Leveraging either SharePoint or FAST search data, KnowledgeLake search interfaces enable the following:

- Search on full-text and column-level data
- Queries can be saved and reused as Web Parts anywhere in the SharePoint user interface
- Search results can be grouped, sorted, and filtered in a simple-to-use interface

Workflow

Once information is efficiently captured into SharePoint, system implementers realize that speedy delivery of these documents to users is critical. As a result, additional software is needed to ensure that the right content is delivered to the right person at the right time. This is accomplished through workflow software designed for this specific purpose. However, dealing with transactional data such as invoices, new account applications, or loan processing paperwork requires a workflow solution that is more than just a rules engine and a designer. This type of transactional content requires what is often referred to as a *transactional content management system*. KnowledgeLake provides these capabilities through the following features:

- An inbox specifically tailored to work with task lists and document libraries used by SharePoint Workflow
- The capability to quickly configure how content should be handled at each step in a workflow, without writing custom code or building customer user interfaces
- A complete history and audit trail of the transactions a document undergoes, from the time it is captured until it reaches its final archival location



For more information about KnowledgeLake, you can browse to their website at www.knowledgelake.com.

Nintex

With headquarters in both the United States and Australia, Nintex is a leading global SharePoint ISV whose core products light up and extend the Microsoft SharePoint platform, addressing workflow and management challenges. As a Microsoft Gold-Certified Partner, Nintex is aligned with Microsoft's strategic and architectural direction to ensure that Nintex and Microsoft products work in harmony, both today and into the future.

Workflow

Chapter 4 dove deeply into the workflow features of SharePoint 2010, demonstrating how its out-of-the-box functionality can be leveraged to solve many business problems. Although the workflow experience in SharePoint has improved with each version, it is a feature that ISVs find crucial to improve upon.

Nintex Workflow is designed to enhance and extend, rather than replace, the SharePoint native workflow environment. The browser-based user interface takes full advantage of the SharePoint Fluent UI user experience. Nintex Workflow enhances SharePoint workflow by providing the following:

- Better usability with an intuitive drag-and-drop visual design environment (see Figure 14-4) within the SharePoint user interface. This single user interface makes moving from design to deployment a much smoother process and empowers SharePoint users to quickly automate their own business processes.
- Better in-process viewing with real-time workflow status reporting and tools to measure and improve your business processes.
- Better management of your workflow processes by including an approval process for changes to workflows, synchronizing these workflow processes across site collections, enabling long-term tracking as well as analysis, verbose logging, and step-through reporting.
- Full support for SharePoint Workflow capabilities.
- Connectivity using industry-standard protocols into Business Connectivity Services, Excel Services, Microsoft CRM, and others.
- Lazy Approval System to respond to requests in real language, even when users are mobile and without access to the SharePoint system.
- Rich workflow notifications using e-mail, instant messaging, and SMS.

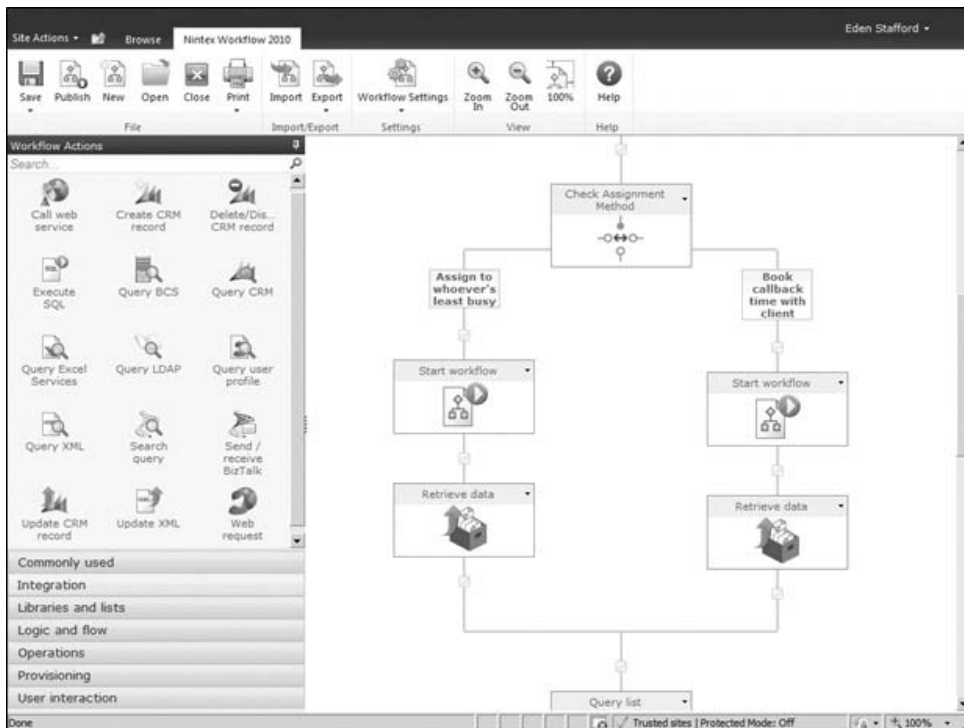


FIGURE 14-4

Manageability

Users new to the SharePoint social community don't have to be online very long to find complaints about SharePoint being difficult to manage, secure, or govern. Not that these are false reports or that SharePoint is the perfect platform, but typically these problems stem from poor planning, lack of policies, and rushed implementations — to name a few culprits. If you are new to the world of ECM and SharePoint is the first implementation you have been involved with, you should consider yourself lucky. Most other on-premise ECM systems require more training, more hardware, and more services for a successful implementation.

Although the SharePoint platform is an improvement to the legacy ECM systems of the past, manageability challenges remain; Nintex Analytics resolves and simplifies these challenges by providing the following:

- Insight into your SharePoint sites and content with reports of creation, inventory, usage, popularity, and growth
- Tools to report and analyze usage trends according to content creators, site activity, site popularity, search terms, CPU activity, and content type
- Extensive report customization
- Integrated reporting and analytics with Nintex Workflow

In the world of ECM, workflow is often document-centric. However, by using Nintex Workflow some of the manageability of your SharePoint system can be automated. Nintex Workflow does this by providing the following:

- Ability to query and update user profiles
- Ability to create, delete, and compile user audiences
- Active Directory account provisioning to add/remove users from groups, create/update/decommission accounts, and create/remove security groups
- Exchange Server user account provisioning
- Exchange Server integration for creating calendar appointments, creating Outlook tasks, and retrieving meeting suggestions



For more information about Nintex, see www.nintex.com.

SUMMARY

The Microsoft Partner Network has been a very successful program for both Microsoft and the thousands of partner members worldwide. When your organization participates in the network, it pays off financially by enabling you to reach new customers and gain access to software, training, marketing, and other materials.

The Microsoft ecosystem is the environment created when Microsoft and its partners rely on each other for continued growth and revenue, much like a biological ecosystem relies on its components for mutual survival. Independent software vendors (ISVs) play a major role in this ecosystem by creating products to round out existing Microsoft platforms with both mainstream and niche software.

The SharePoint ecosystem is made up of a subset of the larger Microsoft ecosystem and the communities that surround SharePoint, such as social networks, technical forums, user groups, and conferences. ISVs play a major role in the success of the SharePoint ecosystem by providing solutions to problems not currently resolved by SharePoint natively.

In the ECM space, many ISVs have found themselves very successful niches by completing solutions needed to create one of the most powerful ECM platforms in existence. Companies such as KnowledgeLake provide a complete document imaging system using SharePoint as the platform. Both Nintex and GimmalSoft enhance existing features such as workflow and records management. Others such as AvePoint provide complete enterprise storage and backup solutions for SharePoint and companies such as ABBYY bring their state-of-the-art OCR technology to enhance the paper document handling capabilities of the platform.

There are hundreds if not thousands of other ISVs in the SharePoint space, each advancing the platform both within and out of the ECM arena. If you are a software developer building solutions for SharePoint, you should take advantage of the ecosystem to build the necessary partnerships to reach more customers.

If you are an organization needing solutions to problems not handled by native SharePoint, there is a very comprehensive SDK available, but you should look into ISVs before setting out to resolve the problem on your own. They may save you money and net you a faster return on investment than going at it alone.

15

Guidance for Successful ECM Projects

WHAT'S IN THIS CHAPTER?

- Comparing legacy ECM and SharePoint ECM solutions
- Migrating content to a document library
- Managing SharePoint 2010 boundaries
- Defining a large repository upgrade strategy

Every now and then, a SharePoint ECM solution will be a brand-new “greenfield” deployment. These deployments are often encountered when the ECM needs of an organization have grown at a very slow pace, a new company is implementing an ECM solution, or perhaps management has been reluctant to trust the digital world with highly important customer documents. Greenfield solutions are great to work with because there are no existing expectations from the future user base and there usually isn’t an existing production system with functionality that needs to be duplicated. However, there is often some system in place, which means that it will likely be necessary to perform some sort of upgrade operation or content migration from a legacy system into a new SharePoint 2010 ECM solution.

This chapter begins by covering various techniques and protocols for importing content into SharePoint, followed by a few tips and tricks to avoid a few common issues that occur during import operations. Finally, the chapter covers a few key concepts regarding the best ways to upgrade an existing ECM content repository.

MIGRATING TO SHAREPOINT 2010

SharePoint has continued to evolve to the point where the platform is now wide enough and deep enough to truly become the “portal” for a wide range of mission-critical solutions in the enterprise. Even with SharePoint 2007, it was common to see organizations moving to the SharePoint platform as the portal solution of choice. Given the extensive improvements with SharePoint 2010, it is only natural to expect an increasing number of migrations from legacy or unstructured platforms to SharePoint. This section presents techniques for migrating content into SharePoint. Then, after learning how legacy system features can be expressed in a SharePoint solution, you’ll look at some solutions to a few of the common issues that can occur during a migration.

Migrating content into SharePoint is not performed with a few clicks of the mouse. It is a process that must be planned. Several concepts need to be addressed before the migration solution can begin.

Identifying Content for Migration

This concept may seem obvious, but it is often overlooked. Just because it is possible to migrate all of the existing content to SharePoint, that doesn’t always mean it is necessary. Many enterprises will often have very old content that is no longer relevant to the business. Also, it is common for end users to create documents that never end up being beneficial to business operations. It would be a waste of storage space to migrate any content that does not have business value.

Consider launching an effort to review content in the legacy system and sift out anything that is unnecessary. It is a great time to inventory content and eliminate anything irrelevant. This effort will also help determine the corpus profile, which can be used to drive information architecture and farm hardware requirements in later stages of preparation. Take care, however, to ensure that the review process and content housecleaning are decisive and efficient; otherwise, you risk dragging down the migration effort.

Extracting Content from the Source System

This is the wildcard question for every legacy solution. Some systems facilitate content export very easily and others are very difficult. It is usually best to work with a consulting firm that specializes in migration from a particular content source to SharePoint. There are many potential dangers that a migration novice won’t consider.

While it isn’t possible to provide export examples for every legacy system available, it is possible to provide general migration guidance for a few common content sources. In most cases, the best migration solution involves an export process in which document metadata is temporarily stored in a relational database that can then be used to drive import processing.

File Shares

Migrating from a file share is both a blessing and a curse. The good news is that it is relatively easy to devise a content migration from a file share into SharePoint. The bad news is that the source content is almost always unstructured. This makes it difficult to derive meaningful metadata that

can be used to categorize content such that it can be migrated appropriately and then located by search afterwards.

The best-case scenario is actually fairly common. Oftentimes, organizations will devise a hierarchical folder structure that essentially categorizes the content. These folder levels can be used to drive metadata for the child contents of each folder. In this paradigm, a utility application could be developed that recursively iterates through all files in the file share. For each file, the basic file metadata, as well as a full UNC path to the file, might be stored in a Microsoft SQL Server database. With the file paths in the database, it becomes possible to build SQL scrub scripts that parse the file paths and generate categorical metadata that can be stored in additional SQL Server columns for a given data row. A scrub script is a collection of SQL statements that parse available metadata to add new metadata or modify existing metadata. For example, the scripts may join in additional metadata from other lines of business systems or perhaps they may be used to clean illegal characters from existing metadata. In some cases, they can even be used to remove or alter data that doesn't conform to a specific pattern. For instance, different spellings of the same word may be standardized to support the migration of metadata values to a SharePoint choice list. In this case, the parent folder structure is flushed through a set of business rules, represented as SQL statements, that drive at least some meaningful metadata that can be used during import of each document.

Unfortunately, metadata can't always be generated from the parent folder hierarchy. In this case, the only solution is to have real people, possibly temporary employees, categorize the content based on document familiarity or by actually opening each file. This is indeed a daunting task; but keep in mind that the longer the process is delayed, the more content there will be to categorize later, so it is important to get started!

Categorization is essentially the process of determining the content type for each document. That is enough information to at least begin a migration. Once the content is in SharePoint, it is easy to set up views that users can use to determine which documents need additional metadata applied. They can then review those documents over time and apply additional metadata.

While content is being categorized, or even after it has been migrated to SharePoint with relatively little metadata, SharePoint Enterprise Search can full-text index most documents regardless of whether they are located on the file system or they have been migrated to SharePoint. Therefore, there are ways to locate content regardless of whether it is located on a file share or in SharePoint.

Internally Developed Document Management Solutions

Many organizations have internally developed a document management system that has capable internal developers. These solutions usually employ Microsoft SQL Server or other relational databases to store metadata and drive a simple search system while document content is stored on a server file system.

These solutions are usually the easiest legacy systems to migrate to SharePoint. Metadata is usually available and the developers usually have a good idea of how the solution database is structured. This information can be used to construct database scripts or import jobs that flatten document data for storage in a migration database. This migration database can then be used to drive an import into SharePoint. In this scenario, the migration success rate is typically very high due to the minimal complexity of the source system.

Other Legacy Document Management Solutions

The more complex migrations typically involve legacy document management system (DMS) content sources. These solutions are often “closed,” meaning internal database tables and database relationships are rarely published. Furthermore, trying to get additional information from the vendor so that a migration away from their platform can be performed is rarely successful.

The best-case scenario is a DMS system with published APIs that support content export. These APIs can often be used by a utility application to iterate through logical documents in order to populate a migration database. Alternatively, it may be possible to interrogate the DMS back-end database structure in order to determine the relevant relationships. This information can then be used to construct database scripts or import jobs that flatten document data for storage in a migration database. Again, the migration database will drive content import processing.

The worst-case scenario is a DMS system that is completely closed, with no usable API for content access and an extremely complex database structure that is very difficult to decipher. In this case, it may even be possible to damage the solution if attempts are made to manipulate the DMS engine in order to export content. In this situation, using a consulting firm that has experience migrating from this specific DMS platform offers the best chance of success.

Preparing Content for Importing

The export concepts just discussed alluded to the use of a Microsoft SQL Server or other relational migration database to drive import processing. This isn’t absolutely necessary, but it is usually the most effective intermediate destination for flattened document metadata. From this location, it is possible to construct scrub scripts that can be used to manipulate the existing metadata before the content is uploaded to SharePoint. There are several reasons why metadata might need to be scrubbed.

Setting the Content Type

The source content has likely been categorized in some way by this point. Therefore, it makes sense to use this category information to set a specific content type for each document. This content type can be used during the upload process to ensure that appropriate metadata fields are exposed for each document. The content type can also be used by the Content Organizer to drive the final destination of a given document.

In some cases, the content type may need to be changed. It is commonplace to identify a set of documents that need an alternate content type based on specific metadata associated with the documents. The scrub scripts can be designed to take this into account in order to redirect this subset of documents to an alternate content type.

Metadata Merge

It is also a common scenario to use the metadata from a line-of-business (LOB) system to further enhance document metadata. In many cases, you can use one or more existing metadata fields from the migration database to look up additional details regarding each document. These details can

then be merged into the metadata database as additional column information. This information can then be passed into SharePoint during the upload process.

Controlling the Import Process

Another excellent use of the scrub scripts is to use document metadata to prioritize the order in which content is imported into SharePoint. These scripts can be customized to set an “order by” field value based on a predetermined set of business rules. This is a useful technique for organizations that need recently created or modified documents to be available in SharePoint quickly, allowing the migration to continue for older content later.

General Metadata Cleanup

All scrub scripts should include commands for processing important metadata that might contain illegal characters. Offending characters in the metadata of a given document will cause the SharePoint upload of that item to fail. You can proactively avoid these failures by ensuring that illegal characters in metadata are either stripped or replaced by legal characters.

Scrub scripts should also include commands that ensure that required values are available for required fields. If certain fields will be required, then the scrub scripts need to insert a migration-specific value such as [unknown] or [not available] in key required fields. Alternatively, the fields can be set to “Optional” in SharePoint during the migration and then changed back to “Required” after the migration.

Scrub scripts can also be used to clean up values that will be imported into an options list. In the source DMS system, users may have been allowed free-text metadata entry. These same values should be cleaned up and standardized into a single valid value. For example, one user might enter a state name of “IL” while another user enters a state name of “Illinois” and a third user enters a state name of “ILLINOIS.” These entries could be standardized — for example, to “IL” — to create consistency in a list of options for a State field.

Importing Content into SharePoint

Continuing with the general migration process, you need an application or process that uses the information in a migration database, or possibly even simple metadata XML documents stored on the file system, to drive the SharePoint content upload process.

For large migrations that will move hundreds of thousands or millions of documents to SharePoint, sophisticated multi-threaded migration utilities will best serve to facilitate the migration. These utilities are the result of a significant time investment and development effort to ensure that content is sent into SharePoint reliably and as quickly as possible. As they say, “time is money.” When there are 10 million documents to migrate, a simple utility application consisting of 50 lines of code is probably not the most cost-effective way to execute the migration. For a migration of this magnitude, a migration expert should be consulted.

For smaller migrations, if skilled developers are available, it may be cost effective to build the migration utility internally. The next section includes code samples that can be used as the starting point of a solution that facilitates uploading single-version and multiple-version documents into SharePoint.

Protocols for Importing Content

Once the documents have been exported from the source system and the metadata has been scrubbed, several techniques can be used to upload content to a SharePoint 2010 document library. Documents can be uploaded to a Drop Off Library or they can be directly uploaded to a specific document library destination. The following sections describe and provide examples for several ways to send your valuable content to SharePoint, regardless of its final destination.

Web Services

Perhaps the easiest way to upload documents to a SharePoint 2010 library from a remote client is to take advantage of the `copy.asmx` web service. Fortunately, this service can do more than just copy or move a document from one place in SharePoint to another. It can also be used to upload a new document along with related metadata. The advantage of using the `copy.asmx` web service is that the application can be run from any computer that has network access to the SharePoint server. The disadvantage of the web service upload technique is that it will run slower than a SharePoint Server object model solution running directly on a SharePoint server. Listing 15-1 demonstrates the process of using an XML metadata file to upload a document using the `copy.asmx` web service.



LISTING 15-1: Using `copy.asmx` to upload a document

Available for
download on
Wrox.com

```
private void importDoc(FileInfo metaFile)
{
    string filePath = string.Empty;
    string fileName = string.Empty;
    string destUrl = string.Empty;

    // Initialize web service
    string serviceUrl = siteUrl + @"_vti_bin/copy.asmx";
    WSCopy = new WSCopy.Copy();
    WSCopy.Url = serviceUrl;
    WSCopy.Credentials = System.Net.CredentialCache.DefaultCredentials;

    // The FieldInformation object array is constructed
    // using the field values in the metadata Xml file.
    WSCopy.FieldInformation[] fieldInfo = getFieldInfo(metaFile, out
    filePath, out fileName);

    // In this example, we will assume that the library name has not been
    // altered since the library was originally created. To guarantee that
    // the library Url is correct, it is necessary to invoke the Lists.asmx
    // web service and call to SharePoint for a list definition. This
    // definition will include a URL for the document library that can be
    // trusted.
    destUrl = siteUrl + libName + @"/" + fileName;

    byte[] fileBytes = getFileBytes(filePath);

    WSCopy.CopyResult[] copyResults = null;

    // This is the method call that actually uploads the document (with
    // metadata) to SharePoint. Note that because we're not actually copying
```

```

// a document from one SharePoint location to another, the SourceUrl
// parameter should be populated with a single space (" "). The call
// will fail if an empty string or null is passed in.
UInt32 completed = wsCopy.CopyIntoItems(" ", new string[] {destUrl},
    fileInfo, fileBytes, out copyResults);

if (completed != 0)
    throw new Exception("Unable to complete the upload request for document: "
        + destUrl);

if (copyResults == null)
    throw new Exception("Unable to verify the successful upload of document: "
        + destUrl + ". Manual verification is recommended.");

if (copyResults[0].ErrorCode != WSCopy.CopyErrorCode.Success)
    throw new Exception("An error occurred during upload for document: "
        + destUrl + ". Error: " + copyResults[0].ErrorMessage);

// By adding the ".processed" extension to the file, we ensure that
// previously uploaded documents are not uploaded again during subsequent
// executions. This also provides a rudimentary "resume" capability.
metaFile.MoveTo(metaFile.FullName + ".processed");
}

```

Even though the fully functional project code can be downloaded, it is helpful to understand how the source files (see Figure 15-1) are uploaded to SharePoint.

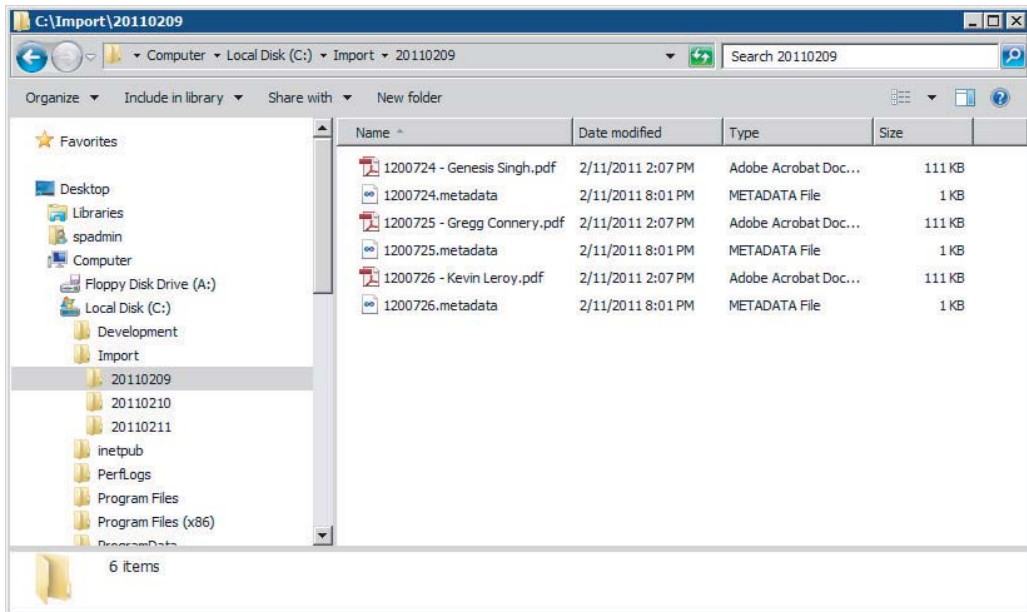


FIGURE 15-1

The application uses a special import XML file to drive the import process (see Figure 15-2). For larger-scale migrations, it is much more efficient and reliable to use a centralized migration database to drive the migration.

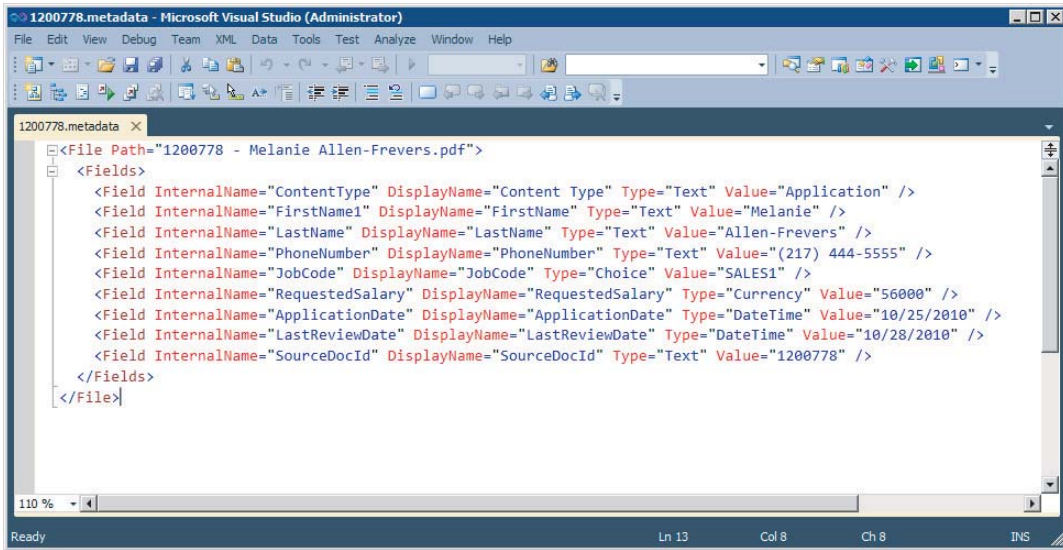


FIGURE 15-2

In this example, the utility constructs an array of property values and uploads them to SharePoint along with the document binary using the `copy.aspx` web service (see Figure 15-3).

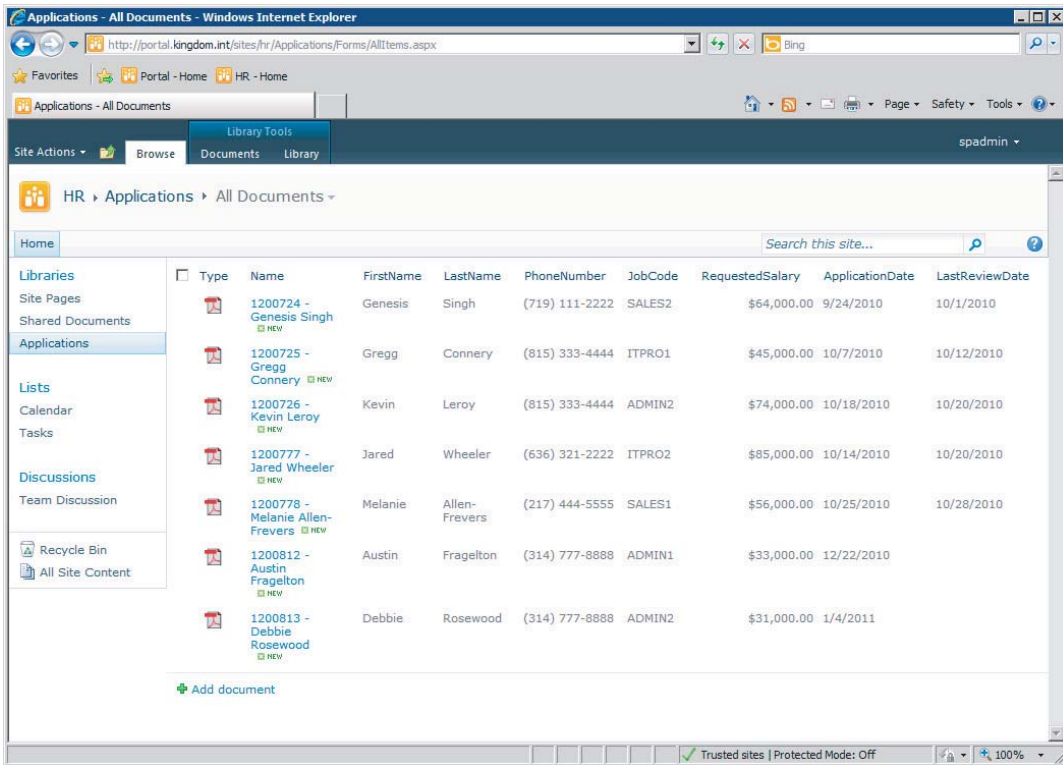


FIGURE 15-3

The application uses `System.Diagnostics.Trace.WriteLine()` method calls to send information messages to a trace listener. Using the SysInternals DebugView application from Microsoft, it is easy to monitor migration progress and save migration processing audit trail information for long-term archiving (see Figure 15-4).

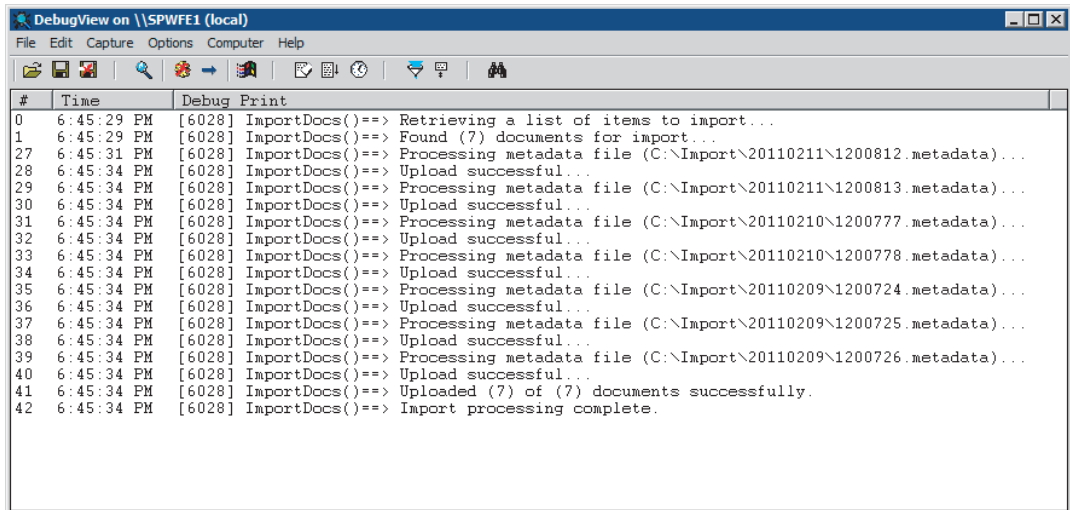


FIGURE 15-4

SharePoint Server Object Model

Using the SharePoint Server object model in a .NET managed code utility application is a very easy way to upload content to SharePoint. The advantage of this solution is that the upload will operate at a faster rate than a solution based on web services. The disadvantage is that the application must run on a SharePoint server in the farm. This is often not an acceptable compromise. Listing 15-2 demonstrates a SharePoint Server object model–based solution.

In this example, after the XML metadata file is parsed, the utility opens a reference to the appropriate SPSite object and then the necessary SPWeb object. In the SharePoint Server object model, the SPWeb object represents the site that contains the destination library. After a reference to the destination library is made, the utility uploads the document to the root folder (SPFolder) of the library. Once the file has been uploaded, the utility populates the metadata values into the SPLListItem associated with the new SPFile. Finally, the Update() method must be called on the SPLListItem to ensure that the metadata is saved to the SPLListItem.

LISTING 15-2: Using the SharePoint Server object model to upload a document

```
private void importDoc(FileInfo metaFile, string siteUrl, string libName)
{
    string filePath = string.Empty;
    string fileName = string.Empty;
    string destUrl = string.Empty;
    SPFile spFile = null;
```



continues

LISTING 15-1 *(continued)*

```

XmlDocument metaXml = new XmlDocument();

metaXml.Load(metaFile.FullName);

filePath = getFilePath(metaXml, metaFile.DirectoryName);

FileInfo sourceFileInfo = new FileInfo(filePath);
if (!sourceFileInfo.Exists)
    throw new Exception("A document could not be found at location:"
        + filePath);

fileName = sourceFileInfo.Name;

destUrl = siteUrl + libName + @"/" + fileName;

byte[] fileBytes = getFileBytes(filePath);

// Always be sure to properly dispose of SPSite and SPWeb objects
using (SPSite site = new SPSite(siteUrl))
{
    using (SPWeb web = site.OpenWeb())
    {
        SPList list = web.Lists[libName];

        spFile = list.RootFolder.Files.Add(destUrl, fileBytes);

        spFile.Item["Title"] =
            sourceFileInfo.Name.Replace(sourceFileInfo.Extension, "");

        XmlNodeList fields =
            metaXml.DocumentElement.SelectNodes("Fields/Field");
        object itemVal = null;
        string itemAttr = string.Empty;

        if (fields != null && fields.Count > 0)
        {
            for (int i = 0; i < fields.Count; i++)
            {
                XmlElement fieldElem = (XmlElement)fields[i];

                itemAttr = getAttributeVal(fieldElem, "DisplayName", "");
                itemVal = getFieldValue(fieldElem);

                if (itemVal != null && itemVal.ToString().Length > 0)
                    spFile.Item[itemAttr] = itemVal;
            }
        }
        spFile.Item.Update();
    }
}
}

```

After the utility uploads all documents to SharePoint, notice in Figure 15-5 that the Modified By field shows “System Account.” In order for this to work, the import utility application must be executed as the same account as the application pool account for the web application that hosts the upload site.

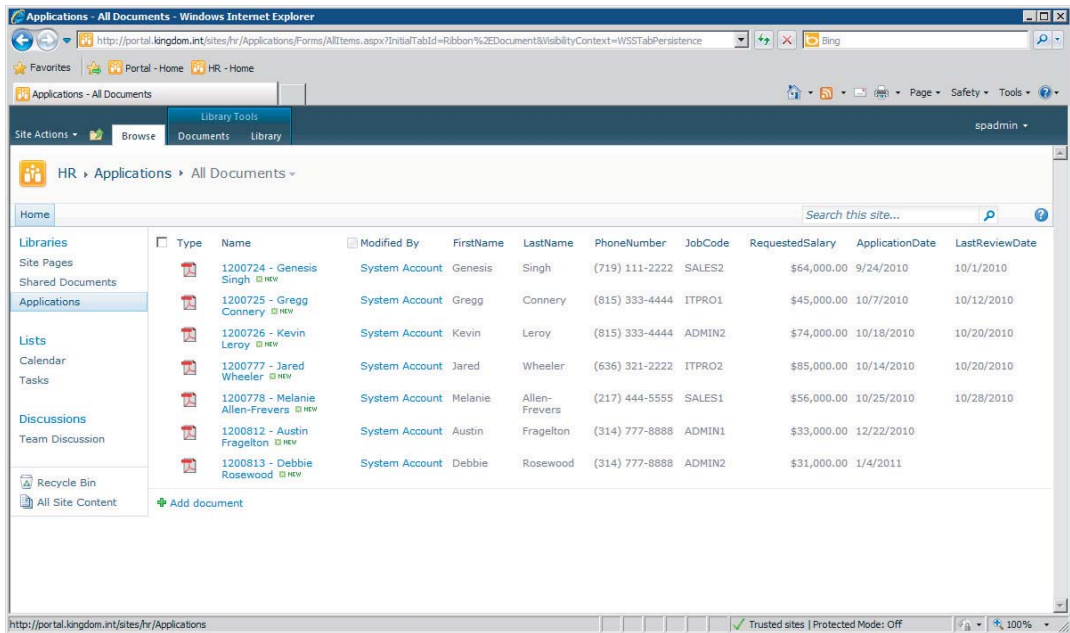


FIGURE 15-5

FrontPage Remote Procedure Calls (FPRPC)

Front Page Remote Procedure Calls (FPRPC), or just (RPC) for short, is by far the most complex technique for uploading to SharePoint 2010. In order to use RPC to upload a document, the developer must carefully construct a complex call and then execute an HTTP POST to `[SiteURL]/_vti_bin/_vti_aut/author.dll`. As part of the call, the document binary is streamed to SharePoint. SharePoint will process the call and return a response in the form of an HTML document that needs to be parsed in order to determine the success or failure of the upload.

For small migrations, using this technique to upload content to SharePoint is not cost effective. The time to develop a working solution will be significant. However, if the time and effort are cost justified, there are significant benefits to using RPC for migrations. RPC is capable of uploading a document to SharePoint and setting the metadata in a single call. Using RPC, it is also possible to create subfolder structure in libraries on-the-fly during document uploads. This is very handy when it is necessary to upload documents to specific folders based on metadata. Another advantage is that RPC can be used from a remote client workstation. Most important, however, RPC is easily the fastest and most efficient protocol for uploading content to SharePoint.

Protocols for Updating SharePoint Content

During a migration, it may be necessary to update existing documents in SharePoint after they have been initially uploaded. The most common reason to update existing documents is to recreate, in SharePoint, document version structures that existed in the legacy system. A less common reason might be to update documents in SharePoint with new metadata that was generated in a LOB system.

Regardless of the reason, this section provides a couple of examples demonstrating how to update an existing document with new binary and/or metadata content.

SharePoint Server Object Model

The SharePoint Server object model provides the easiest and fastest way to update existing content in SharePoint. Once again, the disadvantage is that the application must run on a SharePoint server in the farm. Listing 15-3 demonstrates a code solution that could be used to add a new version to an existing document in SharePoint using the Server object model.



LISTING 15-3: Using the SharePoint Server object model to update a new document version

Available for
download on
Wrox.com

```
public void UpdateDoc(string UpdateMetaPath)
{
    SPSite site = null;
    SPFile spFile = null;
    string filePath = string.Empty;
    string updateUrl;

    updateMetaPath = UpdateMetaPath;

    traceMsg("UpdateDoc()", "Retrieving the update metadata file ...");
    FileInfo metaFileInfo = new FileInfo(updateMetaPath);

    XmlDocument metaXml = new XmlDocument();
    metaXml.Load(updateMetaPath);

    updateUrl = getAttributeVal(metaXml.DocumentElement, "UpdateUrl", "");
    if (updateUrl.Trim().Length.Equals(0))
        throw new Exception("UpdateUrl value must exist in the metadata file.");

    filePath = getFilePath(metaXml, metaFileInfo.Directory.FullName);
    FileInfo updateFileInfo = new FileInfo(filePath);
    if (!updateFileInfo.Exists)
        throw new Exception("Source file [" + filePath + "] could not be found!");

    // Be sure to properly dispose of SPSite and SPWeb objects as soon as
    // possible.
    // This code will be executed in the context of the logged on user.
}
```

```

using (SPSite site = new SPSite(updateUrl))
{
    using (SPWeb web = site.OpenWeb())
    {

        spFile = web.GetFile(updateUrl);
        spFile.CheckOut();

        byte[] fileBytes = getFileBytes(filePath);

        traceMsg("UpdateDoc()", "Uploading new document version to: " +
            updateUrl);
        spFile.SaveBinary(fileBytes);

        traceMsg("UpdateDoc()", "Updating document metadata...");
        spFile.Item["Title"] =
            updateFileInfo.Name.Replace(updateFileInfo.Extension, "");

        XmlNodeList fields =
            metaXml.DocumentElement.SelectNodes("Fields/Field");
        object itemVal = null;
        string itemAttr = string.Empty;

        if (fields != null && fields.Count > 0)
        {
            for (int i = 0; i < fields.Count; i++)
            {
                // Be sure to grab the correct (zero based) element node.
                XmlElement fieldElem = (XmlElement)fields[i];

                itemAttr = getAttributeVal(fieldElem, "DisplayName", "");
                itemVal = getFieldValue(fieldElem);

                if (itemVal != null && itemVal.ToString().Length > 0)
                    spFile.Item[itemAttr] = itemVal;
            }
        }
        spFile.Item.Update();
        spFile.CheckIn("Updated using ServerAPIVersionUpdater.");
    }
    traceMsg("UpdateDoc()", "Version update processing complete.");
}
}

```

After opening the appropriate SPSite and SPWeb objects, the code acquires a reference to the existing document using the GetFile() method of the SPWeb object. The file is then checked out, updated, and checked back in. This technique is used when the binary document, the item metadata, or both need to be updated. Notice in Figure 15-6 that DebugView again shows the success or failure of each operation related to the update process.

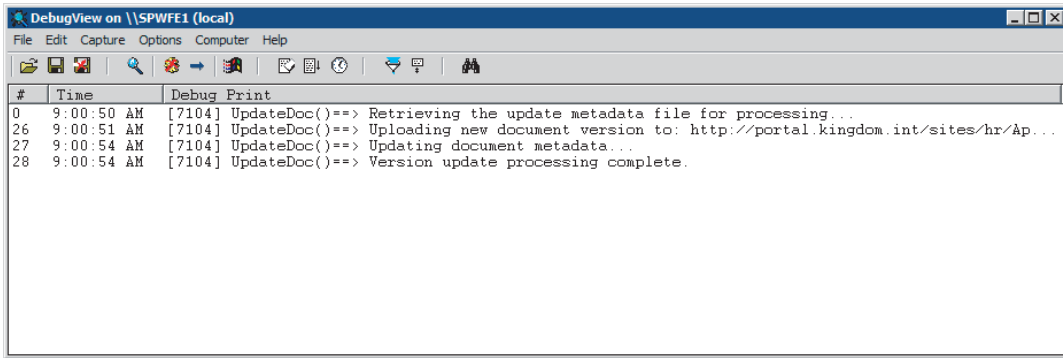


FIGURE 15-6

After DebugView indicates success, refreshing the document library page indicates that some of the metadata has been changed, and the version number has been updated to 3.0 for one of the documents (see Figure 15-7).

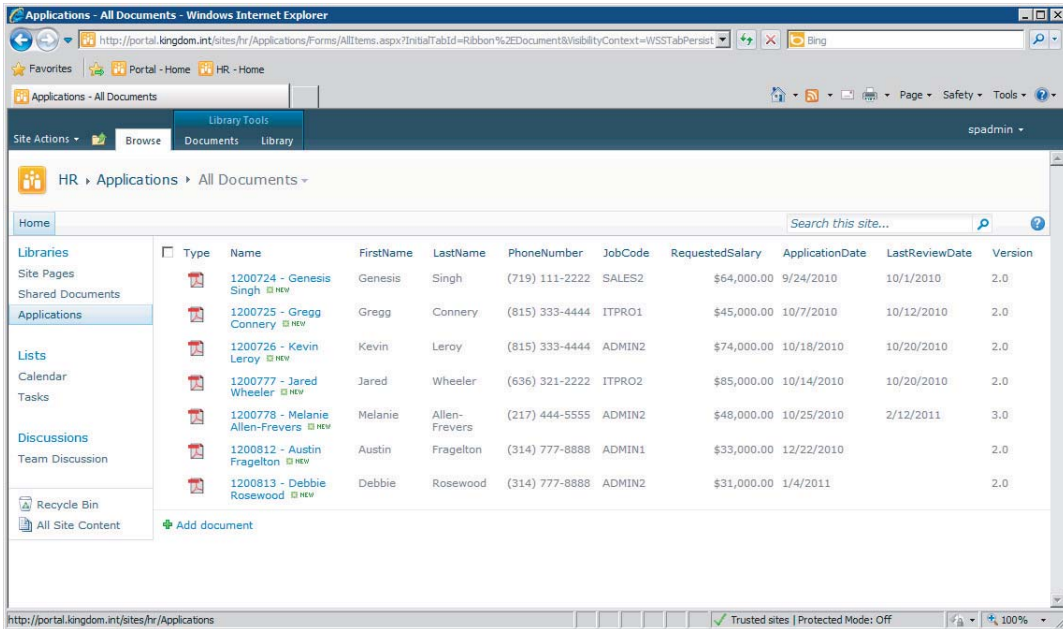


FIGURE 15-7

Viewing the version history for the document shows that four metadata fields have been updated and the binary file is now almost 200KB larger. The version comments indicate that the ServerAPIVersionUpdater utility application successfully updated the document version (see Figure 15-8).

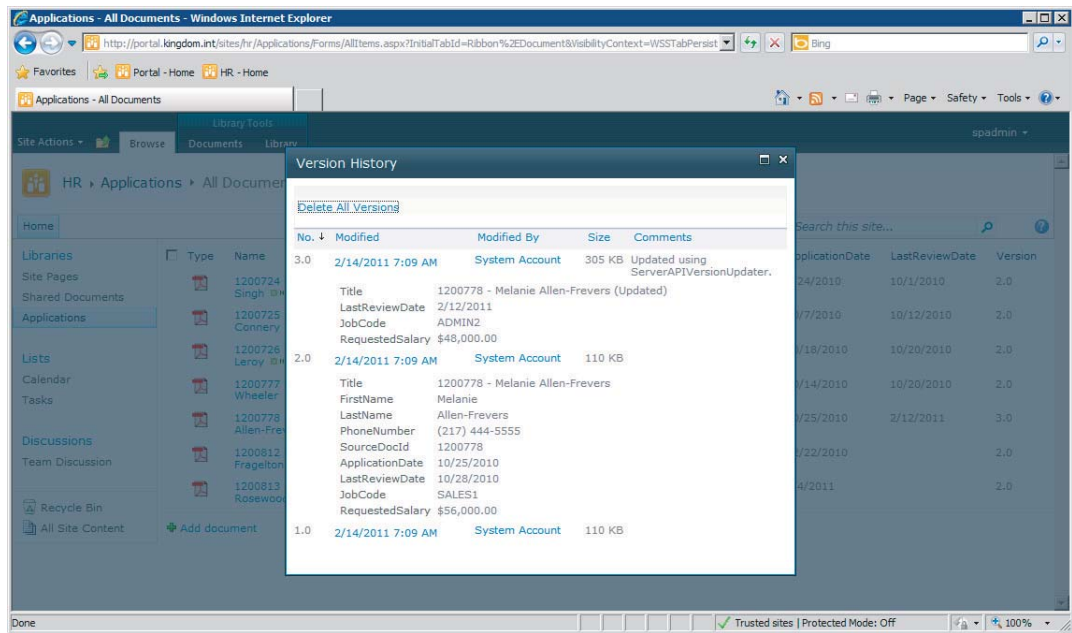


FIGURE 15-8

SharePoint Client Object Model

Using the SharePoint client object model is a little more difficult but definitely a viable way to update existing content in SharePoint. The advantage of this solution is that the application does not have to run on a SharePoint server in the farm. Also, because operations can be queued and executed in a batch, processing can be more efficient than other remote management coding techniques such as using web services. Listing 15-4 demonstrates adding a new version to an existing document in SharePoint using the SharePoint client object model.



LISTING 15-4: Using the client object model to update a new document version

Available for
download on
Wrox.com

```
public void UpdateDoc(string UpdateMetaPath, ICredentials Creds)
{
    Microsoft.SharePoint.Client.File spFile = null;
    List list = null;
    string siteUrl = string.Empty;
    string libName = string.Empty;
    string updateSourcePath = string.Empty;
    string updateRelativeUrl = string.Empty;
    string newContentTypeName = string.Empty;

    traceMsg("UpdateDoc()", "Retrieving update metadata file for processing...");
```

continues

LISTING 15-4 *(continued)*

```

FileInfo metaFileInfo = new FileInfo(UpdateMetaPath);

XmlDocument metaXml = new XmlDocument();

metaXml.Load(UpdateMetaPath);

// Load the update information variables using the update metadata Xml
// document
getUpdatePathInfo(metaXml, metaFileInfo.Directory.FullName, out siteUrl,
    out libName, out updateRelativeUrl, out updateSourcePath,
    out newContentTypeName);

FileInfo updateFileInfo = new FileInfo(updateSourcePath);
if (!updateFileInfo.Exists)
    throw new Exception("Update source file ["
        + updateSourcePath + "] could not be found!");

traceMsg("UpdateDoc()", "Acquiring context for site: " + siteUrl +
    "...");
using (ClientContext context = new ClientContext(siteUrl))
{
    spFile = context.Web.GetFileByServerRelativeUrl(updateRelativeUrl);

    list = context.Web.Lists.GetByTitle(libName);

    // Get a reference to the content type specified in the metadata
    // update document.
    context.Load(list.ContentTypes);
    IEnumerable<ContentType> newContentTypes =
        context.LoadQuery(list.ContentTypes.Where(c => c.Name ==
            newContentTypeName));

    context.Load(spFile);
    context.Load(spFile.ListItemAllFields);
    spFile.CheckOut();

    traceMsg("UpdateDoc()", "Checking out list item and file at: "
        + updateRelativeUrl + "...");
    // Nothing actually happens until we "Execute".
    context.ExecuteQuery();

    ContentType newContentType = newContentTypes.First();

    // We can now begin updating now that all required objects have been
    // populated.
    traceMsg("UpdateDoc()", "Updating document metadata...");
    spFile.ListItemAllFields["Title"] =
        updateFileInfo.Name.Replace(updateFileInfo.Extension, "");

    // Handle content type changes a bit differently than just a field
    // update.

```



```

if (spFile.ListItemAllFields.ContentType != newContentType)
    spFile.ListItemAllFields["ContentTypeId"] = newContentType.Id;

XmlNodeList fields =
    metaXml.DocumentElement.SelectNodes("Fields/Field");
object itemVal = null;
string itemAttr = string.Empty;

if (fields != null && fields.Count > 0)
{
    for (int i = 0; i < fields.Count; i++)
    {
        XmlElement fieldElem = (XmlElement)fields[i];

        // Be sure to use internal names when updating using the
        // Client Object Model.
        itemAttr = getAttributeVal(fieldElem, "InternalName", "");
        itemVal = getFieldValue(fieldElem);

        if (itemVal != null && itemVal.ToString().Length > 0 &&
            itemAttr != "ContentType")
            spFile.ListItemAllFields[itemAttr] = itemVal;
    }
}

// Don't forget to call the update.
spFile.ListItemAllFields.Update();

traceMsg("UpdateDoc()", "Uploading new file version from: "
    + updateSourcePath + "...");
byte[] fileBytes = getFileBytes(updateSourcePath);
FileSaveBinaryInformation binInfo = new FileSaveBinaryInformation();
binInfo.Content = fileBytes;

spFile.SaveBinary(binInfo);

traceMsg("UpdateDoc()", "Checking in...");
spFile.CheckIn("Updated by ClientAPIVersionUpdater.",
    CheckinType.MajorCheckIn);

traceMsg("UpdateDoc()", "Executing changes...");
context.ExecuteQuery();

traceMsg("UpdateDoc()", "Version update processing complete.");
}
}

```

As before, the utility is executed, and iterates through XML metadata, using the information to drive the version update process. `DebugView` is again used to show additional information regarding the stages of the update process. Using the client object model can be tricky due to the `Load` and `ExecuteQuery` code pattern and the concept of executing these commands using a previously opened `ClientContext` object. The additional trace messaging provided by `DebugView` can be very helpful (see Figure 15-9).

```

DebugView on \\SPWF1 (local)
File Edit Capture Options Computer Help
# Time Debug Print
0 12:48:42 PM [2400] UpdateDoc()==> Retrieving the update metadata file for processing...
1 12:48:42 PM [2400] UpdateDoc()==> Acquiring context for site: http://portal.kingdom.int/sites/hr...
2 12:48:42 PM [2400] UpdateDoc()==> Checking out list item and file at: /sites/hr/Applications/1200778%20-%20Melanie...
3 12:48:44 PM [2400] UpdateDoc()==> Updating document metadata...
4 12:48:44 PM [2400] UpdateDoc()==> Uploading new file version from: C:\UpdateSource\1200778 - Melanie Allen-Frevers...
5 12:48:44 PM [2400] UpdateDoc()==> Checking in...
6 12:48:44 PM [2400] UpdateDoc()==> Executing changes...
7 12:48:45 PM [2400] UpdateDoc()==> Version update processing complete.

```

FIGURE 15-9

Similar to the ServerAPIVersionUpdater, the version history for the document shows the updated metadata fields and larger document size. But this time, the version comments indicate that the ClientAPIVersionUpdater utility application successfully updated the document version (see Figure 15-10).

| No. | Modified | Modified By | Size | Comments |
|-----|--------------------|----------------|--------|-------------------------------------|
| 3.0 | 2/15/2011 10:48 AM | spadmin | 305 KB | Updated by ClientAPIVersionUpdater. |
| 2.0 | 2/12/2011 6:11 PM | System Account | 110 KB | |
| 1.0 | 2/12/2011 6:10 PM | System Account | 110 KB | |

FIGURE 15-10

Mapping Legacy ECM Features to SharePoint Solutions

SharePoint 2010 is an extremely comprehensive solution platform. A single SharePoint 2010 farm can replace a document imaging system, document management system, transactional content processing system, corporate intranet portal, public internet website, and document collaboration

portal — just to name a few. Thanks to workflow, custom development hooks, and alerts systems, it can also be the basis for a fully functional line-of-business system.

The biggest advantage of moving to a SharePoint 2010 platform for enterprise content management is to realize a significant reduction in *total cost of ownership* (TCO). Even though SharePoint can effectively supplant several legacy solutions, in many cases replacing just one of them with SharePoint 2010 can result in an immediate licensing and maintenance cost savings. Once this concept is understood, the next step is to map legacy ECM solution features to SharePoint 2010 features to determine whether a replacement effort will result in reduced functionality to business users. This section discusses the features of a traditional ECM solution and how they might be implemented in SharePoint 2010.

Document Versions

As discussed earlier, several legacy document management solutions provide versioning functionality in order to ensure that deprecated content that has been removed or replaced can be retrieved and viewed later. This is often even a regulatory requirement.

SharePoint 2010 supports major and minor version check-out and check-in capabilities, and adds publishing functionality. Users may have the rights to check out, modify, and check in a document; but final approval and publishing can be limited to one or more specific individuals. The versioning features in SharePoint 2010 are supported for both manual end-user operations as well as in the API (see Listings 15-3 and 15-4 for code samples). Because of the API-level support for versioning, it is usually possible to migrate all versions of a document from a legacy system to a SharePoint 2010 document library.

Metadata-based Security

Over the years, developers have devised custom event receivers that monitor document metadata when documents are added or modified and attempt to manipulate permissions on the item. This is not a recommended practice. It may be possible to get away with this on very small-scale solutions, but solutions of any reasonable size will likely experience performance degradation.

As defined by Microsoft, a *scope* is the security boundary for a securable object and any of its children that do not have a separate security boundary defined. If permissions are set on an individual item in a list or library, that item will have its own security scope. There is a built-in limit of 50,000 security scopes per list or library. Although this limit can be increased, it is generally not recommended.

There is a very practical solution to this limitation. The best way to implement metadata-based security is to upload content to folders that are named based on metadata. For example, if document processors were only allowed to see documents that belong to a specific country, then a required metadata field named “CountryCode” might be included in a document content type. The Content Organizer can then use the CountryCode metadata value to automatically shuffle uploaded documents into the appropriate folder whose name is the same as the CountryCode metadata value, such as US, GB, or NL. Permissions can then be set on these folders to limit document access to a security group that is defined for each country code. Effectively, the CountryCode value of the document determines who is allowed access to the document, which is essentially metadata-based security.

Document Protection and Rights Management

Through Microsoft Information Rights Management (IRM), SharePoint 2010 is able to control not only access to a document, but also what the end user is able to do with the document after it has been downloaded. For example, a user may be allowed to view a document but not print it or run VBA or other custom code in the file. These actions are governed by an *IRM protector* that is registered for specific file types. SharePoint can also prevent files from being uploaded when an appropriate IRM protector doesn't exist.

Annotations and Redaction

In some legacy ECM solutions, annotations are used to enhance or highlight information in a given document. These annotations can either be “floating,” such that they are layered above the document, or, in the case of images, they can be “burned in” to become a permanent part of the document.

Most annotation solutions are proprietary to the legacy platform, but in many cases they expose the annotation information through an SDK of some sort. Annotations are most relevant to image-based documents. Using the Tagged Image File Format (TIFF), annotation information can be embedded in the tags of the TIFF header. Therefore, if the legacy platform can provide annotation information, the annotations can be recreated in the header of a TIFF image. SharePoint 2010 does not provide a TIFF viewer that supports annotations, but this capability has been provided by third-party solutions.

Alternatively, if burning the annotations into an image is an option, it is possible to permanently add annotation data, provided by the legacy platform, to an image using an imaging toolkit.

Search

The topic of searching in a SharePoint-based ECM solution is discussed at length in Chapter 6. Typically, legacy ECM search solutions provide either metadata-only searches or a combination of metadata and full-text search functionality. Both SharePoint 2010 Enterprise Search and FAST Search for SharePoint 2010 enable users to search content using keywords (full-text) or specific metadata.

Scanning and Indexing

In the context of an ECM document management system, the term *indexing* refers to the process of applying metadata to a document — either manually or programmatically. Most legacy ECM solutions do not have a core component that provides document scanning and indexing functionality. Scanning and indexing solutions are often add-on components supplied by third-party vendors. The same paradigm exists with SharePoint 2010. SharePoint provides extensive functional capabilities in terms of content storage and management. However, as with most legacy platforms, SharePoint 2010 requires the addition of a third party when the solution includes requirements for scanning and indexing.

Records Retention and Disposition

Many legacy ECM solutions also include support for document retention and disposition. SharePoint 2010 does not disappoint in this area either. It can serve as a fully featured records management solution. SharePoint 2010 records management is covered in detail in Chapter 8.

Workflow

Some legacy ECM solutions include integrated support for workflow, although many do not. SharePoint 2010 shines again with a fully functional workflow platform based on Windows Workflow Foundation. In addition to the capability to create usable no-code workflows for simpler tasks, SharePoint supports custom workflow development in order to handle a wide variety of workflow processes. The topic of SharePoint 2010 workflow is covered in detail in Chapter 4.

Avoiding the Pitfalls

When replacing a legacy ECM solution with SharePoint 2010, there are a few key areas that can trip up the move process. It is not always necessary to mitigate these issues, but most of them will arise sooner or later during a major platform migration. Awareness is the key so that proper planning can take place.

Capacity Planning

One of the biggest issues that can come up during a content migration into SharePoint is capacity planning. The problem stems from a lack of proper planning with respect to content storage. IT staff can usually determine how much storage is being used to store the existing document content. This is an easy task, particularly when all documents are stored in some sort of managed file system. It is then assumed that when the content is migrated to SharePoint 2010, the same amount of storage space will be required.

This is completely false. SharePoint introduces significant storage overhead as content is migrated into the system. This point is covered in detail in Chapter 12, which presents concepts for designing a scalable ECM architecture.

Illegal Characters in Metadata

Earlier in this chapter, the issue of illegal characters in metadata was mentioned. This issue is of concern because many legacy ECM solutions allow characters that SharePoint defines as illegal. For example, a carriage return might be represented differently in the legacy system than it is in SharePoint, which results in a strange whitespace character that causes upload errors. This problem can present itself in a very cryptic way. SharePoint may return a somewhat vague error such as “Invalid Request” or “Object reference not set” or “Input string was not in a correct format.” Depending on the condition of source metadata and the error-handling capabilities of the migration application, a significant percentage of the source documents may not migrate correctly. In most cases, it is useful to define a base character set such as alphabetic upper and lowercase characters

along with numbers, the space character, and basic punctuation characters. Each character in a metadata value can be compared against the base character set to ensure that all other potentially illegal characters are stripped from metadata values.

Missing Required Data

When preparing metadata during a migration process, it is wise to pay careful attention to required fields. If a value is not provided for a required field, the document will upload and all provided metadata will be applied, but the document will be placed in a *checked out* state such that only the uploading user account can see the document in SharePoint. Unfortunately, SharePoint doesn't usually provide an error when this occurs, so the migration application just continues as if the item were migrated successfully. In order to avoid this situation, identify all required fields in advance and interrogate relevant metadata to ensure that a value is available in the required field for all items.

Content Database Transaction Log Growth

If the recovery mode of a content database is set to Full, then migrating a large volume of documents into the content database will rapidly inflate the content database transaction log. Oftentimes, migration processing will consume not only all available log space, but also all space on the storage volume.

If a transaction fails to commit to the transaction log, the item upload will fail, causing migration processing to abruptly come to a halt. To avoid this, consider performing the following steps:

1. Execute a full backup of the content database.
2. Set the content database recovery model to Simple.
3. Execute the migration.
4. Set the content database recovery model back to Full.
5. Execute a full backup of the content database.

Considering the fact that a migration can always be performed again, if the content database fails, no data has been lost. However, for long migrations, it may be beneficial to occasionally pause the migration and execute a full backup of the content database in order to ensure that an extensive amount of time is not lost in the event of a failure.

Managing Upload File Size Restrictions

Another common issue a migration can run into is when SharePoint throws an error because the source file is too large to be uploaded based on current settings. You can find the upload size setting in the general settings of a given web application in Central Administration. The default upload file size limit is 50MB. This value can be raised significantly, but it's not the only configuration change that needs to be made. In IIS 7.0 and above, it is also necessary to set the `maxAllowedContentLength` attribute in a `<requestLimits/>` element in the `web.config` file. This is a configuration setting that is often overlooked.

UPGRADING A SHAREPOINT ECM FARM TO SHAREPOINT 2010

It is not uncommon to have a MOSS 2007 ECM farm that contains hundreds of thousands or even tens of millions of documents. Unless EBS has been implemented to externalize binary content, content databases will typically be very large — ideally, not exceeding 100GB, but in some circumstances they may even be larger than that. This section discusses factors that determine the best approach for upgrading a MOSS 2007 ECM farm to SharePoint 2010.

Know Your Farm

During an upgrade, the administrator needs to run the *preupgradecheck* operation using STSADM. This tool processes a set of rules that help administrators discover issues that will cause an upgrade process to stumble. Essentially, among other things, this tool was created to help administrators understand all the custom components.

In addition, it is important to understand how a MOSS 2007 ECM farm is being used. Typically, the farm will serve in one or more of the following roles:

- Document management or content archive farm with little or no customizations
- Collaboration farm that likely has customizations and third-party components
- Hybrid farm that accommodates both collaboration and content archive needs
- Public-facing Windows Content Management (WCM) farm

The upgrade pattern is likely to be different for each of these farm types, so it is imperative that administrators understand exactly how the farm is being used.

SharePoint Portal Server 2003 and WSS v2.0

Unfortunately, there is no direct upgrade path from SPS 2003 to SharePoint 2010. There are also significant differences in topology and taxonomy between 2003 and the 2007 or 2010 releases of SharePoint. For this reason, in most cases it is best to only upgrade the content. It is possible to upgrade to SharePoint 2010, but only by way of an intermediate upgrade to MOSS 2007. To do so, follow these steps:

1. Detach the content databases from the 2003 farm.
2. Attach the content database to a working 2007 farm and verify that it has been properly upgraded.
3. Detach the content database from the 2007 farm.
4. Attach the content database to a working 2010 farm and verify that it has been properly upgraded.

Microsoft Office SharePoint Server 2007 and WSS v3.0

Once the administrator understands how the farm is being used, has executed the *preupgradecheck*, and mitigated any *preupgradecheck* issues, a decision must be made regarding the best method for upgrade. The following scenarios provide general guidance depending on how the MOSS 2007 ECM farm is being used.

Imaging or Archive-Only Farm with No Customization

If the MOSS 2007 farm is primarily used as a high-volume content archive or document management platform, there is a good chance that there will be very few customizations with the farm. In this scenario, the DB attach method is the easiest and safest upgrade path. A backup of the content databases can be restored to a SharePoint 2010 test farm.

Using the test farm, the administrator can use the `test-spccontentdatabase` PowerShell cmdlet to ensure that no unexpected customizations are missing from the environment. If there are missing customization components, then the administrator has an opportunity to install them on the SharePoint 2010 test farm.

Next, the administrator can test the upgrade process by attaching the restored content database in the SharePoint 2010 test farm. If the attach is successful and the viability of the upgraded content database is verified, the administrator can be assured that a production farm upgrade process can be completed successfully.

After the production upgrade has been completed, it is a good time to evaluate the benefits of Remote BLOB Storage (RBS). In many cases, content databases that contain a large volume of document content can benefit from an RBS implementation.

Collaboration Farm with Customizations

A collaboration farm that has customizations should be carefully analyzed using *preupgradecheck*. In most cases, an *in-place* upgrade will be sufficient as long as the farm servers meet the minimum hardware requirements for SharePoint 2010.

The safest way to ensure the success of an in-place upgrade is to construct a test environment that is as similar as possible to the production MOSS 2007 farm. Then progress through an in-place upgrade using the test farm while documenting steps and upgrade timing. This process will likely identify any issues that could cause the upgrade to fail. It is much better to fail during an upgrade of a test farm than it is to fail during an upgrade of a production farm.

Before executing an upgrade of the production farm, be sure to have a proven backup and restore solution for the farm.

Collaboration Farm with Large Imaging or Archive Site Collections

In many cases, the MOSS 2007 farm serves in more than one role. It is quite common to have a farm that is used for both collaboration and large-scale document management or content archive. In this scenario, a hybrid approach is often the best upgrade solution.

First, the administrator deploys a MOSS 2007 test farm that is as similar as possible to the production MOSS 2007 farm, with particular attention paid to the collaboration portion of the farm. Large-scale document management content databases should be excluded. An in-place upgrade is then executed to ensure that the test farm can be upgraded to SharePoint 2010. Once the upgrade of the collaboration functionality has been verified, the administrator can then restore a backup of the production farm document archive content databases to the upgraded SharePoint 2010 test farm. Finally, all aspects of the upgraded farm are verified to ensure that everything is working properly.

Again, before executing the upgrade of the production farm, ensure that reliable backups have been performed for both the production farm servers and all farm data.

SUMMARY

This chapter identified SharePoint 2010 as an extremely flexible solution platform that is capable of replacing several legacy document management systems. However, before that can happen, careful planning should take place to ensure that the end user experience is improved, not degraded. Also, the path for migrating content from a legacy solution to SharePoint 2010 should be identified because there are several possible ways to send content into SharePoint. These different techniques have been outlined and demonstrated. During a migration, it is important to watch out for certain issues that can cause the content transfer process to stall. These pitfalls were also described. Finally, when upgrading an ECM solution from an older version of SharePoint to SharePoint 2010, it is vital to understand the functional design of the farm in order to determine the most effective and reliable upgrade technique.

INDEX

A

- ABBYY, 455–457
- access control lists (ACLs), 16
- accessibility standards, 216–217
- Acrobat Reader, 14, 443
- action rules, 366–367
- actions
 - InfoPath, 366–367
 - workflow
 - SharePoint Designer, 111
 - Visio, 101–103
- activation dependencies, 220
- Active Directory (AD) service account, 386–387
- activities. *See also specific activities*
 - social tagging, 143
 - WF, 88–89
- Activity Feed Timer Job, 157
- activity feeds, 159
 - programmatically creating, 163–165
 - programmatically retrieving, 162–163
- ActivityEvent, 159
- ActivityEventCollection, 159
- ActivityManager, 159
- ActivityPreference, 159
- ActivityPreferenceCollection, 159
- ActivityPreferencePerType, 159
- ActivityTemplate, 159
- ActivityTemplateCollection, 159
- ActivityType, 159
- ActivityTypeCollection, 159
- AD (Active Directory) service account, 386–387
- Add and Customize Pages permission, 74
- advanced content processing, FAST Search, 204
- advanced property-based searches, 182, 187, 199–202
- Advanced Search Box Web Part, 200, 201, 341–342, 355
- AIIIM (Association for Information and Image Management)
 - collaborative life cycle conceptual stages, 134, 172
 - definitions
 - collaboration, 134
 - ECM, 4
 - ECM components, 7
 - records management, 244
 - PDF/A, 445
- AJAX, 216, 463
- AllowedContentTypeCollection, 72
- Amazon S3, 11
- analytics
 - Nintex, 467
 - PerformancePoint Analytics, 18
 - WCM, 241–242
 - Web Analytics service, 399, 400, 413
- analyzing content, records management, 246–247
- AND, 190
- annotations. *See* markup
- API based queries, 203
- APIs (application programming interfaces). *See* SharePoint APIs
- Application Registry Service, 398
- application servers, farm deployment, 412–413
- applications, forms *v.*, 358
- Approval workflow, 94, 95–99
- Approvers group, 222
- archived formats, 421, 437–438, 450
- archive-only MOSS farm, 492
- ARMA International, 13
- arrays. *See* storage
- Article Page content type, 224
- ASP.NET, 23, 26, 119, 120, 226, 236, 287, 297
- asset libraries, 276, 284
- association columns, 110
- Association for Information and Image Management. *See* AIIIM
- association phase, workflow, 93

association workflow pages, 119–122, 126
 audio podcasting, 284
 audit reports, 271–272
 audit trail, 34, 83–84, 465, 477
 auditing, 270–271. *See also* records management
 authoring. *See* content authoring
 auto-folding, 85, 259
 autogrowth, database, 386, 388–390, 392, 399, 403
 AvePoint, 458–460
 awareness stage, collaboration, 134
 Azure, 11

B

backup, RBS, 407
 base content types, 45
 Basel II, 244
 BCS (Business Connectivity Service), 24, 205, 360, 362, 398, 463
 binary externalization, 404, 458, 491
 binary large objects. *See* BLOBs
 Bing, 21
 bit rate throttling, 289–292
 BlobCache element attributes, 288
 BLOBs (binary large objects). *See also* remote BLOB storage
 bit rate throttling, 289–292
 caching, 287–288
 defined, 286
 WORM devices, 405–406
 Blog Site template, 153, 169–170
 blogs, 13, 133, 134, 152, 153, 169–170, 240, 284
 bookmarklets, 135, 141–143
 Boolean operators
 AND, 190
 OR, 190
 NOT, 190
 BPM (business process management), 6, 12.
 See also workflows
 branding
 DAM, 281–282
 WCM, 240
 browsers
 bookmarklets, 135, 141–143
 browser-compatible form exercise, 374–380
 cross-browser support, 217
 viewing of content, 14

building blocks, document taxonomy, 35–36
 Business Connectivity Service (BCS), 24, 205, 360, 362, 398, 463
 business process management (BPM), 6, 12.
 See also workflows

C

CAML (Collaborative Application Markup Language), 26, 178, 228, 229
 capability categories, SharePoint, 19–21
 capacity planning. *See* database storage and capacity planning
 capture. *See also* document imaging
 ABBYY, 455
 AvePoint, 458
 documenting imaging, 294
 ECM component, 7–9
 electronic forms, 9
 e-mail, 8–9
 KnowledgeLake, 463
 paper, 7–8
 reports, 9
 transformation technologies, 14–15
 capture application. *See* WPF-based capture application
 CAS devices (content addressable storage), 404, 405
 Central Administration
 database, 400
 defined, 24–25
 ChangedGroup, 52
 ChangedItem, 52
 ChangedItemCollection, 52
 ChangedItemType, 52
 ChangedOperationType, 52
 ChangedTerm, 52
 ChangedTermSet, 52
 ChangedTermStore, 52
 check-in/check-out, 12, 21, 34, 73, 79–81, 215, 487
 Chief Compliance Officer, 246
 classified ad form fields, 374–375
 client-side object models. *See also* server-side object model
 columns value updated, 43
 content type assigned to document, 46–47
 defined, 35
 new document version, 483–486

- reading groups, 78–79
- server-side object model compared to, 37
- upload document to document library, 38–39
- cloud storage, 11, 405
- clouds, tag, 137, 152
- CodeDOM objects, 105
- COLD (computer output to laser disc), 4, 6, 9, 15
- collaboration, 133–172. *See also* social networking
 - blogs, 13, 133, 134, 152, 153, 169–170, 240, 284
 - defined, 13, 133
 - document collaboration, 13, 486
 - ECM management component, 13
 - My Sites, 151–157
 - configuring, 154–157
 - infrastructure, 154–157
 - Outlook, 171–172
 - sections, 151
 - Office, 170
 - SharePoint as, 134
 - SharePoint Workspace, 18, 20, 134, 170–171, 360, 369
 - social networking, 13, 133, 134, 165–170, 172, 453–454, 468
 - social tagging, 134–151
 - privacy concerns, 143–144
 - programming model, 144–151
 - security concerns, 143–144
 - Tags and Notes feature, 135, 136, 137, 138, 142, 143, 152
 - stages, 134
 - User Profile Service Application, 145, 151, 156, 157, 161, 165–168
 - user profiles, 157–165
 - policies, 158
 - programming model, 159–165
 - wikis, 239–240
 - benefits, 169
 - collaboration, 133, 134
 - Enterprise Wiki Page content type, 224
 - Enterprise Wiki site template, 168–169, 219
 - Enterprise WikiLayouts folder, 221
 - SharePoint communities category, 20
 - WCM, 216, 218
- collaboration farms, 492–493
- Collaborative Application Markup Language. *See* CAML
- Colleagues tab, My Profile, 152
- Collect Feedback workflow, 94
- Collect Signatures workflow, 94
- columns, 39–43. *See also* site columns
 - Article Page content type, 224
 - association columns, 110
 - defined, 36, 39
 - index columns, 204, 205, 211
 - Load Columns button, 301, 355
 - loading, WPF-based capture application, 301–303
 - managed metadata, 50, 53, 396
 - Page content type, 223
 - programming model, 41–43
 - term sets in, 50
 - types, 40–41
 - updating, with client-side object model, 43
- communities category, SharePoint, 20
- compliance, 248–249
 - documentation, 247
 - GrimmalSoft, 460–461
 - officer, 246
 - Sarbanes-Oxley, 5, 244, 246, 248
 - scenarios, SharePoint, 249
- composites category, SharePoint, 19–20
- computer output to laser disc (COLD), 4, 6, 9, 15
- condition actions, 102
- conditions, rule types and, 364
- content. *See also* ECM; migrating to SharePoint 2010; web content management
 - analyzing, records management, 246–247
 - ECM, 4
 - FAST Search advanced content processing, 204
 - semi-structured, 8, 239
 - SharePoint content category, 21
 - structured, 7
 - unstructured, 8, 456
- content addressable storage (CAS devices), 404, 405
- Content and Structure Reports List, 225
- content authoring
 - custom workflow services, 127–130
 - defined, 235
 - SharePoint Designer workflows, 111–114
 - WCM, 216, 235–239
- content corpus. *See* corpus
- content databases, 395–396
 - DAM, 285
 - defined, 25
 - size supported limits, 416–417

- transaction log growth, 490
- content deployment, 238–239
- content management system, transactional, 465
- content manager, 246
- Content Organizer. *See also* Drop Off Library
 - auto-folding, 85, 259
 - DAM, 285
 - defined, 29, 84
 - records management, 238, 255–258
 - Rules, 238, 256–257
 - scalable taxonomy, 414–416
 - Settings page, 238, 254, 255, 256, 257
- content query Web Part (CQWP), 217, 233–235, 240, 280
- Content Search Service Application, 180, 205
- content storage size factors, 384–385
- Content tab, My Profile, 152
- content type hubs, 28, 56–57, 414–415, 416
- content type syndication, 56–58, 414–415, 418
 - defined, 56
 - programming model, 57–58
- content types, 44–47. *See also specific content types*
 - base, for document taxonomy, 45
 - defined, 22, 44
 - digital asset, 277
 - document imaging system, 339
 - policies for, 264–265
 - programmatically publishing, 57–58
 - programming model, 45–47
 - SharePoint object hierarchy, 22–23
 - site content types gallery, 44
 - WCM, 223–225
- ContentType node, 232
- ContentTypePublisher, 57–58
- conversion, living document, 444
- copy.asmx web service, 474–477
- corpus
 - change rate, 177–178
 - defined, 176, 401
 - index partition storage, 401
 - profile, 176–178
 - search architectures
 - 3-million-item, 208
 - 10-million-item, 208
 - 40-million-item, 208–210
 - 100-million-item, 210
 - 500-million-item, 211

- size
 - Enterprise Search, 204
 - FAST Search, 204
- correlation
 - tokens, 118–119
 - workflows, 91
- CQWP. *See* content query Web Part
- crawl databases, 393–395
- crawled properties, 175, 179, 184, 187, 338
- crawling
 - corpus profile, 176
 - FAST Search, 205
 - iFilters, 176–177
 - improving, 213
 - MOSS, 179
- cross-browser support, 217
- CSS, 74, 215, 225, 240, 288
- custom advanced search property XML, 342–343
- custom code, InfoPath, 370–371
- custom form exercise, 374–380
 - layout, 374–375
 - publishing, 379–380
 - rules, 376
 - submission, 376–378
- custom record declaration handler, 261–262
- custom workflow services, 127–130

D

- DAM. *See* digital asset management
- DAS (direct-attached storage), 383–384
- data. *See* metadata; *specific data*
- data binding, WPF, 297, 300
- data connections, 368–369
- data files. *See* database data files
- databases. *See also* content databases; service application databases; SQL Server; storage architecture; *specific databases*
 - autogrowth, 386, 388–390, 392, 399, 403
 - crawl, 393–395
 - performance, 381–382
 - property, 396–397
 - storage solution, 10
- database data files
 - management, 402–403
 - number
 - content databases, 395
 - crawl databases, 393–394

- property databases, 396
 - TempTB, 390–391
 - pre-sizing, 388–390
- database storage and capacity planning, 385–400. *See also* farm deployment; storage architecture
 - autogrowth adjustment, 388–390
 - content databases, 395–396
 - crawl databases, 393–395
 - disk allocation priority, 400
 - instant file initialization, 386–388
 - legacy ECM solutions, 489
 - log files, 392–393
 - optimizing storage array utilization, 386
 - pre-sizing database files, 388–390
 - property databases, 396–397
 - service application databases, 397–400
 - SQL Server supporting concepts, 386–390
 - TempDB, 390–392
- debugging, 26, 116, 117
- DebugView, 477, 481, 482, 485
- declarative workflows, 99–114
 - SharePoint Designer, 92, 105–114, 125–126
 - Visio, 92, 99–105
 - XML, 99, 105
- deep refiners, FAST Search, 204
- DefaultDocument, 72
- DefaultDocumentCollection, 72
- DelayActivity, 89
- DeletedSocialComment, 144
- DeletedSocialData, 144
- DeletedSocialRating, 144
- DeletedSocialTag, 144
- delivery. *See also* search; viewing
 - AvePoint, 459–460
 - defined, 459
 - as ECM component, 13–16, 421
 - search, 13–14
 - transformation technologies, 14–15
 - viewing of content, 14
- dependencies, activation, 220
- dependency properties, 121–122, 297
- deployment. *See also* publishing
 - content, 238–239
 - electronic forms, 361–362
 - Silverlight-based viewer Web Part, 327
- Designer. *See* InfoPath; SharePoint Designer
- Designers group, 222
- designing forms, 359–360
- designing records management solution, 247–248
- dialogs, 364. *See also* forms
- digital asset management (DAM), 275–292
 - bit rate throttling, 289–292
 - brand management, 281–282
 - components, 276–280
 - defined, 29, 275
 - marketing, 281–282
 - media development project, 282–283
 - online training center, 283–284
 - performance optimization, 287–292
 - rich media, 15, 217
 - solution scenarios, 280–284
 - storage considerations, 285–287
 - Web Parts, 278–280
- digital assets
 - content types, 277
 - defined, 275
 - reusable, 284
 - site columns, 276–277
- digital rights management, 16
- digital signatures, 16, 94, 360, 367
- direct-attached storage (DAS), 383–384
- disk allocation priority, 400
- disk counters, 402
- disks in array, 382–383
- disposition, 13, 131, 215, 244, 247, 267, 489
- Disposition Approval workflow, 94
- DMS. *See* document management systems
- documents (electronic documents). *See also* archived
 - formats; document imaging; living documents
 - document imaging *v.*, 6
 - Office, 8
 - paper storage *v.*, 293–294
- document classification, 35
- document collaboration, 13, 486
- document control, 73–84
 - audit trail, 34, 83–84, 465, 477
 - check-in/check-out, 12, 21, 34, 73, 79–81, 215, 487
 - defined, 73, 85
 - security, 73–79
 - permissions, 73–75
 - programming model, 76–79
 - version history, 34, 69, 81–83, 482, 486
 - versioning, 12, 73, 81–84, 215, 487
- document ID service, 66–69
 - defined, 29, 66
 - programming model, 68–69

- document identifier, 67–68
 - document imaging, 293–355
 - benefits, 4–5
 - concepts, 294–295
 - defined, 294
 - document management, 12
 - early history, 4–5
 - ECM, 295
 - electronic documents *vs.*, 6
 - optical storage, 11
 - paperless office, 5, 9, 295, 422
 - records management, 12
 - search, 294
 - security, 5
 - document imaging system on SharePoint, 295–355
 - Advanced Search Box Web Part, 341–342, 355
 - content type creation, 339
 - document library creation, 340
 - search configuration, 338–339
 - Search Core Results Web Part, 341, 343, 355
 - Search Core Results XSLT (code listing), 343–354
 - setting up scenario, 295–296
 - SharePoint infrastructure setup, 338–342
 - Silverlight Web Part, 325, 327, 341
 - Silverlight-based viewer Web Part, 322–338
 - architecture, 323
 - code listing, 331–338
 - deployment, 327
 - design, 323
 - image loader, 323–324
 - imaging HTTP handler, 325
 - imaging web service, 324, 328–331
 - implementation, 323–327
 - JavaScript, 322, 326–327
 - solution data flow diagram, 296
 - viewer application, 325–326
 - solution data flow diagram, 295–296
 - Web Parts, 340–342
 - WPF-based capture application, 298–322
 - architecture, 299
 - code listing, 305–322
 - deployment, 322
 - design, 299
 - implementation, 299–322
 - loading columns, 301–303
 - main window, 301–305
 - MVVM infrastructure, 299–300
 - releasing to SharePoint, 304–305
 - scanning an image, 303–304
 - screenshot, 298
 - solution data flow diagram, 296
 - target dialog, 300–301
 - document libraries, 36–39
 - defined, 36
 - document imaging system, 340
 - programming model, 36–39
 - retention policy, 265–266
 - document management, 33–85
 - defined, 12, 34
 - ECM management component, 12
 - features, 34
 - records management compared to, 12–13, 243
 - SharePoint APIs, 35
 - document management solutions, legacy, 471–472
 - document management systems (DMS), 3, 12, 13, 34, 73, 83, 85, 174, 493
 - document readers, 14, 443
 - document security level, 16, 73
 - document sets, 69–73
 - custom, 70
 - defined, 28–29, 69
 - implementing, 69
 - object model, 72
 - programming model, 71–73
 - recordization, 259
 - document taxonomy, 35–73
 - building blocks, 35–36
 - defined, 34, 35
 - document versions, 487
 - DocumentRouterAutoFolderSettings, 259
 - DocumentSet, 72
 - DocumentSetTemplate, 72
 - DocumentSetVersion, 72
 - DocumentSetVersionCollection, 72
 - Drop Off Library, 29, 84, 238, 253, 254, 255, 256, 257, 282, 285, 415, 416, 474
- ## E
- EBS (external binary storage), 404, 458, 491
 - ECM (enterprise content management). *See also* legacy
 - ECM solutions
 - collaboration, 134
 - components, 7–16
 - definitions, 3–4
 - document imaging, 295
 - features, in SharePoint, 27–29
 - historical perspective, 4–6

- management components, 12–16
 - scalable ECM farm deployment, 381–418
 - security, 16
 - SharePoint ECM ecosystem, 26, 451–468
 - support concepts, 419
 - WCM, 3
 - workflows, 87–88
 - ECM file formats. *See* file formats
 - ECM projects guidance. *See* migrating to SharePoint 2010
 - ECMA (European Computer Manufacturers Association), 423, 424
 - ECMAScript, 35, 36, 279
 - EcmDocumentRouter, 259
 - EcmDocumentRouterRule, 259
 - EcmDocumentRoutingWeb, 260
 - ecosystems, 451–452. *See also* Microsoft Partner Ecosystem; SharePoint ECM ecosystem
 - EDI, 9
 - eDiscovery, 249, 272–273, 459
 - electronic documents. *See* documents
 - electronic forms. *See* forms
 - electronic signatures, 16, 94, 360, 367
 - e-mail. *See also* Outlook
 - capture, 8–9
 - submit data connections, 369
 - engagement stage, collaboration, 134
 - ENIAC, 357
 - Enron Corporation, 244
 - enterprise content management. *See* ECM
 - enterprise keywords, 28, 48, 233. *See also* keywords
 - enterprise report management, 4, 6, 9, 15
 - Enterprise Search, 180–203
 - architectures, 206–211
 - configuration components, 184–189
 - FAST Search for SharePoint 2010, 180
 - improvements, 179–180
 - topology components, 180–184
 - enterprise wikis. *See* wikis
 - Enterprise Wiki Page content type, 224
 - Enterprise Wiki site template, 168–169, 219
 - Entity, 159
 - EntityTypeIds, 159
 - European Computer Manufacturers Association (ECMA), 423, 424
 - event receivers
 - custom record declaration handler, 261
 - metadata-based security, 487
 - workflow, 130–131
 - exact relevance, 396
 - Excel (Microsoft)
 - electronic documents, 6
 - KnowledgeLake, 463
 - “Microsoft Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats,” 422
 - Office Web Apps, 425, 427, 450
 - remote BLOB service, 404
 - execution phase, workflow, 93–94
 - exporting process, Visio workflows, 103–105
 - extensibility, SharePoint, 26
 - Extensible Markup Language. *See* XML
 - external binary storage (EBS), 404, 458, 491
 - external data, InfoPath, 368–369
- ## F
- Facebook, 133, 453
 - farms
 - collaboration, 492–493
 - MOSS, 491–493
 - SharePoint architecture, 23
 - SharePoint object hierarchy, 22
 - size definitions, 409–412
 - upgrading, to SharePoint 2010, 491–493
 - farm deployment (scalable SharePoint 2010 ECM farm deployment), 381–418
 - farm topology, 408–413
 - application servers, 412–413
 - factors affecting performance, 408–409
 - farm size definitions, 409–412
 - SQL Servers, 412–413
 - web servers, 409, 410, 411, 412
 - key takeaways, 418
 - performance tuning, 417–418
 - resource throttling, 417–418
 - storage architecture, 381–407
 - content storage size factors, 384–385
 - DAS, 383–384
 - database storage and capacity planning, 385–400
 - disks in array, 382–383
 - index partition storage, 401
 - IOPS requirements, 382–383
 - NAS, 383–384
 - performance pitfalls, 382–385
 - remote BLOB, 403–407
 - shared SAN, 383–384
 - tuning and optimization, 401–403

- taxonomy design, 413–417
- FAST (FAST Search and Transfer)
 - Microsoft acquisition, 179, 203
 - search crawlers, 14
 - team, 179–180
- FAST Query Language (FQL), 203, 204
- FAST Search Authorization, 205, 206
- FAST Search Connector, 205, 206
- FAST Search for SharePoint 2010, 203–206
 - development, 179–180
 - Enterprise Search, 180
 - functional overview, 203–205
- features. *See also specific features*
 - defined, 26, 219
 - publishing, 219–222
- Fibre Channel, 383
- field types, metadata navigation, 63
- fielded query, 396
- fields, form, 374–375
- Fields pane, 359
- file formats, 421–450. *See also* HTML; living documents; Microsoft Office Formats; PDF; TIFF; XML; XPS
 - archived, 421, 437–438, 450
 - COLD/enterprise reports, 4, 6, 9, 15
 - Open Document Format, 176, 437
 - transformation technologies, 14–15
- file initialization, instant, 386–390
- file plan, records management, 247
- file plan reports, 272
- file shares, 470–471. *See also* migrating to SharePoint 2010
- file systems, 10
- FILESTREAM RBS provider, 406–407
- Filler, InfoPath, 360, 362, 364, 366, 367, 371, 373
- filters, metadata navigation, 62
- 500-million-item corpus, 211
- Fluent UI, 19, 216, 359, 466. *See also* ribbon
- folders
 - defined, 36
 - partitioning, 256
- folksonomy, 135
- forms, 357–380. *See also* InfoPath
 - applications *v.*, 358
 - capture, 9
 - custom form exercise, 374–380
 - data and layout, 374–375
 - publishing, 379–380
 - rules, 376
 - submission, 376–378
 - deploying, 361–362
 - designing, 359–360
 - dialogs, 364
 - electronic, 9, 357–358
 - fields, 374–375
 - overview, 357–358
 - paper forms *v.*, 9, 357–358
 - paperless office, 5, 9, 295, 422
 - publishing, 361–362, 371–372
 - rules, 364–368
 - action, 366–367
 - common characteristics, 364
 - custom form exercise, 376
 - formatting, 366
 - validation, 364–366
 - saving, 369
 - strategy, 372–374
 - submission, 369, 376–378
- Form Options dialog, 370, 377, 379
- form templates
 - makeup, 362–363
 - publishing, 361–362, 371–372
- formatting rules, 366
- Forms Services, 360–361
- forms-based projects strategy, 372–374
- 40-million-item corpus, 208–210
- forums, SharePoint ecosystem, 453–454
- FPRPC (Front Page Remote Procedure Calls), 479
- FQL. *See* FAST Query Language
- Friendster, 133
- Front Page Remote Procedure Calls (FPRPC), 479
- full-text searching, ABBYY, 455–456

G

- GetInitiationData, 121, 122
- GetPage, 323–324
- GetPageCount, 323–324
- global promotion, 107
- global term sets, 47, 49, 233
- Gold-Certified Partner, 452, 453, 455, 458, 460, 465
- Google, 21
- governance, GrimmelSoft, 462
- graphic equalizer, 414
- green bar paper, 6
- GrimmelSoft, 460–462
- Groove workspace, 170

Group, 52
 GroupCollection, 52
 groups. *See also specific groups*
 creating, with server-side object model, 77–78
 document control, 76
 managing, 76–79
 publishing features, 222
 reading, with client-side object model, 78–79
 guidance for ECM projects. *See migrating to SharePoint 2010*

H

Health Analyzer, 212
 Health Care Compliance Association, 246
 Health Insurance Portability and Accountability Act (HIPAA), 5, 244, 246, 248
 HiddenListFullSyncJobDefinition, 52
 hierarchical storage management (HSM), 10
 Hierarchy Managers group, 222
 high-privilege workflows, 108
 HIPAA (Health Insurance Portability and Accountability Act), 5, 244, 246, 248
 Hold and eDiscovery feature, 253, 272–273
 HSM (hierarchical storage management), 10
 HTML (Hypertext Markup Language)
 Add and Customize Pages permission, 74
 defined, 15
 Forms Services, 360
 FPRPC, 479
 MHTML, 429
 Silverlight HTML Bridge, 326
 web content, 215
 XHTML, 216, 217, 224, 375
 HTTP handler, imaging, 325
 HTTP POST, 479
 Hyper-Threading, 390–391

I

I Like It button, 135, 136
 ICustomRouter, 260
 IfElseActivity, 88–89
 ifilters
 crawling, 176–177
 NULL, 176
 OCR, 176, 438–442, 443, 445
 PDF, 177, 442–443
 PDF/A, 445
 TIFF, 438–440
 XPS, 447–449
 IIS, 23, 24, 36, 289
 Manager, 288, 290
 Media Services, 289–290
 upload size setting, 490
 illegal characters, 471, 473, 489–490
 image management system (IMS), 3
 image viewer Web Part, 280
 ImageLoader, 323
 ImageService, 324
 imaging HTTP handler, 325
 imaging MOSS farm, 492
 imaging web service, 324, 328–331
 importing. *See also migrating to SharePoint 2010*
 content
 preparing, 472–473
 protocols, 474–479
 into SharePoint, 473
 policies, 266–267
 Visio workflows, 105
 ImportManager, 52
 IMS (image management system), 3
 independent software vendors. *See ISVs*
 indexes
 columns, 204, 205, 211
 defined, 488
 document imaging, 294
 legacy ECM solutions, 488
 index partition storage, 401
 indexer rows, 204, 205
 InfoPath, 357–380. *See also forms*
 actions, 366–367
 custom code, 370–371
 Designer, 360
 external data, 368–369
 Filler, 360, 362, 364, 366, 367, 371, 373
 Forms Services, 360–361
 new features, 360
 overview, 358–373
 project planning, 372–374
 workflows, 125–126
 XML, 363, 370
 information management policy, 250, 263–267
 Information Rights Management, Microsoft, 16, 488
 initiation forms, workflow, 120, 126
 initiation phase, workflow, 93
 initiation workflow pages, 119–122

- INotifyPropertyChanged, 297, 299, 326
- in-place records management, 250–252
- insights category, SharePoint, 20
- instant file initialization, 386–388
- internally developed legacy document management systems, 471
- International Compliance Association, 246
- Internet newsgroups, 13
- IOPS (IOs per second) requirements, 382–383
- IRecordDeclarationHandler, 260, 261–262
- IRecordUndeclarationHandler, 260, 261
- ISVs (independent software vendors), 451, 452–453
 - ISV/Software Competency, 452–453
 - solutions, 454–467
- item level scope, workflow, 92

J

- Java Database Connectivity, 205
- JavaScript
 - AJAX, 216
 - bookmarklets, 141, 142
 - Silverlight-based viewer Web Part, 322, 326–327

K

- key filter, 62
- keywords
 - enterprise, 28, 48, 233
 - search, 190–192
 - set, 48
- KnowledgeLake, 462–465

L

- Label, 52
- LabelCollection, 52
- language packs, 241
- large farms, 410, 411
- legacy ECM solutions. *See also* migrating to SharePoint 2010
 - capacity planning, 489
 - content database transaction log growth, 490
 - disposition, 489
 - document management, 471–472
 - pitfalls, 489–490
 - records retention, 489

- scanning, 488
- search, 488
- SharePoint solutions, 486–490
- upload file size, 490
- workflows, 489
- libraries. *See also* document libraries; *specific libraries*
 - lists compared to, 22
 - SharePoint object hierarchy, 22
 - submit data connections, 369
- library services, 10. *See also* repositories
- line-of-business (LOB) systems, 463, 472, 480
- Link, 159
- LinkedIn, 13, 133, 172, 453
- lists. *See also specific lists*
 - defined, 36
 - libraries compared to, 22
 - programmatically adding metadata navigation filters to, 65–66
 - receive data connections, 368
- List, 159
- list items, 36
- List Unique Permissions threshold, 418
- List View Lookup threshold, 417
- List View threshold, 414, 417
- ListPolicySettings, 267
- living documents
 - conversion, 444
 - defined, 422, 450
 - Microsoft Office Formats, 422–436
 - Open Document Format, 176, 437
 - Open XML, 423–425, 429, 450
 - developing, 423–424
 - .NET Framework compared to, 449
 - Open Document Format compared to, 437
 - SDK, 430
 - standardization, 424–425
 - XPS compared to, 446
- Load Columns button, 301, 355
- LoadImage, 303, 324, 326, 327
- LOB (line-of-business) systems, 463, 472, 480
- local term sets, 47, 233
- location-based metadata defaults, 60–62
- log files, 392–393
 - pre-sizing, 392
 - storage volume performance and reliability, 393
 - VLFs, 388

long-term preservation. *See* preservation
 Lotus 1-2-3, 6
 Lotus Notes, 205, 458

M

magnetic storage, 10–11. *See also* RAID
 main window, WPF-based capture application, 301–305
 MainModel, 325
 MainPage, 325
 MainWindow, 299, 301, 303, 304
 managed metadata, 47–59. *See also* term sets
 administering, 48–49
 advantages of, 48
 defined, 27–28, 47
 object model, classes for, 52–53
 programming model, 50–55
 managed metadata columns, 50, 53, 396
 managed metadata service applications, 55–56
 creating, 59
 defined, 55
 deleting, 59
 Term Store Management Tool, 48–49
 updating, 59
 managed metadata service database, 399
 managed properties, 179, 187–188
 managed terms, 28, 233
 management components, ECM, 12–16
 Management Studio, SQL Server, 289, 403
 managing records, compliance scenario, 249–267
 many-to-many content distribution paradigm, 239
 mapping legacy ECM features to SharePoint solutions, 486–490
 marketing, DAM, 281–282
 markup, 442
 annotations, 442
 PDF, 443
 redaction, 488
 standards, 217
 TIFF, 442
 XPS, 449
 master pages, 226–227
 MaxDownloadSize, 177
 maximum upload size, DAM, 286–287
 MCP (Microsoft Certified Professional), 452
 MDF file, 403
 media development project, DAM, 282–283
 media field control, 279–280
 media resource library, 284
 media Web Part, 278–279
 mediation stage, collaboration, 134
 medium farms, 410, 411
 Memberships tab, My Profile, 152
 memory requirements, property databases, 397
 metadata. *See also* managed metadata
 content types, 22
 defined, 34, 233
 illegal characters, 471, 473, 489–490
 location-based metadata defaults, 60–62
 search, 191–192
 support, WCM, 217–218
 Metadata and Taxonomy namespaces, 52
 Metadata and Taxonomy Programming Model, 50–55
 metadata navigation, 62–66
 configuring, 63
 defined, 62
 field types, 63
 filters for, 62
 programming model, 65–66
 search, 14
 metadata-based security, 487
 MetadataNavigationContext, 65
 MetadataNavigationHierarchy, 65
 MetadataNavigationItem, 65
 MetadataNavigationKeyFilter, 65
 MetadataNavigationSettings, 65
 MHTML, 429
 microfiche, 5
 Microsoft acquisition, of FAST, 179, 203
 Microsoft Certified Professional (MCP), 452
 Microsoft Information Rights Management, 16, 488
 Microsoft Office. *See also* documents; Excel; Outlook; PowerPoint; Word
 collaboration, 170
 documents, 8
 Microsoft Office Binary formats, 422–423
 “Microsoft Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats,” 422
 Microsoft Office Formats, 422–436
 Open XML, 423–425, 429, 450
 developing, 423–424
 .NET Framework compared to, 449
 Open Document Format compared to, 437
 SDK, 430
 standardization, 424–425

- XPS compared to, 446
- Word Automation Services, 428–436
 - database, 400
 - defined, 428
 - developing with, 430–436
 - modified class for monitoring (code listing), 434–436
 - sample class (code listing), 432
- Microsoft Partner Ecosystem. *See also* SharePoint ECM ecosystem
 - defined, 451, 468
 - ecosystems compared to, 451–452
 - Gold Certified Partner, 452, 453, 455, 458, 460, 465
 - ISV/Software Competency, 452–453
 - partnerships, 452
- Microsoft Partner Network, 451, 452, 467
- Microsoft SharePoint. *See* SharePoint
- Microsoft.SharePoint namespace, 36
- Microsoft.SharePoint.Client namespace, 36
- migrating to SharePoint 2010, 470–493
 - content
 - extracting content from source system, 470–472
 - identification, 470
 - importing into SharePoint, 473
 - preparing for importing, 472–473
 - protocols for importing, 474–479
 - protocols for updating SharePoint content, 480–486
 - file shares, 470–471
 - upgrading farms, 491–493
- MinimalPerson, 159
- missing required data, 490
- Model component, MVVM, 297, 299, 300
- Model-View-ViewModel pattern. *See* MVVM pattern
- MOSS (SharePoint Server 2007)
 - farm, 491–493
 - search, 174, 178–179, 180, 182, 184, 185
- Most Valuable Professional (MVP) Program, 453
- motivation stage, collaboration, 134
- MSS. *See* SharePoint Server
- MSSDocProps, 397
- MSSDocResults, 397
- MSSDocSdids, 397
- multilingual sites, 19, 241
- multithreading, 177, 390, 473
- MVP (Most Valuable Professional) Program, 453
- MVVM (Model-View-ViewModel) pattern. *See also* document imaging system
 - Model component, 297, 299, 300
 - primer, 296–298
 - Silverlight-based viewer Web Part, 325–326
 - View component, 297, 298, 300
 - ViewModel component, 297, 298, 299, 300, 301, 325
 - WPF-based capture application, 299–300
- My Content, 153
- My Newsfeed, 153–154
- My Profile, 151–152
 - tabs, 152
 - user profiles, 157–165
- My Site Host site collection, 154, 155–156
- My Site Settings section, User Profile Service Application, 167
- My Sites, 151–157
 - configuring, 154–157
 - infrastructure, 154–157
 - Outlook, 171–172
 - sections, 151
- MySpace, 133

N

- namespaces. *See also specific namespaces*
 - Metadata and Taxonomy, 52
 - Microsoft.SharePoint, 36
 - Microsoft.SharePoint.Client, 36
- NAS (network area storage), 11, 383–384
- navigation, WCM, 240–241
- navigation hierarchy filter type, 62
- NDF data files, 403
- NEAR, 190
- .NET
 - ASP.NET, 23, 26, 119, 120, 226, 236, 287, 297
 - Framework
 - CodeDOM objects, 105
 - dependency properties, 122
 - SharePoint, 9, 23, 26, 88
 - Visual Studio, 92, 114
 - Visual Basic .NET, 363, 370
- network area storage (NAS), 11, 383–384
- new document version, 480–486
- New Project dialog, 26, 27, 116
- newsgroups, Internet, 13
- nil attribute, 371
- Nintex, 465–467
- NOT, 190
- notes, 137–139

- creating, 138–139
- Note Board, 137, 138, 139, 152
- programmatically working with, 147–149
- Tags and Notes feature, 135, 136, 137, 138, 142, 143, 152

NULL iFilter, 176

O

OASIS (Organization for the Advancement of Structured Information Standards), 437

object hierarchy, SharePoint, 21–23

object models. *See* SharePoint object models; *specific object models*

OCR (optical character recognition). *See also* iFilters

- iFilters, 176, 438–442, 443, 445
- PDF, 442–443
- PDF/A, 445
- TIFF, 438–442
- XPS, 447–449
- Zonal, 7

Office. *See* Microsoft Office

Office Open XML. *See* Open XML

Office Web Apps, 421, 425–428

100-million-item corpus, 210

OneNote, 176, 425, 427, 450

online training center, DAM, 283–284

OnWorkflowActivated, 117, 121, 122, 125

OOXML. *See* Open XML

Open Document Format, 176, 437

Open XML (Office Open XML), 429, 450

- developing, 423–424
- .NET Framework compared to, 449
- Open Document Format compared to, 437
- SDK, 430
- standardization, 424–425
- XPS compared to, 446

Open XML Paper Specification. *See* XPS

OpenFileDialog, 298, 299, 303

optical character recognition. *See* OCR

optical storage, 6, 9, 11

optimizing storage array utilization, 386

OR, 190

Organization for the Advancement of Structured Information Standards (OASIS), 437

Organization tab, My Profile, 152

Organizations section, User Profile Service Application, 166–167

OSearch14 service application, 177

Outlook (Microsoft)

- KnowledgeLake, 463
- My Sites, 171–172
- Nintex Workflow, 467
- SharePoint integration, 9, 134, 170, 171–172

out-of-the-box workflows, 94–99

- customizing, 109
- InfoPath, 125

Overview tab, My Profile, 152

P

Page content type, 223–224, 227, 228

Page Layout content type, 224

page layouts

- defined, 227
- taxonomy and page layouts exercise, 227–232

Pages Library, 225, 234, 239

paper

- capture, 7–8
- costs, 7
- electronic document storage *v.*, 293–294
- electronic forms *v.*, 9, 357–358
- paperless office, 5, 9, 295, 422

ParallelActivity, 89

parent security level, 16

participation stage, collaboration, 134

partnerships, 452. *See also* Microsoft Partner Ecosystem

Patriot Act, 244

PDF (Portable Document Format)

- defined, 15
- development, 443
- iFilters, 177, 442–443
- living document conversion, 444
- markup, 443
- OCR, 442–443
- viewing, 443–444

PDF/A, 444–446

people search, 21, 203, 205

People section, User Profile Service

- Application, 166

Perform Volume Maintenance Tasks privilege, 387–388

performance

- optimization
 - DAM, 287–292
 - farm deployment, 417–418
 - search, 211–214
 - storage, 401–402

PerformancePoint Analytics, 18

permission levels

- document control, 73, 75

- publishing features, 221–222
- permissions
 - document control, 73–75
 - List Unique Permissions threshold, 418
- persistence, workflows and, 90–91
- picture library slideshow Web Part, 280
- pluggable workflow services. *See* workflow services, pluggable
- podcasting, 284
- policies
 - exporting, 266
 - importing, 266–267
 - information management policy, 250, 263–267
 - retention, for document library, 265–266
 - for site collection, 263–264
- Policy, 267
- PolicyCatalog, 267
- PolicyCollection, 267
- PolicyFeature, 267
- PolicyFeatureCollection, 267
- PolicyItem, 267
- PolicyItemCollection, 267
- Portable Document Format. *See* PDF
- Portal Server. *See* SharePoint Portal Server
- POST, 479
- PowerPoint (Microsoft)
 - asset library, 276
 - KnowledgeLake, 463
 - “Microsoft Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats,” 422
 - Office Web Apps, 425, 427, 450
 - remote BLOB service, 404
- presentation options, CQWP, 234–235
- preservation. *See also* storage
 - archive formats, 437–438
 - AvePoint, 459
 - defined, 9
 - ECM component, 9–11
 - hardware components, 10–11
 - software components, 10
- pre-sizing
 - content databases, 395
 - crawl databases, 394
 - database files, 388–390
 - log files, 392
 - property databases, 396–397
 - TempDB, 391
 - preupgradecheck, 491, 492
 - privacy concerns, social tagging, 143–144
 - ProfileBase, 159
 - ProfileLoader, 160
 - ProfileManagerBase, 160
 - ProfilePropertyManager, 160
 - ProfileSearchManager, 160
 - ProfileSubtype, 160
 - ProfileSubtypeProperty, 160
 - ProfileSubtypePropertyManager, 160
 - ProfileTypeProperty, 160
 - ProfileTypePropertyManager, 160
 - ProfileValueCollectionBase, 160
 - Project Page content type, 224
 - promotion
 - global, 107
 - property, 379
 - Property, 160
 - property catalog, 396
 - property databases, 396–397
 - property promotion, 379
 - PropertyBase, 160
 - property-based searches, 178–179, 182, 187, 192, 199–202
 - PropertyBaseManager (T), 160
 - PropertyConstants, 160
 - PropertyDataType, 160
 - protocols
 - importing content, 474–479
 - updating SharePoint content, 480–486
 - publishing. *See also* deployment
 - features, 219–222
 - forms, 361–362, 371–372
 - Publishing Infrastructure feature, SharePoint Server, 219–220
 - Publishing Master Page content type, 225
 - Publishing Portal template, 219
 - publishing site content, 225–226
 - Publishing Site template, 219
 - Publishing Site with Workflow template, 219
 - PublishingAssociatedContentType node, 232
 - PublishingPreviewImage node, 232

Q

- queries. *See also* search
 - API based, 203
 - fielded, 396

- options, CQWP, 233–234
- per second (QPS), 182, 204, 206, 207, 208, 209, 210
- performance tuning, 213–214
- query matching processes, 204, 205, 206
- Query Search Service Application, 205–206
- Quick Deploy Users group, 222, 239
- Quick Publish option, 371

R

- RAID (redundant array of independent disks), 11
 - RAID 5 storage array, 382, 383, 395
 - RAID 10 storage array, 391, 393, 394, 395, 397, 401
- rank profiles, Fast Search, 204
- ratings, 28, 139–141
 - enable, 140
 - programmatically working with, 149–151
- RBS. *See* remote BLOB storage
- receive data connections, 368–369
- reciprocity stage, collaboration, 134
- RecordDeclarationPermissions, 260
- recordization, 247, 250–263
- records
 - custom record declaration handler, 261–262
 - defined, 13, 244
 - managers, 245
 - programmatically declaring, 260–261
 - retention, legacy ECM solutions, 489
- Records Center, 238, 253–255
- Records class, 260
- records management, 243–273
 - ARMA International, 13
 - auditing, 270–271
 - Content Organizer, 238, 255–258
 - definitions, 12–13, 29, 244
 - designing, 247–248
 - document management compared to, 12–13, 243
 - ECM management component, 12–13
 - file plan, 247
 - GrimmalSoft, 460
 - importance of, 244
 - in-place, 250–252
 - online information, 13
 - planning, 245–248
 - reports, 271–272

- roles, 245–246
- SharePoint, 245
- redaction, 488
- Redirect Page content type, 224
- redundant array of independent disks. *See* RAID
- refinement panel, keyword search, 190–191
- refiners, Fast Search, 204
- reflection stage, collaboration, 134
- Release button, 304, 355
- releasing document, to SharePoint, 304–305. *See also*
 - WPF-based capture application
- relevance, exact, 396
- remote BLOB storage (RBS), 403–407. *See also* BLOBs
 - backup, 407
 - benefits, 404–405
 - DAM, 286
 - provider options, 406–407
 - restore, 407
 - storage architecture, 403–407
 - when to use, 405–406
- remote procedure calls (RPCs), 479
- reports
 - audit, 271–272
 - capturing, 9
 - Content and Structure Reports List, 225
 - enterprise report management, 4, 6, 9, 15
 - file plan, 272
 - records management, 271–272
 - WCM, 241–242
- repositories, 10
- REST, 368
- restore, RBS, 407
- Restricted Readers group, 222
- retention
 - records, legacy ECM solutions, 489
 - retention policies for document library, 265–266
 - schedules, 247, 267–270
- Reusable Content List, 225
- reuse
 - digital assets, 284
 - master pages, 226
 - workflows, 106–107
- ribbon
 - Fluent UI, 19, 216, 359, 466
 - InfoPath Designer, 359
 - SharePoint Designer, 106
- rich media, 15, 29, 217, 276, 282, 285
- Rich Media Assets, 276, 277

- rich-text editor, 29, 235, 236
- rights management, 16, 488
- roles, records management, 245–246
- routing rules
 - creating, 257–258
 - programmatically managing, 262–263
- rows, indexer, 204, 205
- RPCs (remote procedure calls), 479
- Rule Inspector, 367–368
- rules, 364–368
 - action, 366–367
 - common characteristics, 364
 - custom form exercise, 376
 - formatting, 366
 - validation, 364–366

S

- SANs (storage area networks), 11, 382, 383–384
- Sarbanes-Oxley (SOX), 5, 244, 246, 248
- saving forms, 369
- scalability
 - FAST Search, 204–205
 - SharePoint, 23–24
- scalable ECM architecture. *See* farm deployment
- scalable taxonomy
 - Content Organizer, 413–417
 - design exercise, 415–416
- scanning
 - legacy ECM solutions, 488
 - WPF-based capture application, 303–304
- SCOM (System Center Operations Manager), 212
- scopes
 - search, 192–198
 - term sets, 233
 - workflow, 92–93
- search, 173–214
 - ABBY
 - full-text, 455–456
 - unstructured, 456–457
 - advanced property-based, 182, 187, 199–202
 - API, 203
 - crawl databases, 393–395
 - crawling
 - corpus profile, 176
 - crawled properties, 175, 179, 184, 187, 338
 - FAST Search, 205
 - iFilters, 176–177
 - improving, 213
 - MOSS, 179
 - defined, 28
 - delivery, 13–14
 - document imaging, 294
 - document imaging system, 338–339
 - Enterprise Search, 180–203
 - architectures, 206–211
 - configuration components, 184–189
 - FAST Search for SharePoint 2010, 180
 - improvements in, 179–180
 - topology components, 180–184
 - FAST (FAST Search and Transfer)
 - Microsoft acquisition, 179, 203
 - search crawlers, 14
 - team, 179–180
 - FAST Query Language, 203, 204
 - FAST Search Authorization, 205, 206
 - FAST Search Connector, 205, 206
 - FAST Search for SharePoint 2010, 203–206
 - development of, 179–180
 - Enterprise Search, 180
 - functional overview, 203–205
 - FQL, 203, 204
 - history, 14, 174, 178–179
 - importance, 173, 214
 - introduction, 173–174
 - KnowledgeLake, 464–465
 - legacy ECM search solutions, 488
 - metadata, 191–192
 - metadata navigation, 14
 - MOSS, 174, 178–179, 180, 182, 184, 185
 - people, 21, 203, 205
 - performance tuning, 211–214
 - property-based, 178–179, 182, 187, 192, 199–202
 - queries
 - performance tuning, 213–214
 - QPS, 182, 204, 206, 207, 208, 209, 210
 - query matching processes, 204, 205, 206
 - Query Search Service Application, 205–206
 - retrieval, 174–175
 - as SharePoint category, 21
 - user adoption of, 174–175
 - WCM, 240–241
- search administrator responsibilities, 175
- search center, 189–203
 - defined, 189
 - keyword search, 190–192
 - metadata search, 191–192
 - new features, 190
- Search Core Results Web Part, 195, 341, 343, 355

- Search Core Results XSLT (code listing), 343–354
- search scopes, 192–198
- Secure Store service, 374, 399
- security
 - digital rights management, 16
 - digital signatures, 16, 94, 360, 367
 - document control, 73–79
 - document imaging, 5
 - document level, 16, 73
 - document management feature, 34
 - ECM, 16
 - Information Rights Management, 16, 488
 - metadata-based, 487
 - parent level, 16
 - publishing features, 221–222
 - social tagging, 143–144
 - trimming, 143, 144, 203, 206
 - type-level, 16
- self-synchronization stage, collaboration, 134
- semi-structured content, 8, 239
- Send to Connections, 254, 255, 257, 414, 415, 416, 418
- sequential workflows, 90
- ServerAPIVersionUpdater utility application, 482, 486
- server-side object model. *See also* client-side object models
 - client-side object model compared to, 37
 - defined, 35
 - document library creation with, 38
 - groups created with, 77–78
 - new document version, 480–483
 - site column, 42–43
 - site content type created with, 45–46
- service application databases, 397–400
 - Application Registry Service, 398
 - Business Connectivity Service, 24, 205, 360, 362, 398, 463
 - Subscription Settings, 398
 - Usage and Health Data Collection, 398, 400
 - User Profile, 398
- service applications, 23–24. *See also specific service applications*
- shared SANs, 383–384
- SharedFieldCollection, 72
- SharePoint (Microsoft SharePoint 2010)
 - architecture, 23–26
 - capability categories, 19–21
 - collaboration, 134
 - compliance scenarios, 248–249
 - ECM features, 27–29
 - extensibility, 26
 - farms upgraded to, 291–293
 - history, 18
 - migrating to, 470–490
 - object hierarchy, 21–23
 - platform, 17–30
 - records management, 245
 - scalability, 23–24
 - scalable ECM farm deployment, 381–418
 - service applications, 23–24
 - 64-bit, 177
 - TCO, 174, 487
 - workflow's role in, 91–94
- SharePoint APIs, 35. *See also* SharePoint object models
- SharePoint Designer
 - declarative workflows, 92, 105–114, 125–126
 - Ribbon, 106
 - task process, 107–108
 - taxonomy and page layouts exercise, 227–232
- SharePoint document imaging system. *See* document imaging system
- SharePoint ECM ecosystem, 26, 451–468. *See also* Microsoft Partner Ecosystem
 - defined, 468
 - ISV solutions, 454–467
 - technical community, 453–454
- SharePoint Foundation, 18, 35, 74, 76, 272, 406, 425, 429
- SharePoint infrastructure setup. *See* document imaging system
- SharePoint object models, 35, 37, 144, 145, 147, 149, 159, 160, 379. *See also* client-side object models; server-side object model
- SharePoint Portal Server, 13, 18, 178, 454, 463, 491
- SharePoint Server 2007 (MOSS)
 - farm, 491–493
 - search, 174, 178–179, 180, 182, 184, 185
- SharePoint Server 2010 (MSS), 35. *See also* Enterprise Search
 - history, 18
 - levels, 35
 - Publishing feature, 220
 - Publishing Infrastructure feature, 219–220
- SharePoint server object model. *See* server-side object model
- SharePoint Services, Windows, 13, 16, 18, 178, 491–492
- SharePoint Workspace, 18, 20, 134, 170–171, 360, 369
- signatures, digital, 16, 94, 360, 367

- Silverlight
 - HTML Bridge, 326
 - media Web Part, 278
 - MVVM, 296
 - Web Part, 325, 327, 341
- Silverlight-based viewer Web Part, 322–338
 - architecture, 323
 - code listing, 331–338
 - deployment, 327
 - design, 323
 - image loader, 323–324
 - imaging HTTP handler, 325
 - imaging web service, 324, 328–331
 - implementation, 323–327
 - JavaScript, 322, 326–327
 - in solution data flow diagram, 296
 - viewer application, 325–326
- sites
 - defined, 36
 - in SharePoint object hierarchy, 22
 - SharePoint sites category, 19
- Site Collection Documents Library, 225
- site collections
 - content type hubs, 56–57
 - policies for, 263–264
 - in SharePoint object hierarchy, 22
- site columns
 - creating, with server side object model, 42–43
 - defined, 39
 - digital asset, 276–277
 - gallery, 39, 44
- site content types
 - creating, with server-side object model, 45–46
 - gallery, 44
- site level scope, workflow, 93
- site templates, for WCM, 218–219
- SiteDataQuery, 178
- 64-bit SharePoint, 177
- small farms, 409–410, 411
- SOAP, 75, 368, 369
- Social Connector Plugin, 171–172
- social media. *See* social networking
- social networking. *See also* collaboration
 - collaboration, 13, 133, 134, 165–170, 172, 453–454, 468
 - Facebook, 133, 453
 - Friendster, 133
 - LinkedIn, 13, 133, 172, 453
 - MySpace, 133
 - SharePoint ECM ecosystem, 453–454
 - Twitter, 13, 133, 453
 - YouTube, 13, 133
- social tagging, 134–151. *See also* bookmarklets; notes; ratings; tags
 - privacy concerns, 143–144
 - programming model, 144–151
 - security concerns, 143–144
 - Tags and Notes feature, 135, 136, 137, 138, 142, 143, 152
- SocialComment, 144
- SocialCommentManager, 144
- SocialData, 144
- SocialDataManager, 144
- SocialRating, 144
- SocialRatingAverage, 144
- SocialRatingManager, 144
- SocialTag, 144
- SocialTagManager, 144
- SocialTerm, 144
- SocialUrl, 144
- solutions. *See also* legacy ECM solutions; *specific solutions*
 - data flow diagram, document imaging system, 295–296
 - defined, 26
 - ISV, 454–467
 - scenarios
 - DAM, 280–284
 - records management, 247–248
- SOX (Sarbanes-Oxley), 5, 244, 246, 248
- SPBultInContentTypeId, 45
- SPContentType, 45
- SPContentTypeCollection, 45
- SPDocumentLibrary, 37
- SPField, 41
- SPFieldBoolean, 41
- SPFieldCalculated, 41
- SPFieldChoice, 41
- SPFieldCollection, 41
- SPFieldCurrency, 41
- SPFieldDateTime, 41
- SPFieldLink, 45
- SPFieldLookup, 41
- SPFieldLookupValue, 41
- SPFieldMultiChoice, 42
- SPFieldMultiChoiceValue, 42
- SPFieldMultiLineText, 42
- SPFieldNumber, 42

- SPFieldText, 42
- SPFieldUrl, 42
- SPFieldUser, 42
- SPFieldUserValue, 42
- SPGroup, 76
- SPGroupCollection, 76
- SPList, 37
- SPListCollection, 37
- SPListItem, 37
- SPRoleAssignment, 77
- SPRoleAssignmentCollection, 77
- SPRoleDefinition, 76
- SPRoleDefinitionBindingCollection, 77
- SPRoleDefinitionCollection, 77
- SPSite, 37
- SPUser, 76
- SPUserCollection, 76
- SPWeb, 37
- SPWorkflowActivationProperties, 117, 121
- SPWorkflowExternalDataExchangeService, 127, 128, 129
- SQL Server
 - AD service account, 386–387
 - crawl databases, 393–395
 - database storage and capacity planning, 386–390
 - farm deployment, 412–413
 - instant file initialization, 386–388
 - licensing, RBS, 407
 - log files, 392–393
 - Management Studio, 389, 403
 - optimizing storage array utilization, 386
 - Perform Volume Maintenance Tasks privilege, 387–388
 - pre-sizing database files, 388–390
 - receive data connections, 368
 - SharePoint architecture with, 25
 - TempDB, 390–392
- Stack Exchange, 133
- state machine workflows, 90
- storage, 9–11. *See also* preservation
 - AvePoint, 458–459
 - BLOBs
 - bit rate throttling, 289–292
 - caching, 287–288
 - defined, 286
 - WORM devices, 405–406
 - CAS devices, 404, 405
 - cloud, 11, 405
 - DAM, 285–287
 - DAS, 383–384
 - as document management feature, 34
 - EBS, 404, 458, 491
 - as ECM component, 9–11
 - Fibre Channel, 383
 - hardware components, 10–11
 - NAS, 11, 383–384
 - optical, 6, 9, 11
 - paper *v.* electronic document storage, 293–294
 - performance monitoring, 401–402
 - preservation, 9–11
 - RAID, 11
 - RAID 5 storage array, 382, 383, 395
 - RAID 10 storage array, 391, 393, 394, 395, 397, 401
 - RBS, 403–407
 - backup, 407
 - benefits, 404–405
 - DAM, 286
 - provider options, 406–407
 - restore, 407
 - storage architecture, 403–407
 - when to use, 405–406
 - SANs, 11, 382, 383–384
 - software components, 10
- storage architecture, 381–407. *See also* farm deployment
 - content storage size factors, 384–385
 - DAS, 383–384
 - database storage and capacity planning, 385–400
 - autogrowth adjustment, 388–390
 - content databases, 395–396
 - crawl databases, 393–395
 - disk allocation priority, 400
 - instant file initialization, 386–388
 - log files, 392–393
 - optimizing storage array utilization, 386
 - pre-sizing database files, 388–390
 - property databases, 396–397
 - service application databases, 397–400
 - SQL Server supporting concepts, 386–390
 - TempDB, 390–392
 - disks in array, 382–383
 - index partition storage, 401
 - IOPS requirements, 382–383
 - NAS, 383–384
 - performance pitfalls, 382–385
 - remote BLOB, 403–407
 - shared SAN, 383–384

- tuning and optimization, 401–403
- storage area networks (SANs), 11, 382, 383–384
- storage array utilization, 386
- storage volume performance and reliability
 - content databases, 395–396
 - crawl databases, 394–395
 - log files, 393
 - property databases, 397
 - TempDB, 391–392
- StringMatchOption, 52
- structured content, 7
- STSADM, 491
- Style Library, 222, 225, 234
- style sheets, 74, 222, 225
- Style Source Readers group, 222
- submission, forms, 369, 376–378
- submit data connections, 369
- Subscription Settings database, 398
- supported limits. *See also* thresholds
 - content database size, 416–418
 - defined, 414
 - thresholds compared to, 414
- Synchronization section, User Profile Service
 - Application, 168
- SysInternals DebugView, 477, 481, 482, 485
- System Center Operations Manager (SCOM), 212
- System.Diagnostics.Trace.WriteLine(), 477

T

- Tagged Image File Format. *See* TIFF
- tags, 135–137. *See also* social tagging
 - creating, 135–136
 - defined, 135
 - programmatically working with, 145–147
 - tag clouds, 137, 152
 - Tags and Notes feature, 135, 136, 137, 138, 142, 143, 152
- target dialog, WPF-based capture application, 300–301
- TargetDialog, 300–301
- task process, 107–108
- task properties, workflow, 122–125
- taxonomy
 - DAM, 284–285
 - defined, 284
 - design, farm deployment, 413–417
 - document, 35–73
 - folksonomy compared to, 135
 - page layouts and taxonomy exercise, 227–232
 - scalable, Content Organizer for, 414–416
- TaxonomyField, 53
- TaxonomyFieldControl, 53
- TaxonomyFieldEditor, 53
- TaxonomyFieldValue, 53
- TaxonomyFieldValueCollection, 53
- TaxonomyItem, 53
- TaxonomyRights, 53
- TaxonomySession, 53
- TaxonomyWebTaggingControl, 53
- TCO (total cost of ownership), 174, 487
- TempDB, 390–392
- templates. *See specific templates*
- 10-million-item corpus, 208
- Term, 53
- terms, 28, 47
- term sets
 - in columns, 50
 - creating, 53–54
 - data, reading, 54–55
 - defined, 28, 47, 233
 - enterprise keywords, 28, 48, 233
 - global, 47, 49, 233
 - local, 47, 233
 - scopes, 233
 - Term Set Management Tool, 49
- term stores
 - defined, 48
 - Term Store Management Tool, 48–49
- TermCollection, 53
- terminators, 101
- TermSet, 53
- TermSetCollection, 53
- TermSetItem, 53
- TermStore, 53
- TermStoreCollection, 53
- test-spcontentdatabase, 492
- 3-million-item corpus, 208
- Three-state workflow, 94, 109
- thresholds
 - defined, 414
 - resource throttling, 417–418
 - supported limits compared to, 414
- TIFF (Tagged Image File Format), 438–442
 - defined, 15
 - development, 442
 - iFilters, 438–442
 - markup, 442
 - NULL iFilter, 176

OCR, 438–442
 Silverlight-based viewer Web Part, 322
 time to first byte, 287
 timer jobs, 25, 26, 52, 66, 139, 140, 157, 168, 416
 Title node, 232
 tokens, correlation, 118–119
 total cost of ownership (TCO), 174, 487
 training center, DAM, 283–284
 transaction log growth, content database, 490
 transaction logs. *See* log files
 transactional content management system, 465
 transformation technologies, 14–15. *See also* file formats
 TreeControl, 53
 trimming, security, 143, 144, 203, 206
 tunable relevance, Fast Search, 204
 Turnbull, 244
 Twitter, 13, 133, 453
 type-level security, 16

U

unstructured content, 8, 456
 unstructured searching, ABBYY, 456–457
 updating SharePoint content, 480–486
 upgrading farms, to SharePoint 2010, 491–493. *See also*
 migrating to SharePoint 2010
 upload size
 DAM, 286–287
 legacy ECM solutions, 490
 Usage and Health Data Collection, 398, 400
 user groups, SharePoint ECM ecosystem, 454
 user profiles, 157–165
 activity feeds, 159
 programmatically creating, 163–165
 programmatically retrieving, 162–163
 policies, 158
 programming model, 159–165
 User Profile Databases, 398
 User Profile Service Application, 145, 151, 156, 157,
 161, 165–168
 user-impersonation step, 108
 UserProfile, 160
 UserProfileConfigManager, 160
 UserProfileManager, 160
 users
 document control, 76
 managing, 76–79

V

validating process, Visio workflows, 103
 validation rules, 364–366
 variations, 224, 241
 versioning, 12, 73, 81–84, 215
 document versions, 487
 new document version, 480–486
 version history, 34, 69, 81–83, 482, 486
 video podcasting, 284
 View component, MVVM, 297, 298, 300
 viewing. *See also* browsers
 delivery, 14
 document image, 294
 as ECM component, 14
 KnowledgeLake, 463–464
 Office Web Apps, 421, 425–428
 PDF, 443–444
 PDF/A, 446
 ViewModel component, MVVM, 297, 298, 299, 300,
 301, 325
 virtual log files (VLFs), 388
 virtual servers, 211
 Visio declarative workflows, 92, 99–105
 Visio Services, 20, 24, 109
 Visual Basic .NET, 363, 370
 Visual Studio
 debugging, 26, 116, 117
 features, 26–27
 improvements, 115–116
 taxonomy and page layouts exercise, 227–232
 Tools for Applications, 370
 workflows, 92, 114–126
 visualization
 tag clouds, 137
 workflow, 99, 109–110
 VLFs (virtual log files), 388
 .vwi, 104–105

W

WCAG (Web Content Accessibility Guidelines), 216–217
 WCM. *See* web content management
 Web Analytics service, 399, 400, 413
 web applications
 defined, 36
 IIS compared to, 24
 SharePoint architecture, 24
 Web Apps. *See* Office Web Apps

- web browsers. *See* browsers
- Web Content Accessibility Guidelines (WCAG), 216–217
- web content management (WCM), 215–242
 - analytics, 241–242
 - branding, 240
 - content authoring process, 216, 235–239
 - content types, 223–225
 - defined, 12, 29, 215
 - ECM as, 3
 - as ECM management component, 12
 - improvements, 216–218
 - navigation, 240–241
 - overview, 215–216
 - reporting, 241–242
 - search, 240
 - site templates, 218–219
 - usage scenarios, 218
 - wikis, 218
 - workflows, 239
- web front end (WFE) servers, 39, 207, 208, 209, 210, 211, 287
- Web Parts. *See also specific Web Parts*
 - DAM, 278–280
 - document imaging system, 340–342
- web servers, farm deployment, 409, 410, 411, 412
- web services
 - copy.asmx, 474–477
 - imaging, 324, 328–331
 - submit data connections, 369
- weblogs. *See* blogs
- Welcome Page content type, 224
- WelcomePageFieldCollection, 72
- WF. *See* Workflow Foundation
- WFE. *See* web front end servers
- Wikipedia, 133, 218, 239
- wikis
 - benefits of, 169
 - collaboration, 133, 134
 - Enterprise Wiki Page content type, 224
 - Enterprise Wiki site template, 168–169, 219
 - Enterprise WikiLayouts folder, 221
 - SharePoint communities category, 20
 - WCM, 216, 218, 239–240
- wildcard prefix, 190
- Windows Forms, 430
- Windows Presentation Foundation. *See* WPF
- Windows SharePoint Services. *See* WSS
- Windows Workflow Foundation. *See* Workflow Foundation
- Word (Microsoft)
 - asset library, 276
 - KnowledgeLake, 463
 - “Microsoft Office Compatibility Pack for Word, Excel and PowerPoint 2007 File Formats,” 422
 - multi-author editing capabilities, 20
 - Office Web Apps, 425, 427, 450
 - remote BLOB service, 404
- Word Automation Services, 428–436
 - database, 400
 - defined, 428
 - developing with, 430–436
 - modified class for monitoring (code listing), 434–436
 - sample class (code listing), 432
- Word Perfect, 6
- WordStar, 6
- workflows, 87–131
 - actions
 - SharePoint Designer, 111
 - Visio, 101–103
 - Approval, 94, 95–99
 - BPM, 6, 12
 - Collect Feedback, 94
 - Collect Signatures, 94
 - correlation, 91
 - defined, 12, 28
 - Disposition Approval, 94
 - early systems, 4, 6
 - ECM management component, 12
 - high-privilege, 108
 - InfoPath, 125–126
 - KnowledgeLake, 465
 - legacy ECM solutions, 489
 - modes, 90
 - Nintex, 465–466
 - out-of-the-box, 94–99
 - customizing, 109
 - InfoPath, 125
 - persistence, 90–91
 - phases, 93–94
 - recordization, 258–259
 - reusable, 106–107
 - role of, in SharePoint, 91–94
 - scopes, 92–93
 - sequential, 90

- state machine, 90
- Three-state, 94, 109
- types, 94
- Visual Studio, 92, 114–126
- visualization, 99, 109–110
- WCM, 239
- workflow event receivers, 130–131
- Workflow Foundation (WF), 88–91
- workflow initiation forms, 120, 126
- workflow pages, 119–122, 126
- workflow services, pluggable, 126–130
- workflow task properties, 122–125
- Workspace, SharePoint, 18, 20, 134, 170–171, 360, 369
- workspaces, 170–171
- WorldCom, 244
- WORM devices, 405–406
- WPF (Windows Presentation Foundation)
 - data binding, 297, 300
 - MVVM, 296
- WPF-based capture application, 298–322
 - architecture, 299
 - code listing, 305–322
 - deployment, 322
 - design, 299
 - implementation, 299–322
 - loading columns, 301–303
 - main window, 301–305
 - MVVM infrastructure, 299–300
 - releasing document to SharePoint, 304–305
 - scanning an image, 303–304
 - screenshot, 298
 - in solution data flow diagram, 296
 - target dialog, 300–301
- WSS (Windows SharePoint Services), 13, 16, 18, 178, 491–492
- WYSIWYG editor, 216

X

- XAML, 280, 297, 300
- XAP file, 327
- XHTML, 216, 217, 224, 375
- XML (Extensible Markup Language). *See also* CAML; InfoPath; Open XML
 - advanced property-based searches, 200
 - copy.asmx web service, 474–477
 - custom advanced search property XML (code listing), 342–343
 - declarative workflows, 99, 105
 - defined, 15
 - InfoPath, 363, 370
 - receive data connections, 368
- XML Paper Specification. *See* XPS
- XPS (Open XML Paper Specification), 446–450
 - defined, 15
 - development, 449–450
 - iFilters, 447–449
 - markup, 449
 - OCR, 447–449
 - Open XML compared to, 446
- XSLT, 217, 225, 234, 362
- XSLT listing (Search Custom Core Results XSLT), 343–354

Y

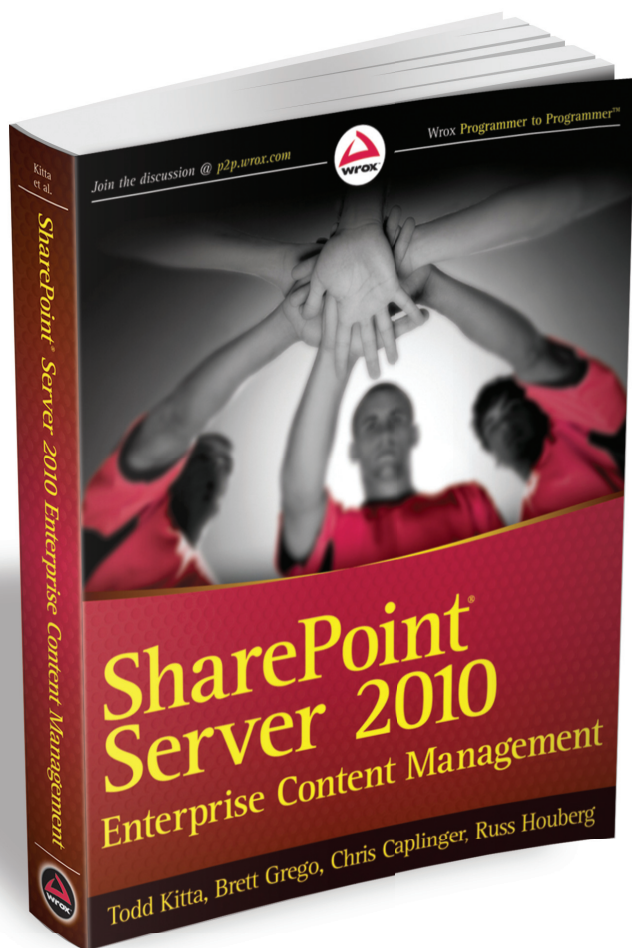
- Yammer, 133
- YouTube, 13, 133

Z

- zip files, 105
- Zonal OCR, 7
- zones, 24, 25, 278, 462

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 www.safaribooksonline.com/wrox9 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.



An Imprint of **WILEY**
Now you know.

Safari >>
Books Online

www.it-ebooks.info