

**Jeff Conrad**

*Software test engineer for Access and former Microsoft MVP*

**John L. Viescas**

*Popular Access author and Microsoft MVP*

Microsoft®  
**Access® 2010**

**INSIDE  
OUT**

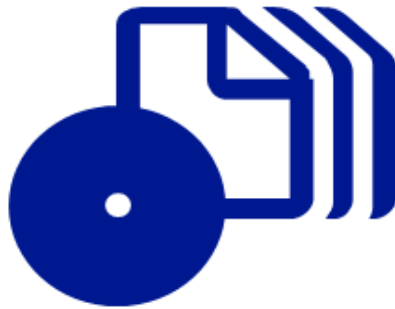
CD INCLUDES:

- Sample database applications
- Bonus chapters and technical articles
- Complete eBook

- The ultimate, in-depth reference
- Hundreds of timesaving solutions
- Supremely organized book and CD



# How to access your CD files



The print edition of this book includes a CD. To access the CD files, go to <http://aka.ms/626850/files>, and look for the Downloads tab.

Note: Use a desktop web browser, as files may not be accessible from all ereader devices.

Questions? Please contact: [mspinput@microsoft.com](mailto:mspinput@microsoft.com)

Microsoft Press





**Microsoft®**

# Microsoft® Access® 2010 Inside Out

Jeff Conrad  
John Viescas

Copyright © 2010 Jeff Conrad and John Viescas.

Complying with all applicable copyright laws is the responsibility of the user. All rights reserved. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without express written permission of Microsoft Press, Inc.

ISBN: 978-0-7356-2685-0

Printed and bound in the United States of America.

3 4 5 6 7 8 9 10 11 QG 7 6 5 4 3 2

Microsoft Press titles may be purchased for educational, business or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or send comments to [mshpinput@microsoft.com](mailto:mshpinput@microsoft.com).

Microsoft, Microsoft Press, ActiveX, Excel, FrontPage, Internet Explorer, PowerPoint, SharePoint, Webdings, Windows, and Windows 7 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

This book expresses the author's views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the author, Microsoft Corporation, nor their respective resellers or distributors, will be held liable for any damages caused or alleged to be caused either directly or indirectly by such information.

**Acquisitions and Development:** Juliana Aldous and Kenyon Brown

**Production Editor:** Rachel Monaghan

**Editorial Production:** Custom Editorial, Inc. and Nancy Kotary

**Technical Reviewers:** Andrew Couch and Jim Bailie

**Indexing:** Potomac Indexing, LLC

**Cover:** Karen Montgomery

**Compositor:** Nellie McKesson

**Illustrator:** Robert Romano

*For my family, Cheryl, Amy, Aaron, and Arica, for your love and support.  
You are the reason I have written this book and I couldn't have done it  
without you.*

*—Jeff Conrad*

*To Jeff, who adopted my book and made it his own.*

*—John Viescas  
Paris, France  
June 2010*







# Contents at a Glance

## Part 1: Understanding Access

Chapter 1	
What Is Access? .....	3
Chapter 2	
Exploring the Access 2010 Interface .....	21
Chapter 3	
Access 2010 Overview .....	125

## Part 2: Creating a Database and Tables

Chapter 4	
Designing Client Tables .....	167
Chapter 5	
Modifying Your Table Design .....	237
Chapter 6	
Designing Web Tables .....	287
Chapter 7	
Creating Table Data Macros .....	361
Chapter 8	
Importing and Linking Data .....	445

## Part 3: Building Queries

Chapter 9	
Creating and Working with Simple Queries ..	545
Chapter 10	
Building Complex Queries .....	621
Chapter 11	
Modifying Data with Action Queries .....	703

## Part 4: Creating Forms

Chapter 12	
Using Forms in an Access Application .....	737
Chapter 13	
Building a Form .....	785
Chapter 14	
Customizing a Form .....	841
Chapter 15	
Advanced Form Design .....	945

## Part 5: Working with Reports

Chapter 16	
Using Reports .....	1023
Chapter 17	
Constructing a Report .....	1045
Chapter 18	
Advanced Report Design .....	1107

## Part 6: Automating an Access Application Using Macros

Chapter 19	
Understanding Event Processing .....	1167
Chapter 20	
Automating a Client Application Using Macros .....	1191
Chapter 21	
Automating a Web Application Using Macros .....	1253

## Part 7: Working with the Web

Chapter 22

Using Web Applications in a Browser. . . . . 1287

Chapter 23

Using Business Connectivity Services. . . . . 1347

Appendix

Installing Your Software . . . . . 1375

Index. . . . . 1389



*Bonus Content on the Companion CD*

## Part 8: Automating an Access Application Using Visual Basic

Chapter 24

Understanding Visual Basic Fundamentals . 1451

Chapter 25

Automating Your Application  
with Visual Basic. . . . . 1583

## Part 9: After Completing Your Application

Chapter 26

The Finishing Touches. . . . . 1665

Chapter 27

Distributing Your Application . . . . . 1715

Article 1

Designing Your Database Application . . . . 1743

Article 2

Understanding SQL . . . . . 1773

Article 3

Exporting Data . . . . . 1823

Article 4

Function Reference . . . . . 1831

Article 5

Color Names and Codes . . . . . 1841

Article 6

Macro Actions. . . . . 1849



# Table of Contents

Acknowledgments .....	.xxv
About the CD .....	.xxvii
What's on the CD .....	.xxvii
Sample Applications .....	.xxvii
System Requirements .....	.xxviii
Support Information .....	.xxx
Conventions and Features Used in This Book .....	.xxxi
Text Conventions .....	.xxxi
Design Conventions .....	.xxxi
Syntax Conventions .....	.xxxii
Introduction .....	.xxxv
Getting Familiar with Access 2010 .....	.xxxvi
About This Book .....	.xxxvi

## Part 1: Understanding Access

Chapter 1:	<b>What Is Access?</b> .....	<b>3</b>
	What Is a Database? .....	4
	Relational Databases .....	4
	Database Capabilities .....	6
	Access as an RDBMS .....	6
	Data Definition and Storage .....	7
	Data Manipulation .....	9
	Data Control .....	12
	Access as an Application Development System .....	13
	Deciding to Move to Database Software .....	15
	Extending the Power of Access to the Web .....	17

---

### What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

[microsoft.com/learning/booksurvey](http://microsoft.com/learning/booksurvey)



Chapter 2:	<b>Exploring the Access 2010 Interface</b>	<b>21</b>
	Opening Access for the First Time	21
	Getting Started with Access 2010	23
	Opening an Existing Database	25
	Exploring the Microsoft Office Backstage View	27
	Taking Advantage of the Quick Access Toolbar	39
	Understanding Content Security	47
	Enabling a Database That Is Not Trusted	49
	Understanding the Trust Center	52
	Enabling Content by Defining Trusted Locations	55
	Understanding the Office Fluent Ribbon	57
	Home Tab	58
	Create Tab	59
	External Data Tab	60
	Database Tools Tab	61
	Customizing the Ribbon	63
	Understanding the Navigation Pane	70
	Exploring Navigation Pane Object Views	72
	Working with Custom Categories and Groups	78
	Sorting and Selecting Views in the Navigation Pane	99
	Manually Sorting Objects in the Navigation Pane	101
	Searching for Database Objects	102
	Using the Single-Document vs. the Multiple-Document Interface	107
	Modifying Global Settings via the Access Options Dialog Box	112
Chapter 3:	<b>Access 2010 Overview</b>	<b>125</b>
	The Architecture of Access	125
	Exploring a Desktop Database—Housing Reservations	128
	Tables	133
	Queries	138
	Forms	142
	Reports	149
	Macros	156
	Modules	158
	What Happened to Project Files (ADP)?	161
	The Many Faces of Access	161

## Part 2: Creating a Database and Tables

Chapter 4:	<b>Designing Client Tables</b>	<b>167</b>
	Creating a New Database	168
	Using a Database Template to Create a Database	169
	Creating a New Empty Database	174
	Creating Your First Simple Table by Entering Data	176

Creating a Table Using Application Parts .....	178
Creating a Table Using Data Type Parts .....	182
Creating a Table in Design View .....	186
Defining Fields .....	187
Understanding Field Data Types .....	190
Setting Field Properties .....	193
Completing the Fields in the Companies Table .....	200
Defining Simple Field Validation Rules .....	201
Defining Input Masks .....	204
Defining a Primary Key .....	208
Defining a Table Validation Rule .....	209
Understanding Other Table Properties .....	212
Defining Relationships .....	215
Defining Your First Relationship .....	217
Creating a Relationship on Multiple Fields .....	220
Adding Indexes .....	222
Single-Field Indexes .....	223
Multiple-Field Indexes .....	224
Setting Table Design Options .....	226
Creating a Default Template for New Databases .....	230
Printing a Table Definition .....	233
Database Limitations .....	235
<b>Chapter 5: Modifying Your Table Design .....</b>	<b>237</b>
Before You Get Started .....	239
Making a Backup Copy .....	239
Checking Object Dependencies .....	242
Deleting Tables .....	244
Renaming Tables .....	245
Changing Field Names .....	247
Moving Fields .....	251
Inserting Fields .....	255
Copying Fields .....	257
Deleting Fields .....	260
Changing Data Attributes .....	261
Changing Data Types .....	262
Changing Data Lengths .....	266
Dealing with Conversion Errors .....	267
Changing Other Field Properties .....	268
Reversing Changes .....	269
Using the Table Analyzer Wizard .....	270
Taking a Look at Lookup Properties .....	275
Working with Multi-Value Lookup Fields .....	280
Changing the Primary Key .....	283
Compacting Your Database .....	285

Chapter 6:	<b>Designing Web Tables . . . . .</b>	<b>287</b>
	Working with the Web . . . . .	288
	Creating a New Web Database . . . . .	290
	Using a Database Template to Create a Web Database. . . . .	291
	Creating a New Empty Web Database . . . . .	295
	Creating Your First Simple Web Table by Entering Data. . . . .	297
	Creating a Web Table Using Application Parts. . . . .	300
	Using Data Type Parts. . . . .	304
	Creating Web Tables in Datasheet View . . . . .	306
	Defining Web Fields . . . . .	307
	Understanding Web Field Data Types. . . . .	311
	Setting Field Properties for Web Databases. . . . .	315
	Completing the Fields in the Vendors Web Table. . . . .	318
	Creating Calculated Fields. . . . .	319
	Defining Field Validation Rules for Web Databases . . . . .	327
	Defining a Table Validation Rule for Web Databases . . . . .	332
	Defining a Primary Key for Web Databases . . . . .	335
	Understanding Other Web Table Properties. . . . .	336
	Creating Lookup Fields in a Web Database. . . . .	337
	Creating Relationships Using Lookup Fields . . . . .	341
	Defining a Restrict Delete Relationship. . . . .	343
	Defining a Cascade Delete Relationship. . . . .	349
	Using the Web Compatibility Checker. . . . .	351
	Analyzing the Web Compatibility Issues Table . . . . .	352
	Preparing a Client Database for the Web. . . . .	357
Chapter 7:	<b>Creating Table Data Macros . . . . .</b>	<b>361</b>
	Uses of Data Macros . . . . .	362
	The Data Macro Design Facility—An Overview . . . . .	364
	Redesigning the Macro Window from Previous Versions of Access . . . . .	364
	Touring the New Logic Designer . . . . .	366
	Working with Before Events. . . . .	369
	Before Change. . . . .	370
	Preventing Duplicate Records Across Multiple Fields. . . . .	392
	Before Delete. . . . .	396
	Working with After Events . . . . .	398
	After Insert . . . . .	398
	After Update . . . . .	404
	After Delete . . . . .	409
	Working with Named Data Macros . . . . .	411
	Creating Named Data Macros . . . . .	412
	Saving Named Data Macros . . . . .	415
	Calling Named Data Macros. . . . .	416

Renaming and Deleting Named Data Macros . . . . .	418
Analyzing Errors in the USysApplicationLog Table . . . . .	420
Using Parameters . . . . .	424
Using Local Variables . . . . .	427
Working with Return Variables . . . . .	429
Debugging Data Macros . . . . .	437
Understanding Recursion in Data Macros . . . . .	441
Sharing Data Macro Logic . . . . .	442
<b>Chapter 8: Importing and Linking Data . . . . .</b>	<b>445</b>
A Word About Open Database Connectivity (ODBC) . . . . .	445
Creating a Data Source to Link to an ODBC Database . . . . .	448
Importing vs. Linking Database Files . . . . .	451
Importing Data and Databases . . . . .	452
Importing dBASE Files . . . . .	453
Importing SQL Tables . . . . .	456
Importing Access Objects . . . . .	459
Importing Spreadsheet Data . . . . .	462
Preparing a Spreadsheet . . . . .	463
Importing a Spreadsheet . . . . .	464
Fixing Errors . . . . .	468
Importing Text Files . . . . .	471
Preparing a Text File . . . . .	471
Importing a Text File . . . . .	474
Fixing Errors . . . . .	479
Modifying Imported Tables . . . . .	481
Linking Files . . . . .	481
Security Considerations . . . . .	482
Performance Considerations . . . . .	482
Linking Access Tables . . . . .	484
Linking dBASE Files . . . . .	487
Linking Text and Spreadsheet Files . . . . .	488
Linking SQL Tables . . . . .	490
Modifying Linked Tables . . . . .	491
Unlinking Linked Tables . . . . .	492
Using the Linked Table Manager . . . . .	492
Collecting Data via Email . . . . .	493
Collecting Data via HTML Forms . . . . .	494
Collecting Data Using InfoPath Forms . . . . .	514
Importing and Linking SharePoint Data . . . . .	531
Importing a List from a SharePoint Site . . . . .	532
Linking a SharePoint List into Access . . . . .	535
Saving Import Procedures . . . . .	539



## Part 3: Building Queries

Chapter 9:	<b>Creating and Working with Simple Queries</b>	<b>545</b>
	Selecting Data from a Single Table	548
	Specifying Fields	551
	Setting Field Properties	553
	Entering Selection Criteria	555
	Using Expressions	562
	Using the Expression Builder	572
	Specifying Field Names	582
	Sorting Data	583
	Testing Validation Rule Changes	586
	Checking a New Field Validation Rule	586
	Checking a New Table Validation Rule	588
	Working in Query Datasheet View	589
	Moving Around and Using Keyboard Shortcuts	590
	Working with Subdatasheets	592
	Changing Data	596
	Working with Hyperlinks	603
	Sorting and Searching for Data	607
Chapter 10:	<b>Building Complex Queries</b>	<b>621</b>
	Selecting Data from Multiple Tables	622
	Creating Inner Joins	622
	Building a Query on a Query	630
	Using Outer Joins	634
	Using a Query Wizard	641
	Summarizing Information with Totals Queries	644
	Totals Within Groups	645
	Selecting Records to Form Groups	650
	Selecting Specific Groups	652
	Building Crosstab Queries	652
	Using Query Parameters	660
	Customizing Query Properties	663
	Controlling Query Output	664
	Working with Unique Records and Values	665
	Defining a Subdatasheet	669
	Other Query Properties	673
	Editing and Creating Queries in SQL View	674
	Limitations on Using Select Queries to Update Data	680
	Creating PivotTables and PivotCharts from Queries	681
	Building a Query for a PivotTable	682
	Designing a PivotTable	685
	Designing a PivotChart	690
	Creating Queries for the Web	695

Chapter 11:	<b>Modifying Data with Action Queries</b>	<b>703</b>
Updating Groups of Rows		704
Testing with a Select Query		704
Converting a Select Query to an Update Query		706
Running an Update Query		707
Updating Multiple Fields		709
Creating an Update Query Using Multiple Tables or Queries		711
Creating a New Table with a Make-Table Query		714
Creating a Make-Table Query		714
Running a Make-Table Query		719
Inserting Data from Another Table		721
Creating an Append Query		721
Running an Append Query		725
Deleting Groups of Rows		725
Testing with a Select Query		725
Using a Delete Query		728
Deleting Inactive Data		729
Troubleshooting Action Queries		730
Solving Common Action Query Errors and Problems		730
Looking at an Error Example		731

## **Part 4: Creating Forms**

Chapter 12:	<b>Using Forms in an Access Application</b>	<b>737</b>
Uses of Forms		737
A Tour of Forms		738
Headers, Detail Sections, and Footers		739
Multiple-Page Forms		740
Continuous Forms		741
Split Forms		742
Subforms		742
Pop-Up Forms		743
Modal Forms		744
Special Controls		745
PivotTables and PivotCharts		761
Understanding Web Form Limitations		763
Moving Around on Forms and Working with Data		763
Viewing Data		763
Adding Records and Changing Data		768
Adding a New Record		768
Changing and Deleting Data		776
Searching for and Sorting Data		777
Performing a Simple Search		777
Using the Search Box		779
Performing a Quick Sort on a Form Field		780
Adding a Filter to a Form		780
Printing Forms		783

Chapter 13:	<b>Building a Form</b>	<b>785</b>
	Forms and Object-Oriented Programming	785
	Starting from Scratch—A Simple Input Form	789
	Building a New Form with Design Tools	789
	Building a Simple Input Form for the tblCompanies Table	806
	Customizing Colors and Checking Your Design Results	816
	Working with Quick Create and the Form Wizard	819
	Creating a Form with the Quick Create Commands	819
	Creating the Basic Products Form with the Form Wizard	823
	Modifying the Products Form	826
	Simplifying Data Input with a Form	829
	Taking Advantage of Combo Boxes and List Boxes	829
	Using Toggle Buttons, Check Boxes, and Option Buttons	833
	Working with Application Part Forms	835
Chapter 14:	<b>Customizing a Form</b>	<b>841</b>
	Aligning and Sizing Controls in Design View	841
	Sizing Controls to Fit Content	844
	Adjusting the Layout of Controls	849
	“Snapping” Controls to the Grid	850
	Lining Up Controls	852
	Enhancing the Look of a Form	856
	Lines and Rectangles	857
	Colors and Special Effects	860
	Fonts	862
	Setting Control Properties for Client Forms	865
	Formatting Properties	865
	Adding a Scroll Bar	876
	Enabling and Locking Controls	876
	Setting the Tab Order	877
	Adding a Smart Tag	879
	Understanding Other Control Properties for Client Forms	881
	Setting Client Form Properties	886
	Allowing Different Views	887
	Setting Navigation Options	888
	Defining a Pop-Up and/or Modal Form	888
	Controlling Edits, Deletions, Additions, and Filtering	890
	Defining Window Controls	891
	Setting the Border Style	891
	Understanding Other Client Form Properties	892
	Setting Client Form and Control Defaults	897
	Changing Control Defaults	897
	Defining a Template Form	897
	Working with Web Forms in Layout View	899
	Starting from Scratch—A Simple Input Web Form	900
	Understanding Control Layouts and Control Anchoring	901
	Lining Up Controls	903
	Moving Controls Within a Control Layout	905

Formatting a Column of Controls .....	909
Resizing Controls .....	910
Removing a Control Layout for Client Forms .....	911
Splitting and Merging Cells .....	912
Inserting Rows and Columns .....	914
Using Web-Compatible Controls .....	916
Adding Gridlines to Web Forms .....	918
Adding Some Space with Control Padding .....	921
Creating a Title .....	922
Moving Controls to Different Sections on Web Forms .....	923
Setting Control Properties for Web Forms .....	925
Setting Web Form Properties .....	927
Using Themes .....	928
Working with Shared Resources .....	938
<b>Chapter 15: Advanced Form Design .....</b>	<b>945</b>
Basing a Form on a Multiple-Table Query .....	946
Creating a Many-to-One Form .....	946
Creating and Embedding Subforms .....	952
Specifying the Subform Source .....	954
Designing the Innermost Subform .....	956
Designing the First Level Subform .....	962
Embedding a Subform .....	964
Specifying the Main Form Source .....	968
Creating the Main Form .....	969
Creating a Subdatasheet Subform .....	973
Displaying Values in an Option Group .....	976
Using Conditional Formatting in Client Forms .....	978
Working with the Tab Control .....	985
Creating Multiple-Page Client Forms .....	992
Working with Client PivotChart Forms .....	996
Building a Client PivotChart Form .....	996
Embedding a Linked PivotChart .....	998
Working with Navigation Controls .....	1000
Using Web Browser Controls .....	1014

## Part 5: Working with Reports

<b>Chapter 16: Using Reports .....</b>	<b>1023</b>
Uses of Reports .....	1024
A Tour of Reports .....	1024
Print Preview—A First Look .....	1026
Headers, Detail Sections, Footers, and Groups .....	1028
Subreports .....	1030
Objects in Reports .....	1033
Report View—A First Look .....	1035
Printing Reports .....	1039
Print Setup .....	1039



Chapter 17:	<b>Constructing a Report</b>	<b>1045</b>
	Starting from Scratch—A Simple Report	1045
	Building the Report Query	1046
	Designing the Report	1048
	Grouping, Sorting, and Totaling Information	1050
	Completing the Report	1059
	Using the Report Command	1066
	Using the Report Wizard	1069
	Specifying Report Wizard Options	1069
	Viewing the Result	1074
	Working with Web Reports in Layout View	1076
	Modifying a Report Command-Created Web Report in Layout View	1077
	Completing the Web Report	1083
	Building a Web Report in Layout View	1087
	Starting with a Blank Web Report	1087
	Adding Grouping and Sorting	1089
	Working with Control Layouts	1095
	Adding Totals to Records	1098
	Using Gridlines	1100
	Adding the Final Formatting Touches	1102
	Understanding Web Report Limitations	1105
Chapter 18:	<b>Advanced Report Design</b>	<b>1107</b>
	Building a Query for a Complex Report	1108
	Creating the Basic Facility Occupancy By Date Report	1109
	Defining the Grouping and Sorting Criteria	1111
	Setting Section and Report Properties	1114
	Section Properties for Reports	1115
	Report Properties	1119
	Using Calculated Values	1128
	Adding the Print Date and Page Numbers	1129
	Performing Calculations	1132
	Hiding Redundant Values and Concatenating Text Strings	1138
	Calculating Percentages	1141
	Using Running Sum	1143
	Taking Advantage of Conditional Formatting	1146
	Creating and Embedding a Subreport	1151
	Understanding Subreport Challenges	1152
	Building a Client Report with a Subreport	1155
	Adding a PivotChart to a Client Report	1160
	Designing the PivotChart Form	1160
	Embedding a PivotChart in a Client Report	1161

## Part 6: Automating an Access Application Using Macros

Chapter 19:	<b>Understanding Event Processing</b> .....	<b>1167</b>
	Access as a Windows Event-Driven Application .....	1167
	Understanding Events in Windows .....	1167
	Using Access Events to Build an Application .....	1168
	Summary of Form and Report Events .....	1169
	Understanding Event Sequence and Form Editing .....	1185
Chapter 20:	<b>Automating a Client Application Using Macros</b> .....	<b>1191</b>
	Uses of Macros .....	1193
	The Macro Design Facility—An Overview .....	1194
	Working with the Logic Designer .....	1194
	Saving Your Macro .....	1199
	Testing Your Macro .....	1199
	Defining Multiple Actions .....	1201
	Working with Submacros .....	1204
	Understanding Conditional Expressions .....	1207
	Using Embedded Macros .....	1209
	Editing an Embedded Macro .....	1209
	Creating an Embedded Macro .....	1212
	Deleting an Embedded Macro .....	1215
	Using Temporary Variables .....	1216
	Trapping Errors in Macros .....	1219
	Understanding Macro Actions That Are Not Trusted .....	1226
	Making Your Application Come Alive with Macros .....	1229
	Referencing Form and Report Objects .....	1229
	Opening a Secondary Form .....	1231
	Synchronizing Two Related Forms .....	1235
	Validating Data and Presetting Values .....	1239
Chapter 21:	<b>Automating a Web Application Using Macros</b> .....	<b>1253</b>
	Creating Web Macros .....	1254
	Using Macro Objects for Common Functionality .....	1258
	Working with Web Form and Control Events .....	1260
	Passing Parameters to Forms and Reports .....	1261
	Exploring the Invoice Audit Web Form Macros .....	1266
	Using the SetProperty Action with Web Form Controls .....	1266
	Calling Named Data Macros and Using Return Variables .....	1271
	Using BrowseTo Macro Actions to Browse to Forms and Reports .....	1275
	Trapping Error Messages .....	1278
	Checking SharePoint User Permission Group Levels .....	1279
	Performing Different Actions When Opening a Web Form in a Browser .....	1281
	Avoiding Type Coercion Issues .....	1282

## Part 7: Working with the Web

Chapter 22:	<b>Using Web Applications in a Browser . . . . .</b>	<b>1287</b>
	Working with SharePoint . . . . .	1288
	Publishing Your Database to an Access Services Site . . . . .	1288
	Assigning a Web Display Form . . . . .	1289
	Understanding the Publish Process . . . . .	1290
	Analyzing Publish Errors . . . . .	1297
	Working with Your Application in a Web Browser . . . . .	1299
	Using Forms . . . . .	1299
	Using Reports . . . . .	1304
	Using Datasheet Forms . . . . .	1306
	Waiting for Server Processing . . . . .	1309
	Understanding Session Management . . . . .	1311
	Exploring the Access Services Shell . . . . .	1312
	Downloading a Web Application Back into Access . . . . .	1314
	Setting Site Permissions . . . . .	1315
	Reviewing the Modify Application Page . . . . .	1318
	Working with the Recycle Bin . . . . .	1321
	Extending Your Access Services Application . . . . .	1322
	Using Your Published Web Database in Access . . . . .	1328
	Making Changes to a Published Web Application . . . . .	1330
	Resolving Synchronization Conflicts . . . . .	1335
	Working Offline . . . . .	1337
	Saving a Web Application as a Local Database . . . . .	1341
	Instantiating an Access Services Template . . . . .	1342
	Using an Installed Web Template . . . . .	1343
	Instantiating a Custom Template . . . . .	1344
Chapter 23:	<b>Using Business Connectivity Services . . . . .</b>	<b>1347</b>
	Understanding Web Services . . . . .	1347
	Introducing Business Connectivity Services . . . . .	1349
	Using XML . . . . .	1350
	Exploring XML . . . . .	1350
	Well-Formed XML . . . . .	1351
	Working with BDC Model Definition Files . . . . .	1352
	Generating Entities . . . . .	1355
	Connecting Data Services in Access . . . . .	1366

Appendix:	<b>Installing Your Software</b> .....	<b>1375</b>
	Installing the Office System .....	1376
	Choosing Options When You Have No Previous Version of the Office System...	1377
	Choosing Options to Upgrade a Previous Version of the Office System .....	1381
	Converting from a Previous Version of Access .....	1383
	Conversion Issues .....	1384
	Installing the Office 64-Bit Version .....	1385
	Installing the Sample Files .....	1387
	<b>Index</b> .....	<b>1389</b>



### *Bonus Content on the Companion CD*

## **Part 8: Automating an Access Application Using Visual Basic**

Chapter 24:	<b>Understanding Visual Basic Fundamentals</b> .....	<b>1451</b>
	The Visual Basic Development Environment .....	1452
	Modules .....	1452
	The Visual Basic Editor Window .....	1455
	Working with Visual Basic Debugging Tools .....	1463
	Variables and Constants .....	1474
	Data Types .....	1475
	Variable and Constant Scope .....	1477
	Declaring Constants and Variables .....	1479
	Const Statement .....	1479
	Dim Statement .....	1480
	Enum Statement .....	1483
	Event Statement .....	1485
	Private Statement .....	1486
	Public Statement .....	1488
	ReDim Statement .....	1489
	Static Statement .....	1491
	Type Statement .....	1492
	Collections, Objects, Properties, and Methods .....	1494
	The Access Application Architecture .....	1494
	The DAO Architecture .....	1497
	The ADO Architecture .....	1502
	Referencing Collections, Objects, and Properties .....	1505
	Assigning an Object Variable—Set Statement .....	1509
	Object Methods .....	1512
	Functions and Subroutines .....	1525
	Function Statement .....	1525
	Sub Statement .....	1527

Understanding Class Modules . . . . .	1529
Property Get . . . . .	1530
Property Let . . . . .	1532
Property Set . . . . .	1535
Controlling the Flow of Statements . . . . .	1538
Call Statement . . . . .	1538
Do...Loop Statement . . . . .	1539
For...Next Statement . . . . .	1540
For Each...Next Statement . . . . .	1541
GoTo Statement . . . . .	1543
If...Then...Else Statement . . . . .	1543
RaiseEvent Statement . . . . .	1545
Select Case Statement . . . . .	1545
Stop Statement . . . . .	1547
While...Wend Statement . . . . .	1547
With...End Statement . . . . .	1548
Running Macro Actions and Menu Commands . . . . .	1549
DoCmd Object . . . . .	1549
Executing an Access Command . . . . .	1550
Actions with Visual Basic Equivalents . . . . .	1551
Trapping Errors . . . . .	1551
On Error Statement . . . . .	1552
Some Complex Visual Basic Examples . . . . .	1553
A Procedure to Randomly Load Data . . . . .	1553
A Procedure to Examine All Error Codes . . . . .	1568
Working with 64-Bit Access Visual Basic for Applications . . . . .	1574
Using Declare Statements . . . . .	1575
Using LongPtr Data Types . . . . .	1576
Using PtrSafe Attributes . . . . .	1577
Supporting Older Versions of Access . . . . .	1577
Understanding Pointer Valued Functions and LongPtr Type Coercion . . . . .	1578
Using LongLong Data Types . . . . .	1579
Working with .MDE and .ACCDE files in 64-Bit Environments . . . . .	1580

## Chapter 25: Automating Your Application with Visual Basic . . . . . 1583

Why Aren't We Using Macros? . . . . .	1583
When to Use Macros . . . . .	1584
When to Use Visual Basic . . . . .	1584
Assisting Data Entry . . . . .	1585
Filling In Related Data . . . . .	1585
Handling the NotInList Event . . . . .	1590
Fixing an E-Mail Hyperlink . . . . .	1594
Providing a Graphical Calendar . . . . .	1595
Working with Linked Photos . . . . .	1602
Validating Complex Data . . . . .	1604
Checking for Possible Duplicate Names . . . . .	1605
Testing for Related Records When Deleting a Record . . . . .	1607

Verifying a Prerequisite .....	1608
Maintaining a Special Unique Value .....	1610
Checking for Overlapping Data .....	1611
Controlling Tabbing on a Multiple-Page Form .....	1613
Automating Data Selection .....	1615
Working with a Multiple-Selection List Box .....	1615
Providing a Custom Query By Form .....	1619
Selecting from a Summary List .....	1627
Filtering One List with Another .....	1628
Linking to Related Data in Another Form or Report .....	1631
Linking Forms Using a Filter .....	1631
Linking to a Report Using a Filter .....	1632
Synchronizing Two Forms Using a Class Event .....	1635
Automating Complex Tasks .....	1639
Triggering a Data Task from a Related Form .....	1639
Linking to a Related Task .....	1643
Calculating a Stored Value .....	1648
Automating Reports .....	1649
Allowing for Used Mailing Labels .....	1649
Drawing on a Report .....	1652
Dynamically Filtering a Report When It Opens .....	1655
Calling Named Data Macros .....	1658

## **Part 9: After Completing Your Application**

Chapter 26: <b>The Finishing Touches .....</b>	<b>1665</b>
Creating Custom Ribbons with XML .....	1665
Creating a USysRibbons Table .....	1667
Creating a Test Form .....	1670
Building the Ribbon XML .....	1671
Loading Ribbon XML .....	1679
Syntax .....	1680
Notes .....	1680
Using Ribbon Attributes .....	1682
Creating VBA Callbacks .....	1691
Dynamically Updating Ribbon Elements .....	1692
Loading Images into Custom Controls .....	1695
Hiding Options on the Backstage View .....	1696
Adding Options to the Backstage View .....	1697
Creating a Custom Quick Access Toolbar .....	1703
Setting Focus to a Tab .....	1704
Disabling Layout View .....	1705
Controlling How Your Application Starts and Runs .....	1706
Setting Startup Properties for Your Database .....	1706
Starting and Stopping Your Application .....	1708
Creating an AutoKeys Macro .....	1712
Performing a Final Visual Basic Compile .....	1713

Chapter 27:	<b>Distributing Your Application</b>	<b>1715</b>
	Using Linked Tables in a Desktop Database	1716
	Taking Advantage of the Database Splitter Wizard	1717
	Creating Startup Code to Verify and Correct Linked Table Connections	1719
	Understanding Runtime Mode	1724
	Creating a Database Template	1727
	Creating Custom Data Type Parts	1731
	Creating an Execute-Only Database	1732
	Creating an Application Shortcut	1733
	Encrypting Your Database	1737
	Packaging and Signing Your Database	1739
Article 1:	<b>Designing Your Database Application</b>	<b>1743</b>
	Application Design Fundamentals	1743
	Step 1: Identifying Tasks	1744
	Step 2: Charting Task Flow	1744
	Step 3: Identifying Data Elements	1745
	Step 4: Organizing the Data	1745
	Step 5: Designing a Prototype and a User Interface	1745
	Step 6: Constructing the Application	1746
	Step 7: Testing, Reviewing, and Refining	1746
	An Application Design Strategy	1747
	Analyzing the Tasks	1749
	Selecting the Data	1751
	Organizing Tasks	1753
	Data Analysis	1754
	Choosing the Database Subjects	1754
	Mapping Subjects to Your Database	1756
	Database Design Concepts	1757
	Waste Is the Problem	1757
	Normalization Is the Solution	1760
	Efficient Relationships Are the Result	1768
	When to Break the Rules	1770
	Improving Performance of Critical Tasks	1770
	Capturing Point-in-Time Data	1771
	Creating Report Snapshot Data	1772
Article 2:	<b>Understanding SQL</b>	<b>1773</b>
	SQL SELECT Queries	1774
	Aggregate Functions: AVG, CHECKSUM_AGG, COUNT, MAX, MIN, STDEV, STDEVP, SUM, VAR, and VARP	1775
	BETWEEN Predicate	1775
	Column-Name	1776
	Comparison Predicate	1777
	EXISTS Predicate	1778
	Expression	1779
	FROM Clause	1782

	Syntax .....	1782
	GROUP BY Clause .....	1785
	HAVING Clause .....	1786
	IN Clause .....	1787
	IN Predicate .....	1788
	LIKE Predicate .....	1790
	NULL Predicate .....	1791
	ORDER BY Clause .....	1792
	PARAMETERS Declaration .....	1794
	Quantified Predicate .....	1796
	Search Condition .....	1797
	SELECT Statement .....	1799
	Subquery .....	1806
	TRANSFORM Statement .....	1810
	UNION Query Operator .....	1811
	WHERE Clause .....	1813
	SQL Action Queries .....	1815
	DELETE Statement .....	1815
	INSERT Statement (Append Query) .....	1816
	SELECT ... INTO Statement (Make-Table Query) .....	1819
	UPDATE Statement .....	1820
Article 3:	<b>Exporting Data .....</b>	<b>1823</b>
	Exporting to Another Access Database .....	1823
	Exporting to a Spreadsheet or to a dBASE File .....	1824
	Exporting to a Text File .....	1825
	Exporting to a Mail Merge Document in Word .....	1826
	Exporting to an ODBC Database .....	1827
	Exporting Data to SharePoint .....	1828
Article 4:	<b>Function Reference .....</b>	<b>1831</b>
Article 5:	<b>Color Names and Codes .....</b>	<b>1841</b>
Article 6:	<b>Macro Actions. ....</b>	<b>1849</b>

---

### What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

[microsoft.com/learning/booksurvey](https://microsoft.com/learning/booksurvey)





# Acknowledgments

Nearly every member of the Microsoft Access development team provided invaluable technical support as I worked through the finer details in Microsoft Access 2010. The program managers, developers, and test engineers on the team helped with suggestions, tips and tricks, and reviewing my material. Special thanks to program manager Ric Lewis, who helped write the Business Connectivity Services chapter. You folks make an author's job so much easier. But any errors or omissions in this book are ultimately mine.

A book this large and complex requires a top-notch team to get what I put into Microsoft Word documents onto the printed pages you are now holding. I had some of the best in the business at both Microsoft Press and O'Reilly Media to get the job done. Many thanks to Kenyon Brown at O'Reilly Media for serving as Acquisitions and Development Editor and to Juliana Aldous as Acquisitions Editor for Microsoft Press. Special thanks also to Linda Allen and Rachel Monaghan for handling copy and production editing and to Andrew Couch and Jim Bailie for technical reviewing. I couldn't have done it without you!

And last, but certainly not least, I thank my wife and soulmate, Cheryl. She not only patiently stood by me as I cranked through 1,800 pages of manuscript, but also helped behind the scenes reviewing and editing what I did.

—Jeff Conrad  
Redmond, Washington  
July 2010



# About the CD

The companion CD that ships with this book contains many resources to help you get the most out of your Inside Out book.

## What's on the CD

Your Inside Out CD includes the following:

- **Sample client and web database applications.** Includes query, form, and report examples.
- **Four bonus chapters.** Here you'll find coverage of Visual Basic fundamentals, adding the finishing touches, and distributing your application.
- **Six technical articles.** This section includes an overview of SQL, exporting data, a function reference, color names and codes, and macro actions.
- **Complete eBook.** In this section, you'll find the entire electronic version of this title.

## Sample Applications

Throughout this book, you'll see examples from four sample Access applications included on the sample CD:

- *Back Office Software System Restaurant Management Application (BOSS.accdb).* This application is a hybrid Access application designed for use in both client and web. It demonstrates how a restaurant might manage food orders, maintain employee records, and create weekly work schedules. This application can be published to a server running SharePoint 2010 and Access Services and then used within a web browser. You'll also find BOSSDataCopy.accdb and BOSSDataCopy2.accdb files that contain many of the query, form, and report examples.
- *Conrad Systems Contacts (Contacts.accdb and ContactsData.accdb).* This application is both a contacts management and order entry database—two samples for the price of one! This sample database demonstrates how to build a client/server application using only desktop tools as well as how to “upscale” an application to create an Access project and related SQL Server tables, views, stored procedures, and functions. You'll also find a ContactsDataCopy.accdb file that contains additional query, form, and report examples.

- *Housing Reservations (Housing.accdb)*. This application demonstrates how a company housing department might track and manage reservations in company-owned housing facilities for out-of-town employees and guests. You'll also find *HousingDataCopy.accdb* and *HousingDataCopy2.accdb* files that contain many of the query, form, and report examples.
- *Wedding List (WeddingMC.accdb and WeddingList.accdb)*. This application is an example of a simple database that you might build for your personal use. It has a single main table where you can track the names and addresses of invitees, whether they've said that they will attend, the description of any gift they sent, and whether a thank-you note has been sent. Although you might be tempted to store such a simple list in an Excel spreadsheet or a Word document, this application demonstrates how storing the information in Access makes it easy to search and sort the data and produce reports. The *WeddingMC* database is automated entirely using macros, and the *WeddingList* database is the same application automated with Visual Basic.

You can find these databases on the companion CD provided with this book. Please note that the person names, company names, email addresses, and web addresses in these databases are fictitious. Although we pre-loaded all databases with sample data, the *Housing Reservations* and *Conrad Systems Contacts* databases also include a special form (*zfrmLoadData*) that has code to load random data into the sample tables based on parameters that you supply.

The examples in this book assume you have installed the 32-bit version of Microsoft Office 2010, not just the 32-bit version of Access 2010. The sample databases included with the companion CD have not been modified to work with the 64-bit version of Access 2010. Several examples also assume that you have installed all optional features of Access through the Office 2010 setup program. If you have not installed these additional features, your screen might not match the illustrations in this book or you might not be able to run the samples from the companion CD. A list of the additional features you will need to run all the samples in this book is included in the Appendix.

## System Requirements

Following are the minimum system requirements necessary to run the CD:

- A Pentium 500 megahertz (MHz) or faster processor (Pentium III is recommended as a minimum), and 1 gigahertz (GHz) is required for Microsoft Outlook with Business Contact Manager.
- Microsoft Windows XP with Service Pack (SP) 3 (32-bit), Windows Vista with SP1 (32-bit or 64-bit), Windows Server 2003 R2 (32-bit or 64-bit) with MSXML 6.0 installed,

Windows Server 2008 (32-bit or 64-bit), or Windows 7 (32-bit or 64-bit). Terminal Server and Windows on Windows (WOW), which allows installing 32-bit versions of Office 2010 on 64-bit operating systems, are supported.

- At least 256 megabytes (MB) of random access memory (RAM); 512 MB is recommended.
- A hard drive with at least 527 MB of free space for a minimum installation when your network administrator has set up an install package for you on a server. When you perform a local installation, you need up to 3.5 gigabytes (GB) on your primary hard drive for the installation files and programs. At the end of the Office system install, you have the option to leave some of the installation files on your hard drive, which requires up to an additional 240 MB of free space.
- A CD-ROM or DVD-ROM drive. (A DVD-ROM is recommended.) If you are installing over a network, no disc drive is required.
- A mouse or other pointing device.
- A 1024 × 768 or greater monitor display.

Other options required to use all features include the following:

- A multimedia computer for sound and other multimedia effects.
- Dial-up or broadband Internet access.
- Certain inking features require running Windows XP Table PC edition or later. Speech recognition functionality requires a close-talk microphone and an audio output device. Information Rights Management features require access to a Windows Server 2003 with SP1 or later running Windows Rights Management Services.
- Connectivity to Microsoft Exchange 2000 Server or later is required for certain advanced functionality in Outlook 2010. Instant Search requires Windows Desktop Search 3.0. Dynamic Calendars require server connectivity.
- Microsoft Internet Explorer 6 or later, 32-bit browser only. Internet functionality requires Internet access (fees might apply).
- Connection to an Internet service provider or a local copy of Microsoft Internet Information Services (IIS) installed.
- 512 MB of RAM or higher recommended for Outlook Instant Search. Grammar and contextual spelling in Microsoft Word 2010 is not turned on unless the computer has 1 GB memory.

### Note

An internet connection is necessary to access the hyperlinks on the companion CD. Connect time charges may apply.

## Support Information

Every effort has been made to ensure the accuracy of the contents of the book and of this CD. As corrections or changes are collected, they will be added to a Microsoft Knowledge Base article. Microsoft Press provides support for books and companion CDs at the following website: [www.microsoft.com/learning/support/books/](http://www.microsoft.com/learning/support/books/).

If you have comments, questions, or ideas regarding the book or this CD, or questions that are not answered by visiting the site above, please send them via e-mail to [mspinput@microsoft.com](mailto:mspinput@microsoft.com).

You can also click the Feedback or CD Support links on the Welcome page. Please note that Microsoft software product support is not offered through the above addresses.

If your question is about the software, and not about the content of this book, please visit the Microsoft Help and Support page or the Microsoft Knowledge Base at <http://support.microsoft.com>.

# Conventions and Features Used in This Book

This book uses special text and design conventions to make it easier for you to find the information you need.

## Text Conventions

Convention	Meaning
Abbreviated menu commands	For your convenience, this book uses abbreviated menu commands. For example, “Click Tools, Track Changes, Highlight Changes” means that you should click the Tools menu, point to Track Changes, and click the Highlight Changes command.
<b>Boldface type</b>	<b>Boldface</b> type is used to indicate text that you enter or type.
Initial Capital Letters	The first letters of the names of menus, dialog boxes, dialog box elements, and commands are capitalized. Example: the Save As dialog box.
<i>Italicized type</i>	<i>Italicized</i> type is used to indicate new terms.
Plus sign (+) in text	Keyboard shortcuts are indicated by a plus sign (+) separating two key names. For example, Ctrl+Alt+Delete means that you press the Ctrl, Alt, and Delete keys at the same time.

## Design Conventions

### INSIDE OUT

This Statement Illustrates an Example of an “Inside Out” Heading

These are the book’s signature tips. In these tips, you’ll get the straight scoop on what’s going on with the software—inside information about why a feature works the way it does. You’ll also find handy workarounds to deal with software problems.

### Sidebars

Sidebars provide helpful hints, timesaving tricks, or alternative procedures related to the task being discussed.



## TROUBLESHOOTING

This statement illustrates an example of a “Troubleshooting” problem statement.

Look for these sidebars to find solutions to common problems you might encounter. Troubleshooting sidebars appear next to related information in the chapters. You can also use the Troubleshooting Topics index at the back of the book to look up problems by topic.

Cross-references point you to other locations in the book that offer additional information about the topic being discussed.

### Note

Notes offer additional information related to the task being discussed.

### CAUTION !

Cautions identify potential problems that you should look out for when you’re completing a task or problems that you must address before you can complete a task.



When an example has a related file that is included on the companion CD, this icon appears in the margin. You can use these files to follow along with the book’s examples.

## Syntax Conventions

The following conventions are used in the syntax descriptions for Visual Basic statements in Chapter 24, “Understanding Visual Basic Fundamentals,” Chapter 25, “Automating Your Application with Visual Basic,” SQL statements in Article 2, “Understanding SQL,” and any other chapter where you find syntax defined. These conventions do not apply to code examples listed within the text; all code examples appear exactly as you’ll find them in the sample databases.

You must enter all other symbols, such as parentheses and colons, exactly as they appear in the syntax line. Much of the syntax shown in the Visual Basic chapter has been broken into multiple lines. You can format your code all on one line, or you can write a single line of code on multiple lines using the Visual Basic line continuation character (`_`).

Convention	Meaning
<b>Bold</b>	Bold type indicates keywords and reserved words that you must enter exactly as shown. Microsoft Visual Basic understands keywords entered in uppercase, lowercase, and mixed case type. Access stores SQL keywords in queries in all uppercase, but you can enter the keywords in any case.
<i>Italic</i>	Italicized words represent variables that you supply.
Angle brackets < >	Angle brackets enclose syntactic elements that you must supply. The words inside the angle brackets describe the element but do not show the actual syntax of the element. Do not enter the angle brackets.
Brackets [ ]	Brackets enclose optional items. If more than one item is listed, the items are separated by a pipe character ( ). Choose one or none of the elements. Do not enter the brackets or the pipe; they're not part of the element. Note that Visual Basic and SQL in many cases require that you enclose names in brackets. When brackets are required as part of the syntax of variables that you must supply in these examples, the brackets are italicized, as in <i>[MyTable].[MyField]</i> .
Braces { }	Braces enclose one or more options. If more than one option is listed, the items are separated by a pipe character ( ). Choose one item from the list. Do not enter the braces or the pipe.
Ellipsis ...	Ellipses indicate that you can repeat an item one or more times. When a comma is shown with an ellipsis (...), enter a comma between items.
Underscore _	You can use a blank space followed by an underscore to continue a line of Visual Basic code to the next line for readability. You cannot place an underscore in the middle of a string literal. You do not need an underscore for continued lines in SQL, but you cannot break a literal across lines.



# Introduction

Microsoft Access 2010 is just one part of Microsoft's overall data management product strategy. Like all good relational databases, it allows you to link related information easily—for example, customer and order data that you enter. But Access 2010 also complements other database products because it has several powerful connectivity features. As its name implies, Access can work directly with data from other sources, including many popular PC database programs (such as dBASE), with many SQL (Structured Query Language) databases on the desktop, on servers, on minicomputers, or on mainframes, and with data stored on Internet or intranet web servers. Access also fully supports Microsoft's ActiveX technology, so an Access application can be either a client or a server for all the other 2010 Microsoft Office system applications, including Word, Excel, PowerPoint, Outlook, Publisher, and OneNote.

Access provides a very sophisticated application development system for the Microsoft Windows operating system. This helps you build applications quickly, whatever the data source. In fact, you can build simple applications by defining forms and reports based on your data and linking them with a few macros or Microsoft Visual Basic statements; there's no need to write complex code in the classic programming sense. Because Access uses Visual Basic, you can use the same set of skills with other applications in the Microsoft Office system or with Visual Basic.

For small businesses (and for consultants creating applications for small businesses), the Access desktop development features are all that's required to store and manage the data used to run the business. Access coupled with Microsoft SQL Server—on the desktop or on a server—is an ideal way for many medium-size companies to build new applications for Windows quickly and inexpensively. To enhance workgroup productivity, you can use Access 2010 to create an Access Services web application running on a server with SharePoint 2010. Users of your application can view, edit, and delete data from your application directly in their web browser. For large corporations with a big investment in mainframe relational database applications as well as a proliferation of desktop applications that rely on personal computer databases, Access provides the tools to easily link mainframe and personal computer data in a single Windows-based application. Access 2010 includes features to allow you to export or import data in XML format (the lingua franca of data stored on the web).

## Getting Familiar with Access 2010

If you have never used a database program—including Access—you'll find Access 2010 very approachable. Using the results of extensive productivity lab tests, Microsoft has revamped the user interface in all the Microsoft Office programs. The Backstage view and ribbon technology makes it much easier for novice users to get acquainted with Access and easily discover its most useful features. To get a new user jump-started, Microsoft has provided over a dozen local client and web database templates that load onto your hard disk when you install Access. In addition, you'll find many additional database templates available for easy download from the Microsoft Office website directly from within Access. Microsoft plans to continue to add templates after Access 2010 is released to further enhance your productivity.

But if you have used any version of Access prior to 2007, you're in for a big surprise. Menus and toolbars are gone—all replaced by the Backstage view and ribbon. The Database window has been replaced by the Navigation pane. When you first start using Access 2010, you'll probably notice a decrease in productivity, but it won't take you long to get comfortable with the new interface. You'll probably soon discover features that you didn't know were there. Nearly all the old familiar objects are around—tables, queries, forms, reports, macros, and modules, and you'll find that the standard design and data views you've come to know and love are still around. You'll also quickly learn that the Layout and Report views and macro Logic Designer rapidly increase your productivity.

## About This Book

If you're developing a database application with the tools in Access 2010, this book gives you a thorough understanding of "programming without pain." It provides a solid foundation for designing databases, forms, and reports and getting them all to work together. You'll learn that you can quickly create complex applications by linking design elements with macros or Visual Basic. This book will also show you how to take advantage of some of the more advanced features of Access 2010. You'll learn how to build an Access web database that you can then publish to a server running SharePoint 2010 and Access Services. You'll also learn about new data macros that allow you to attach business logic to your tables and how to work with the revamped Logic Designer to design macros.

If you're new to developing applications, particularly database applications, this probably should not be the first book you read about Access. We recommend that you first take a look at *Microsoft Access 2010 Plain & Simple* or *Microsoft Access 2010 Step By Step*.

*Microsoft Access 2010 Inside Out* is divided into nine major parts:

- Part 1 provides an overview of Access 2010 and provides you with a detailed look at the user interface, including the new Backstage view.
  - Chapter 1 explains the major features that a database should provide, explores those features in Access, and discusses some of the main reasons why you should consider using database software.
  - Chapter 2 thoroughly explores the user interface introduced in the Office 2010 release. The chapter also explains content security, working with the Backstage view, ribbon, and the Navigation pane, and setting options that customize how you work with Access 2010.
  - Chapter 3 describes the architecture of Access 2010, gives you an overview of the major objects in an Access database by taking you on a tour through two of the sample databases, and explains the many ways you can use Access to create an application.
- Part 2 shows you how to create your desktop or web database and tables.
  - Chapter 4 teaches you how to design client databases and tables.
  - Chapter 5 shows you the ins and outs of modifying tables even after you've already begun to load data and build other parts of your application.
  - Chapter 6 focuses on designing tables for use in a web database.
  - Chapter 7 discusses the new feature of table data macros and how to work with the new Logic Designer to create your macro logic.
  - Chapter 8 explains how to link to or import data from other sources.
- Part 3 focuses on how to build queries to analyze and update data in your tables.
  - Chapter 9 shows you how to build simple queries and how to work with data in Datasheet view.
  - Chapter 10 discusses how to design client and web queries to work with data from multiple tables, summarize information, build queries that require you to work in SQL view, and work with the PivotTable and PivotChart views of queries.
  - Chapter 11 focuses on modifying sets of data with queries—updating data, inserting new data, deleting sets of data, or creating a new table from a selection of data from existing tables.

- Part 4 discusses how to build and work with forms in client and web applications.
  - Chapter 12 introduces you to forms—what they look like and how they work.
  - Chapters 13, 14, and 15 teach you all about form design in client and web applications, from simple forms you build with a wizard to complex, advanced forms that use embedded forms and navigation and web browser controls. You'll also learn how to use Layout view to design web forms that you can open in a web browser using Access Services.
- Part 5 explains how to work with reports in client and web applications.
  - Chapter 16 leads you on a guided tour of reports and explains their major features.
  - Chapters 17 and 18 teach you how to design, build, and implement both simple and complex reports in your client or web application.
- Part 6 shows you how to make your client and web applications “come alive” using macros.
  - Chapter 19 discusses the concept of event processing in Access, provides a comprehensive list of events, and explains the sequence in which critical events occur.
  - Chapter 20 covers macro design for client applications in depth and explains how to use error trapping and embedded macro features.
  - Chapter 21 focuses on how to create web macros, work with web forms and control events, and perform actions in a web browser.
- Part 7 is all about using Access tools and working with the web.
  - Chapter 22 teaches you how to publish your web database to an Access Services site and use your application in a browser. It discusses making changes to a web application, synchronizing your changes with the server, working in offline mode, and instantiating a web template.
  - Chapter 23 covers features in Access that handle XML and how to use Business Connectivity Services.
- The Appendix explains how to install the Office 2010 release, including which options you should choose for Access 2010 to be able to open all the samples in this book.



- Part 8, on the companion CD, shows you how to use the programming facilities in Microsoft Visual Basic to integrate your database objects and automate your application.

- Chapter 24 is a comprehensive reference to the Visual Basic language and object models implemented in Access. It presents two complex coding examples with a line-by-line discussion of the code. The final section shows you how to work with the new 64-bit Access Visual Basic.
- Chapter 25 thoroughly discusses some of the most common tasks that you might want to automate with Visual Basic. Each section describes a problem, shows you specific form or report design techniques you must use to solve the problem, walks you through the code from one or more of the sample databases that implements the solution, and discusses calling named data macros.



- Part 9, on the companion CD, covers tasks you might want to perform after completing your application.

- Chapter 26 teaches you how to automate custom ribbons, create a custom Backstage view, and how to set Startup properties.
- Chapter 27 teaches you tasks for setting up your application so that you can distribute it to others. It also shows you how to create your own custom Data Type Parts, Application Parts, and application templates.



- The CD provides an additional six Articles that contain important reference information:

- Article 1 explains a simple technique that you can use to design a good relational database application with little effort. Even if you're already familiar with Access or creating database applications in general, getting the table design right is so important that this article is a "must read" for everyone.
- Article 2 is a complete reference to SQL as implemented in desktop databases. It also contains notes about differences between SQL supported natively by Access and SQL implemented in SQL Server.
- Article 3 discusses how to export data and Access objects to various types of other data formats from your Access application.
- Article 4 lists the functions most commonly used in an Access application categorized by function type. You'll also find a list of functions that you can use with web databases.
- Article 5 lists the color names and codes you can use in Access.
- Article 6 lists the macro actions for both client and web applications you can use in Access.



# Designing Web Tables

Working with the Web .....	288	Defining a Table Validation Rule for Web Databases ..	332
Creating a New Web Database.....	290	Defining a Primary Key for Web Databases.....	335
Creating Your First Simple Web Table by Entering Data..	297	Understanding Other Web Table Properties .....	336
Creating a Web Table Using Application Parts .....	300	Creating Lookup Fields in a Web Database .....	337
Using Data Type Parts .....	304	Creating Relationships Using Lookup Fields .....	341
Creating Web Tables in Datasheet View.....	306	Using the Web Compatibility Checker .....	351

The process of defining web tables in a Microsoft Access 2010 web database (.accdb file) is similar to designing client tables in a desktop database. In general, you create the fields you need, assign properties to those fields, and create relationships between the tables. The differences in the process of table creation for web databases lie in the design surface options you can use and the available data types and properties for web fields. In this chapter, we'll show you how to begin creating your first web application by starting with the tables. You'll learn how to:

- Create a new web database application using a web database template
- Create a new empty web database for your own custom application
- Create a simple web table by entering data directly in the table
- Get a jump start on defining custom web tables by using Application Parts
- Create new web fields by using Data Type Parts
- Define your own web tables and web fields from scratch by using Datasheet view
- Create Calculated fields in your web tables
- Set validation rules for your web fields and web tables
- Create lookups to other web fields to define relationships between your web tables
- Run the Compatibility Checker to verify your fields and tables are web-legal
- Prepare a client database for the web

**Note**

All the screen images in this chapter were taken on a Windows 7 system with the Access color scheme set to Silver.

## INSIDE OUT

### Take Time to Learn About Table Design

At the start of Chapter 4, “Designing Client Tables,” we mentioned the importance of planning up front the tasks you want to perform, the data structures you need to support those tasks, and the flow of tasks within your client database application. The importance of this planning stage is the same when designing a web application. Proper planning at the start of the web application building process can save you from constantly redesigning your application over and over. If you haven’t already, we encourage you to learn at least the fundamentals of table and application design by reading Article 1, “Designing Your Database Application,” that you can find on the companion CD.

## Working with the Web

The topic of web databases in Access 2010 is a very broad subject—certainly not a topic we can fully cover in just one chapter. In fact, a large portion of the Access development team at Microsoft worked solely on all the various features of web databases during the Access 2010 development cycle. The process of developing a web application is very much the same as developing a client application—you identify the tasks you want to accomplish with the application, chart the flow of tasks, identify the data elements, organize the data elements, design a user interface for the application, construct the application, and then test and refine the application. We’ve organized this book to closely follow the application building process for both client and web databases. We start by building the fields and tables of an application and then continue with building queries, forms, reports, macros, and modules throughout the various chapters.

This chapter begins the discussion of developing a web database by starting with creating fields and tables—the foundation of your web application. We’ll continue the discussion of creating a fully functional web database in subsequent chapters. You’ll learn how to create data macros and attach them to table events, create queries to pull out the data you need, create forms and reports that run in a web browser, and how to create macros to automate your web application. In Chapter 22, “Using Web Applications in a Browser,”

we'll pull everything together and show you how to publish your application to a Microsoft SharePoint 2010 server running Access Services, work with your objects in a web browser, and how to make modifications to your web application.

Before we start the discussion of building tables for web databases, we should first discuss some terminology you'll be hearing throughout this chapter and subsequent chapters to follow. A *web database* (or web application) is a database that has web tables—tables that will successfully publish to a SharePoint server running Access Services. A web database must be in the .accdb file format and created in Access 2010 or later. A *client database* is a standard Access database with client tables that you've been working with for many Access releases. A client database can be in the .mdb or .accdb file format and can be created in Access 2010 or earlier versions.

A web database can have client objects, such as queries, forms, reports, macros, and modules, as well as web objects. Tables are the one exception to this rule. In a web database, you can only have web tables. The tables in a web database, also called the *schema* of the database, must be compatible with SharePoint server lists. You are allowed to have links to other data sources in a web database because they are not local tables. However, linked tables will not work in your web browser—you will not be able to use the data from those external data sources in your queries, forms, and reports that run in the browser. When a web database contains client objects as well as web objects, Microsoft uses the term *hybrid application* to describe this type of database.

It's important to note that you don't need a SharePoint server to use a web database in Access. You can use a web database in Access 2010 and never publish the database to a SharePoint server. The web database will function just fine in Access, and you can continue to modify the application as your needs grow. However, if you want to take advantage of all the extra features that publishing your web database to a server can offer, you'll need to set up a SharePoint server running Access Services within your business. If you are in a corporate domain, your IT department might already have a SharePoint server installed and running Access Services. You should check with your network administrator to see if this is the case. If you do not want to take the time and expense to setup and install a SharePoint server within your business, you can also use a third party that offers SharePoint hosting services. There are many third party companies, including Microsoft, that can host your Access Services applications.

When you publish your web database to a SharePoint server, you can use the application from within Access 2010 and you can use the application within a web browser. When you use or design the database from within Access, Microsoft often uses the term *rich client* in these discussions. Web databases, as you just learned, can contain web objects as well as client objects. The client objects do not run in the browser, but they will run in Access 2010. When you see the term *rich client*, we are referring to designing or using the web database from within Access 2010 where you have all the design facilities available to you, including

the ability to work with client objects. Access Services on SharePoint does not have any design facilities to modify your web database and the server cannot run or execute something it doesn't understand. Rich client refers to the fact that Access 2010 has all the rich design and functional capabilities that the server might not have available.

If you're already familiar with creating client databases in previous versions of Access, you're already well on your way to understanding how to create web databases. In general, the server has less functionality than the rich client so when you are designing web objects, Access 2010 presents design surfaces that only show options, properties, controls, and other design mechanisms that will seamlessly move to the server. We use the term *web-legal* to describe those objects, controls, properties, etc. that can move successfully to the server.

## Creating a New Web Database

When you first start Access 2010, you see the New tab on the Microsoft Office Backstage view, as shown in Figure 6-1. We explored the New tab in detail in Chapter 2, "Exploring the Access 2010 Interface," and in Chapter 4, "Designing Client Tables."

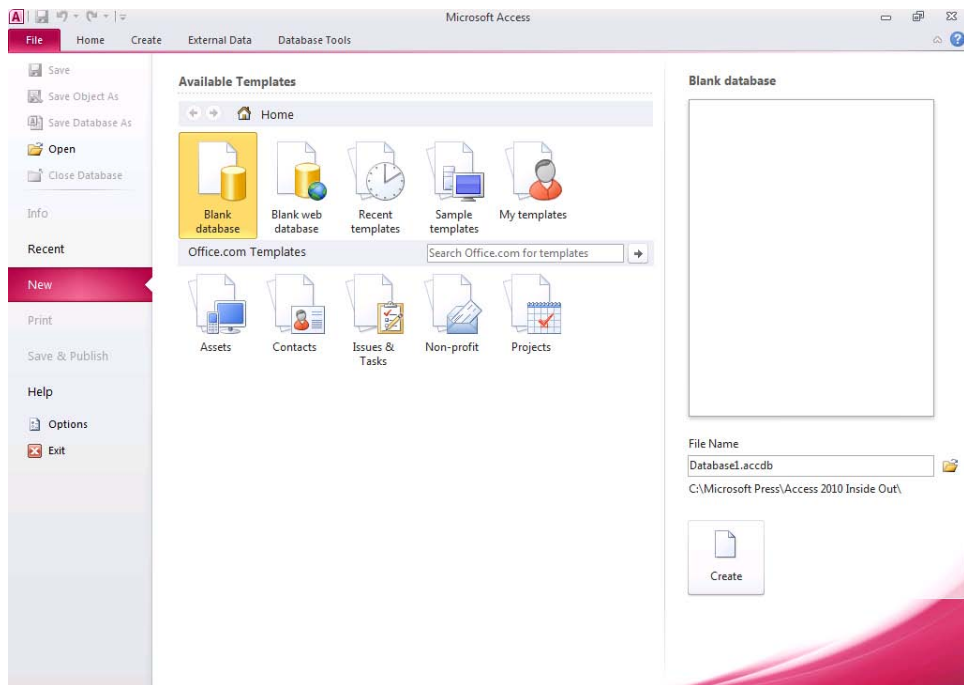
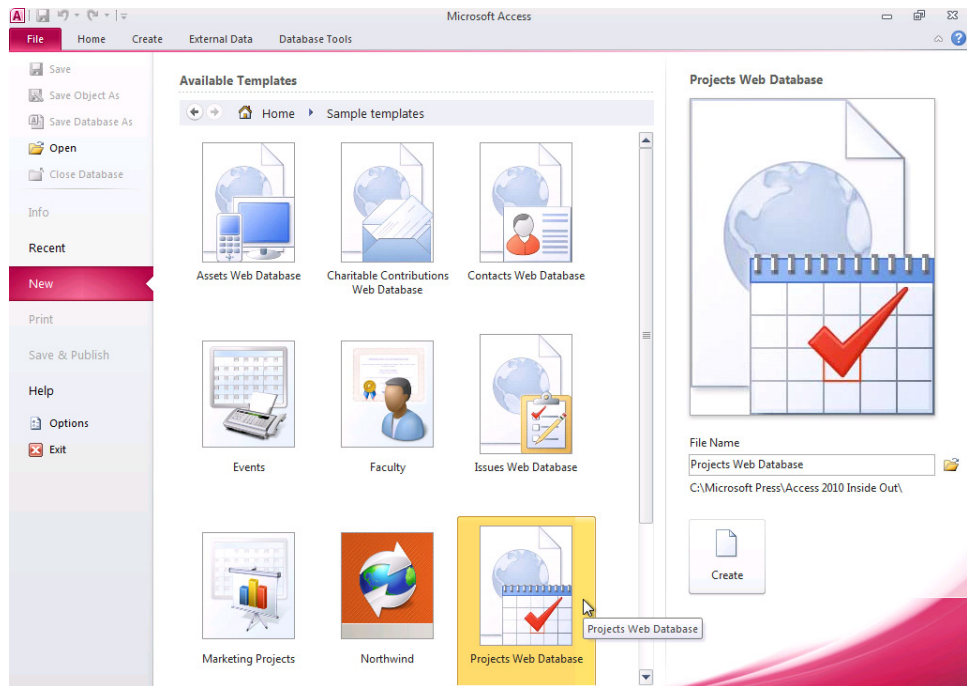


Figure 6-1 You can see the New tab of the Backstage view when you first launch Access 2010.

## Using a Database Template to Create a Web Database

Web databases are a new template feature for Access 2010, so even if you're an experienced developer, you might find that studying the built-in web application templates will help you understand how to design and work with web databases. You might find that one of these applications meets most of your needs without any modifications. You can also build on and customize the basic web application design and add new features as your application needs grow.

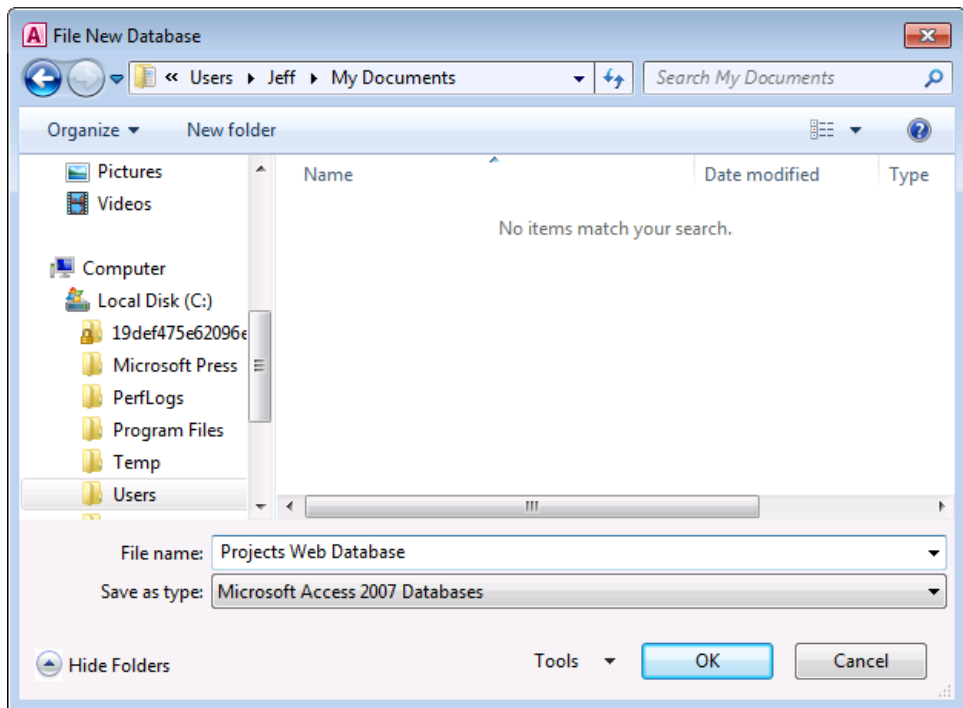
On the New tab of the Backstage view, you can access the built-in locally installed web templates by clicking Sample Templates under Available Templates in the middle of the screen. When you click Sample Templates, the center section of the New tab changes to show a graphic representing each of the local database templates available in Access 2010, as shown in Figure 6-2.



**Figure 6-2** You can choose to work with any of the five web templates from the Sample Templates section.

Access 2010 includes over 10 locally installed database templates, five of which are web templates—Assets, Charitable Contributions, Contacts, Issues, and Projects. You can differentiate which templates under Sample Templates are web templates by looking for the world icon in the template graphic and by the word Web included in the file name. When you click one of the web template graphics in the center of the New tab, Access displays the file name and a larger graphic in the right task pane. Click the Projects Web Database template in the middle of the screen to see the file name and graphic for the Projects Web Database Template, as shown in Figure 6-2. You can work with all five local web templates from the New tab in the same way. This example will show you the steps that are needed to build a Projects web database.

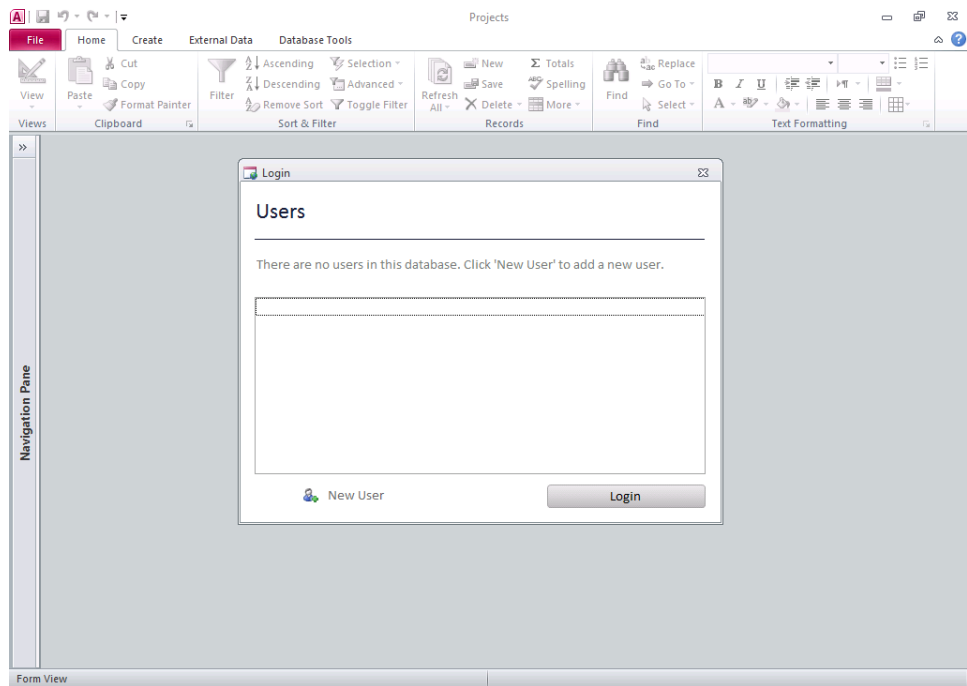
Access suggests a name for your new web database in the File Name text box and a location to save the file beneath the File Name text box. You can modify the name of this web database by typing in the File Name text box. If you want to change the suggested save location, click Browse to open the File New Database dialog box, as shown in Figure 6-3.



**Figure 6-3** Use the File New Database dialog box to select a folder for saving the new local web database template.

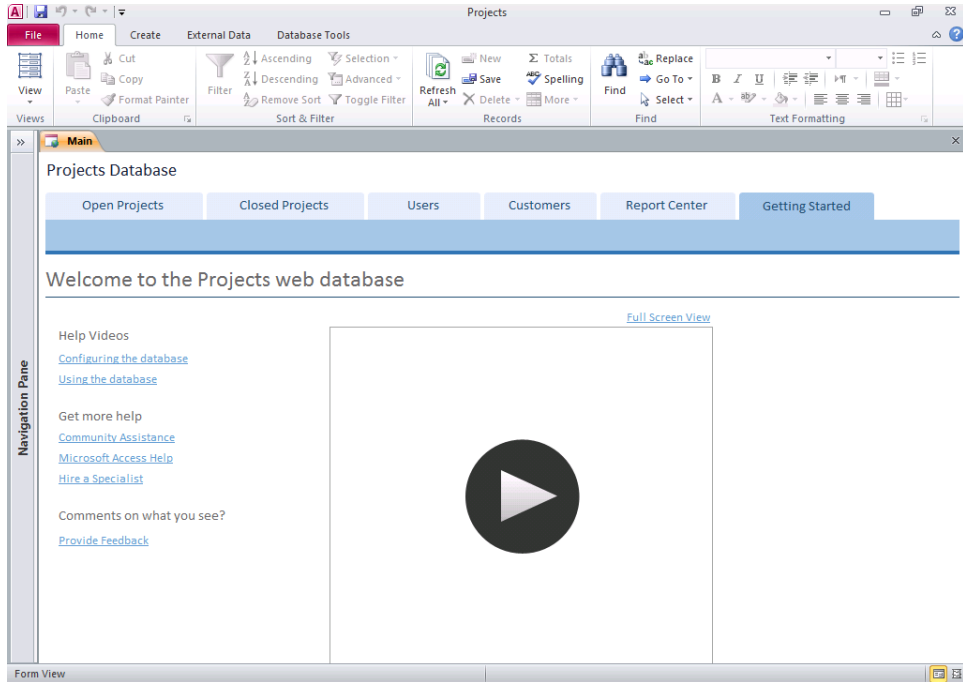
You can select the drive and folder you want by clicking the links on the left and browsing to your destination folder. After you select the specific folder to which you want to save this new web database, click OK to return to the New tab on the Backstage view. Your new folder location is shown beneath the File Name text box. If you decide at this point not to create the web database, click the Home button near the top of the screen to return to the main Home page of the New tab to stop the process. Click Create to start the template instantiation process.

A progress bar appears on the screen informing you to please wait while Access creates the web template. After a few seconds of preparation, Access opens the new Projects web database and displays the Login form, as shown in Figure 6-4.



**Figure 6-4** After you create the Projects web database from a template, Access opens the database and displays the Login form.

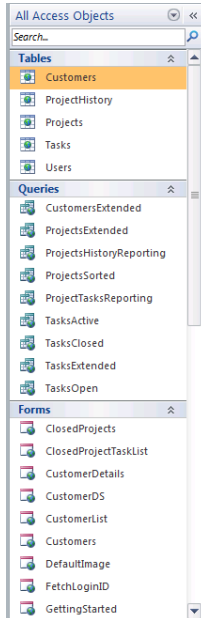
Dismiss this Login form for now by clicking the X button in the upper-right corner of the form window. After you close this form, Access displays the Main navigational form for the Projects web database, as shown in Figure 6-5.



**Figure 6-5** When you close the Login form, Access displays the Main navigational form in the Projects web database.

This Projects web database is a complete application that can track the progress of your various projects. The database comes with tables, queries, forms, reports, and macros. We'll discuss all the form elements you are currently looking at beginning in Chapter 12, "Using Forms in an Access Application." For now, close this form by clicking the X button in the upper-right corner of the form window or by right-clicking the Main tab and clicking Close Form on the shortcut menu. Expand the Navigation pane now to see all the various objects contained in this database, as shown in Figure 6-6.



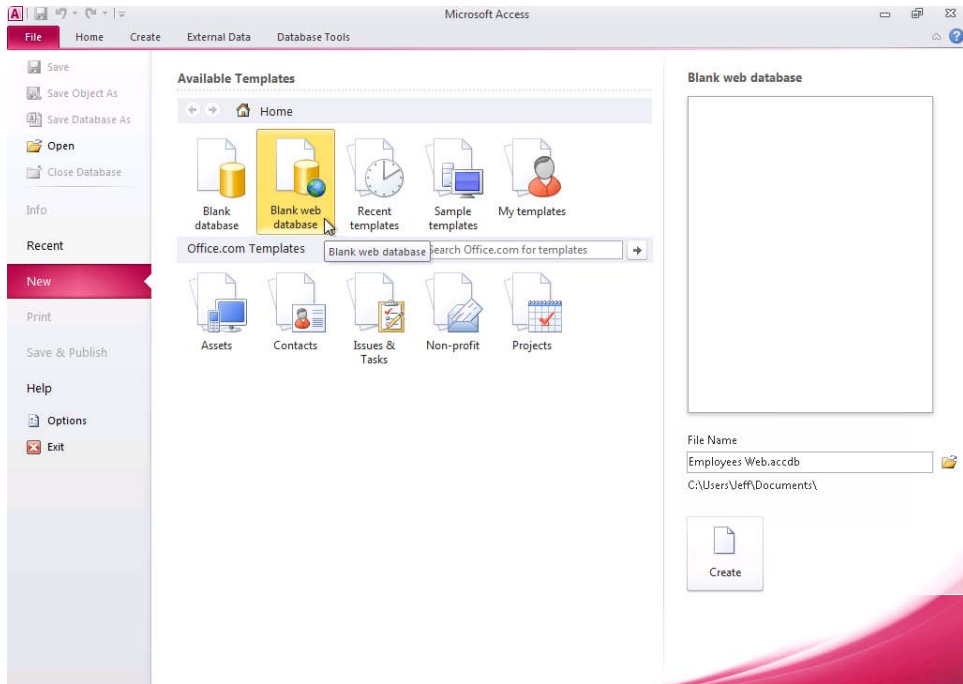


**Figure 6-6** The Projects web database contains many web objects to track the status of your various projects.

In a web database, like the Projects one here, any web objects—tables, queries, forms, reports, and macros—have a globe on the object icon to indicate they are web objects. In this Projects web database, as well as the other four built-in local web templates, all objects are web objects which means they will publish and render in a web browser. You can also use this application in Access client if you do not want to publish to a SharePoint server. We'll discuss designing and working with all the various web queries, forms, reports, and macros in later chapters. For now, close this new web database by clicking the File tab on the Backstage view and then clicking Close Database to return to the New tab.

## Creating a New Empty Web Database

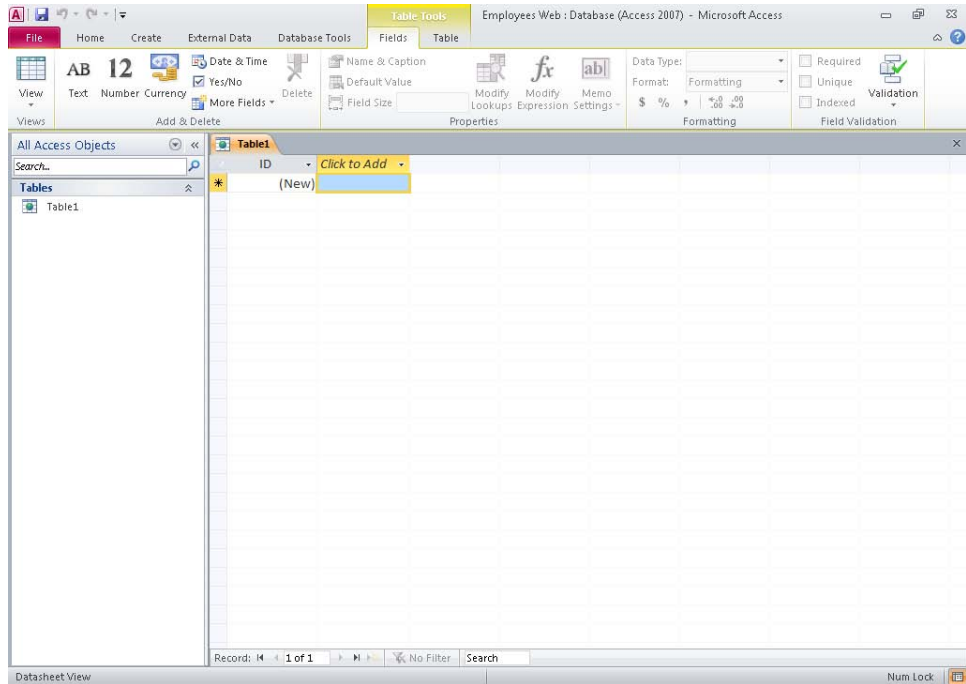
To begin creating a new empty web database when you start Access 2010, go to the Available Templates section in the middle of the New tab (as shown in Figure 6-1) and click Blank Web Database. The right side of the New tab changes to display the Blank Web Database task pane, as shown in Figure 6-7.



**Figure 6-7** From the New tab on the Backstage view, click Blank Web Database to open the Blank Web Database task pane on the right.

You can click Browse to open the File New Database dialog box, shown in Figure 6-3, to select the drive and folder you want. In this example, we selected the Documents folder in Windows 7 for the current user. Next, type the name of your new database in the File Name text box—Access 2010 appends an .accdb extension to the file name for you. For this example, let's create a database with a table containing employee names and addresses. Type **Employees Web** in the File Name box and click Create to create your web database.

Access 2010 takes a few moments to create the system tables in which to store all the information about the tables, queries, forms, reports, and macros that you might create. Access then displays the Navigation pane for your new web database and opens a new blank web table in Datasheet view, shown in Figure 6-8.



**Figure 6-8** Access 2010 opens a new web table in Datasheet view when you create a new blank web database.

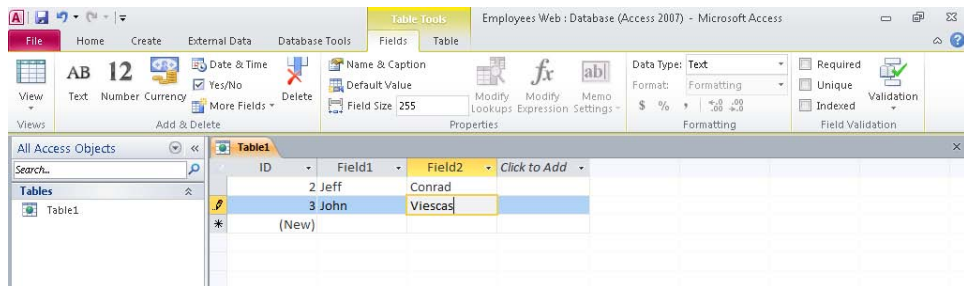
Because this is a new web database and no objects or special startup settings exist yet, you see a Navigation pane with only one object defined. For new web databases, Access, by default, creates a new web table in Datasheet view called Table1 with an ID field already defined. However, Access has not saved this table, so if you do not make any changes to it, Access will not prompt you to save the table if you close it. The following sections show various methods for creating a new table.

## Creating Your First Simple Web Table by Entering Data

If you've been following along to this point, you should still have your new Employees Web database open with Table1 open in Datasheet view, as shown in Figure 6-8. (You can also follow these steps in any open web database.) Access 2010 automatically created the first web field, called ID, in the left column. Leave this web field intact for now. In the second column, Access has placed another web field with the Add New Field heading. Just like client tables, you can enter just about any type of data you want in this web field—text, dates, numbers, or currency.

Let's start with a simple list of employee names so you can become more comfortable working in Datasheet view with web tables. For this quick example, we'll only need two columns containing the employee's first name and last name. Be sure to enter the same type of data in a particular column for every row. For example, enter Jeff's and John's last names in the third column (named Field2 by Access) for every row.

You can see some of the data entered for the employee list in Figure 6-9. Access behaves the same in this situation for web fields and tables just as it does for client fields and tables. When you start to type in a web field in a row, Access displays a pencil icon on the row selector at the far left to indicate that you're adding or changing data in that row. Press the Tab key to move from column to column. When you move to another row, Access saves what you typed. If you make a mistake in a particular row or column, you can click the data you want to change and type over it or delete it. Notice that after you enter data in a column, Access guesses the most appropriate data type and displays it in the Data Type box in the Formatting group of the Fields tab on the ribbon.

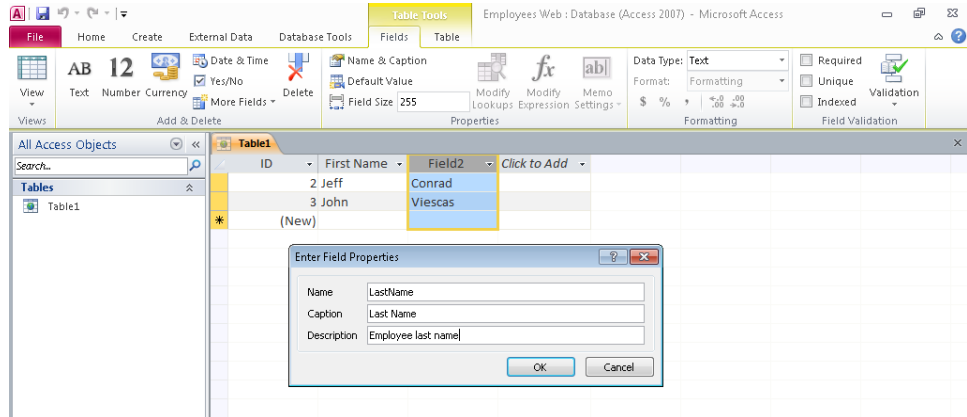


**Figure 6-9** You can create the employee list table by entering data.

If you create a column of data that you don't want, click anywhere in the column and click Delete in the Add & Delete group of the Fields contextual tab on the ribbon. Click Yes when Access asks you to confirm the deletion. If you want to insert a blank column between two columns that already contain data, right-click the column header to the right of where you want to insert the new column and then click Insert Field from the shortcut menu that appears. To move a column to a different location, click the field name at the top of the column to select the entire column, and then click again and drag the column to a new location. You can also click an unselected column and drag your mouse pointer through several adjacent columns to select them all. You can then move the columns as a group.

Access named your columns Field1 and Field2 just as it does if you are creating client tables in Datasheet view. You can change the name of a field by highlighting the column and then clicking the Name & Caption command in the Properties group on the Fields tab. Access opens the Enter Field Properties dialog box, as shown in Figure 6-10. In this dialog box, you can enter the field name you want to use in the Name text box, the field caption in the

Caption text box, and the field description in the Description text box. Note that for web tables, the Enter Field Properties dialog box is your only entry point in the user interface to set and update the field caption and field description for your web fields.

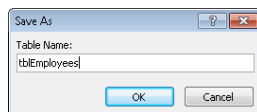


**Figure 6-10** The Enter Field Properties dialog box allows you to change the name, caption, and description for your web fields.

### Note

Just as with client tables in Datasheet view, you can change the name of a column by double-clicking the column's field name or right-clicking a column header and then clicking Rename Field from the shortcut menu that appears.

After you enter several rows of data, it's a good idea to save your table. You can do this by clicking the Save button on the Quick Access Toolbar or by clicking the File tab and then clicking Save. Access 2010 displays a Save As dialog box, as shown in Figure 6-11.



**Figure 6-11** Access 2010 displays the Save As dialog box when you first save a new web table so that you can specify a table name.

### Choosing Web Table Names

Access 2010 gives you lots of flexibility when it comes to naming your web tables; however, there are some restrictions to be aware of. A web table name can be up to 64 characters long, can include any combination of letters, numbers, spaces, and special characters except a period (.), exclamation point (!), square brackets ([]), leading space, leading equal sign (=), or nonprintable character such as a carriage return. The name also cannot contain any of the following characters: / \ ; \* ? " ' < > | # <TAB> { } % ~ &. In general, you should give your web tables meaningful names. You cannot name your web tables the same as any built-in SharePoint list names that Access Services uses to maintain your site. Specifically, you cannot use any of the following web table names: Lists, Docs, WebParts, ComMd, Webs, Workflow, WFTemp, Solutions, Report Definitions, or AppImages.

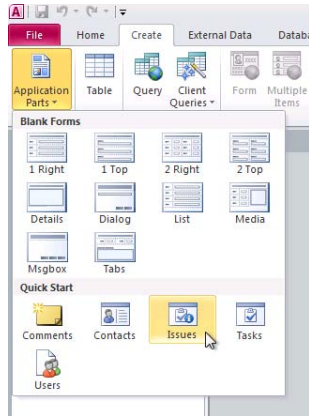
## Creating a Web Table Using Application Parts



If you were to look in the Contacts Map sample web database (ContactsMap.accdb) included on the companion CD, you'll find it to be very simple, with one main table to store contact information. Most databases are usually quite a bit more complex. For example, the built-in Projects sample web template you saw earlier in this chapter contains five main tables, and the Back Office Software System sample web database (BOSS.accdb) included on the companion CD contains nearly two dozen tables. If you had to create every web table manually, it could be quite a tedious process.

Fortunately, Access 2010 comes with a new feature called Application Parts to help you build a few common web tables and other web database objects. In Chapter 4, we first introduced you to this new feature when we discussed creating client tables. The good news with this feature is that you can also use Application Parts when creating web fields, web tables, and other web objects. In this section, we'll show you another feature with Application Parts that we did not discuss in Chapter 4—using Application Parts to create lookup fields with relationships to other web tables.

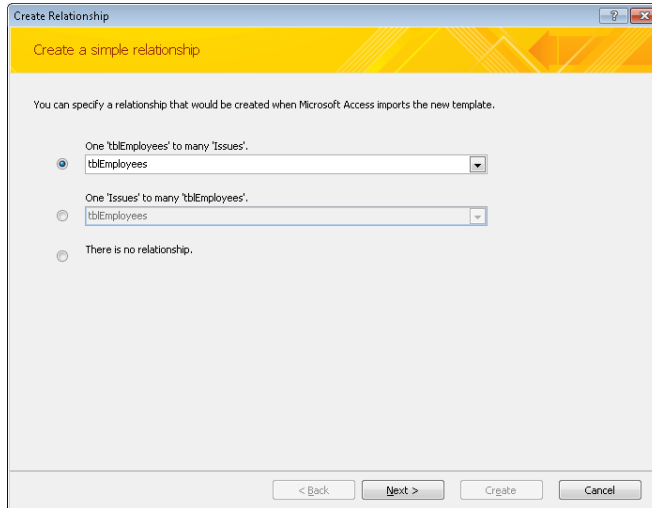
To complement the simple employee web table you created previously, it would be helpful to create another web table that can track any issues assigned to the employees. To build this second web table using one of the Application Parts, close the employees table if you still have it open, click the Create tab on the ribbon, and then click the Application Parts button in the Templates group. Access displays a list of 10 form types under the Blank Forms category and five Application Parts under the Quick Start category, as shown in Figure 6-12. Microsoft uses the term Models to refer to this one-click object creation feature. You'll learn more about using the 10 form Application Parts in Chapter 13, "Building a Form."



**Figure 6-12** Application Parts help you create common types of web database objects.

When you are using the five Application Parts under Quick Start in a web database, Access creates the same fields, tables, and objects as it does for a client database. The only difference when creating these objects in a web database is that Access marks everything as a web object. The five web Application Parts under Quick Start, which represent some of the more common types of web table structures and objects found in databases, are Comments, Contacts, Issues, Tasks, and Users. See “Creating a Table Using Application Parts,” on page 178, for a description of each of these Application Parts.

Click Issues in the Quick Start list, and Access opens the Create Relationship wizard, as shown in Figure 6-13. Whenever you select to build an Application Part under Quick Start in a database that already includes at least one existing table, Access displays the Create Relationship wizard to see if you want to create a relationship between one of your existing tables and the new table. In this case, Access identified the existing employees table you created earlier and now asks if you want to build a relationship with the new Issues table Access is about to create.



**Figure 6-13** Use the Create Relationship wizard to help create relationships between tables through Application Parts.

On the first page of the Create Relationship wizard, Access displays three options. Next to the first two options, Access displays combo boxes that list all the existing saved tables in your database. Above these combo boxes, Access displays text to help identify how it will create the relationship between the two tables. If you select either of these two first options, Access creates a one-to-many relationship between the selected table in the combo box and the new issues table Access will create. Select the first option if you want Access to create a new lookup field in the issues table, representing the many part of the relationship. (We'll discuss more about lookup fields and relationships in web tables later in this chapter.) Select the second option if you want Access to create a new lookup field in the selected table, representing the many part of the relationship. You can choose to select a different table in the combo boxes; by default, Access displays the tables in alphabetical order. Select the third option if you do not want Access to create a relationship between the new issues table and any existing table. Note that if you choose this option, Access does not create an additional lookup field in any tables.

For your example, we want to have the employees table (tblEmployees) represent the one side of the relationship and the issues table represent the many side of the relationship—each employee can have many issues assigned to them. Select the first option and leave the default tblEmployees selected in the combo box. (We only have one existing table in this sample database so you don't need to change any settings on the first page of the wizard.) Click Next to move to the second page of the wizard, as shown in Figure 6-14.



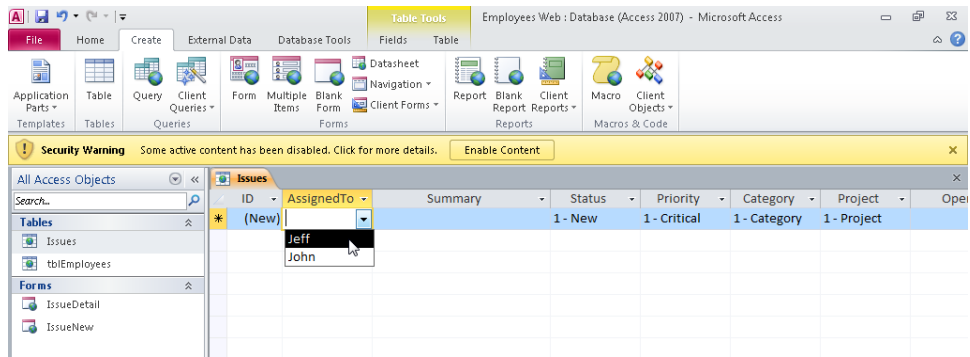
**Figure 6-14** Select options for the lookup field on the second page of the wizard.

On the second page of the wizard, Access displays options where you can customize the new lookup field in the issues table. In the first combo box, Access displays a list of field names from the table you selected on the first page of the wizard. Access displays the field you choose here as the drop-down list of options in the new lookup field. Select `FirstName` in this combo box, selected by default, to display the first name of each employee in the lookup field. The second option, `Sort This Field`, allows you to sort the values of the field you selected in the first combo box option. You can choose `Sort Ascending`, `Sort Descending`, or `None` (the default) to sort the values. Select `Sort Ascending`, for this example, to sort the first names in ascending order. In the third option, enter the name of your new lookup field in the text box provided (`AssignedTo`, for our example). The last option on this page of the wizard—`Allow Multiple Values`—tells Access to create a Multi-Value Lookup Field. We don't want to assign issues to more than one employee, so leave this option cleared. Note that if you selected `There Is No Relationship` on the first page of the wizard, you won't see the second page of the wizard.

### Note

If you click `Cancel` on either page of the `Create Relationship` wizard, Access immediately closes the wizard but does not cancel the creation of the objects associated with the Application Part you selected. Access, in this case, still continues the process of creating the objects but does not create any lookup fields in the tables representing relationships.

Click Create and Access builds a complete table structure for an issues web table, as well as other supporting objects. Open the Issues table and notice Access created 13 fields to identify the data elements for this Issues table, as shown in Figure 6-15. This Issues Application Part includes fields such as AssignedTo (your new lookup field), Summary, Status, Priority, Category, Project, Opened Date, Resolution, and so on to identify a single subject—an issue. The Issues Application Part also automatically defines a data type for each of these fields. Notice, in Figure 6-15, that if you click the drop-down list in the AssignedTo lookup field, you can see the first names of the employees from the tblEmployees table.



**Figure 6-15** The Issues command under Quick Start builds a complete table with appropriate field types and supporting objects.

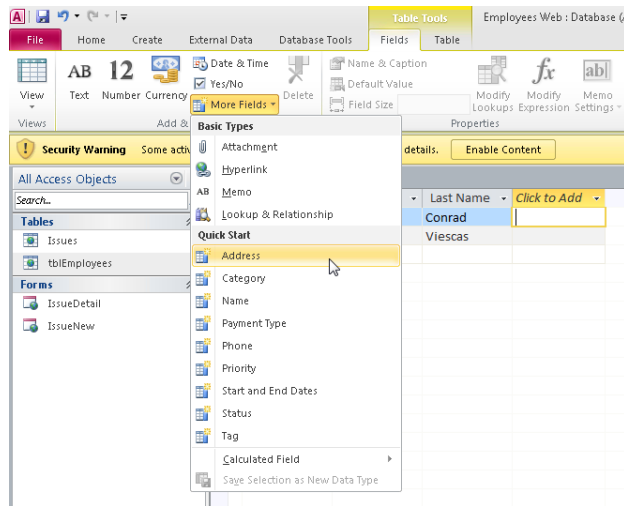
You can save time creating a web table by using an Application Part even if the table structure Access creates does not exactly match your needs. You can rename fields, add new fields, and delete unneeded fields to customize the table to your specific application needs. When you use an Application Part to help you create a web table, you also get the added benefit of Access creating other supporting objects to work with that table. Close the Issues table window now so you can continue with the next section.

## Using Data Type Parts

In Chapter 4, you learned about the new feature in Access 2010, called Data Type Parts, which assist you in creating fields in your tables. Similar to Application Parts, Data Type Parts help give you a jump start on creating your application. Data Type Parts create individual fields or groups of fields in existing tables. As you design more web databases, you

might find yourself needing to create similar field structures in your tables. For example, you'll probably find yourself needing to have fields in at least one table that track address information such as street address, city, state, and ZIP code. Fortunately, this new feature in Access 2010 can also be used when you are creating web tables. However, you only have the Quick Start options for Data Type Parts available for web tables.

If you've been following along to this point, you should still have your test web database open with the tblEmployees web table, Issues web table, and supporting two forms in the Navigation pane. To create fields using one of the Data Type Parts, you first need to have a table opened in Datasheet view. Select the tblEmployees web table you created earlier in the Navigation pane, open it in Datasheet view, and then put your cursor in the Click To Add empty field to the right of the LastName field. Click the More Fields button in the Add & Delete group on the Fields tab, and Access displays various field types, as shown in Figure 6-16.



**Figure 6-16** Click More Fields to see the Data Type Parts you can use to create groups of web fields.

Under the Basic Types category, Access displays four field types you can use in your web table—Attachment, Hyperlink, Memo, and Lookup & Relationship. You'll be creating these types of fields later in this chapter. Beneath the Quick Start category, you can see a list of the nine Data Type Parts—Address, Category, Name, Payment Type, Phone, Priority, Start And End Dates, Status, and Tag. These are the same Data Type Parts you can use in client tables, except in this case, Access creates web fields because you are using a web database. See "Creating a Table Using Data Type Parts," on page 182, for a description of each of these Data Type Parts.

Click Address under the Quick Start category and Access creates five fields to hold address information ready for you to use in your employees table, as shown in Figure 6-17. You can add additional Data Type Parts to your web table by clicking another option under the Quick Start category. If you want any new Data Type Part fields to appear at the end of your web table (as you saw with the address fields you just created), make sure to set the focus in the Click To Add empty field at the far right. Access always adds new Data Type fields to the left of where the current focus is located in the Datasheet view grid.

ID	First Name	Last Name	Address	City	State Province	ZIP Postal	Country Region	Click to Add
2	Jeff	Conrad						
3	John	Viescas						
(New)								

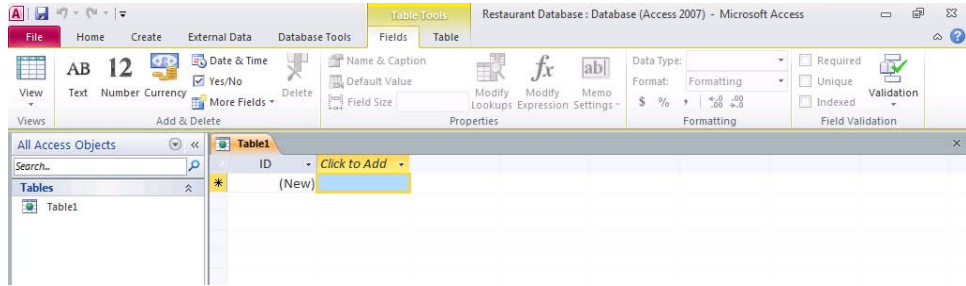
**Figure 6-17** Use the Address Type Part when you need to create fields to track address information.

Using Data Type Parts can save you time when you're designing your web tables in Access 2010, just like for client tables, by giving you a jump start on creating common field types. Close the Table window now and do not save the changes to this table when Access prompts you to save the changes. You can now close this database to begin the next section.

## Creating Web Tables in Datasheet View

You could continue to use Data Type Parts and Application Parts to build web fields and other web tables. However, you'll find in most cases, you need to create your web schema from scratch to meet your specific application needs. Even if you're already very familiar with creating new client tables in Design View, you'll find it necessary to learn the mechanics of building a web table from scratch in Datasheet view because Design view is not available for web objects. You can only use Datasheet View to design web tables. By learning how to create web tables in Datasheet view, you can also use that knowledge to design client tables in Datasheet view.

To begin creating a new web table in Datasheet view, let's start with a new, empty web database. Click the Blank Web Database button on the New tab of the Backstage view, name your new web database Restaurant Database, and then click Create. Access 2010 displays a blank web Table window in Datasheet view, as shown in Figure 6-18.



**Figure 6-18** Access displays an empty web table in Datasheet view when you create a blank web database.

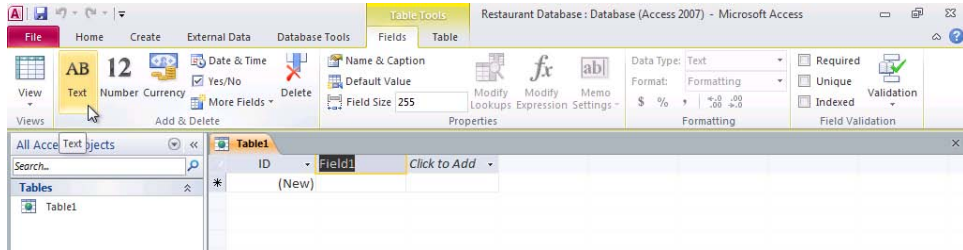
When you work in Design view with client tables, the upper part of the Table Design window displays columns in which you can enter the field names, the data type for each field, and a description of each field. Access allows you to set various field properties in the lower-left section of the Table Design window. When you design web tables in Datasheet view, however, all your design elements are in the two contextual ribbon tabs—Fields and Table—under Table Tools.

For details about web data type values, see “Understanding Web Field Data Types,” on page 311.

## Defining Web Fields



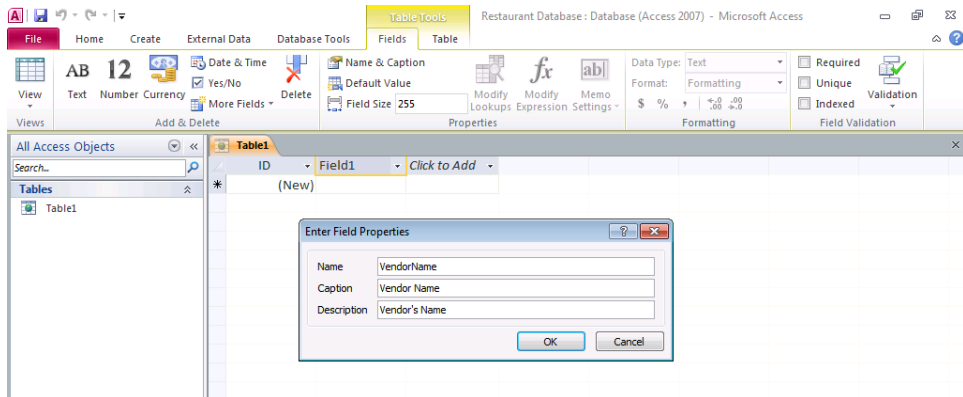
Now you’re ready to begin defining some of the web fields and web tables in this new empty web database that you can find in the Back Office Software System sample web database (BOSS.accdb). The Back Office Software System sample web application is used to track various facets of a restaurant business such as purchases, employee file maintenance, scheduling, and so forth. To understand how to create web tables like the ones you can find in the Back Office Software System, we’ll start by creating a new web table from scratch to use for a vendor list. You should still have your empty Table1 open in Datasheet view that Access created when you opened a new web database. Be sure the insertion point is in the Click To Add field in the first row, and then click the Text command in the Add & Delete group on the Fields contextual tab, as shown in Figure 6-19.



**Figure 6-19** You can add new web fields to your table by clicking the various data type commands in the Add & Delete group.

Access creates a new text field called Field1 to the right of the ID field. You'll notice in Figure 6-19 that Access now displays Text in the Data Type list in the Formatting group and sets the default Field Size to 255 in the Properties group. When you are creating and modifying web fields in Datasheet view, you'll need to look at the Properties, Formatting, and Field Validation groups on the Fields contextual ribbon to view and edit the various properties for each field.

Access sets your focus to the top of the new Field1 column where you can change the name. You can change the default name Field1 Access provided by highlighting the existing text and typing in a new field name. We need to also set a caption and description for this new field so we'll set all of these at the same time. Click the Name & Caption button in the Properties group, and Access opens the Enter Field Properties dialog box, as shown in Figure 6-20.



**Figure 6-20** Click the Name & Caption command to set the field name, caption, and description properties for your new field.

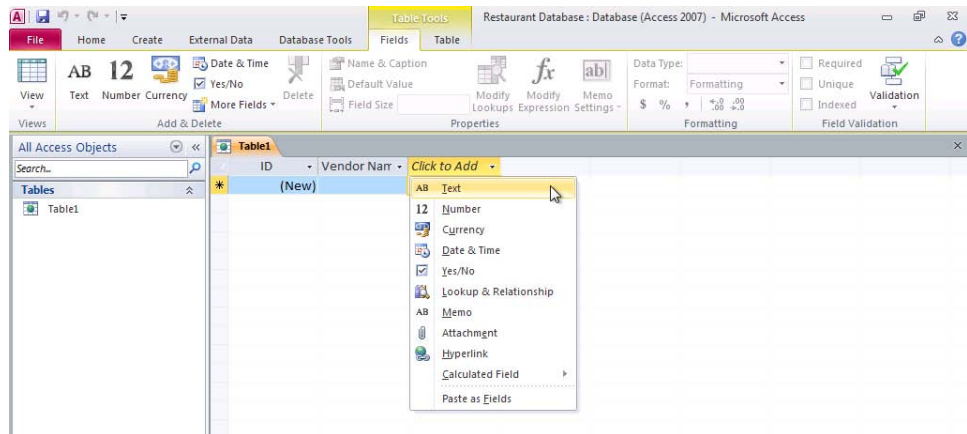
In the Name field, type the name of this first new field, **VendorName**. Press Tab once to move to the Caption column and then type **Vendor Name**. Access displays this caption property at the top of the column when you view this table in Datasheet view. Finally, press Tab once more to move to the Description property. In the Description property for each web field, you can enter a descriptive phrase. Access 2010 displays this description on the status bar (at the bottom of the Access window) whenever you select this field in a query in Datasheet view or in a form in Form view or Datasheet view. For your new VendorName field, enter **Vendor's Name** in the Description property and then click OK to close the dialog box.

## INSIDE OUT

### Why Setting the Description Property Is Important

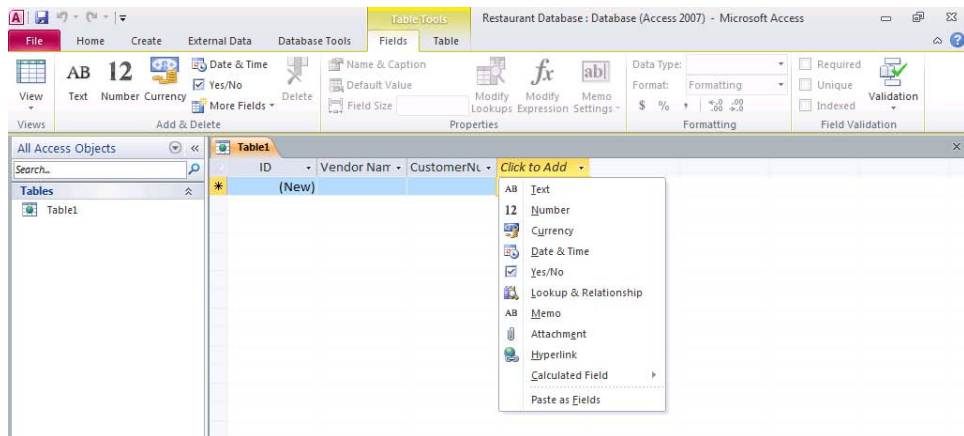
In Chapter 4, we discussed the importance of entering a Description property for every field in your client table to help document your application. Access 2010 also displays the description on the status bar, which can serve as a kind of mini-help for the users of your database. Our advice on using the Description property also applies to web fields in web databases. Documenting your web database is just as important for web databases as it is for client databases because it can help you as well as other people who might need to modify your work in the future.

In addition to using the data type commands in the Add & Delete group to create your new web fields, you can also click the column header of the Click To Add empty field to display a list of the data types you can use in web tables, as shown in Figure 6-21. Click Text or press the shortcut key T to create another new text field for your vendor table. Access creates another new text field and gives it the default name of Field1.



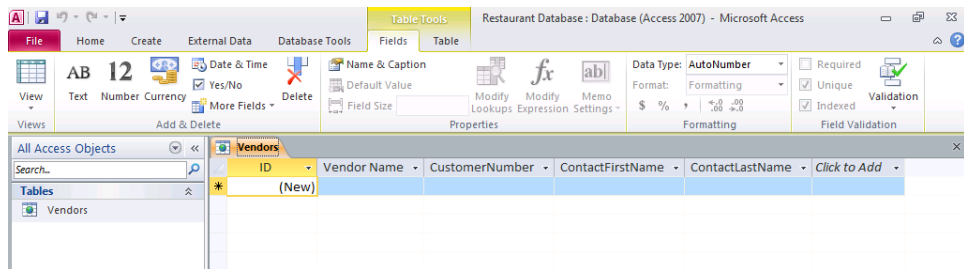
**Figure 6-21** Click the column header on the Click To Add empty field to drop down a list of data types to use for your new web field.

You can use the options in the Click To Add empty field for rapid creation of your web fields and then go back through and add captions and descriptions to your fields later. For example, the focus should currently be on the column header of the new text field you just created. Enter **CustomerNumber** as the name of this new text field. (We're using a text data type for this field, because the customer number can contain alphanumeric characters.) Once you've entered the new field name, press Tab and Access moves the focus to the Click To Add empty field and drops down the list of data types again for you, as shown in Figure 6-22. You can press the shortcut key corresponding to the data type you want to create your next field without ever having to use the mouse. Use this technique to create two new text fields called **ContactFirstName** and **ContactLastName**.



**Figure 6-22** Use the shortcut keys for the corresponding data type options in the Click To Add empty field to quickly create new web fields for your table.

Your table up to this point should now look like Figure 6-23. Click the Save button on the Quick Access Toolbar to save this table in its present state and name the new web table **Vendors**.



**Figure 6-23** Your changes for your new web Vendors table should now look like this.



## Choosing Web Field Names

Access 2010 gives you lots of flexibility when it comes to naming your web fields. A web field name can be up to 64 characters long, can include any combination of letters, numbers, spaces, and special characters except a period (.), exclamation point (!), square brackets ([]), leading space, leading equal sign (=), or nonprintable character such as a carriage return. The name also cannot contain any of the following characters: / \ : ; \* ? " ' " < > | # <TAB> { } % ~ &. In general, you should give your fields meaningful names and should use the same name throughout for a field that occurs in more than one table. You should avoid using field names that might also match any name internal to Access or Microsoft Visual Basic. For example, all objects have a Name property, so it's a good idea to qualify a field containing a name by calling it VendorName or CompanyName. You should also avoid names that are the same as built-in functions, such as Date, Time, Now, or Space. See Access Help for a list of all the built-in function names.

Although you can use spaces within your field names in a web database, you should try to create field names without embedded spaces. Many Structured Query Language (SQL) databases to which Access can link (notably Oracle and Ingres) do not support spaces within names. Although Microsoft SQL Server does allow spaces in names, you must enclose such names in brackets or use quotes and execute a Set Quoted Identifier On command. If you ever want to move your web application to a client/server environment and store your data in an SQL database such as SQL Server or Oracle, you'll most likely have to change any names in your web database tables that have an embedded space character. As you'll learn later in this book, table field names propagate into the queries, forms, and reports that you design using these tables. So any name you decide to change later in a table must also be changed in all your queries, forms, and reports.

If you use reserved words or function names for web field names, Access 2010 catches most of these and displays a warning message. This message warns you that the field name you chose, such as Name or Date, is a reserved word and you could encounter errors when referring to that field in other areas of the database application. Access still allows you to use this name if you choose, but take note of the problems it could cause. To avoid potential conflicts, we recommend you avoid using reserved words and built-in functions for web field names.

## Understanding Web Field Data Types

When you are creating fields for a web database, Access allows you to use only data types that are supported in SharePoint lists. Access 2010 web databases support 10 types of data, each with a specific purpose. You can see the details about each data type in Table 6-1.

The Lookup & Relationship option helps you define the characteristics of foreign key fields that link to other tables or value lists. You'll learn about the Lookup & Relationship option later in this chapter in "Creating Lookup Fields in a Web Database," on page 337.

**Table 6-1** Access Web Data Types

Data Type	Usage	Size
Text	Alphanumeric data.	Up to 255 characters
Number	Numeric data.	All numbers in web tables are 8-byte Double
Currency	Monetary data, stored with four decimal places of precision.	8 bytes
Date/Time	Dates and times.	8 bytes
Yes/No	Boolean (true/false) data; SharePoint stores the numeric value zero (0) for false, and one (1) for true.	1 byte
Lookup & Relationship	When you choose this entry, a wizard starts to help you define either a simple or complex lookup field. A simple lookup field uses the contents of another table or a value list to validate the contents of a single value per row. A complex lookup field allows you to store multiple values of the same data type in each row.	Dependent on the data type of the lookup field
Memo	Alphanumeric data—sentences and paragraphs.	Values are truncated if they contain more than 8,192 characters
Attachment	You can attach files such as pictures, documents, spreadsheets, or charts; each Attachment field can contain an unlimited number of attachments per record, up to the storage limit of the SharePoint list. You can only have one Attachment field per web table.	Varies based on SharePoint list and site settings
Hyperlink	A link "address" to a document or file on the World Wide Web, on an intranet, on a local area network (LAN), or on your local computer.	SharePoint Hyperlink fields can only store 255 characters for the Uniform Resource Locator (URL) and 255 characters for the description
Calculated	You can create an expression that uses data from one or more fields. You can designate different result data types from the expression.	Dependent on the data type of the Result Type property. Text data type result can have up to 243 characters. Memo, Number, Yes/No, and Date/Time should match their respective data types.

For each field in your table, select the data type that is best suited to how you will use that field's data. For character data, you should normally select the Text data type. You can control the maximum length of a Text field by using a field property, as explained later in this chapter. Use the Memo data type only for long strings of text that might exceed 255 characters or that might contain formatting characters such as tabs or line endings (carriage returns).

The Date/Time data type is useful for calendar or clock data and has the added benefit of allowing calculations in seconds, minutes, hours, days, months, or years. For example, you can find out the difference in days between two Date/Time values.

## INSIDE OUT

### Understanding the SharePoint Date/Time Data Type

Use the Date/Time data type to store any date and time value. It's useful to know that SharePoint lists, like Access 2010, store the date as the integer portion of the Date/Time data type and the time as the fractional portion—the fraction of a day, measured from midnight that the time represents. For example, 6:00:00 A.M. internally is 0.25. In SharePoint Date/Time fields, the day number is actually the number of days since January 1, 1900, and cannot display dates prior to that date. The Date/Time data type in Access client tables starts with December 30, 1899, and can be a negative number for dates prior to that date. If you are working in a web database with web tables, however, you cannot store any date values prior to January 1, 1900, because SharePoint Date/Time fields do not support data before that date.

You should generally use the Currency data type for storing money values. Currency has the precision of integers, but with exactly four decimal places.

Use the Yes/No data type to hold Boolean (true or false) values. This data type is particularly useful for flagging accounts paid or not paid or orders filled or not filled. SharePoint Boolean fields store True values as 1, unlike Access Boolean fields, which store True as -1. To work around this disparity, you should only use check boxes to display Boolean fields in web databases so the users of your database are not looking at an integer value for the field. (When you create a Boolean field in a web table, you'll notice Access only gives you the option to display check boxes.) If you attempt to use any restrictions, filters, or comparisons on the integer value of Boolean fields on server forms, you'll see a runtime error. Note that it is still possible to use the integer value of Boolean fields in restrictions, filters, and comparisons in Access client with data published to the server; however, we recommend you always use True and False constants when working with Boolean fields in web databases to ensure consistent behavior in both client and server.

The Hyperlink data type lets you store a simple or complex “link” to an external file or document. (Internally, Hyperlink is a memo data type with a special flag set to indicate that it is a link.) This link can contain a URL that points to a location on the World Wide Web or on a local intranet. It can also contain the Universal Naming Convention (UNC) name of a file on a server on your LAN or on your local computer drives. The link can point to a file that is in Hypertext Markup Language (HTML) or in a format that is supported by an ActiveX application on your computer. If you are using existing data in a Hyperlink field before publishing your database to a SharePoint server, be careful about how many characters are stored in your hyperlinks. Hyperlink fields in client tables (and web tables before publishing to the server) can support more than 255 characters; however, Hyperlink fields in SharePoint lists support only up to 255 characters. Access prevents you from publishing your data to the server if any Hyperlink fields contain more than 255 characters. You need to edit or remove any hyperlinks that contain more than 255 characters before you can successfully publish your web database to the server.

The Attachment data type, introduced in Access 2007, allows you to store multiple attachments in a single record. These files are stored in a binary field in a hidden system table. For the Attachment data type, Access compresses each file, if it isn’t already, and uses the original file rather than a generated thumbnail to minimize the amount of database bloat. You are restricted to only having one Attachment field per web table in web databases. Access displays an error message if you try to create more than one Attachment field in a single web table. SharePoint lists have a default size limit of 50 MB for each attachment. If you have attachments larger than 50 MB, you might encounter problems publishing your application to the server or syncing your data to the server. The attachment size limit is an administrative configurable setting. Contact your SharePoint server administrator if you are experiencing issues publishing larger attachments.

The Calculated data type, newly introduced in Access 2010, can also be published to the server. When you publish a web table to the server that includes a Calculated data type, Access creates a Calculated column in the SharePoint list and sets the expression to match what you defined in Access client. Calculated fields allow you to create a calculated result using an expression. The expression you use can include data from one or more fields in the same table. If you have a number field, for example, that holds quantity information for products purchased and a currency field that holds the price of a product, you can create a calculated field that multiplies the quantity and price fields and stores it with a result type of currency. You could also create a calculated field that concatenates a first name, middle name, and last name fields and stores it with a result type of text for a field called Full Name. Access recalculates the value of the calculated field any time the dependent fields are changed.

You might have noticed that you cannot create an AutoNumber field in web tables. All SharePoint lists include an ID field which increments sequentially like Access AutoNumber data types. Whenever you create a new web table, Access automatically creates an ID field for you because SharePoint lists require it. You cannot delete the ID field from your web tables; however, you can rename the field if you choose.

In client databases, you have a hard limit of 255 fields allowed in a table, but for web databases, you have a hard limit of 220 fields (including the ID field) because SharePoint lists have many hidden fields. Unlike Access tables, SharePoint lists have default limits on the number of columns for each data type. SharePoint lists also have a logical row-size limit of 8,000 KB and a configurable limit of physical rows per list item. This means that you might encounter publishing errors if you have more fields of one data type allowed in a web table, you are over the limit of row size for all fields combined, or a combination of the two. As a baseline, you can publish 48 Date/Time fields, 71 Number or Currency fields, 96 Yes/No fields, 191 Hyperlink fields, 191 Memo fields, or 219 Text fields in a single web table with default SharePoint list settings.

## Setting Field Properties for Web Databases

You can customize the way Access 2010 stores and handles each field in web databases by setting specific properties. These properties vary according to the data type you choose and are available in the Properties, Formatting, and Field Validation groups on the Fields contextual ribbon tab. Table 6-2 lists all the possible properties that can appear for fields in a web database displayed in Datasheet view, and the data types that are associated with each property.

**Table 6-2** Field Properties on the Fields Contextual Tab

Property	Data Type	Options, Description
Field Size	Text	Text can be from 0 through 255 characters long, with a default length of 255 characters.
Format	Number	<p><b>General Number (default).</b> No commas or currency symbols; the number of decimal places shown depends on the precision of the data.</p> <p><b>Standard.</b> Two decimal places and separator commas.</p> <p><b>Percent.</b> Moves displayed decimal point two places to the right and appends a percentage (%) symbol.</p>
	Currency	<p><b>Currency.</b><sup>1</sup> Currency symbol (from Regional And Language Options in the Control Panel) and two decimal places.</p> <p><b>Euro.</b> Euro currency symbol (regardless of Control Panel settings) and two decimal places.</p>
	Date/Time	<p><b>General Date (default).</b> Combines Short Date and time (for example, 7/1/2010 5:30:10 PM).</p> <p><b>Short Date.</b><sup>2</sup> Uses Short Date Style from Regional And Language Options (for example, 7/1/2010).</p>
	Calculated	Format property options for calculated fields depend on the Result Type. The format property options and defaults for the Result Type align with the other data types.
Increase/Decrease Decimals	Number and Currency	You can specify the number of decimal places that Access displays. The default specification is to display two decimal places for the Currency, Standard, and Percent formats and the number of decimal places necessary to show the current precision of the numeric value for General Number format. You can request a fixed display of decimal places by clicking these buttons.
Caption	All	You can enter a more fully descriptive field name that Access displays in form labels and in report headings. (Tip: If you create field names with no embedded spaces, you can use the Caption property to specify a name that includes spaces for Access to use in labels and headers associated with this field in queries, forms, and reports.)
Default Value	Text, Number, Currency, Date/Time, and Boolean	You can specify a default value for the field that Access automatically uses for a new row if no other value is supplied. If you don't specify a Default Value, the field will be Null if the user fails to supply a value. (See also the Required property.)

Property	Data Type	Options, Description
Validation Rule	Text, Number, Currency, and Date/Time	You can supply an expression that must be true whenever you enter or change data in this field. For example, <100 specifies that a number must be less than 100. You can also check for one of a series of values. For example, you can have Access check for a list of valid cities by specifying "Boston" Or "Seattle" Or "Los Angeles". In addition, you can specify a complex expression that includes any of the built-in functions in Access. See "Defining Simple Field Validation Rules," on page 201, for details.
Validation Text	Text, Number, Currency, and Date/Time	You can specify a custom message that Access displays whenever the data entered does not pass your validation rule.
Required	All except Boolean and Calculated	If you don't want to allow a Null value in this field, select this property.
Unique	Text, Number, Currency, and Date/Time	If you don't want to allow duplicates in this field, select this property.
Indexed	Text, Number, Currency, and Date/Time	You can ask that an index be built to speed access to data values.
Memo Settings	Memo	<b>Plain Text (default).</b> The text in the Memo field is stored and displayed as plain text.
		<b>Append Only.</b> You can specify to see column history for this Memo field. When you change the memo field's data, the data change and time stamp are recorded and appended to the version history of the field.
		<b>Rich Text.</b> You can specify that the data in the memo field can be formatted as rich text. Access applies HTML formatting tags to your data.
Modify Expression	Calculated	The expression used to calculate the value for this column. The expression can use the value of one or more fields in the same table and can be up to 65,000 characters in length.

- 1 Note that Currency and Euro formats always display two decimal places regardless of the number of actual decimal places in the underlying data. Access rounds any number to two decimal places for display if the number contains more than two decimal places.
- 2 To help alleviate problems with dates spanning the start of the century, we recommend that you select the Use Four-Digit Year Formatting check box in Access. Click the File tab on the Backstage view, click Options, and then scroll to the General section in the Client Settings category to find this option. You should also be sure that your Short Date Style in the Regional And Language Options dialog box uses a four-digit year. (This is the default in Windows XP, Windows Vista, and Windows 7; you can double-check your settings by accessing Regional And Language Options within Control Panel.)

## INSIDE OUT

### Don't Specify a Validation Rule Without Validation Text

If you specify a validation rule but no validation text, Access 2010 generates an ugly and cryptic message that your users might not understand:

"One or more values are prohibited by the validation rule '<your expression here>' set for '<table name.field name>'. Enter a value that the expression for this field can accept."

Unless you like getting lots of support calls, we recommend that you always enter a custom validation text message whenever you specify a validation rule.

## Completing the Fields in the Vendors Web Table



You now know enough about field data types and properties for web databases to finish designing the Vendors table in this example. (If you'd like to compare your work to the Back Office Software System web database, you can take a look at the tblVendors web table in the BOSSDataCopy.accdb sample web database on the companion CD.) Use the information listed in Table 6-3 to complete the web table shown in Figure 6-24.

**Table 6-3** Field Definitions for the Vendors Table

Field Name	Data Type	Description	Field Size
VendorID	ID	ID Field (AutoNumber)	
VendorName	Text	Vendor's name	50
CustomerNumber	Text	Customer number used by vendor	50
ContactFirstName	Text	Contact person's first name	50
ContactLastName	Text	Contact person's last name	50
ContactTitle	Text	Contact person's title	50
ContactCellNumber	Text	Contact person's cell phone number	30
Address	Text	Address of vendor	255
Address2	Text	Additional address information of vendor if needed	255
City	Text	City	50
PostalCode	Text	Postal/Zip Code	20
PhoneNumber	Text	Telephone number of vendor	30
PhoneNumberExtension	Text	Telephone extension if applicable	30



Field Name	Data Type	Description	Field Size
FaxNumber	Text	Fax number of vendor	30
EmailAddress	Text	Email address of contact or vendor	100
Notes	Memo-Rich Text	Additional notes for internal use	
Website	Hyperlink	Company website	
Active	Yes/No	Yes/No field whether this is an active vendor	
RelatedFiles	Attachment	Vendor specific files	

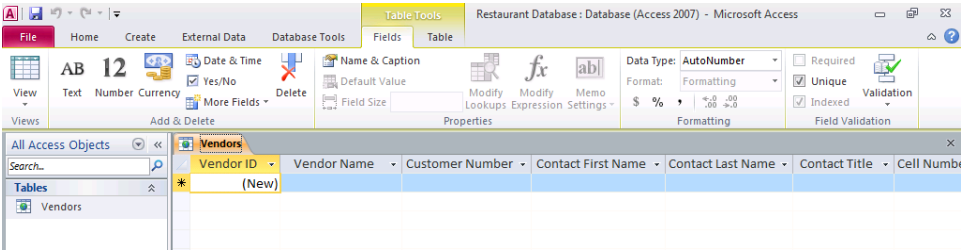


Figure 6-24 Your fields in the Vendors web table should now look like this.

You might have noticed that we did not have you create two fields found in the tblVendors table in the Back Office Software System sample web database—ContactFullName and State. We'll discuss creating these two fields in the next sections.

## Creating Calculated Fields

Access 2010 introduces a new data type called Calculated. SharePoint lists already have a Calculated data type, so if you create a Calculated field in a web database, the field and data will move to the server if you publish the database. This new data type can be used in client databases as well as web databases so the steps we show you here to create Calculated fields in web tables can be applied to client tables.

The Calculated data type allows you to create a calculated result using an expression. The expression you use can include data from one or more fields in the same table, but you cannot reference fields in other tables. If you have a number field, for example, that holds quantity information for products purchased and a currency field that holds the price of a product, you can create a calculated field that multiplies the quantity and price fields and stores it with a result type of currency. Access recalculates the value of the calculated field any time the dependent fields are changed. If the calculation for a specific row evaluates

to an error, Access stores the error for the Calculated field. For example, if your expression result divides a number by zero, Access displays the error #Div/0 in the Calculated field.

In client databases, you can reference all data types except Multi-Value and OLE Object fields in Calculated field expressions. In web databases, you cannot reference Multi-Value, OLE Object, Memo, Lookup, and AutoNumber fields for Calculated field expressions to maintain parity with SharePoint Calculated columns.

Calculated field expressions can reference functions and use operators that are supported in SharePoint Calculated columns. You cannot use volatile functions, such as Date() and Now(), in Calculated fields, because the stored value of the field would always be incorrect. You cannot use functions that look up outside the current record of the table, such as DSum, DCount, and DLookup, because Access cannot track when field dependencies change in those scenarios. You also cannot create an index or a relationship on a Calculated field. You can, however, use a Calculated field as a display column for a lookup field. You'll learn how to create relationships with lookup fields later in this chapter.

In Access 2010, you can generally use Calculated fields anywhere you can use other data types in your database. You can sort, filter on, and group by Calculated fields in client and web queries, forms, and reports. If your Calculated field evaluates to an error and you are filtering or grouping on that field, your query or report will fail to run. Access sorts Calculated field errors as Null values in table datasheets.

### Why Use Calculated Fields?

If you are familiar with data normalization rules, you might be asking yourself why you would use a Calculated field to store a value that is derived from other fields. As you'll learn in Chapter 9, expressions in queries can do everything a Calculated field can do and more. So why would you want to add a Calculated field in your table? Calculated fields in tables can provide potential performance gains when running large queries with the same expression. For a Calculated field, the result of the expression is stored at the time Access saves the record. When you're fetching data from a table using a query, the Calculated field value is fetched just like other data rather than calculated for each record. If you're running large queries against data stored in an Access Services site, you might see performance improvements by storing the value of an expression instead of calculating the value for each row.

However, you cannot upsize any table that has a Calculated field to SQL Server. If you plan to upsize your database in the future to SQL Server, you should consider not creating Calculated fields in your tables.

In the Vendors table you have been building in the Restaurant Database, let's add a Calculated field that concatenates, or combines, the vendor contact's first and last name. Open your Vendors table in Datasheet view, if it isn't already, and tab into the ContactLastName field. (Access creates the new field to the right of the field that currently has focus in Datasheet view so we are starting you off in the last name field.) Next, click the More Fields button in the Add & Delete group on the Fields contextual tab, highlight the Calculated Field option near the bottom of the drop-down list, and then click Text on the shortcut menu, as shown in Figure 6-25.

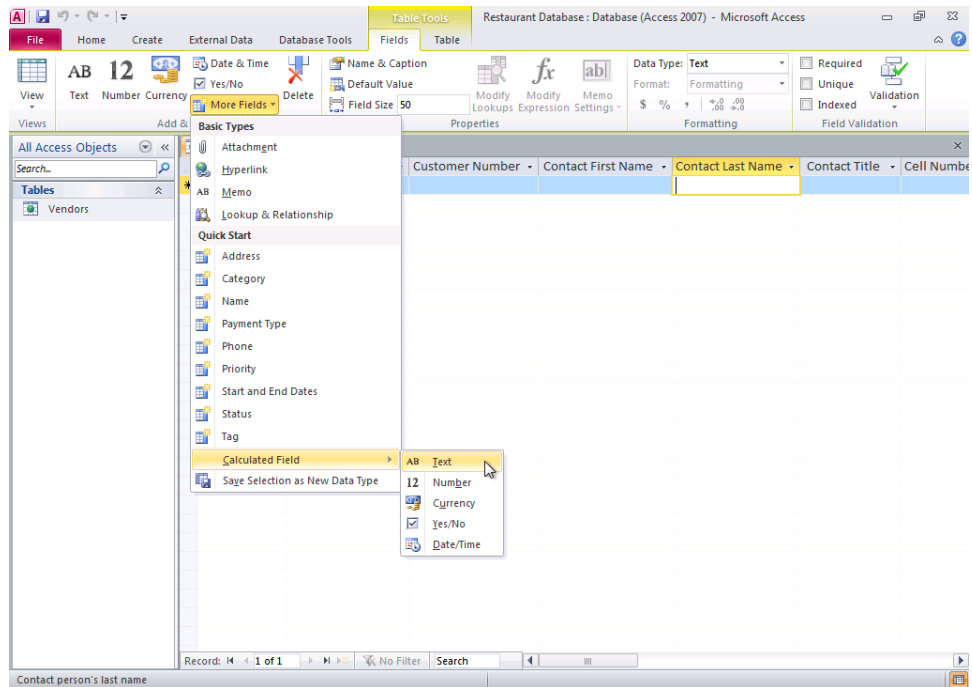
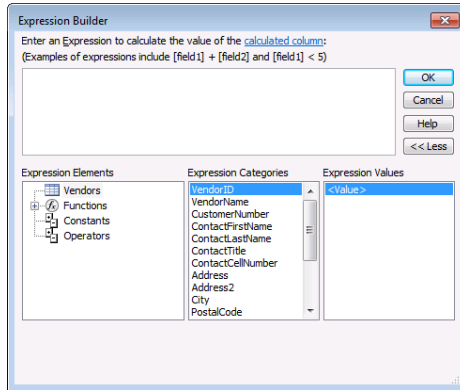


Figure 6-25 Select Text as the result type from the drop-down list for your new Calculated field.

Access opens the Expression Builder dialog box, as shown in Figure 6-26. In the upper part of the dialog box is a blank text box in which you can build an expression. You can type the expression yourself, but it's sometimes more accurate to find field names, operators, and function names in the three panes in the lower part of the dialog box.



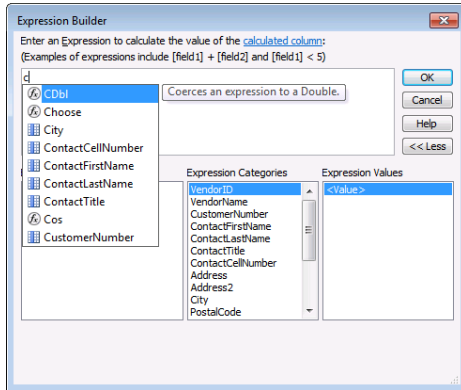
**Figure 6-26** The Expression Builder dialog box helps you build simple and complex expressions.

The expression you need to build, which we'll walk you through in detail in just a moment, will look like this:

```
[ContactFirstName] & " " & [ContactLastName]
```

You can use the Expression Builder to help you correctly construct the expression we need for our new Calculated field. The Expression Builder exists in previous of Access, but in Access 2010, Microsoft enhanced the dialog box with some new capabilities. The first major enhancement to the Expression Builder in Access 2010 is a concept called *progressive disclosure*. In previous versions of Access, the Expression Builder showed all available functions, constants, and operators for expressions regardless of context. In Access 2010, the Expression Builder shows you only the elements applicable to the current context. For example, in Figure 6-26, Access only shows you the list of functions, constants, and operators you can use for Calculated fields in web tables under the Expression Elements pane on the far left. The middle pane, Expression Categories, lists all the current fields in your Vendors table that you can refer to in your Calculated field. The right pane, Expression Values, shows you all possible values you can use based on the currently highlighted category selected in the middle pane. In this context, you can only refer to the field value of existing fields for new Calculated fields.

Start by typing the letter **c**, which is the first letter of the ContactFirstName field in the text box at the top of the Expression Builder, as shown in Figure 6-27. Access now shows the second enhancement to the Expression Builder in Access 2010—the addition of *IntelliSense*. IntelliSense helps you create expressions faster by showing you a drop-down list of words that match an object (such as a field or form name), function, or parameter once you type enough characters for Access to filter to the term. You can either accept the suggestion Access provides in the drop-down list by pressing Enter or Tab, or you can continue manually typing the name you want.



**Figure 6-27** IntelliSense helps you create expressions faster by providing a list of drop-down choices as you type to use in your expression.

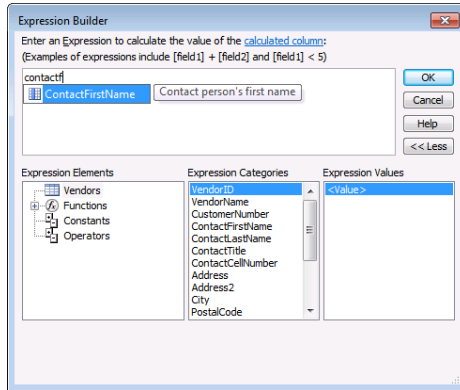
In Figure 6-27, you'll notice that by typing the letter **c**, Access displays three functions—CDbl, Choose, and Cos—and six fields in your Vendors tables, all of which start with the letter c. If you pressed Enter right now, Access selects the function CDbl and adds that to the text box. You'll also notice in Figure 6-27 that Access displays a tooltip (also called a ScreenTip) for the CDbl function. (You can find a list of the most useful functions and their descriptions in Article 4, "Function Reference," on the companion CD.)

## INSIDE OUT

### Using Keyboard Shortcuts to Display or Hide IntelliSense Options

When you are working in the Expression Builder, you can press **Ctrl+Space** to show the IntelliSense list of available items based on where your insertion point is located. If no characters precede your insertion point, Access displays all available items. Press the **Esc** key to dismiss the IntelliSense list at any time.

To select the **ContactFirstName** field from the drop-down list, you can use the down arrow key to highlight the field and then press **Enter** or **Tab**, or you can double-click the field name in the drop-down list. Alternatively, you can continue typing the letters **contactf**, as we did, until that is the only option in the drop-down list, as shown in Figure 6-28. Notice that Access displays the field's description in the tooltip next to the field. Press **Enter** or **Tab** now to add the **ContactFirstName** field to the text box. (You could also choose to simply double-click the field name you want to use in the Expression Categories pane to add the field to the top text box.)



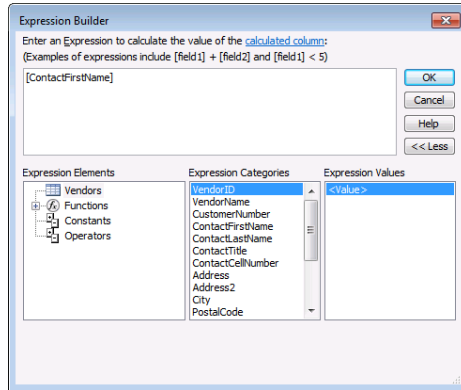
**Figure 6-28** Access displays descriptions next to the field names in the IntelliSense drop-down list.

## INSIDE OUT

### Increasing the Font Size in the Expression Builder

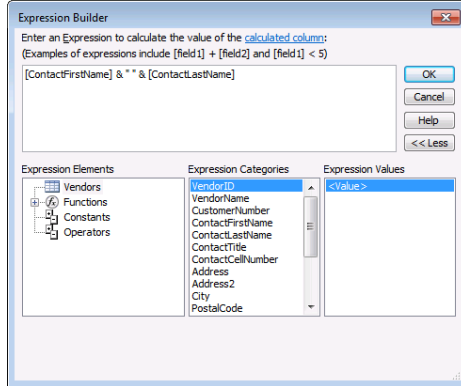
If the font size in the upper part of the Expression Builder seems too small for you to read, you can adjust the size of the font by holding down the Ctrl key and then scrolling with your mouse wheel. When you scroll up, Access increases the font size, and when you scroll down, Access decreases the font size. Access returns the font size to the default size whenever you open the Expression Builder dialog box.

You'll notice that after you select the field, the Expression Builder pastes [ContactFirstName] into the expression area, not just ContactFirstName, as shown in Figure 6-29. The Expression Builder adds brackets ([ ]) around all field names in case they might have embedded spaces in their names. If you design your field names without any blank spaces, you can leave out the brackets, but it's always good practice to include them.



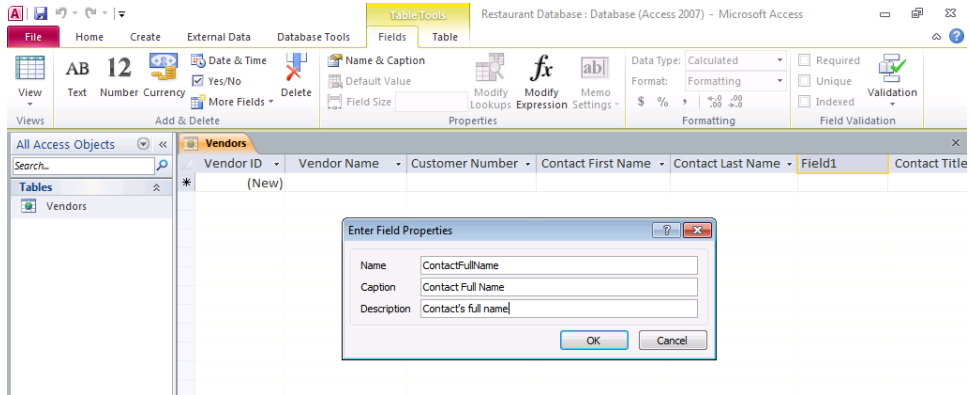
**Figure 6-29** The Expression Builder automatically places brackets around field names after you add them to the text box area.

Continue with the rest of the expression we need by typing **& " " &** and then adding the ContactLastName field by typing it manually, using the IntelliSense drop-down list as you type the letters, or double-clicking the field name in the Expressions Categories pane. Your completed expression should now look like Figure 6-30.



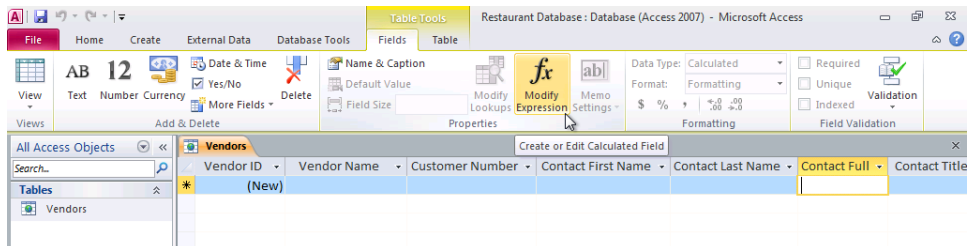
**Figure 6-30** Your completed expression for the new Calculated field should now look like this.

Click OK to save your expression changes and dismiss the Expression Builder dialog box. Access returns focus to the Vendor datasheet and names your new Calculated field Field1, by default. Let's give our new Calculated field a proper name. Click the Name & Caption button in the Properties group to open the Enter Field Properties dialog box, as shown in Figure 6-31. Type **ContactFullName** for the Name, **Contact Full Name** for the caption, and **Contact's full name** for the description. Click OK in the dialog box to save your changes.



**Figure 6-31** You can enter a proper name for the new Calculated field in the Enter Field Properties dialog box.

Your new Calculated field will now concatenate the first and last names of the vendor contact with a space between the two names for each record automatically. For example, if the contact's first name is Jeff and the contact's last name is Conrad, Access displays the text Jeff Conrad in your ContactFullName field. Note that all Calculated fields are read-only—you cannot type anything into these fields, nor can you adjust their data through any other means such as using Visual Basic code. Remember that Access recalculates the Calculated field whenever any dependent field values are changed. If you need to adjust the expression used for your Calculated field, highlight your field in Datasheet view and then click the Modify Expression button in the Properties group, as shown in Figure 6-32. Access opens up the Expression Builder dialog box, where you can adjust the expression to your needs. (Note that if you are using a client database, you can also adjust the expression for Calculated data types on the Expression field property setting in table Design view.) Save your changes to the Vendors table by clicking the Save button on the Quick Access Toolbar.



**Figure 6-32** Click the Modify Expression button in the Properties group if you need to adjust the expression used for your Calculated field.



**CAUTION!**

Don't use the plus sign (+) for string concatenation in web databases. Traditional Access client databases support the use of both the ampersand (&) and the plus sign (+) for concatenation of strings. However, the server only supports the use of the ampersand (&) for string concatenation. If you attempt to use the plus sign (+) in an expression with string values, you'll receive an error in any controls that use the expression after you publish your database to a SharePoint server and work with your database objects in a web browser.

## Defining Field Validation Rules for Web Databases

To define a simple check on the values that you allow in a field in a web database, you can enter an expression in the Validation Rule property for the field just as you can for client databases. Access won't allow you to enter a field value that violates this rule. Access performs this validation for data entered in a Table window in Datasheet view, in an updateable query, or in a form. If you publish your web database to a SharePoint server, the server will also enforce your field validation rules.

In general, a field validation expression consists of an operator and a comparison value. If you do not include an operator, Access assumes you want an "equals" (=) comparison. You can specify multiple comparisons separated by the Boolean operators OR and AND.

It is good practice to always enclose text string values in quotation marks. If one of your values is a text string containing blanks or special characters, you must enclose the entire string in quotation marks. For example, to limit the valid entries for a City field to two of the largest cities in the state of Washington, enter "**Seattle**" Or "**Tacoma**". If you are comparing date values, you must enclose the date constants in pound sign (#) characters, as in #07/1/2010#.

You can use the comparison symbols to compare the value in the field to a value or values in your validation rule. Comparison symbols for web databases are summarized in Table 6-4. (Unlike client databases, you cannot use the Between operator in validation rules in web databases.) For example, you might want to ensure that a numeric value is always a positive number. To do this, enter **>0**. You can use one or more pairs of comparisons to ask Access to check that the value falls within certain ranges. For example, if you want to verify that a number is in the range of 100 through 200, enter either **>=100 And <=200**. Another way to test for a match in a list of values is to use the IN comparison operator. For example, to test for states surrounding California, enter **In ("Oregon", "Nevada", "Arizona")**. If all you need to do is ensure that the user enters a value, you can use the special comparison phrase Is Not Null.

**Table 6-4** Comparison Symbols Used in Validation Rules for Web Databases

Operator	Meaning
NOT	Use before any comparison operator except IS NOT NULL to perform the converse test. For example, NOT > 10 is equivalent to <=10.
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater than or equal to.
=	Equal to.
<>	Not equal to.
IN	Test for <i>equal to</i> any member in a list; comparison value must be a comma-separated list enclosed in parentheses.
LIKE	Test a Text field to match a pattern string.
IS NOT NULL	Requires the user to enter a value in the field.

If you need to validate a Text field against a matching pattern (for example, a postal code or a phone number), you can use the LIKE comparison operator in web databases. You can provide a text string as a comparison value that defines which characters are valid in which positions. Access understands a number of *wildcard characters*, characters which you can use to define positions that contain any single character, zero or more characters, or any single number. These characters are shown in Table 6-5 and are identical to what you can use for client databases.

**Table 6-5** LIKE Wildcard Characters

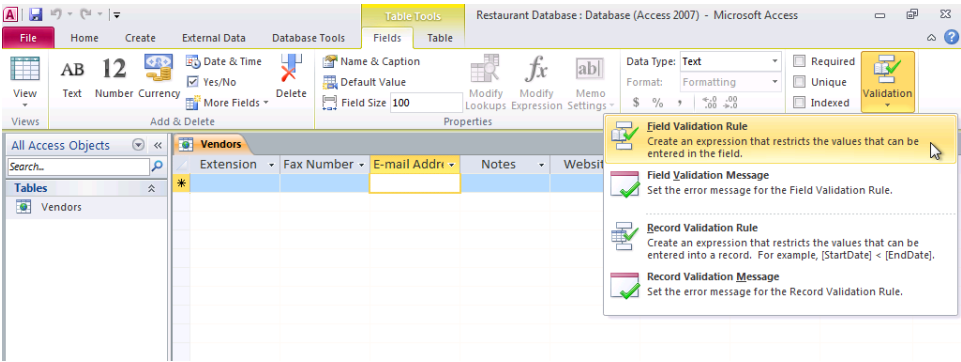
Character	Meaning
?	Any single character.
*	Zero or more characters; use to define leading, trailing, or embedded strings that don't have to match any specific pattern characters.
#	Any single digit.

You can also specify that any particular position in the Text field can contain only characters from a list that you provide. You can specify a range of characters within a list by entering the low value character, a hyphen, and the high value character, as in [A-Z] or [3-7]. If you want to test a position for any characters *except* those in a list, start the list with an exclamation point (!). You must enclose all lists in brackets ([ ]). You can see examples of validation rules using LIKE in web databases here.

Validation Rule	Tests For
LIKE "#####" or	A U.S. 5-digit ZIP Code
LIKE "#####-####"	A U.S. 9-digit ZIP+ Code
LIKE "[A-Z]#[A-Z] #[A-Z]#"	A Canadian postal code
LIKE "###-##-####"	A U.S. Social Security number
LIKE "Smith*"	A string that begins with <i>Smith</i> <sup>1</sup>
LIKE "*smith##"	A string that contains <i>smith</i> followed by two numbers, anywhere in the string
LIKE "??00####"	An eight-character string that contains any first two characters followed by exactly two zeros and then any four digits
LIKE "[!0-9BMQ]*####"	A string that contains any character other than a number or the letter <i>B</i> , <i>M</i> , or <i>Q</i> in the first position and ends with exactly four digits

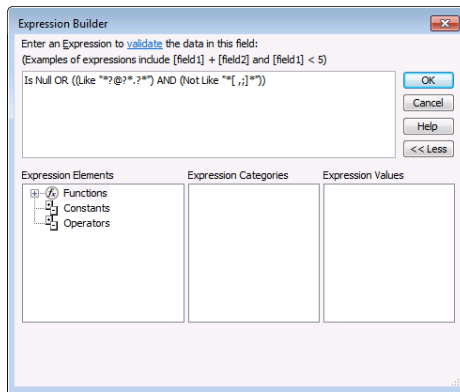
1 Character string comparisons in Access and on the server are case-insensitive, so *smith*, *SMITH*, and *Smith* are all equal.

In the Vendors table of the Restaurant Database you've been building, the EmailAddress field could benefit from the use of a validation rule. Open the Vendors table in Datasheet view, if it isn't already open, tab over to the EmailAddress field, click the Validation button in the Field Validation group, and then click the Field Validation Rule option from the drop-down list, as shown in Figure 6-33.



**Figure 6-33** Click the Validation button on the Fields contextual tab to add or edit field validation rules.

Access opens the Expression Builder dialog box, as shown in Figure 6-34. In the EmailAddress field, we want to be sure the email address provided by the user appears to be a valid email address. We can verify the email address meets most standards of valid syntax by using a combination of the LIKE operator and wildcard characters in a field validation rule. In the blank text box at the top of the Expression Builder dialog box, type Is Null OR ((Like "\*"@?\*.?\*"") AND (Not Like "\*"[,;]\*")) for the field validation rule, as shown in Figure 6-34. This field validation rule ensures that every email address provided by the user starts with at least one character followed by the @ symbol, contains at least one more character following the @ symbol, and contains the dot symbol followed by at least one more character after the dot symbol. Also, this field validation rule does not allow a space, a comma, or a semicolon anywhere in the email address.



**Figure 6-34** Type your field validation rule into the Expression Builder dialog box.

Click OK to save your changes to the field validation rule and dismiss the Expression Builder dialog box. You should now add an appropriate custom validation text to display to users if they provide data to the EmailAddress field that does not pass your new field validation rule. Make sure the focus is still in the EmailAddress field, click the Validation button in the Field Validation group, and then click the Field Validation Message option from the drop-down list, as shown in Figure 6-35. If you look closely at Figure 6-35, you'll notice that Access adds an orange square around the graphic next to the Field Validation Rule option. Access highlights these graphics on the drop-down list as a visual cue if you define a property for that specific option.

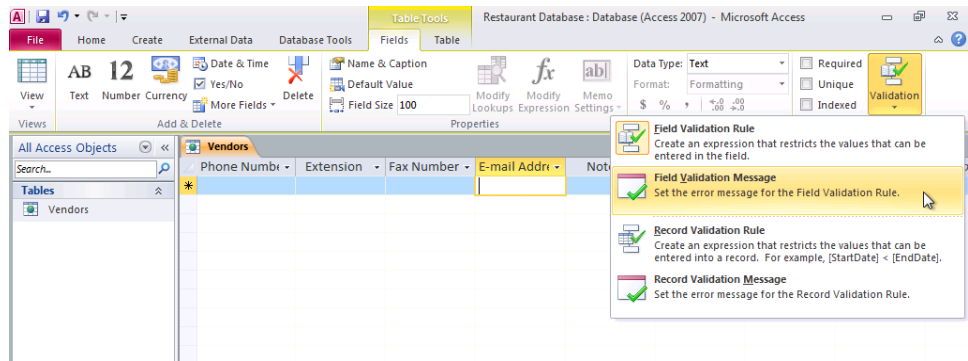


Figure 6-35 Click the Field Validation Message option to add a custom field validation text.

Access now opens the Enter Validation Message dialog box, as shown in Figure 6-36. Type the following custom message into the dialog box—**The email address you provided does not appear to be valid.** Click OK to save your changes to this property and dismiss the dialog box. You now have a completed field validation rule and message for the EmailAddress field that will be enforced whether you use the database in client or on the server. Be sure to click the Save button on the Quick Access Toolbar to save this latest change to your web table definition.

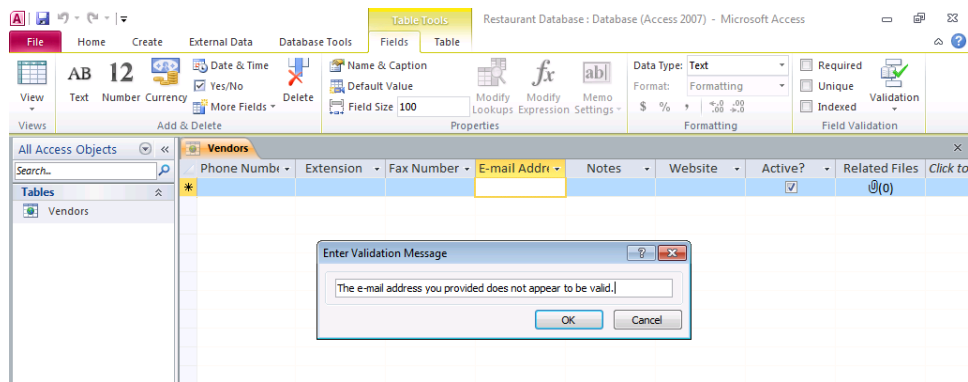


Figure 6-36 Enter your custom validation text into the Enter Validation Message dialog box.

## Defining a Table Validation Rule for Web Databases

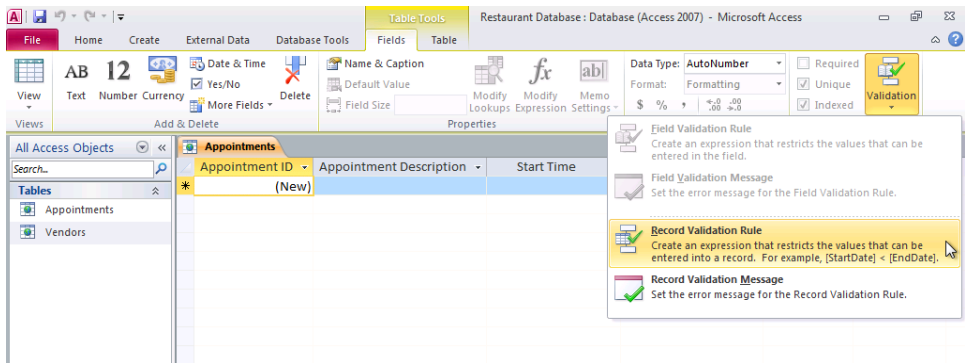
You can create table validation rules in web databases just like you can for client databases. Although field validation rules get checked as you enter each new value, Access checks a table validation rule only when you save or add a complete record. Table validation rules are handy when the values in one field are dependent on what's stored in another field. You need to wait until the entire row is about to be saved before checking one field against another.

In the Restaurant Database you have been creating, we need an Appointments table to track our day to day appointments of managing the restaurant. This table requires a table validation rule. Click the Table button in the Tables group on the Create Ribbon tab to get started. Define that table now using the specifications in Table 6-6. Be sure to rename the ID field Access provides for you to **AppointmentID** and then save the table and name it **Appointments**. Be sure to also set both the StartTime and EndTime fields as required fields by selecting the Required option in the Field Validation group. You can set the Format property for the fields in the Format text box in the Formatting group on the Fields tab.

**Table 6-6** Field Definitions for the Appointments Table

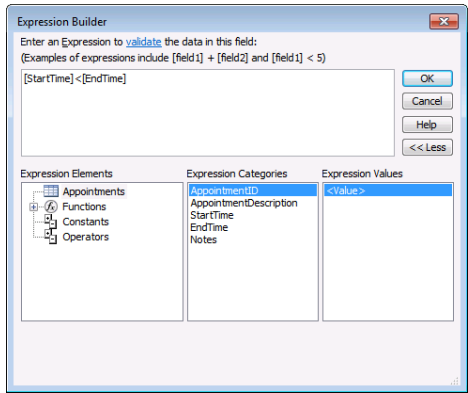
Field Name	Data Type	Description	Field Size	Format
AppointmentID	ID	Unique appointment identifier		
AppointmentDescription	Text	Description of appointment	100	
StartTime	Date/Time	Start time of appointment		General Date
EndTime	Date/Time	End time of appointment		General Date
Notes	Memo	Extended notes from appointment		Rich Text

To define a table validation rule in a web database, open the table in Datasheet view, click the Validation button in the Field Validation group on the Fields contextual tab, and then click the Record Validation Rule option from the drop-down list, as shown in Figure 6-37.



**Figure 6-37** You can define table validation rules in web databases by clicking the Record Validation Rule option.

Access opens the Expression Builder dialog box, as shown in Figure 6-38. For this web table, we want to ensure the start time of the appointment provided by the user comes before the end time. (It certainly would not make sense to have an appointment end before it even started.) We can accomplish this by using the less than (<) operator in a table validation rule. In the blank text box at the top of the Expression Builder dialog box, type **[StartTime]<[EndTime]** for the table validation rule, as shown in Figure 6-38. This table validation rule ensures that every record in the Appointments web table contains a start time that comes before the end time.

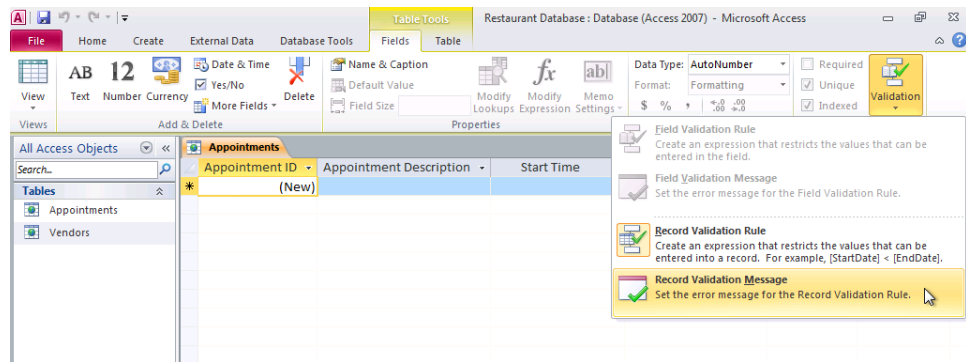


**Figure 6-38** Type your table validation rule into the Expression Builder dialog box.

Click OK to save your new table validation rule and dismiss the Expression Builder dialog box. You should now add a descriptive message to display to users if they add data to a record that violates the rule in this web table. You should be careful to word this message

so that the user clearly understands what is wrong. If you enter a table validation rule and fail to specify validation text, Access displays the following message when the user enters invalid data: "One or more values are prohibited by the validation rule '< your validation rule expression here >' set for '<table name>'. Enter a value that the expression for this field can accept." Although the message does include the complete expression, it's not very pretty, is it? If you publish this web table to the server without a validation text, the default message you'll see on the server is, "List Data Validation failed," so we recommend you always add a descriptive validation text to accompany your table validation rules.

To create a table validation message, click the Validation button in the Field Validation group, and then click the Record Validation Message option from the drop-down list, as shown in Figure 6-39. You'll notice that Access adds an orange square around the graphic next to the Record Validation Rule option to indicate that you already defined a property for that specific option.



**Figure 6-39** Click the Record Validation Message option to add a custom table validation text to your web table.

Access now opens the Enter Validation Message dialog box, as shown in Figure 6-40. Type the following custom message into the dialog box—**The End Time for the appointment cannot be before the Start Time.** Click OK to save your changes to this property and dismiss the dialog box. You now have a completed table validation rule and message for the Appointments web table. Access enforces this rule if you are adding or editing in client, and SharePoint enforces the rule after you publish the web database to the server. Be sure to click the Save button on the Quick Access Toolbar to save this latest change to your web table definition.



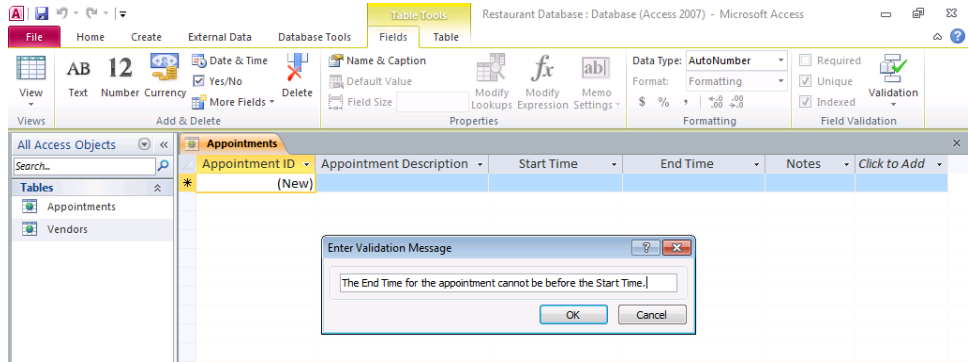


Figure 6-40 Enter your custom validation text into the Enter Validation Message dialog box.

## Defining a Primary Key for Web Databases

Every table in a relational database should have a primary key, and web databases are no exception. In Chapter 4, you learned that you have many options available to use for a primary key in client databases. In web databases, however, you must follow the structure of SharePoint lists, which only allows the SharePoint ID field as the primary key. From a certain perspective, this makes defining a primary key in web databases quite simple—Access automatically creates the primary key field, the ID field, for you whenever you create a new web table. You cannot delete this field from your web table, but you can rename the ID field to something more to your liking, such as VendorID or InvoiceID.

If you're an experienced Access developer, you might find the ID primary key restriction to be very limiting. What if you want to create a multi-field primary key and set it to be unique, as you can for client databases? While it's true you cannot define a unique multi-field primary key in SharePoint lists, you can achieve the same functionality by using table events, which is a new feature in Access 2010. In Chapter 7, "Creating Table Data Macros," you'll learn how to create a data macro that can give you the functionality of a multi-field unique primary key for a web table.

## Understanding Other Web Table Properties

Access 2010 provides several additional web table properties that you can set in Datasheet view. To edit these properties, open in Datasheet view the Appointments web table you finished creating in the previous section. Click the Table Properties button in the Properties group on the Table contextual tab, as shown in Figure 6-41. Note that we'll discuss the other commands on this contextual tab in Chapter 7, when we discuss data macros.

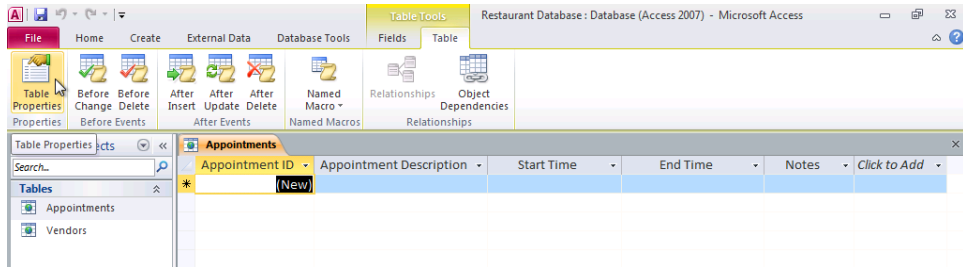


Figure 6-41 Click Table Properties to view the table properties for web tables.

Access opens the Enter Table Properties dialog box, as shown in Figure 6-42. You can use the Order By property to define one or more fields that define the default display sequence of rows in this web table when in Datasheet view. If you don't define an Order By property for web tables, Access displays the rows in primary key sequence. The Filter By property lets you predefine criteria to limit the data displayed in the Datasheet view of this table.

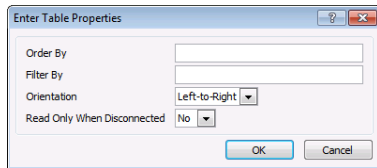


Figure 6-42 You can edit web table properties by using the Enter Table Properties dialog box.

You can use the Orientation property to specify the reading sequence of the data of your web table in Datasheet view. The default in most versions of Access is Left-to-Right. In versions that support a language that is normally read right to left, the default is Right-to-Left. When you use Right-to-Left, field and table captions appear right-justified, the field order is right to left, and the tab sequence proceeds right to left.

The Read Only When Disconnected property by default is set to No, which means you can still update or add new records to a web table that is linked to a Microsoft SharePoint Services site when you are offline. We'll discuss working with data in an Access Services application in Chapter 22.

## Note

If you apply a filter or specify a sorting sequence when you have the web table open in Datasheet view, Access 2010 saves the filter in the Filter By property and the sorting sequence in the Order By property. However, Access does not use the Filter By property when you open a web table datasheet in web databases because the Filter On Load table property is set to No for web databases. To apply the filter to your datasheet, click the Toggle Filter button in the Sort & Filter group on the Home tab.

# Creating Lookup Fields in a Web Database

If you've been following along in this chapter, creating the Vendors web table in your Restaurant Database, you'll remember we are still missing one more field found in the tblVendors web table in the Back Office Software System sample web database on the companion CD—the State field. If you open the tblVendors web table in the BOSSDataCopy.accdb database in Datasheet view and tab to the State field, you'll notice that Access displays a drop-down list of states, as shown in Figure 6-43.

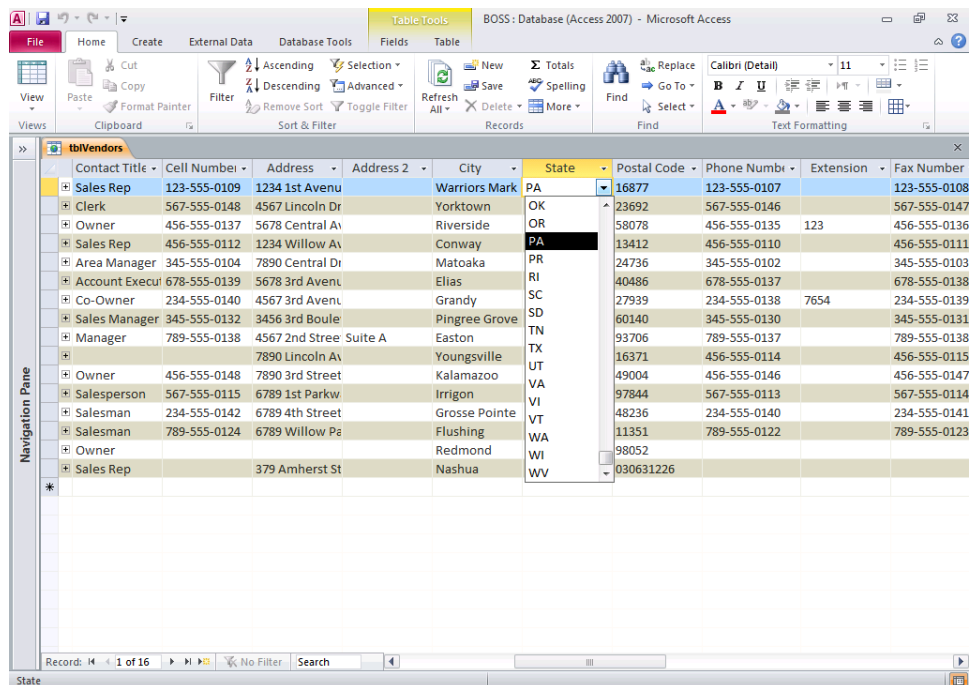
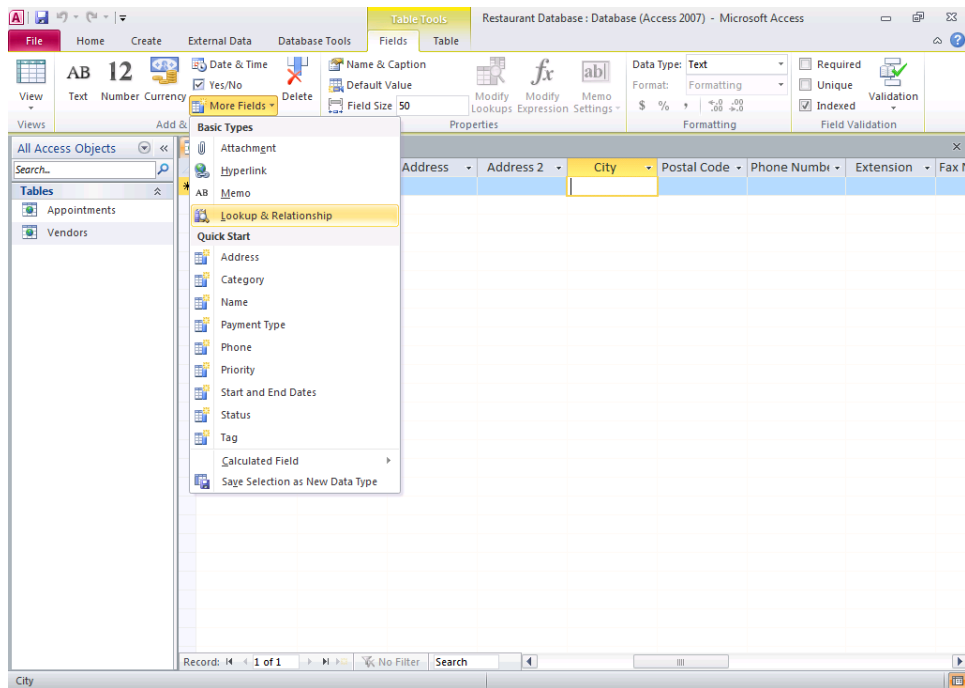


Figure 6-43 The State field in the tblVendors web table is a Lookup Field that displays a list of state abbreviations.

We defined the State field in this web table to be a lookup field. A lookup field can generally be classified as one of two types—a field that “looks up” its data from a predefined list stored with the field itself, or a field that looks up its data from a different table. The State field in the tblVendors web table looks up its data from a pre-defined list we provided when we created the field. Microsoft also uses the term *value list* to describe this type of field because the field gets its data from a list of values.

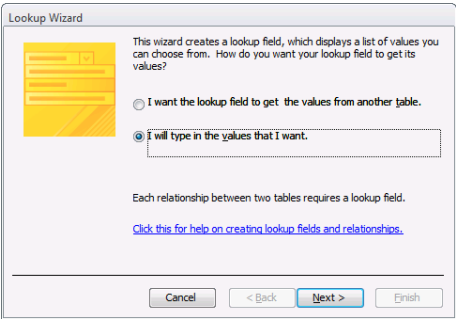
Let’s create this State lookup field in the Vendors table that you’ve been working on. Open the Restaurant Database file again if you closed it and then open the Vendors web table in Datasheet view. Next, tab over to the City field so the focus is in that field. (Remember that Access creates the new field to the right of the field that currently has focus.) Now click the More Fields button in the Add & Delete group on the Fields contextual tab, and then click Lookup & Relationship from the drop-down list of options, as shown in Figure 6-44.



**Figure 6-44** Click Lookup & Relationship under the More Fields option to start creating your lookup field.

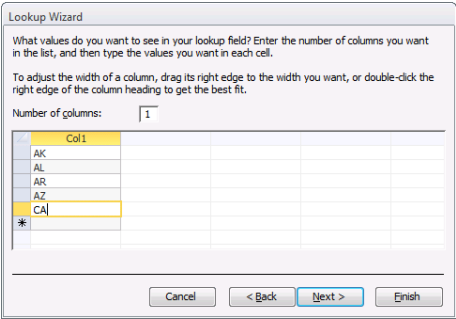
Access opens the Lookup Wizard, shown in Figure 6-45, and displays the first page. You must use this wizard if you want to create lookup fields in web databases. The first page of the wizard needs to know where you want to get the values for the field. You can either

choose to have the values come from another table or type in the values yourself. (We'll discuss the first option in the next section of this chapter.) Select the second option—I Will Type In The Values That I Want—and then click Next to proceed to the next page of the wizard.



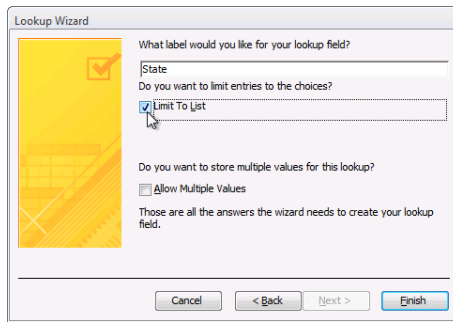
**Figure 6-45** The Lookup Wizard walks you through the steps necessary to create a lookup field for your web database.

The second page of the Lookup Wizard, shown in Figure 6-46, now needs to know specifically what values you want displayed for this field. By default, Access shows one column in the drop-down list of choices for the lookup field. If you want to display more than one column, enter the number of columns you want to display in the Number Of Columns text box. In the bottom half of this page, Access displays a datasheet where you can type in the specific values you need—one per row. In Figure 6-46, you'll notice we typed in the first six state abbreviations in alphabetical order. You can resize the column by dragging its right edge to the size you want or you can double-click the right edge of the column and Access adjusts the width to just fit the data you provide. Enter several state abbreviations now and then click Next to proceed to the next page of the wizard.



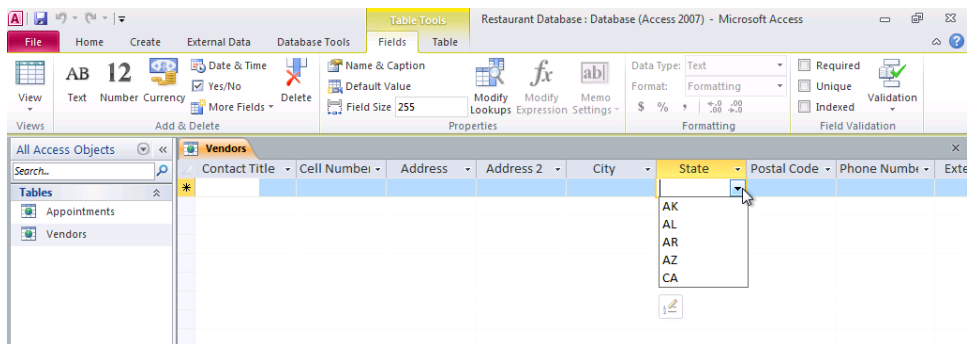
**Figure 6-46** Type in the values you want to see displayed for your lookup field on the second page of the Lookup Wizard.

Access now displays the final page of the Lookup Wizard, as shown in Figure 6-47. On this page, you can provide the name you want to use for this lookup field. Type **State** as the name of your new field in the text box. Select the Limit To List check box—cleared by default—if you want Access to not allow users to enter any other data into this lookup field other than the values you provided. The last option—Allow Multiple Values—tells Access to create a Multi-Value Lookup Field. In Chapter 5, “Modifying Your Table Design,” you learned that Multi-Value Lookup Fields handle complex data. If you select this option, Access allows you to select multiple options from the drop-down list of choices you provided. In our case, it does not make sense to have a vendor address with more than one state abbreviation, so leave this option cleared.



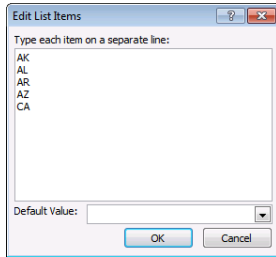
**Figure 6-47** On the last page of the Lookup Wizard, you can name your field, decide if you want to restrict the choices, and decide if you want to store multiple values.

Click Finish to save your changes and dismiss the Lookup Wizard. Access creates your new State field to the right of the City field. When you tab or click into the State field, Access displays a down arrow on the right edge of the field. When you click that arrow, Access displays all the state abbreviations you typed into the Lookup Wizard, as shown in Figure 6-48.



**Figure 6-48** Your completed lookup field now displays the list of values you provided in the Lookup Wizard.

In Figure 6-48, you'll notice that just below the drop-down list of state abbreviations is the Edit List Items button. Click this button, and Access opens the Edit List Items dialog box, as shown in Figure 6-49. In this dialog box, you can add, edit, or delete items from this value list lookup field. At the bottom of the dialog box, you can also select a default value to use for this field for all new records.



**Figure 6-49** You can edit the values in your lookup field in the Edit List Items dialog box.

In addition to using the Edit List Items dialog box to change the values of your lookup field, you can also make changes to the values and settings by clicking the Modify Lookups button in the Properties group on the Fields contextual tab. Access reopens the Lookup Wizard, where you can make adjustments to your settings and save the changes. Be sure to click the Save button on the Quick Access Toolbar to save this latest change to your web table definition.

## INSIDE OUT

### Value List Lookup Fields Create Choice Data Types in SharePoint

When you create a value list lookup field in a web database and then publish it to a SharePoint server, Access creates a Choice data type in the SharePoint list for that field. If you create a multi-value lookup field in a web database and publish it to the server, Access creates a Choice data type with the Checkboxes display property set to True to allow multiple selections.

## Creating Relationships Using Lookup Fields

The process of creating relationships between tables in web databases is very different from creating relationships in client databases. In client databases, you generally create all the tables and fields you need and then create relationships between the various tables using the Relationship Window. In web databases, however, you do both of these steps at the

same time through the Lookup Wizard. What this means to you as an Access developer is that you cannot fully complete child tables before you complete the parent tables. For example, in a client database, you could create a vendor field in a table of invoices to hold the vendor ID even before you created the vendor table itself. After you create the vendors table, you could then link the two tables on the appropriate fields using the Relationship Window. In a web database, the vendor table must exist before you can actually create fields in child tables that you intend to link to the vendor table. You cannot add relationships to existing fields in web databases; you must create the relationship at the time you create the field.

One other important difference between relationships in client and web databases is that client databases allow you to create relationships between tables with data types other than number. You can, for example, create a relationship in a client database between two tables using a Text data type (so long as both fields are defined as Text data types). In a web database, you can create relationships only on the SharePoint ID field.

In versions of SharePoint before SharePoint 2010, you could not create relationships between SharePoint lists. In SharePoint 2010, Microsoft added the ability to create relationships between different lists using the ID field. To ensure that the relationships you create in a web database are compatible with the SharePoint relationship structures, you must use the Lookup Wizard in Access 2010 to create or modify your web table relationships.

Thus far in this chapter, you have seen how to build one of the main subject tables of the Back Office Software System web database—Vendors. You also created a generic Appointments table in your Restaurant Database file. To show you how to create relationships in web databases, we first need to create another main subject table—Report Groups—and then a parent and child table—Invoice Headers and Invoice Details—to track the invoices that list all our food purchases. We'll first create all the fields for these three tables and then add the *linking* field last. Table 6-7 shows you the fields you need to create for the Report Groups table that hold the information for the report groups we use to track all the various expenditures for the restaurant.

**Table 6-7** Field Definitions for the Report Groups Table

Field Name	Data Type	Description	Field Size
ReportGroupID	ID	ID Field (AutoNumber)	
ReportGroupName	Text	Report group name	50
AccountNumber	Text	Account number of report group	50
AccountDescription	Text	Optional description for additional information	255

The Report Groups main table has all the fields we need, but the Invoice Details table depends on this table, so you need to create this table first. After you define all the fields, save the table as ReportGroups.



Table 6-8 shows you the fields you need to define for the Invoice Headers table that holds the parent information about each invoice the restaurant receives.

Table 6-8 Field Definitions for the Invoice Headers Table

Field Name	Data Type	Description	Field Size	Format
InvoiceID	ID	ID Field (AutoNumber)		
InvoiceDate	Date/Time	Date of the invoice		Short Date
InvoiceNumber	Text	Invoice number shown on invoice	50	
InvoiceAmount	Currency	Total invoice amount		Currency
Comments	Memo	Any additional comments		Rich Text
Attachments	Attachment	Invoice specific files		
IsBalanced	Yes/No	Invoice balanced?		

The Invoice Headers table needs to know which vendor this invoice came from. We'll create that field and relationship to the Vendors table in just a moment. Save this table as InvoiceHeaders after you create the necessary fields.

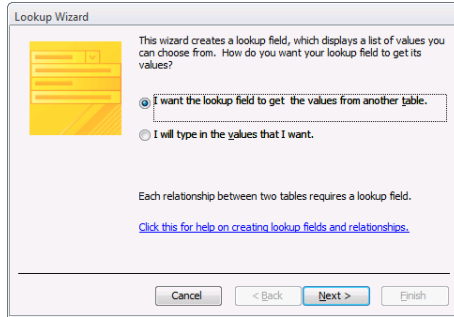
You need one last table, the Invoice Details table, to track the invoices for Restaurant Database. Table 6-9 shows the fields you need to create. This table needs the InvoiceID from the Invoice Headers table and the ReportGroupID from the Report Groups table to track the all the line items from the invoice. We'll create those fields in just a moment. Save this last table as InvoiceDetails.

Table 6-9 Field Definitions for the Invoice Details Table

Field Name	Data Type	Description	Format
InvoiceDetailsID	ID	ID Field (AutoNumber)	
ReportGroupAmount	Currency	Amount for this report group	Currency

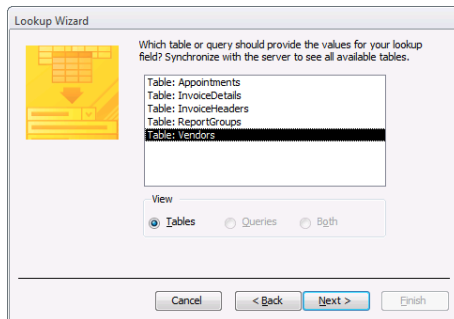
## Defining a Restrict Delete Relationship

Now, you're ready to start defining relationships between these web tables. Each vendor in our Restaurant Database can have more than one invoice. This means Vendors and InvoiceHeaders have a one-to-many relationship. Before you begin, make sure the Vendors table is closed. When you use the Lookup Wizard, Access needs to lock both tables in order to create the new lookup field. If you have the Vendors table open in Datasheet view and try to create a lookup field that uses that table as its source, you'll get an error when Access tries to create the field during the last step. To create the relationship you need, open the InvoiceHeaders table in Datasheet view and place the focus in the InvoiceID field so your new field appears to the right of the ID field. Next, click the More Fields button in the Add & Delete group on the Fields contextual tab and then click Lookup & Relationship from the drop-down list of options. Access opens the Lookup Wizard dialog box, as shown in Figure 6-50.



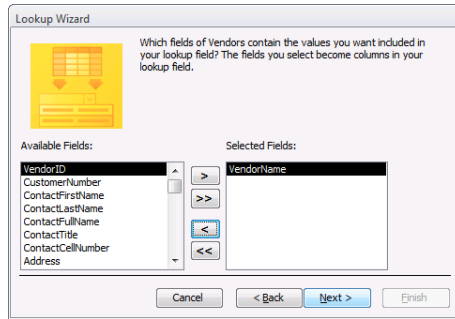
**Figure 6-50** To create a new lookup field with a relationship to another web table, you need to select the first option on this page of the Lookup Wizard.

On the first page of the wizard, Access needs to know where you want to fetch the values for this new lookup field. To create a new field that has a relationship between a different web tables, select the first option—I Want The Lookup Field To Get The Values From Another Table. Click Next to proceed to the second page of the wizard, as shown in Figure 6-51.



**Figure 6-51** Select the Vendors web table to provide a source of data for your new lookup field.

On the second page of the wizard, Access needs to know which web table you want to use to provide the values for your new lookup field. (Note that in web databases, you cannot create relationships between a table and query.) We want to store the vendor who produced the invoice in the InvoiceHeaders web table, so select the Vendors table from the list and then click Next. Access opens the third page of the wizard, as shown in Figure 6-52.



**Figure 6-52** You can select more than one column to display in your lookup field by selecting fields from the Available Fields list.

On the third page of the wizard, you can select which field or fields to display in your lookup field. You can select any field in the Available Fields list and click the single right arrow (>) button to copy that field to the Selected Fields list. You can also click the double right arrow (>>) button to copy all available fields to the Selected Fields list. If you copy a field in error, you can select the field in the Selected Fields list and click the single left arrow (<) button to remove the field from the list. You can remove all fields and start over by clicking the double left arrow (<<) button.

## INSIDE OUT

### Using Calculated Fields as Display Values for Lookup Fields

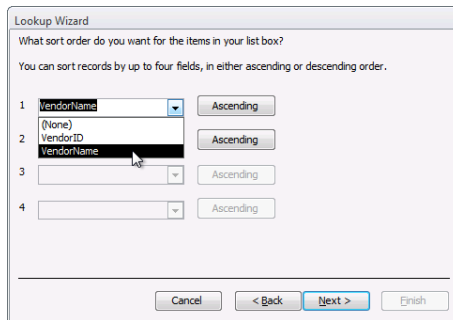
If you want to use a Calculated field as one of the display columns in your Lookup Field for a web database, you can only use Calculated fields that have Text result types. In web databases, Access does not show you any Calculated fields with result types other than Text on the third page of the Lookup Wizard dialog box to choose display columns. However, if you are using a client database, you can use Calculated fields with result types other than text for display values in a Lookup Field.

By default, Access always sets the ID field as the first column in your lookup field. You cannot change this behavior because it is a SharePoint requirement. SharePoint enforces the relationships on the server through the ID field. For your new lookup field, select the VendorName field. (The users of your database will find it much easier to choose a vendor name from a list rather than just a list of vendor ID numbers.) Click Next to proceed to the fourth page of the wizard, as shown in Figure 6-53.

## INSIDE OUT

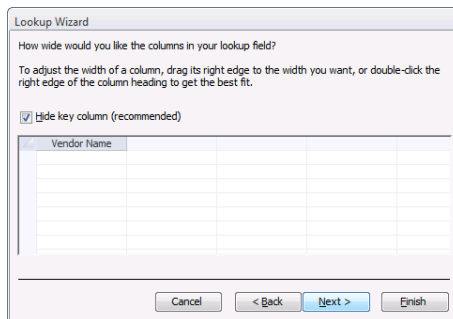
### Self Join Relationships Are Supported in Web Databases

If you want to create a self join relationship in a web database, select the same table that you are creating the new lookup field in on the second page of the Lookup Wizard. You can then select another field you want to use for your new lookup field. A self join relationship could be useful, for example, when you have a table of employees and one of the fields contains the name of the employee they report to in the organization.



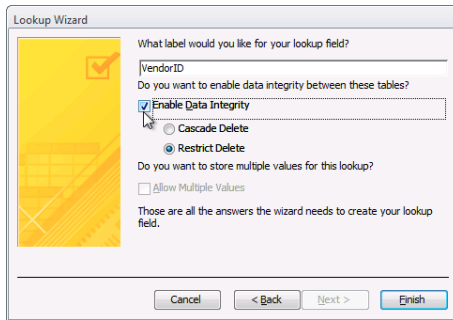
**Figure 6-53** Select the VendorName field in the first box to sort the lookup field values in Ascending order.

On the fourth page of the wizard, Access allows you to select up to four fields to sort either Ascending or Descending. Click the arrow to the right of the first field and then select the VendorName field, as shown in Figure 6-53. The button next to the first box indicates *Ascending*. If you click the button, it changes to *Descending*. For now, let's leave it as Ascending. (You can click the button again to set it back.) Click Next to go to the fifth page of the wizard, as shown in Figure 6-54.



**Figure 6-54** On this page of the wizard, you can select to hide the ID column and resize the display columns for your lookup field.

On the fifth page of the wizard, you can choose to hide the key column, which is selected by default. If you clear this option, Access displays the ID field from the Vendors table in your lookup field drop-down list. On the bottom half of the page, you can resize the Vendor Name column by dragging its right edge to the size you want, or you can double-click the right edge of the column and Access adjusts the width to fit the data. Leave these settings as they are and then click Next to proceed to the last page of the wizard, as shown in Figure 6-55. Note that you could click Finish now to exit the Lookup Wizard, but Access would not enforce referential integrity for this field because it is not the default option on the last page of the wizard.

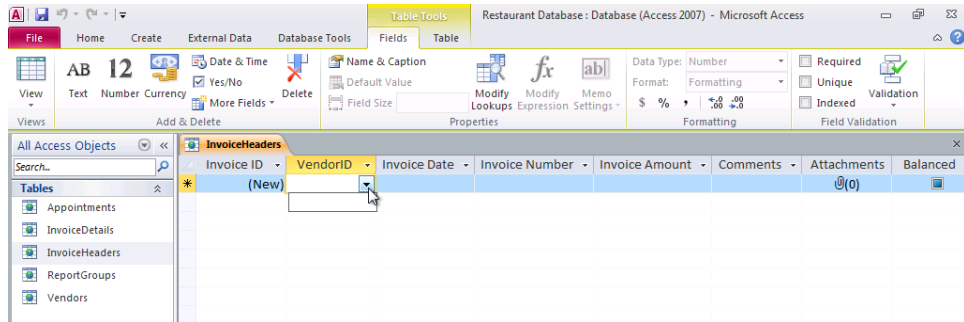


**Figure 6-55** Make sure to select the Enable Data Integrity option on the last page of the wizard to maintain referential integrity.

On the last page of the wizard, you can enter a name for your new lookup field. Type **VendorID** for the name of your new lookup field in the text box at the top of the page. Because you probably don't want any rows created in InvoiceHeaders for a nonexistent vendor, select the Enable Data Integrity check box. (Selecting this check box achieves the same functionality as selecting Enforce Referential Integrity in a client database relationship.) When you do this, Access 2010 ensures that you can't add a row in the InvoiceHeaders web table containing an invalid VendorID. Also, Access won't let you delete any records from the Vendors table if they have invoices still defined.

After you select the Enable Data Integrity check box, Access 2010 makes two additional radio buttons available: Cascade Delete and Restrict Delete. If you select the Cascade Delete check box, Access deletes child rows (the related rows in the many table of a one-to-many relationship) when you delete a parent row (the related row in the one table of a one-to-many relationship). For example, if you remove a vendor from the Vendors web table, Access removes all the related InvoiceHeader rows. If you select the Restrict Delete option (the default), Access prevents you from deleting a vendor from the Vendors table if there are invoices in the InvoiceHeaders table that use that vendor. In this design, we don't want to remove all invoices for accounting purposes so leave the default option, Restrict Delete, set.

The last option on this final page of the wizard—Allow Multiple Values—tells Access to create a Multi-Value Lookup Field. You can only select this option if the Enable Data Integrity check box is cleared. Click Finish to complete the steps necessary to create your lookup field with a relationship to the Vendors table. In Figure 6-56, you can see the completed new VendorID lookup field in the InvoiceHeaders web table. Click the Save button on the Quick Access Toolbar to save these latest definition changes.



**Figure 6-56** You can create a lookup field to the Vendors table and enforce referential integrity for the data.

If you need to modify this lookup field or its relationship, you can highlight the field in Datasheet view, and click the Modify Lookups button in the Properties group on the Fields contextual tab. Access opens the Lookup Wizard to the third page, where you can adjust the field or fields to display in your lookup field. You can adjust the settings for this lookup field on the various wizard pages from this point on and then save the changes. If you want this lookup field to use a different table as its source or use a value list as its source, you'll need to delete the lookup field and recreate a new lookup field by starting over with the Lookup Wizard.

You now know enough to define the additional one-to-many Restrict Delete relationship that you need in the InvoiceDetails web table. You need to include the ReportGroupName field from the ReportGroups web table in the InvoiceDetails web table, so open the InvoiceDetails web table in Datasheet view and set the focus on the InvoiceDetailsID field.

Follow this procedure to build your relationship:

1. Start the Lookup Wizard by clicking More Fields and then clicking Lookup & Relationship.
2. On the first page of the wizard, select the I Want The Lookup Field To Get The Values From Another Table option, and then click Next.
3. On the second page of the wizard, select the ReportGroups web table, and then click Next.

4. On the third page of the wizard, bring over the ReportGroupName field from the Available Fields list to the Selected Fields list, and then click Next.
5. On the fourth page of the wizard, select the ReportGroupName field in the drop-down list to sort Ascending by that field, and then click Next.
6. On the fifth page of the wizard, leave the default options as they are to hide the key column and keep the widths the same. Click Next to go to the final page of the wizard.
7. On the last page of the wizard, name your field **ReportGroupID** and select the Enable Data Integrity option. Leave the default option, Restrict Delete, set, and then click Finish to complete the new field and relationship.
8. Finally, save your changes to the InvoiceDetails web table.

## INSIDE OUT

### SharePoint Lists Support Only 20 Indexes

SharePoint lists support a maximum of 20 indexes, so you need to take this into account when deciding which fields to index. The SharePoint ID field and each Lookup Field you create all count toward the 20-index limit. If you attempt to publish a table with more than 20 indexes to a SharePoint site, Access stops the publish process and reports the error.

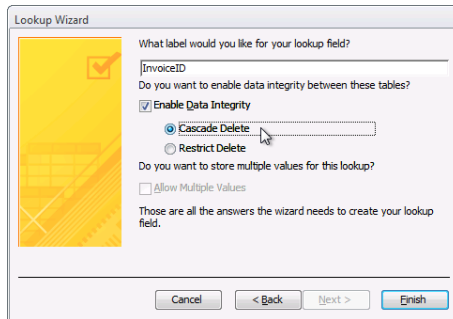
## Defining a Cascade Delete Relationship

There's one last relationship you need to define in the Restaurant Database between InvoiceDetails and InvoiceHeaders. The relationship between these two tables requires a Cascade Delete relationship. We want to ensure that when an invoice is deleted in the InvoiceHeaders web table (the *one* side of the relationship) that all corresponding child records in the InvoiceDetails web table (the *many* side of the relationship) are deleted.

Follow this procedure to build your Cascade Delete relationship:

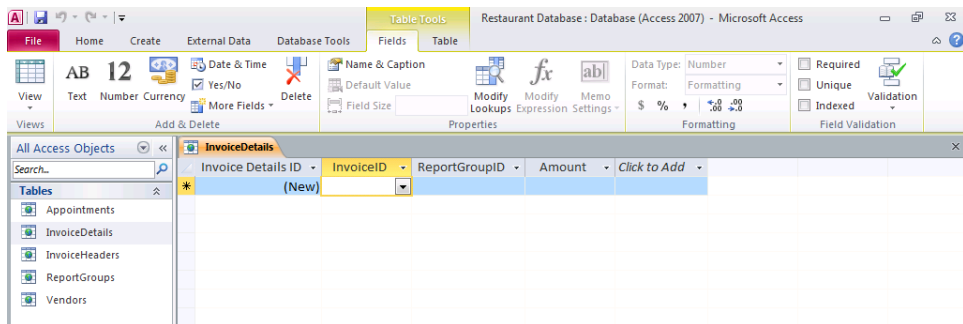
1. You want to include the InvoiceNumber field in the InvoiceDetails web table, so open the InvoiceDetails web table in Datasheet view and set the focus on the InvoiceDetailsID field. Start the Lookup Wizard by clicking More Fields and then clicking Lookup & Relationship.
2. On the first page of the wizard, select the I Want The Lookup Field To Get The Values From Another Table option, and then click Next.

3. On the second page of the wizard, select the InvoiceHeaders web table, and then click Next.
4. On the third page of the wizard, bring over the InvoiceNumber field from the Available Fields list to the Selected Fields list, and then click Next.
5. On the fourth page of the wizard, select the InvoiceNumber field in the drop-down list to sort Ascending by that field, and then click Next.
6. On the fifth page of the wizard, leave the default options as they are to hide the key column and keep the widths the same. Click Next to go to the final page of the wizard.
7. On the last page of the wizard, shown here, name your field **InvoiceID** and select the Enable Data Integrity option. Select the Cascade Delete option, and then click Finish to complete the new field and relationship.



8. Finally, save your changes to the InvoiceDetails web table.

Your completed InvoiceDetails web table should now look like Figure 6-57.



**Figure 6-57** You now have two lookup fields to two different web tables in your InvoiceDetails table.





The sample web database, called `RestaurantSample.accdb`, on the companion CD, contains all the web tables you created in this chapter if you want to compare the results of your work.

## INSIDE OUT

### Viewing Web Relationships Limitation

One of the shortcomings of creating relationships in web databases is that you cannot view the Relationships window as you can in client databases. As a result, you cannot run the Relationships Report option to get a nice graphical printout of all the relationships in your web database.

## Using the Web Compatibility Checker

Access Services does not support all the data types, relationships, properties, controls, objects, and events that the Access rich client supports. When you create a web database from scratch or use a web template, all the design surfaces you see in Access are designed to show you only the elements that are compatible with the server. Microsoft created these design surfaces for web databases in an attempt to keep Access developers from creating controls, objects, etc. that are not web-legal. It is still possible, however, to create elements in a web database that are not web-legal. If you create an element that is not web-legal and publish it to the server, Access Services does not understand what to do with it and you'll most likely see an error. In more extreme cases, you would be unsuccessful in publishing your database to the server.

To help prevent Access developers from publishing elements to the server that are not supported by Access Services, Access 2010 includes a tool called the Web Compatibility Checker. This tool checks the web compatibility of all the tables in a database as well as any web objects. This tool, however, does not check any data in the tables nor does it attempt to check any client objects other than local tables. If you attempt to run the Web Compatibility Checker on a client object other than a table, you'll actually receive no errors, which can be a bit misleading. The tool, in fact, did not even run on the client object. The Compatibility Checker also does not check any linked tables to other data sources.

You can think of the Web Compatibility Checker tool as a gatekeeper, always making sure your objects are web-legal before you publish your database to the server, import objects into a web database, or make changes to existing objects and try to send those changes to the server. If the tool finds issues during any of those actions, Access aborts the process. You'll learn all about publishing a web database to the server and syncing changes in Chapter 22.

## Analyzing the Web Compatibility Issues Table

To run the Web Compatibility Checker on the web database you created, click the File tab on the Backstage view and then click Save & Publish. Next, click Publish To Access Services and then click Run Compatibility Checker, as shown in Figure 6-58.

### Note

All database objects must be closed before the Web Compatibility Checker tool can begin to scan your database. If you have any objects open, the tool displays a message box informing you it needs to close all open objects. Click Yes in this dialog box, and Access closes all open objects and then begins to scan your database.

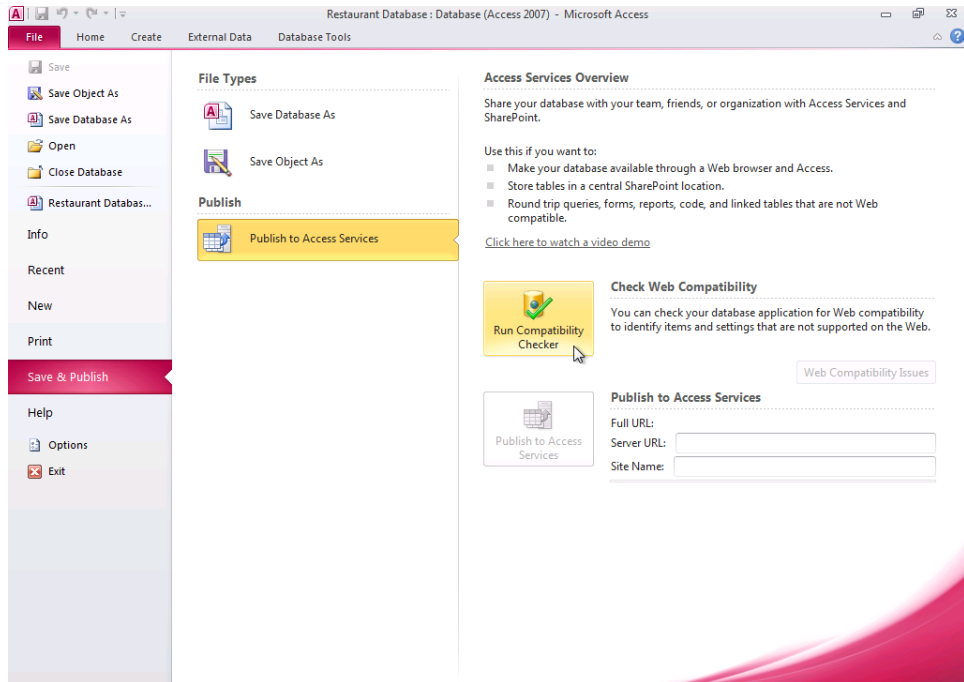
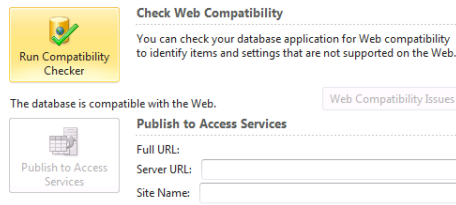


Figure 6-58 You can run the Web Compatibility Checker from the Backstage view.

The Web Compatibility Checker scans each table's schema and every web object, control, and property in your web database. In general, the Web Compatibility Checker checks your database in the following order: table schema, queries, macros, forms, reports, and then

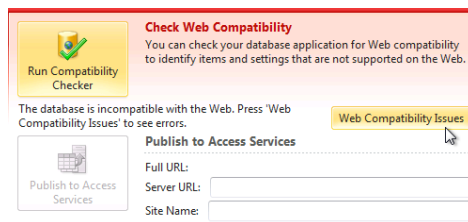
relationships. Note, however, that the order can deviate based on the dependencies of the various objects in your database.

If the Web Compatibility Checker tool finds any issues in your database, the tool creates a new table, called Web Compatibility Issues, and lists any incompatibilities found in your web database in this table. Every time you run the Web Compatibility Checker, Access deletes any existing Web Compatibility Issues table and then creates a new table. If the tool does not find any incompatibilities in your web database, Access deletes the Web Compatibility Issues table when it completes the scan of your database. Access then displays the message, “The database is compatible with the Web,” below the Run Compatibility Checker button on the Backstage, as shown in Figure 6-59.



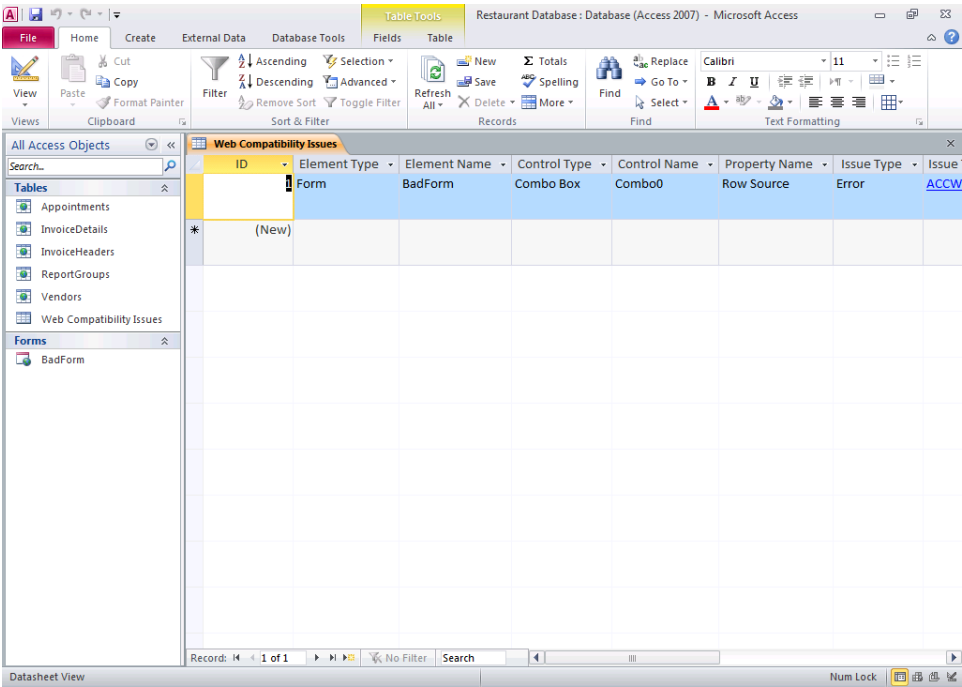
**Figure 6-59** Access displays a success message on the Backstage view if the Web Compatibility Checker did not find any issues with your web database.

If the Web Compatibility Checker finds any issues in your web database, Access changes the background color around the Run Compatibility Checker button on the Backstage to red, as shown in Figure 6-60. Access also displays the message, “The database is incompatible with the Web. Press ‘Web Compatibility Issues’ to see errors,” to indicate it found at least one issue. Finally, Access enables the Web Compatibility Issues button on the Backstage view.



**Figure 6-60** Access informs you if the Web Compatibility Checker found any issues in your web database.

Click Web Compatibility Issues on the Backstage view, and Access closes the Backstage view and opens a new table, called Web Compatibility Issues, in Datasheet view, as shown in Figure 6-61.



**Figure 6-61** The Web Compatibility Issues table list any incompatibilities that the Web Compatibility Checker found in your web database.

The Web Compatibility Issues table contains the following fields: Element Type, Element Name, Control Type, Control Name, Property Name, Issue Type, Issue Type ID, and Description. You cannot see all the information in this table in Figure 6-61, so we’ve reproduced the information in Table 6-10.

**Table 6-10** Sample Web Compatibility Issues Table Results

Field Name	Data
Element Type	Form
Element Name	BadForm
Control Type	Combo Box
Control Name	Combo0
Property Name	Row Source
Issue Type	Error
Issue Type ID	AccWeb103906
Description	The definition of the query is invalid, so the query object cannot be created.

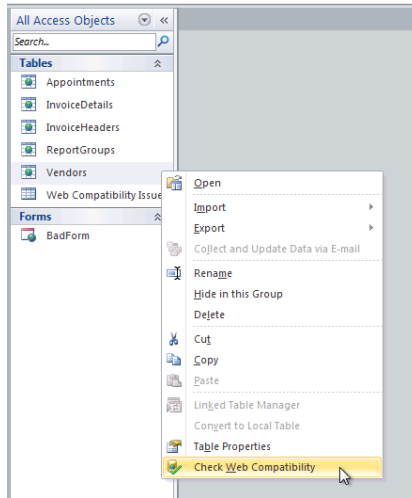
In this example, we created a new web form called BadForm. On this form, we created a new Combo Box control that has a Row Source pointing to a nonexistent web table. (You'll learn all about Combo Box controls in Chapter 12.) If you look at the data in the Web Compatibility Issues table, Access gives specific information to help you understand what you need to fix to make the database web compatible. Access lists the specific object type (a form, in this case), the object name, the control type, the specific control name (Combo0), and the specific property that is causing the error (the Row Source property). The description field lists more detailed information about the specific error.

The Issue Type ID field is a hyperlink that points to a specific error ID assigned to the particular error. When you click on the Issue Type ID hyperlink, Access directs you to a help topic for more information. The help topics concerning web compatibility issues fall into seven categories—General, Schema, Relationships and Lookups, Queries, Forms and Reports, Expressions, and Macros.

You'll need to fix all the errors listed in the Web Compatibility Issues table before your web database can be completely web compatible. You should start by fixing any errors concerning the tables in your database. The reason for fixing table errors first is because the Web Compatibility Checker tool lists an error for any object that depends on another object that is incompatible. For example, suppose you have five forms in a database that all depend on a specific table. Let's also assume that the table is incompatible with the server. When you run the Web Compatibility Checker tool, the Web Compatibility Issues table lists one error for the table as well as one error for each of the five forms because the forms depend on that table. You could be wasting your time trying to fix forms when the real reason for the error is the object the form depends on.

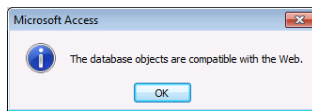
To minimize your time spent trying to fix any errors in the Web Compatibility Issues table, we recommend an iterative process—fix several errors and then run the Web Compatibility Checker tool again. Always start with fixing several table-related errors first and then run the tool again to see the result of your changes. If the tool still finds errors, fix more table issues and then run the tool again. Once you fix all the compatibility errors concerning tables, you might find that everything in your database is now web compatible. If you still have issues remaining, fix any errors concerning query objects next, because forms and reports can depend on queries, and then run the tool again. Finally, move on to fixing any errors concerning forms, reports, and macros.

You can also run the Web Compatibility Checker on a single object in your web database. Right-click a table or web object in the Navigation pane and then click Check Web Compatibility from the shortcut menu, as shown in Figure 6-62.



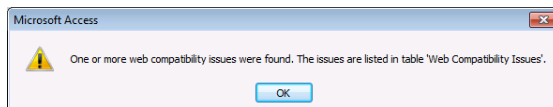
**Figure 6-62** You can run the Web Compatibility Checker tool on a single object from the short-cut menu.

If the Web Compatibility Checker finds no issues with the specific object you selected, Access displays a success message, as shown in Figure 6-63.



**Figure 6-63** The Web Compatibility Checker displays this message if no errors are found in the object you selected to scan.

If the Web Compatibility Checker does find at least one error in the object you selected, Access displays a notification dialog box indicating it found issues with the object, as shown in Figure 6-64. You'll then see a new Web Compatibility Issues table in the Navigation pane, where you can diagnose exactly what the issues are with the object.



**Figure 6-64** The Web Compatibility Checker displays this error message if issues are found in the object you selected.

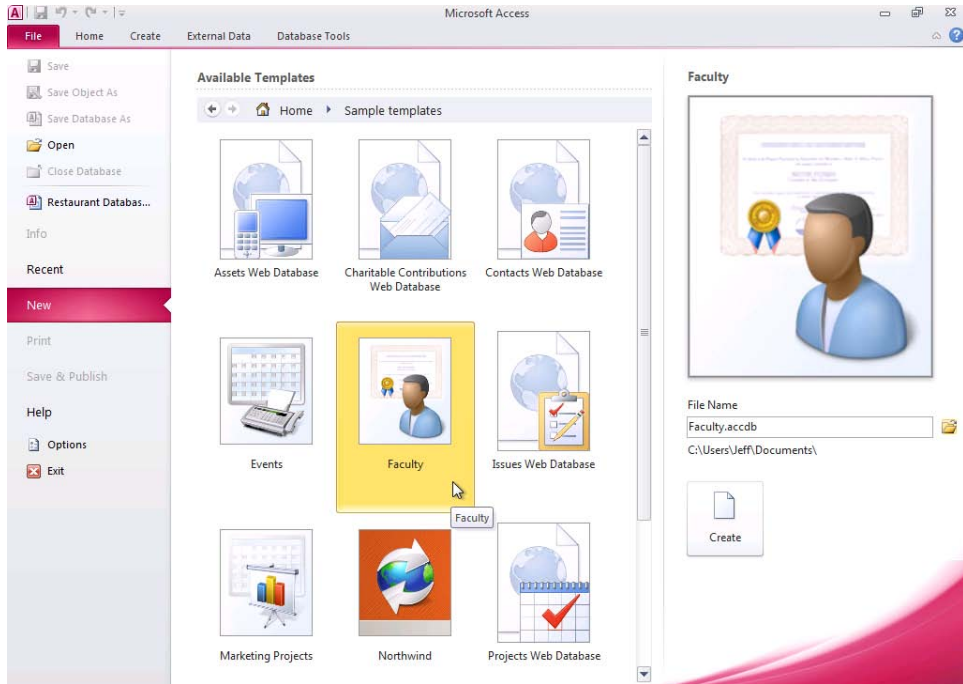
When the Web Compatibility Checker finds no errors in your web database, the tool deletes any existing Web Compatibility Issues table from the Navigation pane. You can also delete this table at any time yourself by highlighting the table in the Navigation pane and pressing Delete.

## Preparing a Client Database for the Web

If you create a new web database from scratch or use a web database template, you are less likely to encounter any errors when you run the Web Compatibility Checker tool because you start out within a confined design surface. But what happens if you have an existing client database and want to publish it to a SharePoint server running Access Services to make it a web application? If you have a client database you want to publish to the server, your first step is to run the Web Compatibility Checker.

In the previous section, we mentioned the Web Compatibility Checker checks all local tables and web objects. In a client database, this means the Web Compatibility Checker tool only checks the local tables because the other objects will remain client objects. To publish to the server, you need all the schema in your database to be web compatible. The Web Compatibility Checker tool helps you migrate existing client databases to the server by showing what changes you need to make to your tables for the database to successfully publish to a SharePoint server. (In Chapter 22, you'll learn how to publish databases to a SharePoint server.)

Let's use one of the built-in client database templates to illustrate this process. Close any databases you have open by clicking the File tab on the Backstage view and then click Close Database. Next, click the Sample Templates button on the New tab to show all the built-in local templates, as shown in Figure 6-65. Finally, click the Faculty client template in the center of the screen, select a location to save this new database, and then click Create to create the new client database.



**Figure 6-65** Click the Faculty button in the Sample Templates section to create a client database.

Access creates the new client database and displays the startup form for the database—the Faculty List form. To run the Web Compatibility Checker on your client database, click the File tab on the Backstage view and then click Save & Publish. Next, click Publish To Access Services and then click Run Compatibility Checker. Note that Access prompts you to close all open objects before it begins to scan the database. Click Yes on the dialog box prompt, and the Web Compatibility Checker tool now scans your client database. Once it completes the scan, Access reports on the Backstage view that your database is incompatible with the web, as shown in Figure 6-60. Click the Web Compatibility Issues button on the Backstage view and Access opens the Web Compatibility Issues table, shown in Figure 6-66. Note that in Figure 6-66, we resized the columns in the table so you can see all the data.



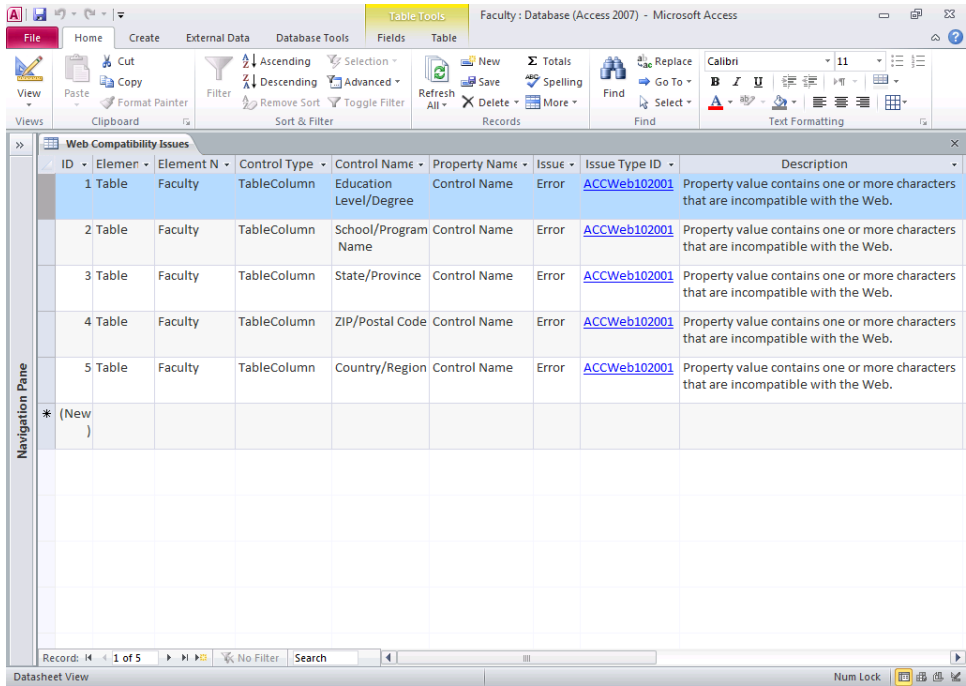


Figure 6-66 The Compatibility Checker reports any issues with tables in client databases.

The Web Compatibility Checker found five issues with the Faculty table in this client database. If you look at the Description field for each error, you'll notice that each error is identical. In this client database, the Web Compatibility Checker found five field names that are incompatible with the server. Specifically, each of these five field names contains the forward slash character (/), which is incompatible with the server. To fix these issues, you can open the Faculty table in Design view or Datasheet view and remove the forward slash character from each of the five field names listed in the Web Compatibility Issues table—Education Level/Degree, School/Program Name, State/Province, ZIP/Postal Code, and Country/Region. After you make these changes to the Faculty table and save your changes, you can run the Web Compatibility Checker tool again. You'll now receive a successful message on the Backstage view that your client database is compatible with the web. Note that because you changed field names in the table, you should verify that the rest of the application still functions.

Most existing client databases have more potential compatibility issues than just invalid column names. The Web Compatibility Checker does a good job of providing specific help topics to inform you what you need to change to make your schema web-legal. Listed below are some of the more common issues found in client databases that you might encounter when trying to prepare your database for the web:

- Incompatible relationships
- Lookup field row sources
- Primary keys that are not Number that have the Field Size property set to Long
- Composite indexes
- Custom field formats
- Subdatasheets
- Other field properties

Before making any changes to your client tables, we recommend creating a backup copy of your database in case you need to refer to the original. In general, most of the issues you'll need to fix involve adding, editing, or removing specific field properties. To fix these issues, you need to open the client table in Design view and make the appropriate changes. For errors concerning relationships, you'll need to take additional steps to fix these errors, especially if you have existing data in your client tables. First, you'll need to open the Relationships window and delete the offending relationships. Next, you'll need to make sure you have an appropriate Primary Key. The simplest solution here is to add a new AutoNumber field to your client table, if one does not already exist, and make that field the Primary Key. You should then use the Lookup Wizard to create relationships to other tables. If you have existing data in your tables, you'll also need to run some action queries to adjust the data so all the parent and child data is mapped correctly. You'll learn all about creating action queries in Chapter 11, "Modifying Data with Action Queries."

In this chapter, you've taken the first steps to learn how to create a web database. You've learned how to create a web database from scratch and create your schema and relationships between different web tables. You've also learned how to use the Web Compatibility Checker tool to verify your tables are web compatible. In Chapter 7, you'll learn how to add logic to tables by creating data macros and attaching them to table events.



# Automating a Client Application Using Macros

Uses of Macros .....	1193	Using Embedded Macros .....	1209
The Macro Design Facility—An Overview .....	1194	Using Temporary Variables .....	1216
Defining Multiple Actions .....	1201	Trapping Errors in Macros .....	1219
Working with Submacros .....	1204	Understanding Macro Actions That Are Not Trusted ..	1226
Understanding Conditional Expressions .....	1207	Making Your Application Come Alive with Macros ..	1229

In Chapter 7, “Creating Table Data Macros,” you learned about the new data macro feature in Microsoft Access 2010. Data macros are attached to table events or the table itself and interact only at the data layer. In this chapter and the next, you’ll learn about user interface macros. In Access 2010, you can define a user interface macro to execute just about any task you would otherwise initiate with the keyboard or the mouse. The unique power of user interface macros in Access is their ability to automate responses to many types of events without forcing you to learn a programming language. The event might be a change in the data, the opening or closing of a form or a report, or even a change of focus from one control to another. Within a user interface macro, you can include multiple actions and define condition checking so that different actions are performed depending on the values in your forms or reports. For the remainder of this chapter and the next, we’ll only use the term macros to refer to user interface macros.

Macros are particularly useful for building small, personal applications or for prototyping larger ones. Macros are also essential if you want to automate a web database and display the application in a web browser using Access Services and SharePoint 2010. (You’ll learn how to automate a web database in Chapter 21, “Automating a Web Application Using Macros.”) As you’ll learn in Chapter 24, “Understanding Visual Basic Fundamentals,” on the companion CD, you probably should use Microsoft Visual Basic for complex applications or for applications that will be shared by several users over a network. However, even if you think you’re ready to jump right into Visual Basic, you should study all the macro actions first. You’ll find that you’ll use nearly all the available macro actions in Visual Basic, so learning macros is an excellent introduction to programming in Access in general.

**Note**

The examples in this chapter are based on the Wedding List Macro (WeddingMC.accdb) sample client database on the companion CD included with this book. The results you see from the samples in this chapter might not exactly match what you see in this book if you have changed the sample data in the files. Also, all the screen images in this chapter were taken on a Windows 7 system with the Access color scheme set to Silver. Your results might look different if you are using a different operating system or a different theme.

In this chapter, you will

- Learn about the various types of actions you can define in macros
- Tour the new Logic Designer and learn how to build both a simple macro and a macro with multiple defined actions
- Learn how to manage the many macros you need for a form or a report by creating submacros inside a macro object
- See how to add conditional expressions to a macro to control the actions that Access performs
- Learn about embedded macros, error trapping in macros, temporary variables, and macro actions that are not trusted
- Learn how to reference other form and report objects in macros
- Understand some of the actions automated with macros in the Wedding List Macro sample client database

**Note**

In Article 6, “Macro Actions,” on the companion CD, you’ll find summaries of the client macro actions and of the web macro actions. You might find that article useful as a quick reference when you’re designing macros for your applications.

# Uses of Macros

Access 2010 provides various types of macro actions that you can use to automate your application. With macros, you can

- Open any table, query, form, or report in any available view or close any open table, query, form, or report.
- Open a report in Print Preview or Report view or send a report directly to the printer.
- Send the output data from a report to a Rich Text Format (.rtf) file or a Notepad (.txt) file. You can then open the file in Microsoft Word 2010 or Notepad.
- Execute a select query or an action query. You can base the parameters of a query on the values of controls in any open form.
- Include conditions that test values in a database, a form, or a report and use the results of a test to determine what action runs next.
- Execute other macros or execute Visual Basic functions. You can halt the current macro or all macros, cancel the event that triggered the macro, or quit the application.
- Trap errors caused during execution of macro actions, evaluate the error, and execute alternate actions.
- Set the value of any form or report control or set selected properties of forms and form controls.
- Emulate keyboard actions and supply input to system dialog boxes.
- Refresh the values in forms, list box controls, and combo box controls.
- Apply a filter to, go to any record in, or search for data in a form's underlying table or query.
- Execute any of the commands on any of the Access ribbons.
- Move and size, minimize, maximize, or restore any window within the Access workspace when you work in multiple-document interface (MDI) mode.
- Change the focus to a window or to any control within a window or select a page of a report to display in Print Preview.

- Display informative messages and sound a beep to draw attention to your messages. You can also disable certain warning messages when executing action queries.
- Rename any object in your database, make another copy of a selected object in your database, or copy an object to another Access database.
- Delete objects in your database or save an open object.
- Import, export, or attach other database tables or import or export spreadsheet or text files.
- Start an application and exchange data with the application using Dynamic Data Exchange (DDE) or the Clipboard. You can send data from a table, query, form, or report to an output file and then open that file in the appropriate application. You can also send keystrokes to the target application.

Consider some of the other possibilities for macros. For example, you can make moving from one task to another easier by using command buttons that open and position forms and set values. You can create very complex editing routines that validate data entered in forms, including checking data in other tables. You can even check something like the customer name entered in an order form and open another form so that the user can enter detailed data if no record exists for that customer.

## The Macro Design Facility—An Overview

In Chapter 7, you learned that Microsoft redesigned the macro design window in Access 2010 for Access developers to be more productive when creating macros in their applications. Although functional, the older macro design window was not the most intuitive interface to work with. For example, when you wanted to create a new action for a macro, you selected the action in the upper part of the window, but you entered the arguments in the bottom part of the window. The Arguments column in the upper part of the window was read-only, so you constantly needed to switch from the top window to the bottom window. You also could not create complex nested logic conditions for your macro execution and the readability of the macro elements was difficult to follow.

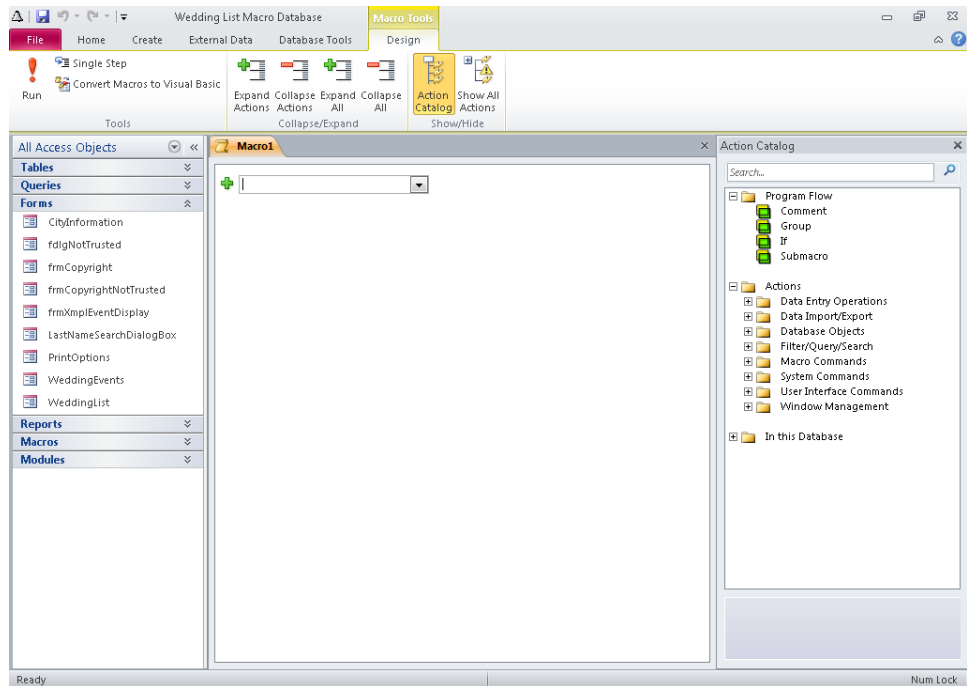
The following sections explain how to work with the macro design facility in Access 2010 in the context of user interface macros.

### Working with the Logic Designer

Open the Wedding List Macro sample database (WeddingMC.accdb) from the folder where you installed the sample files. (The default location is \Microsoft Press\Access 2010 Inside

Out on your C drive.) As you'll discover later in this chapter, a special macro called Autoexec runs each time you open the database. This macro determines whether the database is trusted, opens an informational form for a few seconds, and then tells you which macro to run to start the application. We'll look at that macro in some detail later.

On the Create tab, in the Macros & Code group, click Macro. Access opens the new Logic Designer for creating macros, as shown in Figure 20-1.



**Figure 20-1** This is the new Logic Designer where you create data and user interface macros.

Whenever you need to create or edit data macros or user interface macros in Access 2010, this is the design surface that you use. When you're working with user interface macros, Access enables the commands in the Tools group and displays the macro actions available for user interface macros in the Action Catalog. You'll also notice that Access did not collapse the Navigation pane when you clicked the Macro command on the ribbon, as it does when you are working with data macros. When you are working with user interface macro objects—macros displayed in the Navigation pane—Access does not open the Logic Designer window modally, which means you can open other database objects while working on your macro. (You'll learn later in this chapter that Access opens the Logic Designer modally when you're creating embedded macros.)

As you can see in Figure 20-1, the new Logic Designer layout looks more like a Visual Basic code window. The Expand Actions, Collapse Actions, Expand All, and Collapse All buttons in the Collapse/Expand group selectively expand or collapse the actions listed in the macro designer surface. In the Show/Hide group on the Design tab, you can choose to hide the Action Catalog, shown on the right side of the Logic Designer window, by clicking the Action Catalog toggle button. When you're working with macro objects, Access does not display the Close group on the Design tab as it does for data macros. When you want to save your macro changes, you can click the Save button on the Quick Access Toolbar or press Ctrl+S. Note that if you attempt to close the Logic Designer window with unsaved changes, Access prompts you and asks if you want to save your changes before closing the window.

On the right side of the Logic Designer window is the Action Catalog. The Action Catalog shows a contextual list of the program flow constructs and macro actions that are applicable for user interface macros. If you are working with a client macro, Access displays macro actions available for client macros. Similarly, when you're working with a web macro, Access displays macros actions available for web macros. (We'll discuss the Action Catalog in more detail in the next section, and we'll discuss web macros in Chapter 21.)

## INSIDE OUT

### Searching the Action Catalog Also Searches the Action Descriptions

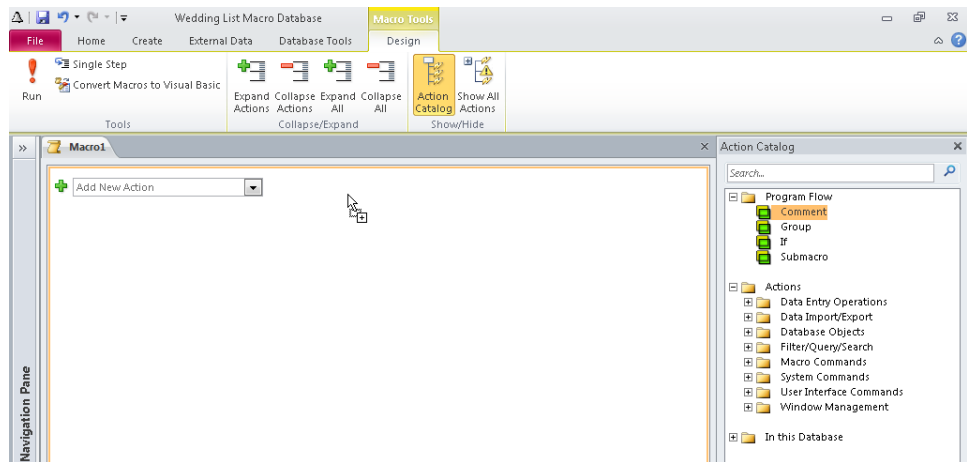
When you type search criteria into the Action Catalog Search box, Access not only looks at the action name for a possible match, but it also searches all the action descriptions for any matching text. For example, if you type **Query** into the Search box, you'll see that Access returns other actions such as **ApplyFilter** and **GoToRecord**. Access shows those results because the word *query* exists in the descriptions for those actions.

In the middle of the Logic Designer window is the main macro designer surface, where you define your macro. You add program flow constructs, macro actions, and arguments to the designer surface to instruct Access what actions to take for the macro. If you have more actions than can fit on the screen, Access provides a scroll bar on the right side of the macro design surface so you can scroll down to see the rest of your actions.

In the lower-right corner of the Logic Designer window is the Help window. Access displays a brief help message in this window, which changes based on where the focus is located in the Action Catalog. (Remember: You can always press F1 to open a context-sensitive Help topic.)



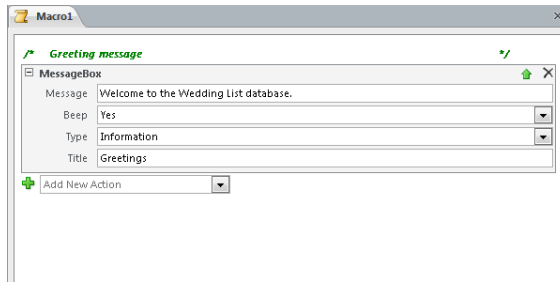
To get you accustomed to using the Logic Designer for macros, let's create a simple macro that displays a message. You can use the MessageBox action to open a pop-up modal dialog box with a message in it. This is a great way to display a warning or an informative message in your database without defining a separate form. To begin, let's add a Comment block to the macro design surface. As you learned in Chapter 7, you'll find the Comment block especially useful for documenting large macros that contain many actions. Click on the word Comments under the Program Flow node in the Action Catalog, hold the mouse key down, drag the comment onto the macro design surface, and then release the mouse button, as shown in Figure 20-2. Note that in Figure 20-2, we collapsed the Navigation pane so you can see more of the macro design surface.



**Figure 20-2** Drag the Comment program flow element from the Action Catalog onto the macro design surface.

Assume that this message will be a greeting, so click inside the Comment block and type **Greeting** message. Click outside the Comment block onto the macro design surface and Access collapses the size of the Comment block to just fit the text you typed and displays the text in green. As you learned in Chapter 7, the `/*` and `*/` symbols mark the beginning and end of a block of comments. Access designates anything written between those symbols as a comment and they are there only to provide information about the purpose of the data macro or particular action to follow.

Click in the Add New Action combo box on the macro design surface now and drop-down the list of macro actions. In the Add New Action combo box, you can specify any of the 84 client macro actions and four program flow constructs provided by Access 2010. Select MessageBox from this drop-down list. After you select an action such as MessageBox, Access displays argument boxes for the specific action you choose, as shown in Figure 20-3, in which you enter the arguments for the action.



**Figure 20-3** Enter arguments for a MessageBox action to display a greeting message.

### Note

In Access 2010, Microsoft renamed the MsgBox action from previous versions of Access to MessageBox. If you have existing databases that use the MsgBox action, Access 2010 can still understand and execute the action.

You use the Message argument box to set the message that you want Access to display in the dialog box you're creating. The setting in the Beep argument box tells Access whether to sound a beep when it displays the message. In the Type argument box, you can choose a graphic indicator, such as a red critical icon, that will appear with your message in the dialog box. In the Title argument box, you can type the contents of your dialog box's title bar. Use the settings shown in Figure 20-3 in your macro.

## TROUBLESHOOTING

### Why doesn't the list include all the available macro actions?

Access 2010 includes 86 client macro actions, but not all these actions can run in a database that is not trusted. By default, Access displays only the macro actions that can run in a trusted database in the Action column. To see the complete list of client macro actions, click the Show All Actions button in the Show/Hide group on the Design tab. When you select an action that can run only in a trusted database, Access displays an exclamation point in the upper-left corner of the action block. If a macro in your application includes actions that can run only in a trusted database, your user must trust your database to be able to run the macro.

## Saving Your Macro

You must save a macro before you can run it. Click the Save button on the Quick Access Toolbar, or click the File tab on the Backstage view and then click Save. When you do so, Access opens the dialog box shown in Figure 20-4. Enter the name **TestGreeting**, and click OK to save your macro.

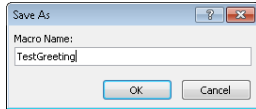


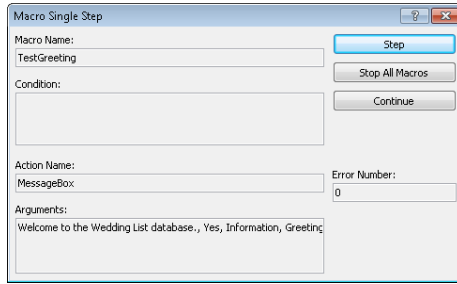
Figure 20-4 Enter a name for this test macro in the Save As dialog box.

## Testing Your Macro

You can run some macros (such as the simple one you just created) directly from the Navigation pane or from the Macro window because they don't depend on controls on an open form or report. If your macro does depend on a form or a report, you must link the macro to the appropriate event and run it that way. (You'll learn how to do this later in this chapter.) However you run your macro, Access provides a way to test it by allowing you to single-step through the macro actions.

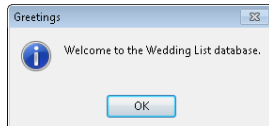
To activate single-stepping, right-click the macro you want to test in the Navigation pane and then click Design View on the shortcut menu. This opens the macro in the Logic Designer. Click the Single Step button in the Tools group on the Design tab. Now, when you run your macro, Access opens the Macro Single Step dialog box before executing each action in your macro. In this dialog box, you'll see the macro name, the action, and the action arguments.

Try this procedure with the TestGreeting macro you just created. Open the Logic Designer, click the Single Step button, and then click the Run button in the Tools group on the Design tab. The Macro Single Step dialog box opens, as shown in Figure 20-5. Later in this section, you'll learn how to code a condition in a macro. The Macro Single Step dialog box also shows you the result of testing your condition.



**Figure 20-5** The Macro Single Step dialog box allows you to test each action in your macro.

If you click the Step button in the dialog box, the action you see in the dialog box will run, and you'll see the dialog box opened by your MessageBox action with the message you created, as shown in Figure 20-6. Click the OK button in the message box to dismiss it. If your macro had more than one action defined, you would have returned to the Macro Single Step dialog box, which would have shown you the next action. In this case, your macro has only one action, so Access returns you to the Logic Designer.



**Figure 20-6** Access displays the dialog box you created by using the MessageBox action in the TestGreeting macro.

If Access encounters an error in any macro during the normal execution of your application, Access first displays a dialog box explaining the error it found. You then see an Action Failed dialog box, which is similar to the Macro Single Step dialog box, containing information about the action that caused the problem. At this point, you can click only the Stop All Macros button. You can then edit your macro to fix the problem. We'll discuss handling errors in "Trapping Errors in Macros," on page 1219.

Before you read on in this chapter, you might want to return to the Logic Designer and click Single Step again so that it's no longer selected. Otherwise, you'll continue to single-step through every macro you run until you exit and restart Access or click Continue in one of the Single Step dialog boxes.

## Defining Multiple Actions

In Access 2010, you can define more than one action within a macro, and you can specify the sequence in which you want the actions performed. The Wedding List Macro database contains several examples of macros that have more than one action. Open the database if it is not open already. Click the Navigation menu at the top of the Navigation pane, click Object Type under Navigate To Category, and then click Macros under Filter By Group to display a list of macros available in the Wedding List Macro database. Right-click the macro named AutoexecXmpl, and then click Design View on the shortcut menu to open the Logic Designer. Figure 20-7 shows the macro.

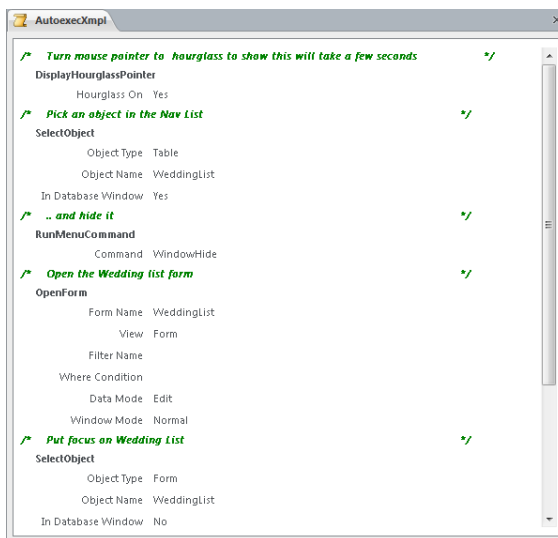


Figure 20-7 The AutoexecXmpl macro defines multiple actions that Access executes when you run the macro.

## INSIDE OUT

### Autoexec Macros—A Special Type of Macro

If you create a macro and name it Autoexec, Access runs the macro each time you open the database in which it is stored. The preferred method to run start-up code is to define a start-up form in the Application Options section of the Current Database category in the Access Options dialog box. For details, see Chapter 26, “The Finishing Touches,” on the companion CD.

In the Wedding List Macro sample database, the Autoexec macro examines the `IsTrusted` property of the `CurrentProject` object (`CurrentProject` defines all the executable code) to see whether the database is trusted. (You can trust the database by placing it in a trusted location.) If the database is not trusted, the macro opens a dialog form with instructions on how to create a trusted folder. If you open this database in a trusted location, the macro displays a copyright form followed by a message box telling you to run the `AutoexecXmpl` macro to start the application.

If this macro were named *AutoExec*, Access would execute each action automatically whenever you open the database. This sample macro is an example of a macro that you might design to start the application when the user opens your database.

We defined eight actions in this macro. First, the `DisplayHourglassPointer` action displays an hourglass mouse pointer to give the user a visual clue that the next several steps might take a second or two. It's always a good idea to turn on this visual cue even if you think the next several actions won't take very long. Next, the `SelectObject` action puts the focus on a known object in the Navigation pane, and the `RunMenuCommand-WindowHide` action hides the selected window (the Navigation pane).

The next action, `OpenForm`, opens the `WeddingList` form. As you can see in Figure 20-7, the `OpenForm` client action has six arguments that you can use to define how it should work. The `Form Name` argument indicates the form you want to open. The `View` argument tells Access what view you want. (The seven choices for the `View` argument are `Form`, `Design`, `Print Preview`, `Datasheet`, `PivotTable`, `PivotChart`, and `Layout`.) You can ask Access to apply a filter to the form when it opens either by specifying the name of a query that defines the filter in the `Filter Name` argument or by entering filter criteria in the `Where Condition` argument. You can click the `Build` button on the `Where Condition` argument to open the `Expression Builder`, which can help you create the filter.

`Edit` is the default for the `Data Mode` argument, which allows the user to add, edit, or delete records while using this form. The other choices for this argument are `Add`, which opens the form in data entry mode, and `Read Only`, which opens the form but does not allow any changes to the data. The default setting for the `Window Mode` argument is `Normal`, which opens the form in the mode set by its design properties. You can override the design property settings to open the form in `Hidden` mode, as an icon in the `Icon` mode, or in the special `Dialog` mode. When you open a form hidden, the user can reveal it only by adding the `Unhide Window` command to the `Quick Access Toolbar` and then clicking the command. When you open a form in `Dialog` mode, Access does not run further actions or Visual Basic statements until you close that form.

When you select the OpenForm action block on the macro design surface, you'll see an Update Parameters hyperlink in the lower-right corner of the action block. You can use this link to pass in parameters to the OpenForm action. For example, if the form you are going to open is based on a query that requires parameters, you can use this link to display text boxes beneath the Window Mode argument, where you can set the parameters necessary for the query. We'll discuss how to pass in parameters with macros in Chapter 21.

Access doesn't always wait for one action to complete before going to the next one. For example, an OpenForm action merely starts a task to begin opening the form. Particularly if the form displays a lot of data, Access might take several seconds to load all the data and finish displaying the form. Because you're running Windows, your computer can handle many tasks at once. Access takes advantage of this by going to the next task without waiting for the form to completely open. However, because this macro is designed to maximize the WeddingList form, the form must be completely open for this to work.

You can force a form to finish opening by telling Access to put the focus on the form. This macro does so by using the SelectObject action to identify the object to receive the focus (in this case, the WeddingList form), followed by the GoToControl action to put the focus on a specific control on the form. After the GoToControl action puts the focus on the control, the MaximizeWindow action sizes the active window (the window containing the object that currently has the focus) to fit the entire screen. The final action in the macro (the DisplayHourglassPointer again) restores the mouse pointer to let the user know that the macro is finished.

### Note

Because macros might be used by inexperienced programmers, Access automatically restores Hourglass when it finishes running a macro. If it didn't do this, the mouse pointer would continue to show an hourglass. The user would think that Access is broken. However, it's good practice to always restore what you turn off, which is why the sample AutoExecXmpl macro includes Hourglass-No at the end, even though it isn't required. As you'll learn in Chapter 24, Visual Basic isn't quite so forgiving. If you turn the mouse pointer to an hourglass in a Visual Basic procedure and forget to turn it back on before your code exits, your mouse pointer will forever display an hourglass!

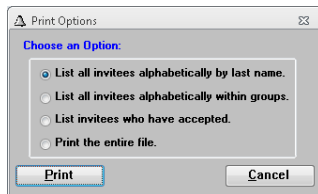
Learning to define multiple actions within a macro is very useful when you want to automate the tasks you perform on a day-to-day basis. Now that you've learned how to do this, the next step is to learn how to group actions by tasks.

## Working with Submacros

You'll find that most of the forms you design for an application require multiple macros to respond to events—some to edit fields, some to open reports, and still others to respond to command buttons. You could design a separate macro saved with its own unique name in the Database window to respond to each event, but you'll soon have hundreds of macros in your application.

You can create a simpler set of more manageable objects by defining *submacros* within named macro objects. A submacro is a named collection of macro actions inside a macro object. One approach is to create one saved macro object per form or report. Another technique is to categorize macros by type of action—for example, one macro containing all the OpenForm actions and another containing all the OpenReport actions.

Let's take a look at a form that depends on submacros. Figure 20-8 shows the PrintOptions form from the Wedding List Macro database in Form view. This form contains two command buttons, Print and Cancel, each of which triggers a different submacro. The two submacros are contained within a macro object called DoReport.



**Figure 20-8** The two command buttons on the Print Options form run submacros.

To look at the macro object, right-click the DoReport macro in the list of macro objects in the Navigation pane, and then click Design view on the shortcut menu to open this macro object in the Logic Designer. Figure 20-9 shows the macro.



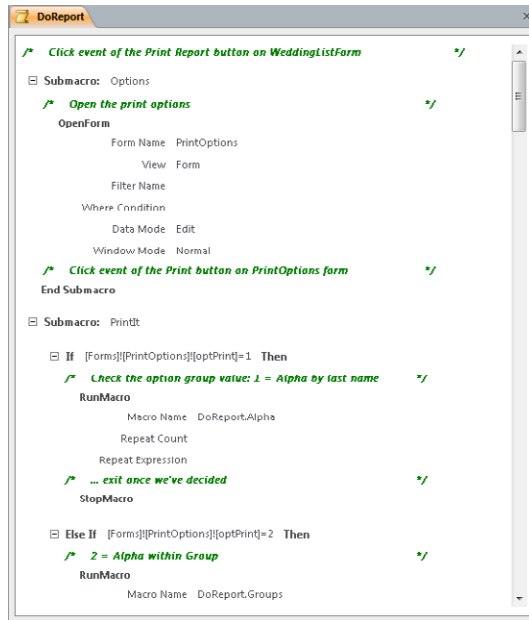


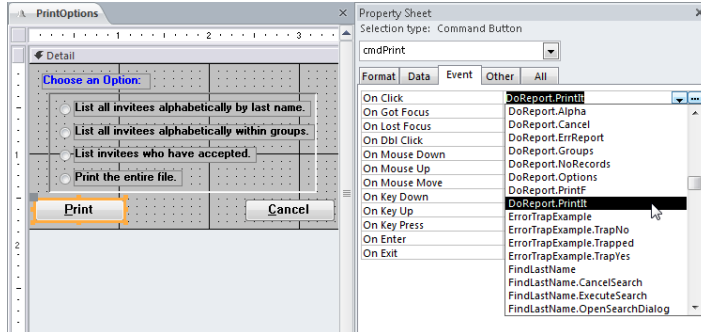
Figure 20-9 The DoReport macro group includes nine individual submacros.

To create a submacro within a macro object, you use the Submacro program flow construct in the Action Catalog. (You can also find the Submacro construct in the drop-down list of actions in the Add New Action combo box on the macro design surface. Access displays the four user interface macro program flow constructs—Comment, Group, If, and Submacro—first in the drop-down list.) You can create a series of actions at the beginning of the macro definition—not inside a submacro block—that you can reference from an event property or a RunMacro action by using only the name of the macro object. As you saw earlier in the AutoexecXmpl macro, naming a macro object in a RunMacro action (without any qualifier) asks Access to run the unnamed actions it finds in that macro object.

To create a set of named submacros within a macro object, you can drag a Submacro construct from the Action Catalog to the macro design surface or select Submacro from the Add New Action combo box. You then need to provide a name for your submacro. Note that Access always places submacro blocks below macro actions on the macro design surface. You cannot place macro actions that are outside a submacro block beneath any submacros. To execute a named submacro within a macro object from an event property or a RunMacro action, enter the name of the macro object, a period, and then the name of the submacro. For example, to execute the PrintIt submacro set of actions in the DoReport macro, enter **DoReport.PrintIt** in the event property or the Macro Name parameter.

In the sample DoReport macro, there are nine submacros within the object. (You must scroll down to see the other submacros.) The first submacro, Options (triggered by the Print Report button on the WeddingList form), opens the Print Options form, and the second submacro, PrintIt, determines which report was selected. The next four submacros (Groups, Alpha, Accepted, and PrintF) display the appropriate report in Print Preview mode, based on the result of the second submacro. The Cancel submacro merely closes the Print Options form if the user clicks the Cancel button. The NoRecords submacro cancels opening a report when the report's record source has no data, and the ErrReport submacro handles errors. As you might have guessed, Access runs a submacro starting with the first action in the submacro block name specified and executes each action in sequence until it encounters a StopMacro action, another submacro, or no further actions. As you'll see later, you can control whether some actions execute by adding conditional tests in the macro. Note that you can click Collapse All in the Collapse/Expand group on the Design tab to collapse all the actions quickly and see the submacro names.

If you open the PrintOptions form in Design view (see Figure 20-10) and look at the properties for each of the command buttons, you'll see that the On Click property contains the name of the submacro that executes when the user clicks the command button. If you open the list for any event property, you can see that Access lists all macro objects and the named submacros within them to make it easy to select the one you want.



**Figure 20-10** You can see that Access lists all macro objects and named submacros in the various event properties.

Remember, the macro name is divided into two parts. The part before the period is the name of the macro object, and the part after the period is the name of a specific submacro within the object. So, for the first command button control, the On Click property is set to DoReport.PrintIt. When the user clicks this button, Access runs the PrintIt submacro in the DoReport macro object. After you specify a macro name in an event property, you can click the Build button next to the property, and Access opens that macro in the Logic Designer.

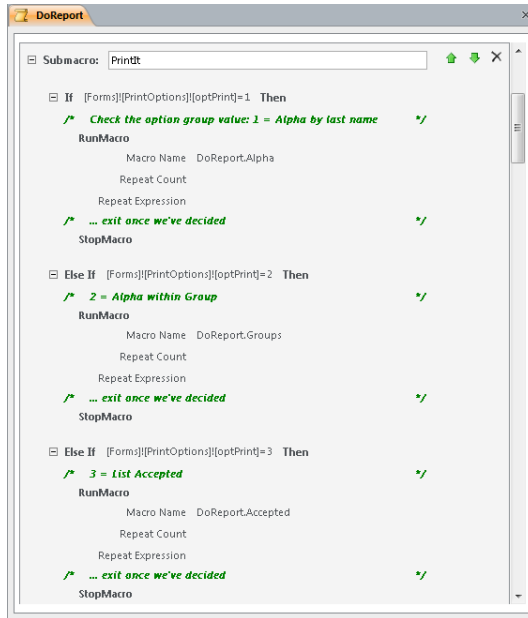
## Understanding Conditional Expressions

In some macros, you might want to execute some actions only under certain conditions. For example, you might want to update a record, but only if new values in the controls on a form pass validation tests; or you might want to display or hide certain controls based on the value of other controls. You can use an If block in macros to test conditions and then perform different actions based on the outcome of the conditional expression.

The PrintIt submacro in the DoReport macro group is a good example of a macro that uses conditions to determine which action should proceed. Right-click the DoReport macro in the Navigation pane, and then click Design View on the shortcut menu to see the Logic Designer, shown in Figure 20-11

As you saw earlier, this macro is triggered by the On Click property of the Print button on the PrintOptions form. This form allows the user to print a specific report by selecting the appropriate option button and then clicking Print. If you look at the form in Design view (see Figure 20-10), you'll see that the option buttons are located within an option group control on the form. Each option button sets a specific numeric value (in this case, 1 for the first button, 2 for the second button, 3 for the third button, and 4 for the fourth button) in the option group, which you can test using an If program flow construct.

As you learned in Chapter 7, when you include an If block in a macro, Access won't run the action on that line unless the condition evaluates to True. The text box next to If is where you type your conditional expression. Each condition is an expression that Access can evaluate to True (nonzero) or False (0 or Null). A condition can also consist of multiple comparison expressions and Boolean operators. If the condition is true, Access executes the action or actions immediately following the Then keyword. If the condition is false, Access evaluates the next Else If condition or executes the statements following the Else keyword, whichever occurs next. If no Else or Else If condition exists after the Then keyword, Access executes the next action following the End If keyword.



**Figure 20-11** In the PrintIt submacro, you can see that we added an If block in the DoReport macro group.

In this particular example, the expressions next to If and the three Else If conditions in the PrintIt submacro test the value of the option group control on the form. You can reference any control on an open form by using the syntax

`FORMS! formname! controlname`

where *formname* is the name of an open form and *controlname* is the name of a control on that form. In this case, the direct reference is `[FORMS]![PrintOptions]![optPrint]`. (*optPrint* is the name of the option group control. You can see this in the Name property on the Other tab of the property sheet for this control.) See “Referencing Form and Report Objects,” on page 1229, for more details about the rules for referencing objects in Access.

## INSIDE OUT

### When to Use Brackets

If your object names do not contain any embedded blanks or other special characters, you don't need to surround *formname* or *controlname* with brackets when you use this syntax to reference a control on a form; Access inserts the brackets as needed.

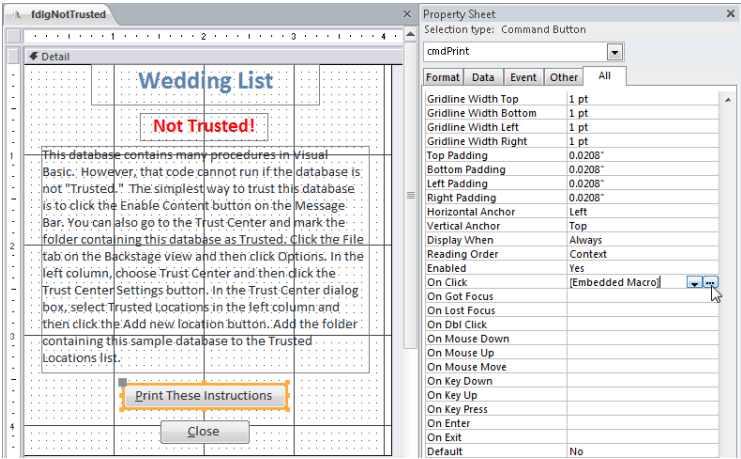
After you understand how to refer to the value of a control on a form, you can see that the PrintIt submacro tests for each of the possible values of the option group control. When it finds a match, PrintIt runs the appropriate named submacro within the macro object to open the requested report and then stops. If you look at the individual report macros, you'll see that each runs a common submacro, DoReport.Cancel, to close the PrintOptions form (which isn't needed after the user chooses a report) and then open the requested report in Print Preview and put the focus on the window that displays the report.

## Using Embedded Macros

Access 2010 includes a feature to create *embedded macros* in the event procedures for forms, reports, and controls. The macros you have been creating and opening thus far in this chapter are macro objects that you can access from the Navigation pane. You save embedded macros, however, within the event procedures for forms and reports. You cannot see or run these macros directly from the Navigation pane. You can think of embedded macros as similar to data macros in that data macros are not objects seen in the Navigation pane; they are attached to table events.

### Editing an Embedded Macro

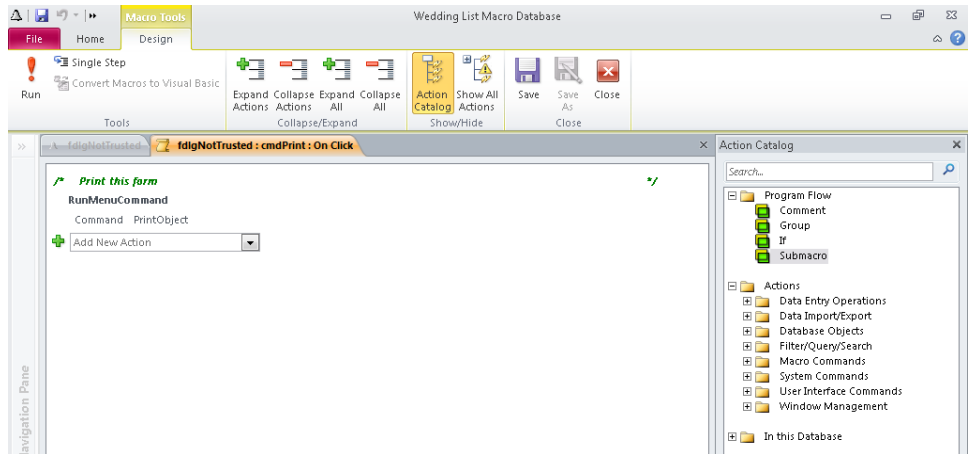
To edit an embedded macro, you must first open a form or report in Design or Layout view. The fdlgNotTrusted form in the Wedding List Macro database contains two embedded macros, each of which is attached to the Click event of one of the two command buttons. Select this form in the Navigation pane, and open it in Design view. Click the Property Sheet button in the Tools group on the Design tab to open the form's property sheet. Next, click the Print These Instructions command button or select cmdPrint from the selection list on the property sheet to view the properties for this command button, as shown in Figure 20-12.



**Figure 20-12** The property sheet lists any embedded macros attached to events.

Notice that [Embedded Macro] appears in the On Click property—this indicates that a macro is stored with the form design that responds to this event. To view and edit the macro attached to this event property, click the Build button on the right side of this property line. Access opens the Logic Designer, as shown in Figure 20-13. Notice that in the tab at the top of the macro design surface, Access displays the name of the form, the object name the embedded macro is attached to (in this case, the cmdPrint command button), and the specific event of the object that runs the embedded macro.

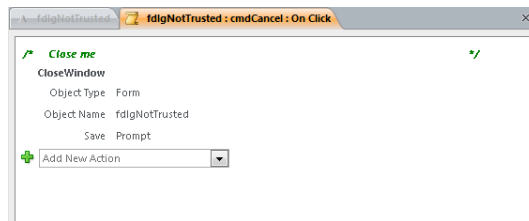
You'll notice Access automatically collapsed the Navigation pane to show you more of the macro designer surface. Access also opens the Logic Designer window modally when you are working with embedded macros, which means you cannot open any other database objects until you close the designer window.



**Figure 20-13** The Logic Designer shows the embedded macro that we created to respond to the Click event of the cmdPrint button on the fdlgNotTrusted form.

With the Logic Designer open, you can now view and edit the macro conditions (not used in this sample macro), actions, arguments, and comments. For the cmdPrint command button, you can see we attached a simple macro that executes the RunMenuCommand action. In the Command argument, we selected the PrintObject command, which tells Access to print the object that has the focus—in this case, the fdlgNotTrusted form. The application displays this form only if the database is not trusted. We provide this print button so that you can print the instructions for creating a trusted location displayed on the form.

Close the Logic Designer for this embedded macro by clicking the Close button in the Close group on the Design tab, and then click the Close command button on the form (or select cmdCancel from the selection list). You'll see [Embedded Macro] displayed in the On Click property for this command button. Click the Build button for this property to open the Logic Designer shown in Figure 20-14. This embedded macro uses the CloseWindow action to tell Access to close the fdlgNotTrusted form when the user clicks this command button.



**Figure 20-14** The Close button on the fdlgNotTrusted form executes an embedded macro to close the form.

We set the Save argument of the Close action to Prompt, which instructs Access to ask the user whether any changes to the form design should be saved on closing. (The form opens in Form view, so the user shouldn't be able to make any changes.) We selected this setting because choosing any other option causes the Close action to be not trusted. We'll discuss actions that are not trusted later in this chapter.

The two embedded macros you've seen on this form are simple macros with only one action each. You're not limited to using only one action in an embedded macro. You can create a very complex macro, such as the DoReport macro you saw previously in this chapter, with several macro actions using several conditions. However, there is one important difference when designing a complex embedded macro. If you create named submacros in an embedded macro, Access executes only the actions defined in the first submacro when the event to which this macro responds occurs. To execute the additional named submacros inside the embedded macro, you must create a call within the first set of actions to tell Access to execute the other submacros—as the DoReport macro object demonstrated earlier in this chapter.

## INSIDE OUT

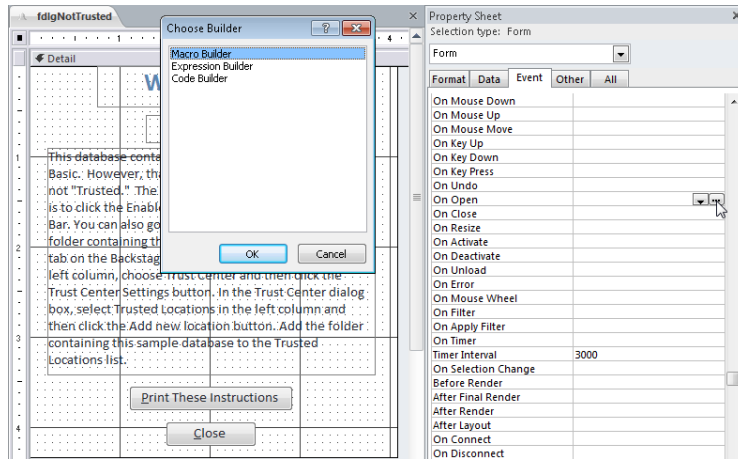
### Embedded Macros Stay with Their Controls

If you attach embedded macros to a specific control on a form or report, Access saves the macro with the control. If you cut or copy this control to the Clipboard and then paste it back on the form or report, Access keeps the embedded macro attached to the control.

## Creating an Embedded Macro

Close the Logic Designer if you still have it open, and let's create a new embedded macro to display a message box when this form opens. From the list under Selection Type near the top of the property sheet, select Form to display all the properties of the form. Click the Event tab, and then click the On Open property. To create a new embedded macro, click the Build button at the right end of the property. Access opens the Choose Builder dialog box, as shown in Figure 20-15.





**Figure 20-15** Select Macro Builder in the Choose Builder dialog box to create an embedded macro.

If you select the Macro Builder option, Access opens the Logic Designer window, where you can create your embedded macro. If you select Expression Builder, Access opens the Expression Builder dialog box, where you can build an expression to enter in the property. If you select Code Builder, Access opens the Visual Basic Editor, where you can write a Visual Basic code procedure for this event property. (We'll discuss Visual Basic in Chapter 24 and Chapter 25, "Automating Your Application with Visual Basic," on the companion CD.) Select the Macro Builder option, and then click OK to begin creating a new embedded macro.

To display a message box, select MessageBox in the Add New Action combo box. In the Message argument, enter the following text:

**This database is not trusted, so it cannot execute all the code needed to automate this application. Please read and follow the instructions displayed in the form that opens after you close this message in order to have the application function properly.**

In the Beep argument, leave the default setting Yes, and change the Type argument to Warning! to provide a visual cue that something is wrong and call attention to the message. In the Title argument, enter **Embedded Macro Test**. Your finished macro should look something like Figure 20-16.

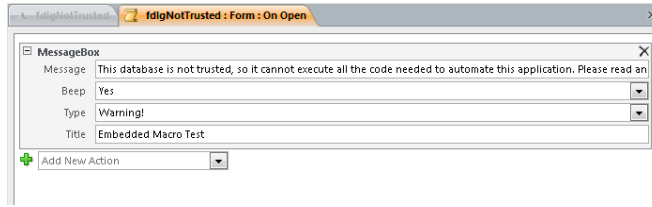


Figure 20-16 The MsgBox action displays a message box in Access.

## INSIDE OUT

### Enabling the Choose Builder Dialog Box

In Chapter 24, you'll learn that you can select [Event Procedure] from the list in an event property and then click the Build button to open the Visual Basic Editor to create the appropriate procedure. Unfortunately, there is no corresponding [Embedded Macro] option that you could use in the same way to create a macro to respond to the event. You also cannot type [Embedded Macro] in the property and click Build. You must leave the property blank, click Build, and choose Macro Builder in the Choose Builder dialog box.

But there's a catch. To see the Choose Builder dialog box, you must not select the Always Use Event Procedures check box in the Form/Report Design View section in the Object Designers category of the Access Options dialog box. (The option is cleared by default.) If you select that check box, Access always opens the Visual Basic Editor window when you click the Build button in any event property. When you're working with client forms and client reports, the only way to create a new embedded macro is to select Macro Builder in the Choose Builder dialog box. If you intend to use embedded macros, you must leave the Always Use Event Procedures option cleared.

Click the Save button on the Quick Access Toolbar to save this new embedded macro, and then close the Logic Designer window. You'll notice that Access now displays [Embedded Macro] on the On Open property line. Note that if you don't click Save before closing the macro design window, Access prompts you to save the changes and update the property.

If you click No, Access does not save the embedded macro. Click Save again to save the changes to the form itself. Switch to Form view (or close the form and then open it in Form view from the Navigation pane), and notice that Access now displays a message box, as shown in Figure 20-17. Click OK in the message box, and Access then displays the not trusted form. Click Close to close the form.

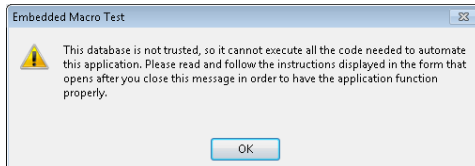


Figure 20-17 Your embedded macro now displays a message box before the form opens.

## Deleting an Embedded Macro

If you need to delete a saved macro object, you can easily delete it in the Navigation pane. For embedded macros, however, you need to delete the contents in the specific property. Open the `fdlgNotTrusted` form again in Design view, and then open the property sheet for the form. To delete the message box embedded macro you just created, find the On Open property on the Event tab, highlight `[Embedded Macro]`, and then press Delete to delete the embedded macro. Click the Save button on the Quick Access Toolbar to save your changes, and then close the form.

### CAUTION!

Access does not warn you that it deletes the macro associated with an event property when you clear the property setting. You also cannot undo clearing the property to get the macro back. If you delete a complex macro that was previously saved in the form design, click the File tab on the Backstage view, and click Save Object As to save the form with a new name (or close the form without saving if you're willing to discard other changes). You can then open the original form in Design view to recover the macro. Remember that when you copy and paste a control from one form to another, Access also pastes any attached embedded macros, so you can copy the control and its macro from the old form to the new one to get the macro back in the new form.

## INSIDE OUT

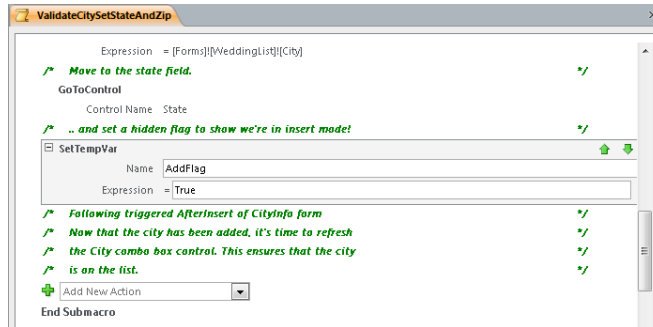
### Embedded Macros Won't Work with Earlier Versions of Access

If you create a database in the .mdb file format, Access 2010 allows you to create embedded macros for forms, reports, and controls just like you can in an .accdb file format database. But if you open the .mdb database with an earlier version of Access—2000, 2002, or 2003—the embedded macros do not function. In fact, you cannot see any [Embedded Macro] entries for event properties when you open an .mdb database with an earlier Access version. If you create an .mdb format database using Access 2010 that will be opened and run with a previous version of Access, do not create embedded macros for your application.

## Using Temporary Variables

You can use a temporary variable in Access to store a value that can be used in other macros, event procedures, expressions, and queries. As you'll learn in Chapter 24, we use a variable to store the user name when you log into the Conrad Systems Contacts and Housing Reservations sample databases. Variables are very useful when you need Access to remember something for later use. You can think of a temporary variable in a macro as writing yourself a note to remember a number, a name, or an address so that you can recall it at a later time. All variables have a unique name. To fetch, set, or examine a variable, you reference it by its name. Temporary variables stay in memory until you close the database, assign a new value, or clear the value.

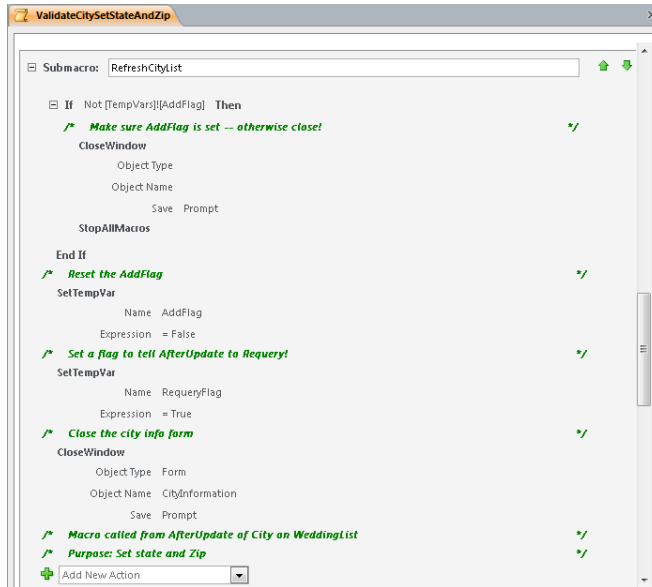
To see an example of using a temporary variable in the Wedding List Macro sample database, open the ValidateCitySetStateAndZip macro in Design view. We'll study this macro in more detail in "Validating Data and Presetting Values," on page 1239, but for now, we'll focus on creating a temporary variable. Creating a temporary variable in a macro is easy—Access creates the variable for you when you reference it for the first time in a SetTempVar action. In Figure 20-18, you can see that in the AskEdit submacro in the ValidateCitySetStateAndZip macro object, we created a new temporary variable called AddFlag and set its value to True in the Expression argument.



**Figure 20-18** The AskEdit submacro in the ValidateCitySetStateAndZip macro uses a temporary variable to indicate that the CityInformation form has been opened in data entry mode.

The AskEdit submacro runs from the BeforeUpdate event of the City combo box on the WeddingList form when the user enters a new city name that isn't in the row source. The macro first executes a MsgBox function in the condition of the first action to ask the user whether the new city should be added. If the user clicks the Yes button in the dialog box displayed by the MsgBox function, the function returns the value 6. (We'll explain more about the MsgBox function later.) If the user clicks No, the macro halts. When the user clicks Yes, the submacro calls the IsFormLoaded custom Visual Basic function (in the modUtility module object) to determine whether the CityInformation form is open. If it is, the submacro closes it. The submacro then opens the CityInformation form in data entry mode and copies the new city name from the WeddingList form to the CityInformation form.

The application uses the AddFlag variable to let code in another macro know that this macro has closed and reopened the CityInformation form in data entry mode. The RefreshCityList submacro that executes in response to the AfterInsert event in the CityInformation form is also stored in the ValidateCitySetStateAndZip macro. The RefreshCityList submacro tests the AddFlag variable set in the AskEdit submacro. Scroll down the macro design surface until you come to the RefreshCityList submacro, as shown in Figure 20-19.



**Figure 20-19** The RefreshCityList submacro in the ValidateCitySetStateAndZip macro tests and sets temporary variables.

In the If conditional expression for the first action in this submacro, you can see the following expression:

`Not [TempVars]![AddFlag]`

This test checks to see whether the AddFlag temporary variable has been set. If not, then the user must be using the CityInformation form to add a new record independent of the WeddingList form, so the submacro closes the form and stops (the StopAllMacros action). If the AddFlag temporary variable is true, the submacro resets the AddFlag temporary variable to False, sets another temporary variable, RequeryFlag, to let the macro that responds to the AfterUpdate event of the City combo box to do a requery, and closes the CityInformation form.

Note the special syntax you need to use to reference a temporary variable anywhere other than in an action specifically related to temporary variables. When you create a temporary variable in a macro, Access adds the variable to the special collection of the database called TempVars. When an object is a member of a collection (Access treats temporary variables as objects), you can reference the object by naming the collection, using an exclamation point separator, and then naming the object. So, to reference a temporary variable in macros, queries, event procedures, and even Visual Basic code, use the following syntax:

`[TempVars]![<name of temporary variable>]`

You can have as many as 255 temporary variables defined at any time in your Access 2010 database. By using temporary variables in the various submacros in the ValidateCitySet-StateAndZip macro object, you can change the way Access executes the various macro actions based on actions taken in other macros and submacros.

If you need to clear the value stored in a temporary variable and delete the variable, you can use the RemoveTempVar macro action. The RemoveTempVar action requires only one argument—Name—and it clears any value stored in the temporary variable of that name and then deletes the variable. If you need to delete all temporary variables from memory, you can use the RemoveAllTempVars action. This action requires no arguments because it clears all temporary variables, similar to what would occur if you closed the database.

Although removing a temporary variable technically deletes it from the TempVars collection, you won't get an error if you attempt to reference a temporary variable that doesn't exist. If you attempt to fetch the value of a nonexistent temporary variable, Access returns the value Null. For this reason, you should be careful when naming and using temporary variables. If you set a temporary variable in one macro and then think you're referencing the same variable in another macro but slightly misspell the variable name, you won't get the results you expect.

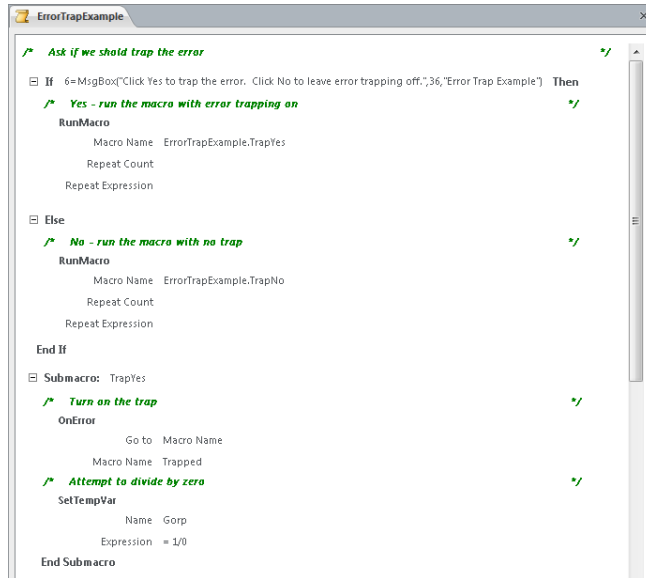
### Note

Access 2010 allows you to create temporary variables in macros if you save your database in the .mdb file format. If, however, you open the .mdb database with an earlier version of Access—2000, 2002, or 2003—the temporary variables do not function, and you will receive error messages when your macros run. If you have users still using previous Access versions, do not use temporary variables created in macros for your application.

## Trapping Errors in Macros

Access 2010 supports trapping and handling errors within macros. During the normal process of running your application, Access can (and most likely will) encounter errors. Access might encounter errors in your code that it cannot resolve—such as a syntax error in a predicate used to filter a form. In those cases, Access cannot proceed further. Other errors might occur that are not quite so catastrophic but happen in the normal processing of your application. For example, you might use the OnNoData event of a report to display a message box saying no records were found. If your code then cancels the report from opening, Access returns an error if a subsequent action attempts to reference the report that didn't open. If there's no error trap in the macro, Access displays an ugly and confusing dialog box to the user.

To see how error trapping works in Access 2010, close the ValidateCitySetStateAndZip macro and then open the ErrorTrapExample macro in Design view. We created this simple macro specifically to show you two things—how Access handles an unexpected error with no error trapping and how you can trap and respond to an error. In Figure 20-20, you can see some of the submacro names, conditional expressions, actions, and arguments for this example macro.



**Figure 20-20** The ErrorTrapExample macro demonstrates error handling in Access 2010.

In the first action of the macro, we call the MsgBox function in the If conditional expression to ask whether you want to use error trapping. (You can learn more about the settings in the MsgBox function in Table 20-4 on page 1243.) If you click the Yes button in the dialog box displayed by MsgBox, the function returns the value 6. So when you click Yes, the condition is True, and the RunMacro action calls the TrapYes submacro. If you click No, the condition is False, so Access moves down to the Else block and executes the second RunMacro that calls the TrapNo submacro.

The first action in the TrapYes submacro uses the OnError macro action to tell Access how it should proceed if any error occurs. The OnError action has two arguments—Go To and Macro Name. The options in the Go To argument are Next, Macro Name, and Fail. If you select Next in the Go To argument, Access does not halt the macro when an error occurs—it simply goes on to the next action. If you select Macro Name in the Go To argument, Access runs the submacro you specify in the Macro Name argument. If you select Fail, you're basically turning error trapping off.



In all cases, Access records the error number and error description information in the `MacroError` object. If you have trapped the error by specifying `Macro Name` or `Next`, you can examine the error in an `If` conditional expression to determine what action, if any, to take. For simple errors (such as an `OpenReport` that might be canceled), you can choose `Next` and check to see whether an error has occurred in an `If` block on the next action. For more complex errors, you should go to another submacro that can test for several potential errors that you plan to handle. In this example, we tell Access to run the `Trapped` submacro if any errors occur.

### Note

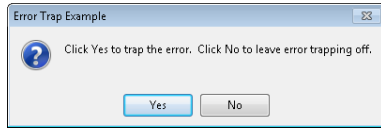
If you specify a submacro name in the `Go To` argument of the `OnError` action, the submacro must exist in the same macro object. You cannot reference a submacro in a different macro object when using the `OnError` action.

The next line in the `TrapYes` macro uses the `SetTempVar` action to create a temporary variable named `Gorp` and set it to an invalid mathematical expression of `1/0`—dividing by zero will cause an error. Because we asked Access to trap any error, Access runs the `Trapped` submacro when the error occurs. Although we could have examined the error and perhaps taken some other action, for this simple example, we used a `MessageBox` action to tell Access to display a message containing the error number and description of the error. Scroll down the macro design surface, select this `MessageBox` action, and notice the following text in the `Message` argument:

```
= "Error Trapped: " & [MacroError].[Number] & ", " & [MacroError].[Description]
```

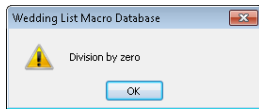
All errors in Access have both a unique error number and a description. When an error occurs in a macro, the `Number` property of the `MacroError` object contains the error number, and the `Description` property of the `MacroError` object contains text describing the error associated with the number. The `Message` argument of the `MessageBox` action asks Access to fetch the `Number` and `Description` properties of the `MacroError` object and display them in the message.

Finally, the `TrapNo` submacro executes the assignment of an invalid value to a temporary variable without first setting an error trap. To see how this process works, click the `Run` button in the `Tools` group on the `Design` tab. Because the first action contains a call to the `MsgBox` function in the `If` conditional expression, Access displays the message box shown in Figure 20-21, asking whether you want to trap the error as part of evaluating the condition.



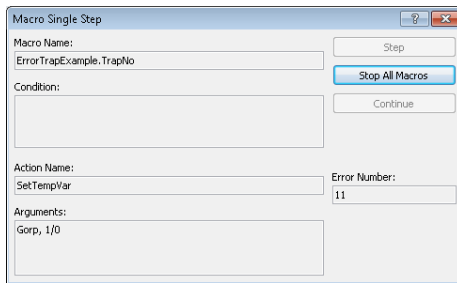
**Figure 20-21** When you run the ErrorTrapExample macro, it first asks you whether you want to trap the error.

Click No to see what happens when the error isn't trapped. First, Access displays a message box telling you the nature of the error, as shown in Figure 20-22.



**Figure 20-22** Access cannot divide a number by zero, so it displays an application error message.

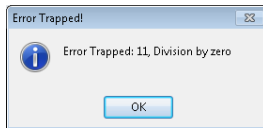
Click OK in this error message, and then Access displays the Macro Single Step dialog box, as shown in Figure 20-23. Not very user friendly, is it? Access displays the Macro Single Step dialog box whenever it encounters an unhandled error while running a macro. Access displays the specifics of where the error occurred in the Macro Name, Condition, Action Name, and Arguments text boxes. Access displays the error number currently stored in the MacroError object in the Error Number text box—in this case, error number 11.



**Figure 20-23** Access displays the Macro Single Step dialog box if it encounters an unhandled error.

The only button you can click in this dialog box is Stop All Macros. When you click this button, Access stops running the macro so that you can continue working in your application. You can imagine the support calls you're going to get from your users if this dialog box displays often in your applications. Click Stop All Macros to close the Macro Single Step dialog box.

Now, let's see what happens when the macro traps the error. Run the macro again, and click Yes when the code asks you whether you want to trap the error. This runs the TrapYes submacro (shown earlier in Figure 20-20), which executes `OnError`, followed by the `SetTempVar` action that generates an error. Access traps the error and executes the Trapped submacro as requested. That submacro asks Access to display another message box with the error number and description, as shown in Figure 20-24. Notice that Access displays the error number (11) and error description (Division By Zero) in the message text.

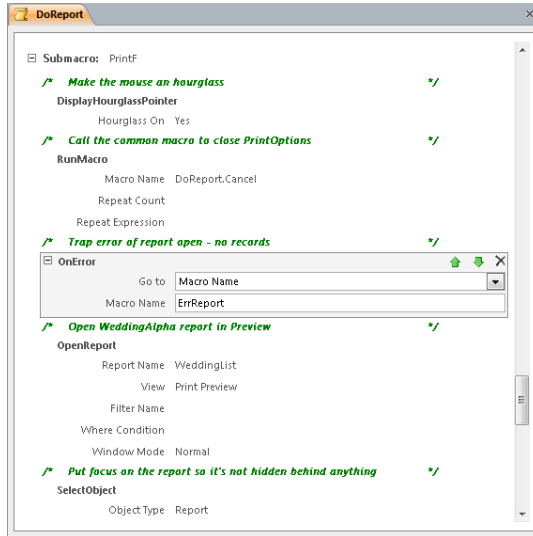


**Figure 20-24** By trapping an error in a macro, you can display a helpful message to the user.

In a completed application, you probably would not need to display the error details to the user, but for debugging your application, the information in the `MacroError` object can be very useful. For a user, it could be more informative to display a message such as “While attempting to calculate a value, the application divided a number by zero. Please recheck the numbers you entered before proceeding.”

Click OK in the message box, and notice what happens when you do trap the error—nothing! Because we trapped the error, Access does not display the confusing Macro Single Step dialog box.

Earlier in this chapter, you saw the `DoReport` macro that is used with the `PrintOptions` form. This macro also uses error trapping to handle the possibility that a report might not contain any records. Close the `ErrorTrapExample` macro, and then open the `DoReport` macro in Design view. In the Groups, Alpha, Accepted, and `PrintF` submacros, you can see that we used the `OnError` action just before each `OpenReport` action. Scroll down until you can see the `PrintF` submacro, as shown in Figure 20-25.



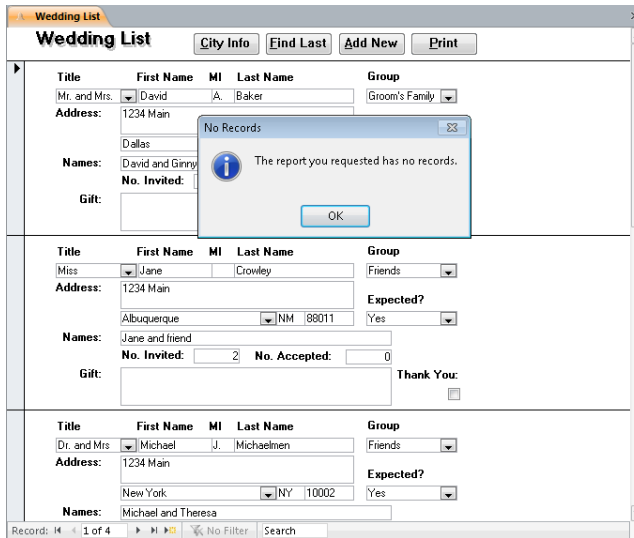
**Figure 20-25** The DoReport macro uses the OnError action to handle the possibility that no records are returned in the report.

The first action of the submacro turns the mouse pointer into an hourglass. The second action calls the Cancel submacro that closes the PrintOptions form and puts the focus back on the WeddingList form. The third action sets the error trap. We selected Macro Name in the Go To argument and ErrReport in the Macro Name argument to tell Access to go to the ErrReport submacro if any errors occur. The fourth action attempts to open the report.

In each of the four reports in this sample database, the On No Data event property specifies the NoRecords submacro. When the report has no records, this macro executes the CancelEvent action to prevent the report from opening. If the report opening is canceled, Access encounters an error on the next action of our submacro—SelectObject. Access cannot put the focus on a report that isn't opened, so we need to plan for this possibility. Because we're trapping all errors, the user won't see the ugly Macro Single Step dialog box. Instead, the ErrReport submacro runs, and this submacro restores the mouse pointer and displays an informative message telling the user that the report requested has no records.

To test how this works, close the DoReport macro, and then open the WeddingList form in Form view. Click the Print button on this form to open the PrintOptions form. On the PrintOptions form, select List Invitees Who Have Accepted. Unless you have changed the sample data, this report should return no records. Click Print to run the PrintIt submacro in the DoReport macro group. This submacro looks at the option you chose on the PrintOptions form and runs the Accepted submacro. That submacro attempts to open the WeddingAccepted report with a filter to return only the records where the value in the Accepted field is greater than zero.

Because no records qualify, the NoData event in the WeddingAccepted report runs the NoRecords submacro and cancels the opening of the report. Next, the submacro attempts to set the focus on the WeddingAccepted report. Because the report is now closed, this causes an error—2489, if you're curious—that Access returns to the submacro that attempted to set the focus. Because we turned on error trapping, the ErrReport submacro displays a message to inform you that no records were found in the report, as shown in Figure 20-26.



**Figure 20-26** The error handling in the DoReport submacro presents an informative message if the report contains no records.

If you want to see what happens when the error isn't trapped, open the DoReport macro object, scroll down to the Accepted submacro, and change the Go To argument to Fail for the OnError action so that the error trap isn't set, and save the macro. Try to run the report again from the WeddingList form, and you'll see the ugly error messages that result when you don't trap the error. Be sure to change the Go To argument back to Macro Name in the OnError action of the Accepted submacro and save it again so that the application works properly.

If you want to use macros in your application, you should add appropriate error handling using the OnError action. A well-designed Access application should always display helpful messages to users when errors occur.

## INSIDE OUT

### Clearing the MacroError Object

The `MacroError` object contains only the information from the last reported error. Access retains this information in the `MacroError` object until either the macro stops running, another error occurs, or you run the `ClearMacroError` action. If you need to continue running your macro after an error is handled and expect to possibly test the `MacroError` object again (perhaps after setting `OnError` to `Next`), you can use the `ClearMacroError` action to clear the contents of the `MacroError` object. The `ClearMacroError` action requires no arguments.

## Understanding Macro Actions That Are Not Trusted

Earlier in this chapter, we mentioned that Access 2010 has trusted and not trusted macro actions. As you might recall from Chapter 2, “Exploring the Access 2010 Interface,” the Trust Center settings in the Access Options dialog box control whether Access disables certain content in your database. If your database is not trusted, Access might silently disable any potentially malicious macros or Visual Basic for Applications (VBA) code depending upon the Trust Center settings you enabled or disabled. So, what exactly is a malicious macro? In Microsoft’s terms, a malicious macro runs an action that could potentially do harm to your computer or files, such as deleting a file.

Access 2010 separates client macro actions into two categories—those that will run in any database, even in a database that is not trusted (trusted macros), and those that can run only in a database that is trusted (not trusted macros). Note that if you select `Enable All Macros` in the Trust Center Macro Settings section (not recommended by Microsoft), Access treats all macro actions as trusted even when the database is not trusted. (In Chapter 21, you’ll learn that all web macro actions are trusted.)

### Note

If you are in a corporate network environment, you should check with your IT department to determine whether your company has established guidelines concerning enabling content in Access databases.

Access 2010 recognizes 27 client macro actions as potentially unsafe to run in a database that is not trusted. Seven of the actions are not trusted only when you select certain arguments. Table 20-1 lists the client macro actions that Access will run only when the database is trusted. The Comments column lists special cases that depend on the arguments you choose or an alternative trusted method you can use.

**Table 20-1** Macro Actions That Are Not Trusted

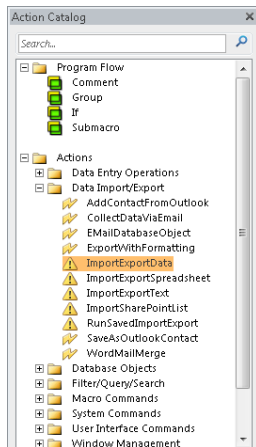
Action	Comments
CloseWindow	Setting the Save argument to Prompt is trusted.
CopyObject	
DeleteObject	
Echo	
ImportExportData	
ImportExportSpreadsheet	
ImportExportText	
ImportSharePointList	
OpenForm	Setting the View argument to Design or Layout is not trusted.
OpenQuery	Setting the View argument to Design is not trusted.
OpenReport	Setting the View argument to Print, Design, or Layout is not trusted.
OpenSharePointList	
OpenSharePointRecycleBin	
OpenTable	Setting the View argument to Design is not trusted.
OpenVisualBasicModule	
PrintOut	
QuitAccess	Setting the Options argument to Prompt is trusted.
RenameObject	
RunApplication	
RunMenuCommand	Commands that affect objects in Design or Layout view are not trusted.
RunSavedImportExport	
RunSQL	
SaveObject	
SendKeys	
SetValue	Use the trusted SetProperty action instead of SetValue to change the Enabled, Visible, Locked, Left, Top, Width, Height, Fore Color, Back Color, and Caption properties of forms, reports, or controls.
SetWarnings	
ShowToolbar	

# INSIDE OUT

## Using Older Macro Action Names

In Access 2010, some macro actions from previous versions of Access have been renamed. For example, the `TransferText` macro action is now named `ImportExportText`, and the `MsgBox` action is now named `MessageBox`. If you open a database created in an earlier version of Access in Access 2010, you'll see the new macro action name displayed on the macro design surface. If you type the older macro name in the Add New Action combo box and press Enter, Access displays the action on the macro design surface. However, you'll note that Access displays the new action name on the action block. Your existing macros created in previous versions will run in Access 2010, but you should use the new action names when designing new macros. See Article 6 on the companion CD for a list of older macro names and their new equivalents in Access 2010.

Note that when you select a client macro action or argument that is not trusted, Access displays an exclamation mark in the upper-left corner of the action block on the macro design surface. Access also displays an exclamation mark to the left of any untrusted action names displayed in the Action Catalog, as shown in Figure 20-27. When you're designing your macros, you can use these visual aides to easily see whether any of your client macro actions will not run in a database that is not trusted.



**Figure 20-27** Client macro actions that are not trusted display an exclamation mark next to the action name in the Action Catalog.



# Making Your Application Come Alive with Macros

Throughout this book, you've learned how to perform common tasks by using ribbon commands or by finding the object you want in the Navigation pane and opening it. In working with your database, you've probably also noticed that you perform certain tasks repeatedly or on a regular basis. You can automate these tasks by creating macros to execute the actions you perform and then associating the macros with various form or control events, such as the Current event of a form, the Click event of a command button, or the DblClick event of a text box. In the following sections, you'll use examples from the Wedding List Macro sample database (WeddingMC.accdb) to understand how macros can help automate your application.

## Referencing Form and Report Objects

As you create macros to automate tasks that you repeat frequently, you'll often need to refer to a report, a form, or a control on a form to set its properties or values. Before we dig into some of the macros in the Wedding List Macro, you need to know how to code these references. You can find the syntax for referencing reports, forms, report and form properties, controls, and control properties in the following sections.

### Rules for Referencing Forms and Reports

You can refer to a form or a report by name, but you must first tell Access which collection contains the named object. Open forms are in the Forms collection, and open reports are in the Reports collection. To reference a form or a report, you follow the collection name with an exclamation point to separate it from the name of the object to which you are referring. You must enclose an object name that contains blank spaces or special characters in brackets ([ ]). If the object name contains no blanks or special characters, you can simply enter the name. However, it's a good idea to always enclose an object name in brackets so that your name reference syntax is consistent.

For example, you refer to a form named WeddingList as follows:

```
Forms! [WeddingList]
```

You refer to a report named WeddingList as follows:

```
Reports! [WeddingList]
```

## Rules for Referencing Form and Report Properties

To reference a property of a form or a report, follow the form or report name with a period and the property name. You can see a list of most property names for a form or a report by opening the form or the report in Design or Layout view and displaying the property sheet while you have the form or the report selected. With macros, you can change most form or report properties while the form is in Form view, or from the Print, Format, and Paint events of a client report as Access prints or displays it.

You refer to the Scroll Bars property of a form named CityInformation as follows:

```
Forms![CityInformation].ScrollBars
```

You refer to the Caption property of a report named CityInformation as follows:

```
Reports![CityInformation].Caption
```

### Note

The names of properties do not contain embedded blank spaces, even though the property sheet shows blanks within names. For example, BackColor is the name of the property listed as Back Color in the property sheet.

## Rules for Referencing Form and Report Controls and Their Properties

To reference a control on a form or a report, follow the form or report name with an exclamation point and then the control name enclosed in brackets. To reference a property of a control, follow the control name with a period and the name of the property. You can see a list of most property names for controls by opening a form or a report in Design or Layout view, selecting a control (note that different control types have different properties), and opening its property sheet. You can change most control properties while the form is in Design view.

You refer to a control named State on the WeddingList form as follows:

```
Forms![WeddingList]![State]
```

You refer to the Visible property of a control named Accepted on a report named WeddingList as follows:

```
Reports![WeddingList]![Accepted].Visible
```

## Rules for Referencing Subforms and Subreports

When you embed a subform in a form or a report, the subform is contained in a *subform control*. A subreport embedded in a client report is contained in a *subreport control*. You can reference a subform control or a subreport control exactly as you would any other control on a form or a report. For example, suppose you have a subform called `RelativesSub` embedded in the `WeddingList` form. You refer to the subform control on the `WeddingList` form as follows:

```
Forms![WeddingList]![RelativesSub]
```

Likewise, you can reference properties of a subform or a subreport by following the control name with a period and the name of the property. You refer to the `Visible` property of the `RelativesSub` subform control as follows:

```
Forms![WeddingList]![RelativesSub].Visible
```

Subform controls have a special `Form` property that lets you reference the form that's contained in the subform control. Likewise, subreport controls have a special `Report` property that lets you reference the report contained in the subreport control. You can follow this special property name with the name of a control on the subform or the subreport to access the control's contents or properties. For example, you refer to the `LastName` control on the `RelativesSub` subform as follows:

```
Forms![WeddingList]![RelativesSub].Form![LastName]
```

You refer to the `FontWeight` property of the `LastName` control as follows:

```
Forms![WeddingList]![RelativesSub].Form![LastName].FontWeight
```

## Opening a Secondary Form

As you learned in Chapter 12, "Using Forms in an Access Application," it's easier to work with data by using a form. You also learned in Chapter 15, "Advanced Form Design," that you can create multiple-table forms by embedding subforms in a main form, thus allowing you to see related data in the same form. However, it's impractical to use subforms in situations such as the following:

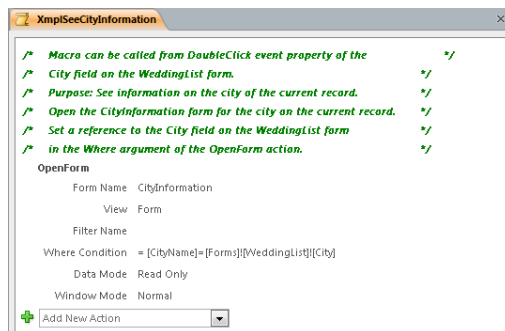
- You need three or more nested subforms to see related data.
- The main form is too small to display the entire subform.
- You need to see the related information only some of the time.

The solution is to use a separate form to see the related data. You can open this form by creating a macro that responds to one of several events. For example, you can use a command button or the `DbtClick` event of a control on the main form to give your users access to the related data in the secondary form. This technique helps reduce screen clutter, makes the main form easier to use, and helps to speed up the main form when you're moving from record to record.

You could use this technique in the `WeddingList` form. It would be simple to create a macro that would respond to clicking the `City Info` button by opening the `CityInformation` form and displaying all records from the `CityNames` table, including the best airline to take and the approximate flying time from each city to Seattle, Washington. However, if you're talking to your friend Jane in Albuquerque, New Mexico, it would be even more convenient for the `CityInformation` form to display only Albuquerque-related data rather than the data for all cities. In the following section, you'll create a macro that opens the `CityInformation` form based on the city that's displayed for the current record in the `WeddingList` form.

## Creating the SeeCityInformation Macro

Open the `Wedding List Macro` sample database (`WeddingMC.accdb`) if you've closed it. Click `OK` in the opening message so that no objects are opened. Click the `Macro` button in the `Macros & Code` group on the `Create` tab to begin creating a new macro object. When the `Logic Designer` window opens, collapse the `Navigation` pane. Figure 20-28 shows the macro you are going to create. (If you simply want to view the macro, it is saved as `XmplSeeCityInformation` in the sample database.)



**Figure 20-28** When triggered from an event on the `WeddingList` form, this macro opens the `CityInformation` form filtered on the city name.

The macro contains only one action, `OpenForm`. The `OpenForm` action not only opens the `CityInformation` form but also applies a filter so that the city that will be displayed matches the city currently displayed in the `WeddingList` form. Add an `OpenForm` action to the macro design surface by dragging the action from the Action Catalog or by selecting `OpenForm` in the Add New Action combo box. In the Where Condition argument, enter the following expression:

```
[CityName]=Forms![WeddingList]![City]
```

The Where Condition argument causes the `OpenForm` action to open the `CityInformation` form showing only the rows in the form's record source whose `CityName` field equals the value currently shown in the City combo box on the open the `WeddingList` form. (Later, you'll learn how to create a macro to synchronize these two forms as you move to different rows in the `WeddingList` form.)

Set the rest of the action arguments for the `OpenForm` action, as shown in Figure 20-28. After you finish creating the action for the macro, it's a good idea to add Comment blocks to the macro design surface to document your macro. Documenting your macro makes it easier to debug, modify, or enhance the macro in the future. It's also easier to read in English what each macro action does rather than have to view the arguments for each action line by line. Refer to Figure 20-28 and enter the information displayed into several Comment blocks. You can see that we've added comments about the macro in general and about the specific action the macro is designed to perform. Click the Save button on the Quick Access Toolbar, and save the macro as `SeeCityInformation`.

Next, you can associate the macro with the City combo box control on the `WeddingList` form. Click the `WeddingList` form in the Navigation pane, right-click the name, and click Design View to open the form in Design view. Click the City combo box control, and then click the Property Sheet button in the Tools group on the Design tab. When the property sheet opens, click the Event tab. You'll want to trigger the `SeeCityInformation` macro you just created from the `DbClick` event, so click the On DbClick property box, and select the macro from the On DbClick event property's drop-down list. You'll find a macro called `SeeCityInfo` already entered here, as shown in Figure 20-29. We created a slightly different version of the macro and saved it in the form so that the application is fully functional when you first open it. You can change the event property to your macro (`SeeCityInformation`) to test what you've built.

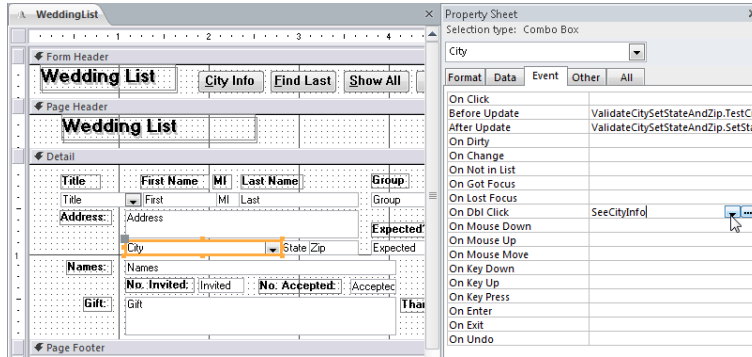


Figure 20-29 Select the macro you created for the DblClick event of the City combo box control.

You can also associate the macro with the City Info button by changing the button's On Click event property to point to the macro. To do this, click Save on the Quick Access Tool-bar to save your changes and then switch to Form view. Scroll down one or two records, and double-click the City combo box. The CityInformation form opens, and the data displayed should be for the city in the current record in the WeddingList form. Your screen should look like Figure 20-30.

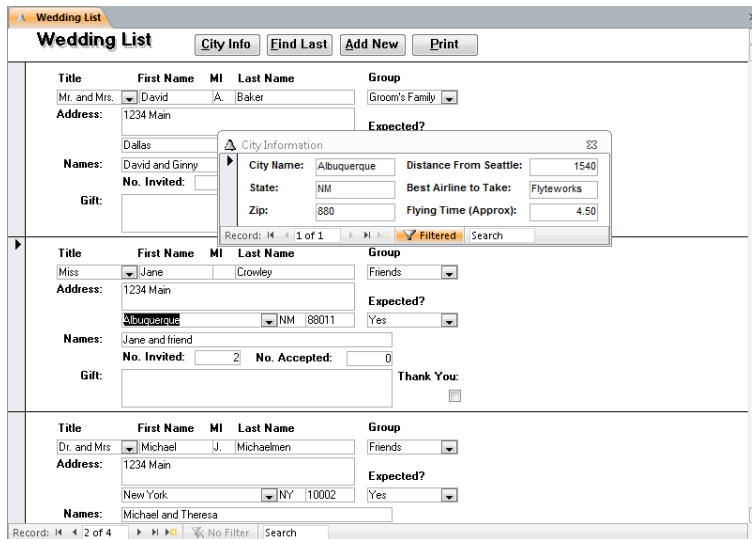


Figure 20-30 The CityInformation form displays a matching city in the WeddingList form.

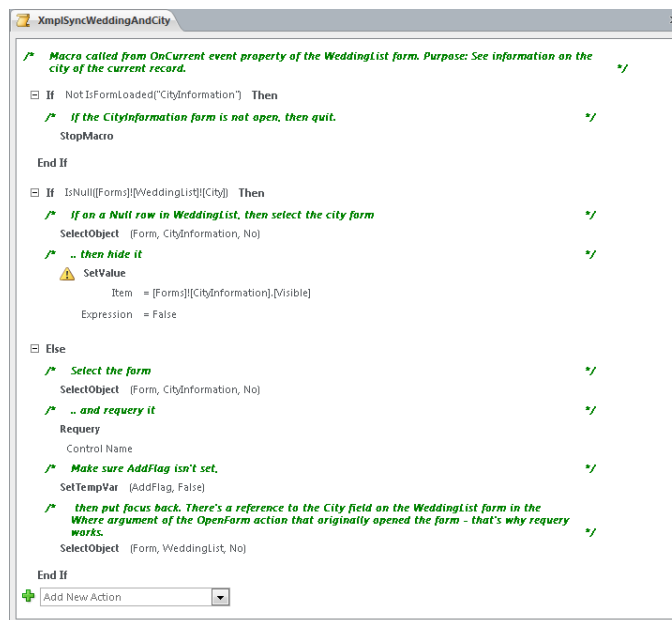
Linking two related forms in this manner is very useful, but what happens to the data displayed in the CityInformation form when you move to a new record in the WeddingList form? Try scrolling through the records using the record selector. You'll find that the data in the CityInformation form changes as you move through records in the WeddingList form. The data changes because we've set one of the events on the WeddingList form to execute a macro that keeps the data displayed on the two forms synchronized. In the next section, you'll walk through the steps to re-create this macro yourself. Close the two forms that are currently open to continue with the next section.

## Synchronizing Two Related Forms

In the previous section, you learned how to open a secondary form from a main form based on matching values of two related fields in the two forms. In the following sections, you'll create a macro that synchronizes the data in a companion form when the selected record changes in a main form.

### Creating the SyncWeddingListAndCity Macro

Click the Macro button in the Macros & Code group on the Create tab to start creating a new macro object. Figure 20-31 shows the actions and arguments you'll create for this macro. Note that in Figure 20-31, we collapsed most of the actions so you can see all the macro logic. (You can find this sample macro saved as XmplSyncWeddingAndCity.)



**Figure 20-31** You'll create these conditions, actions, and comments for the SyncWeddingAndCity macro.

You'll create this macro in the same basic manner that you created the SeeCityInformation macro. Enter the needed conditional expressions into the two If blocks, and add the actions from the Action Catalog (listed in Table 20-2 in the Action column). Type the associated arguments options, listed in the Setting column in Table 20-2, into the action arguments on the macro design surface. (You can ignore typing in the comments for this example.)

### Note

Some code and expression examples in this chapter are too long to fit on a single printed line. A line that ends with the ↵ symbol means that the code shown on the following line should be entered on the same line.

**Table 20-2** Actions, Arguments, and Settings in SyncWeddingAndCity

If Condition	Action	Argument	Setting
Not IsFormLoaded ↵ ("CityInformation")	StopMacro		
IsNull([Forms]!↵ [WeddingList]!↵ [City])	SelectObject	Object Type	Form
		Object Name	CityInformation
		In Database Window	No
	SetValue	Item	[Forms]![CityInformation]. [Visible]
		Expression	False
Else	SelectObject	Object Type	Form
		Object Name	CityInformation
		In Database Window	No
	Requery		
	SetTempVar	Name	AddFlag
		Expression	False
	SelectObject	Object Type	Form
		Object Name	WeddingList
		In Database Window	No

This macro has a couple of If block conditional expressions that determine which parts of the macro execute. The first If block condition uses the IsFormLoaded function, which



is included in the modUtility module of the Wedding List Macro database. This function checks to see whether a form (whose name you've provided to the function) is currently open. (The form can be hidden.) The syntax for the function is `IsFormLoaded("formname")`, where *formname* is the name of the form in question. You must enclose the name of the form in double quotation marks for the function to work. The *Not* before the function expression tells Access to evaluate the converse of the True/False value returned from the function. So, this condition will be true only if the form is not loaded. If the companion CityInformation form isn't open, there's nothing to synchronize, so the macro action inside the If block—`StopMacro`—executes and the macro ends.

Now that we know the companion CityInformation is open, we need to decide whether the value on which that form is filtered is valid. Remember, when you created the `SeeCityInformation` macro that opens the CityInformation form, you included a Where Condition to filter what's displayed in the CityInformation form to match the city in the current record in the WeddingList form. However, it's a bad idea to reference an empty value in a Where Condition argument. In fact, in some cases you'll get an error message. When you move beyond the last row in the WeddingList form or click New Record under the Go To button in the Find group on the Home tab, you'll be in a new blank row in which the City field has no value. In this case, if you force the CityInformation form to refresh, it will go blank because it's a read-only form and there will be no rows returned if the filter compares to an empty value.

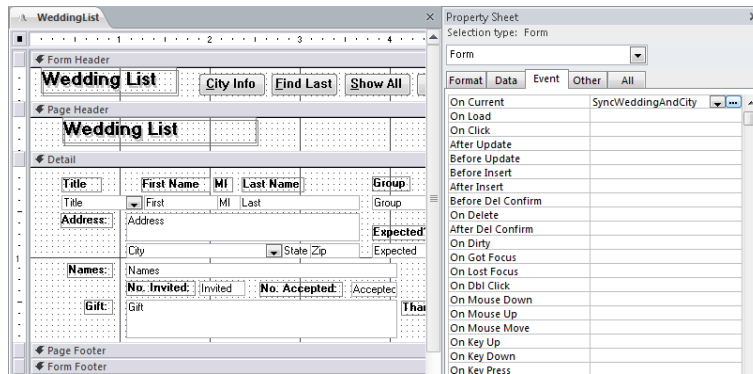
It probably makes more sense to test for an empty, or Null, value and hide the companion form if you're in a new row in the WeddingList form. The second If block conditional expression in this macro uses the `IsNull` built-in function to check for this condition. If City is Null, the macro hides the CityInformation form by using the `SetValue` action. We reference the CityInformation form's `Visible` property in the `Item` argument and use `False` in the `Expression` argument to hide the form. After the macro hides the CityInformation form, the macro ends because the rest of the macro actions exist inside the Else block. Note that the form is still open even though you can't see it. If you move back to a row in the WeddingList form that contains data, this macro executes again, but the actions to hide the CityInformation form will be skipped because the City field won't be Null anymore.

The CityInformation form displays the city details for the current record in the WeddingList form because your macro opened the CityInformation form with a filter pointing to the City control on the WeddingList form. However, the CityInformation form doesn't "know" when you move to a different record in the WeddingList form, so Access never reapplies the filter. Access does save the Where Condition argument you specified in the `Filter` property of the CityInformation form. To display the appropriate city information when the user moves to a new record in the WeddingList form, all you need to do is requery the CityInformation form

to make Access reevaluate the filter. In the Else block of the second If condition, the macro selects the CityInformation form to make sure it has the focus (this also reveals the form if it was hidden) and then executes a Requery action with no value specified in the Control Name argument. With no control name specified, Access knows to requery whatever form or report has the focus.

Finally, the SetTempVar action sets a value that's tested by other macros, and the SelectObject action ensures that the form has the focus after setting the value of the AddFlag temporary variable. We'll explain more about using SetTempVar in "Passing Status Information Between Linked Forms," on page 1245.

After you have the synchronization macro you need, save it as SyncWeddingAndCity. The last step is to associate the macro with the Current event of the WeddingList form. To do that, right-click the WeddingList form in the Navigation pane, and click Design View to open the form in Design view. Click the Property Sheet button in the Tools group on the Design tab to open the property sheet for the form, and then click the On Current property box. Use the list to select your SyncWeddingAndCity macro. (You'll find the example XmplSyncWeddingAndCity macro set in this property in the form.) Your screen should look like the one shown in Figure 20-32.



**Figure 20-32** Associate the SyncWeddingAndCity macro with the On Current event property of the WeddingList form.

When you finish, save and close the form. Open the form in Form view, double-click the City combo box control and move to the second record. Your screen should look like the one shown in Figure 20-30 on page 1234, assuming that Jane Crowley's record is the current one.

Test the macro by moving through the records in the WeddingList form. As you move from record to record, the data in the CityInformation form should change to reflect the city displayed in the current record of the WeddingList form. If you move to the blank record at the end of the recordset, the CityInformation form disappears. Move back to a row containing data, and it reappears.

Using a macro to synchronize two forms containing related data is a technique that works well with almost any set of forms, and you can use it in a number of situations. In the next section, you'll learn how to create a more complex macro set of named submacros within a macro object. When you arrange submacros by task into macro objects, you'll see that this is a good way to organize your work and to keep from cluttering your database with dozens of macro objects.

## Validating Data and Presetting Values

Two tasks you'll commonly automate in your applications are validating data that a user enters in a field and automatically setting values for specific fields. You'll now explore several macro objects saved in the sample database and learn how they perform these tasks on both the WeddingList form and the CityInformation form.

### Validating Data

A problem you'll often encounter when you create database applications is ensuring that the data the users enter is valid. Three types of invalid data are unknown entries, misspelled entries, and multiple versions of the same entry:

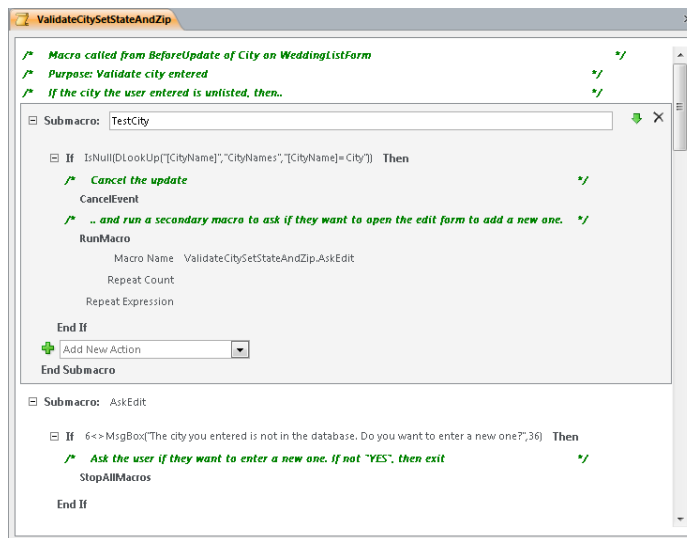
- **Unknown entries** A good example of this error is an entry such as AX in a state field. No state name is abbreviated as AX, but a user who tries to enter AZ might accidentally hit the X key instead of the Z key.
- **Misspelled entries** This sort of error is quite common among users with poor typing or spelling skills and among very fast typists. In this case, you might see entries such as *Settle*, *Seatlle*, or *Saettle* for Seattle.
- **Multiple versions** These errors are common in poorly designed databases and in databases that are shared by a number of users. You might see entries such as *ABC Company, Inc.*; *ABC Company, Incorporated*; *ABC Co., Inc.*; or *A B C Company Inc.*

You can use macros to validate data and help reduce errors. In the next section, you'll create a macro for the WeddingList form that validates the city that the user enters in the City field. If the city doesn't exist in the CityNames table, the macro then executes the following steps:

1. It displays a message indicating that the city is currently unlisted and asks whether the user wants to enter a new city name.
2. If the user wants to create a new city record, another macro runs that opens the CityInformation form in data entry mode and copies the city name that the user just typed.
3. If the user successfully saves a new row, a macro associated with the AfterInsert event of the CityInformation form sets a temporary variable.
4. Back in the WeddingList form, the city name gets revalidated, and if the city entry is a new one, a macro triggered by the AfterUpdate property of the City field sets the combo box to the new name. When the city name is validated, this macro also automatically enters the state name and the first three digits of the ZIP code.

## Understanding the ValidateCitySetStateAndZip Macro

In the Navigation pane, find the ValidateCitySetStateAndZip macro and open it in Design view. Figure 20-33 shows the first submacro and its associated actions.



**Figure 20-33** This figure shows the macro design surface for the first submacro in the ValidateCitySetStateAndZip macro object.

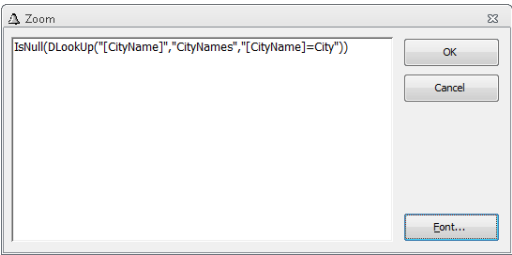
The first three lines of the macro are comments, and TestCity is the name of the first submacro in the object. You can see the actions for this submacro inside the If block listed in Table 20-3.

**Table 20-3** Actions, Arguments, and Settings in the TestCity Submacro

Action	Argument	Setting
CancelEvent		
RunMacro	Macro Name	ValidateCitySetStateAndZip.AskEdit

To understand how this macro works, let's look at the conditional expression in the If block that validates the city name for the TestCity submacro. What we want to do is look up the name just entered in the CityName field to find out whether it exists in the CityNames table. If it doesn't exist, the first action inside the If block of the submacro executes a CancelEvent action. The second action then calls another macro that we'll examine later.

To see this conditional expression easily, click into the expression text box for the If block and then press Shift+F2 to open the expression in the Zoom box, as shown in Figure 20-34.



**Figure 20-34** The conditional expression in the TestCity submacro If block uses the DLookup function to try to find the city in the CityNames table.

This conditional expression uses two built-in functions: DLookup and IsNull. The DLookup function looks up the city name in the CityNames table. The IsNull function checks the return value of the DLookup function. If the DLookup function doesn't find the city name, it returns a Null value. This causes the IsNull function to return a True value because the return value of the DLookup function is indeed Null. If no row in the CityNames table matches the current city name in the WeddingList form, Access then executes the actions inside the If block because the condition evaluated to True. In this case, the CancelEvent macro action tells Access not to store the new value in the City field. So if the city doesn't exist in the CityNames table, the RunMacro action inside the If block calls the AskEdit submacro, which we'll look at in a moment.

On the other hand, if the DLookup function does find the city name, it returns the city name to the IsNull function. The IsNull function then returns a value of False because the return value of the DLookup function is not Null. Access disregards running any actions inside the If block, and since there is no Else or Elself associated with this If block, the submacro ends without taking any further action.

What's the point of all of this? If you open the WeddingList form in Design view, click the City combo box, and look at its event properties, you'll find this macro "wired" into the Before Update property. If you remember from the previous chapter, you can use the BeforeUpdate event of a form or control to verify what's about to be saved. If the data is not valid, you can cancel the event to tell Access not to save the change. This is exactly what the CancelEvent action of this submacro is doing.

When you don't cancel a BeforeUpdate event on a control, Access accepts the changes and gives you a chance to look at the result in the AfterUpdate event. You don't want to use the AfterUpdate event to validate data because the data has already been saved, but it's perfect for filling in other fields on the form based on what the user just entered. As you'll see later, this application uses AfterUpdate on this control to fill in the correct state and part of the ZIP code.

## INSIDE OUT

### Why Aren't We Using the NotInList Event to Test for a New City Name?

The NotInList event occurs when the user types a name that's not in the row source of a combo box. The CityNames table is the row source of the City combo box, so NotInList seems to be an ideal choice to detect a name that's not in the CityNames table. But for NotInList to work properly, you need to be able to return a response code to the event to let Access know whether you've handled the problem and inserted a new city name. You can do that only in Visual Basic code, not in a macro. We set the Limit To List property of the combo box to No so that the NotInList event never happens. By trapping the problem in the BeforeUpdate event of the combo box, we can test the value and take appropriate action without having to return a response code to Access. As you'll learn later in the Visual Basic chapters, the NotInList event is a much better choice so long as you can return a response code.

So what happens if the user enters a city name that's not yet in the database? The AskEdit submacro runs, and the first step it takes is to evaluate the conditional expression in the first If block at the beginning of the submacro. The conditional expression for the first If block is as follows:

```
6<>MsgBox("The city you entered is not in the ↵  
database. Do you want to enter a new one?",36)
```

You’ve seen the `MessageBox` action before. This conditional expression uses a built-in function called `MsgBox` that’s a lot more powerful. The `MsgBox` function lets you not only display a message but also specify what icon you want displayed, and it provides several options for buttons to display in the message box. You set these options by adding number selections and providing the result as the second argument to `MsgBox`. In this case, 36 is the sum of 32, which asks for a question icon, and 4, which requests Yes and No buttons. (Intuitive, isn’t it?) You can find all the option settings by searching for `MsgBox Function` in Access Help. For your convenience, we’ve listed all the option settings for the `MsgBox` function in Table 20-4. In addition, the function returns an integer value that depends on the button the user clicks in the message box. If you look at the `MsgBox Function` help topic, you’ll find out that when the user clicks Yes, `MsgBox` returns the value 6. Table 20-5 shows you the `MsgBox` return value settings. So if the user doesn’t click Yes, the action inside the first block—a `StopAllMacros` action—executes, and the macro ends. If the user does click Yes, the rest of the submacro executes. Table 20-6 lists all the actions and arguments for this submacro.

Table 20-4 Option Settings for the `MsgBox` Function

Value	Meaning
BUTTON SETTINGS (CHOOSE ONE)	
0	OK button only
1	OK and Cancel buttons
2	Abort, Retry, and Ignore buttons
3	Yes, No, and Cancel buttons
4	Yes and No buttons
5	Retry and Cancel buttons
ICON SETTINGS (CHOOSE ONE)	
0	No icon
16	Critical (red X) icon
32	Warning query (?) icon
48	Warning message (!) icon
64	Information message (letter i) icon
DEFAULT BUTTON SETTINGS (CHOOSE ONE)	
0	First button is the default
256	Second button is the default
512	Third button is the default

**Table 20-5** Return Values for the MsgBox Function

Value	Meaning
1	OK button clicked
2	Cancel button clicked
3	Abort button clicked
4	Retry button clicked
5	Ignore button clicked
6	Yes button clicked
7	No button clicked

**Table 20-6** Actions, Arguments, and Settings in the AskEdit Submacro

Action	Argument	Setting
StopAllMacros		
Close	Object Type	Form
	Object Name	CityInformation
	Save	Prompt
OpenForm	Form Name	CityInformation
	View	Form
	Data Mode	Add
	Window Mode	Normal
SetValue	Item	[Forms]![CityInformation]![CityName]
	Value	[Forms]![WeddingList]![City]
GoToControl	Control Name	State
SetTempVar	Name	AddFlag
	Expression	True

The AskEdit submacro contains several actions that Access executes if the user enters the data for a new city name and responds by clicking Yes on the MsgBox that asks whether the user wants to add the new city. The submacro uses the IsFormLoaded function you saw earlier inside the second If conditional expression to determine whether the CityInformation form is open. If it is, the submacro instructs Access to close the form. Next, Access opens



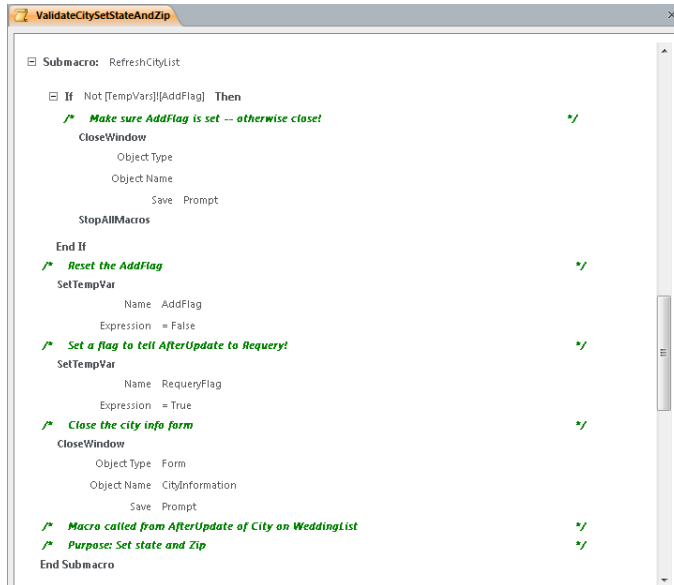
the CityInformation form in Add mode and copies the city name from the WeddingList form to the CityName field of the CityInformation form by using the SetValue action. (Note that SetValue has an exclamation mark icon on the macro design surface to the left of the action name indicating Access will not run this action in a database that is not trusted.) SetValue inserts the city name that the user typed for user convenience and to ensure that the user starts with the city name just entered. After the submacro copies the city name to the CityName field, it tells Access to move the focus to the State field using the GoToControl action. Finally, the submacro creates a temporary variable called AddFlag and sets the value to True to indicate that the CityInformation form is now opened in data entry mode. The submacro attached to the AfterInsert event checks this temporary variable to determine whether it should notify the AfterUpdate event of the City control on the WeddingList form to refresh its list.

## Passing Status Information Between Linked Forms

As you just saw, the AskEdit submacro creates a temporary variable called AddFlag to tell the CityInformation form's AfterInsert event macro that the WeddingList form needs to know whether a new row has been added successfully. Likewise, when the user adds a new row using the CityInformation form, the submacro that runs in response to an AfterInsert event (the event that Access uses to let you know when a new row has been added via a form) needs to check the flag and pass an indicator back to the submacro that responds to the AfterUpdate event of the City combo box on the WeddingList form. You'll learn in later chapters that you can also do this sort of "status indicator" passing by using variables in Visual Basic procedures.

Figure 20-35 shows the submacro that you need to respond to the AfterInsert event of the CityInformation form. You might recall from Chapter 19, "Understanding Event Processing," that Access triggers this event right after it has saved a new row. You could save the row by clicking Save in the Records group on the Home tab, moving to a new row, or closing the form. The If block at the beginning of the RefreshCityList submacro has a conditional expression that tests to be sure that the user asked to add a new row. The conditional expression is as follows:

```
Not [TempVars]![AddFlag]
```



**Figure 20-35** The RefreshCityList submacro sets a temporary variable to indicate that a requery is needed.

If the AddFlag temporary variable is not true, the actions inside the If block close the form, and then the StopAllMacros action causes the submacro to end. If the variable is true, the SetTempVar action creates another temporary variable called RequeryFlag and sets the flag to let the submacro that responds to the AfterUpdate event of the City combo box know that it must refresh the list in the combo box at its earliest opportunity. Finally, the submacro closes the CityInformation form. Remember that the AfterInsert event could be triggered as a result of clicking the form's Close button after entering new data. Normally, you would expect an error if you try to execute a Close command while the form is already in the process of closing (you will get an error in Visual Basic). Access does not generate any error from either of the Close actions in this submacro if this is the case.

If the user triggers the AfterInsert event by moving to another row, closing the form makes sense after adding the one row you need. If the user closes the form without entering any new data, the AfterInsert event won't happen. The user will be back in the WeddingList form with the unmatched city data still typed in the City combo box. If the user attempts to save the unmatched name again, the BeforeUpdate event runs the TestCity submacro that cancels the update when the city isn't in the CityName table. The user must either add the new value or enter a value in the list.

As a final touch, the SetTempVar action in the SyncWeddingAndCity macro that you created in Figure 20-31 (on page 1235) sets the AddFlag temporary variable to False when you move to a new row on the WeddingList form. When you have just moved to a new row, you clearly aren't worried about adding a new row to the CityNames table. Also, there's a SelectObject action in the macro to make sure the focus is back on the WeddingList form after the macro updates the temporary variable.

## Presetting Values

Validating data is just one of the many ways you can ensure data integrity in a database. Presetting values for certain fields is another way. Although you can set the Default property of a field, sometimes you'll need to set the value of a field based on the value of another field in a form. For example, you'll want to set the values of the State field and the Zip field in the WeddingList form based on the value of the City field. You can accomplish this with a macro.

In this section, you'll examine actions in the ValidateCitySetStateAndZip submacros that set the values of the State and Zip fields in the WeddingList form based on the city entered. If you scroll down the macro design surface, you can see the additional actions, as shown in Figure 20-36.

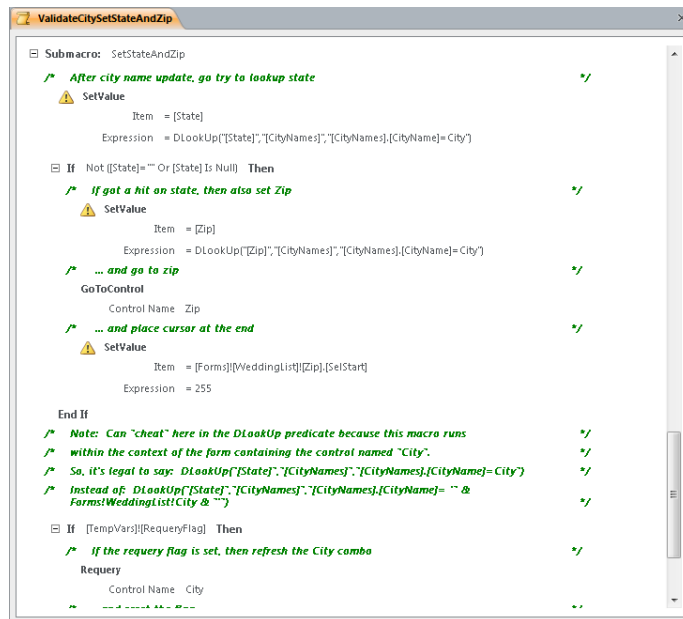


Figure 20-36 The SetStateAndZip submacro uses SetValue actions to automatically fill in the State and Zip controls.

Table 20-7 lists the actions and arguments in this submacro.

**Table 20-7** Actions, Arguments, and Settings in the SetStateAndZip Submacro

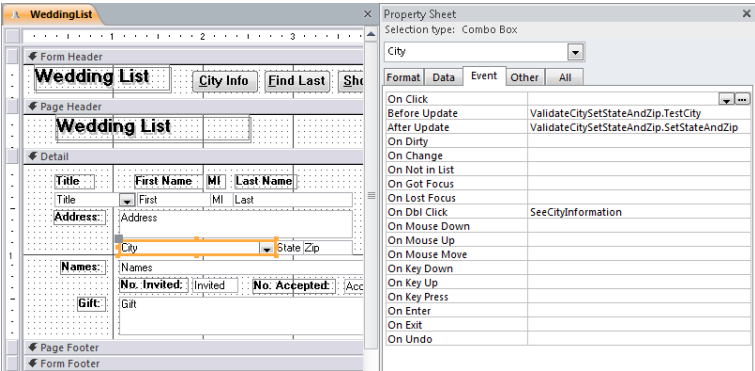
Action	Argument	Setting
SetValue	Item	[State]
	Expression	DLookup("[State]","[CityNames]","[CityNames].[CityName]=City")
SetValue	Item	[Zip]
	Expression	DLookup("[Zip]","[CityNames]","[CityNames].[CityName]=City")
GoToControl	Control Name	Zip
SetValue	Item	[Forms]![WeddingList]![Zip].[SelStart]
	Expression	255
Requery	Control Name	City
SetTempVar	Name	RequeryFlag
	Expression	False

When the user enters a valid city name, the first SetValue action uses the DLookup function to retrieve the matching State value from the CityNames table. If the value for State isn't blank or Null in the conditional expression for the first If block, the second SetValue action retrieves the first three digits of the ZIP code from the table, moves the focus to the Zip control with a GoToControl action, and sets the SelStart property of the Zip control to a high value (255) to place the cursor at the end of the data displayed in the control. Pressing the F2 key after you move to a control also places the cursor at the end of the data in the control, so you could use a SendKeys action here instead. However, setting the SelStart property is faster and more reliable. The user can now enter the last two digits of the ZIP code on the main form before moving on to the Expected field. The conditional expression in the first If block is as follows:

```
Not ([State]=" " Or [State] Is Null)
```

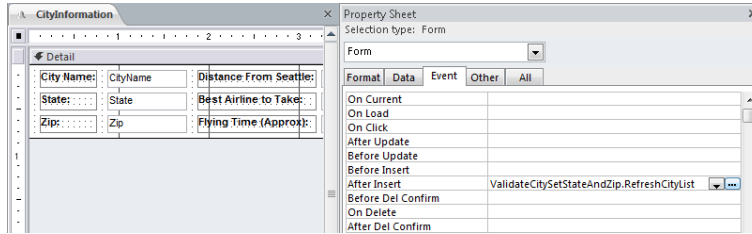
The set of submacros in this macro object is now complete. You can see how these submacros help implement data integrity by validating data and presetting specific values. This decreases the likelihood that users will make errors. Now, you'll see how to associate these submacros with the appropriate events on the WeddingList form and the CityInformation form.

Right-click the WeddingList form in the Navigation pane, and click Design View to open the form in Design view. Click the City combo box control, and then click the Property Sheet button in the Tools group on the Design tab. After the property sheet opens, click the Event tab. You should see the ValidateCitySetStateAndZip.TestCity submacro associated with the BeforeUpdate event of the City combo box. Remember, this is the macro you should run to verify whether the user has entered a valid city name. The AfterUpdate event property should be set to ValidateCitySetStateAndZip.SetStateAndZip. This submacro automatically sets the matching State and Zip values whenever the user specifies a new City value. Figure 20-37 shows the result.



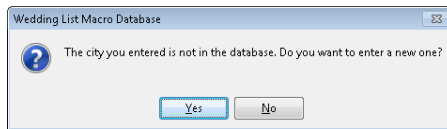
**Figure 20-37** The Before Update and After Update event properties for the City control on the WeddingList form are set to run submacros in the ValidateCitySetStateAndZip macro.

Close the WeddingList form. Open the CityInformation form in Design view, and click the Property Sheet button in the Tools group on the Design tab to open the property sheet. The ValidateCitySetStateAndZip.RefreshCityList macro is set in the form's After Insert event property, as shown in Figure 20-38. Recall from Chapter 19 that you could also use the form's AfterUpdate event to see changed data. However, in this case, you don't care about existing rows that change. The AfterInsert event is more appropriate because Access fires this event only when a new row is saved, not when an existing row is saved.



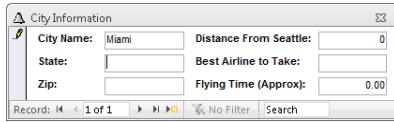
**Figure 20-38** The `ValidateCitySetStateAndZip.RefreshCityList` submacro executes when the After Insert event of the `CityInformation` form occurs.

Close the `CityInformation` form. Now that you've verified that the macros are associated with the appropriate objects and events, you're ready to test how this works. Begin by closing all open objects, and then double-click the `AutoexecXmpl` macro in the Navigation pane to run the macro and open the `WeddingList` form. Move to a new record in the `WeddingList` form, and enter a title, a name, an address, and a group. When the cursor moves to the City combo box, enter **Miami**. After you press Enter or Tab, Access runs the `ValidateCitySetStateAndZip.TestCity` submacro. Because this city doesn't currently exist in the `CityNames` table, the `AskEdit` submacro runs, and Access displays the message box shown in Figure 20-39.



**Figure 20-39** The `AskEdit` submacro displays a message box if you enter a new city.

After you click Yes, Access executes the remaining actions in the submacro. Access opens the `CityInformation` form in data entry mode, copies the city name to the City field of the form, and moves the cursor to the State field. Figure 20-40 shows the result of these actions.



**Figure 20-40** The AskEdit submacro then opens the CityInformation form, where you can enter the details of the new city.

After you enter information in the remaining fields and close the CityInformation form, the AfterInsert event of the form triggers the ValidateCitySetStateAndZip.RefreshCityList submacro. After the form closes, Access moves the focus back to the WeddingList form. When you finally leave the now valid City control, the macro triggered by AfterUpdate requeries the City combo box control and automatically updates the State and Zip fields.

In Chapter 25, we'll introduce you to some enhancements we made to the Wedding List sample database after we converted all the macros to Visual Basic. You can find this version of the database saved as WeddingList.accdb on the companion CD.



Article 6, on the companion CD, summarizes all the actions you can include in client and web macros. You'll find it useful to browse through that article to become familiar with the available actions and events as you automate your applications with macros.

# Index

## Symbols and Numbers

\$ (dollar sign), as character in format strings, 868, 870

3-D objects, setting color for, 791

### 64-bit Access VB applications

about, 1574–1575

LongLong data types, 1475, 1579–1580

LongPtr data types, 1576–1579

PtrSafe attributes, 1577

setting registry key to test upper memory, 1576

supporting previous versions of Access, 1577–1578

understanding pointer valued functions, 1578–1579

using .accde files, 1580–1581

using Declare statements, 1575–1576

VBA7 language updates, 1579–1580

### 64-bit version of Office 2010

installing, 1385–1387

### .accdb files (Access database)

about, 6

after publishing databases, 1295

architecture of, 125–128

compacting, 285–286

components of, 1494–1497

constructing literal expressions, 1780

crosstab queries in, 1774

DAO architecture and, 1497

DAO FindFirst vs. ADO Find methods, 1638

Database Tools tab and, 61–62

deleting, 1294–1295

extension to file name, 174

for web databases, 289

linking, 452, 482, 1716

making backup copies, 239–242

packaging and signing, 1741

photo size limit of, 1602

saving databases, 34

storing character fields using Unicode, 197

using data types in, 193

using SQL aggregate functions, 1780, 1837

Visual Basic procedures and, 1500

### .accde files

32-bit version compatibility, 1387

as execute-only files, 34, 1732–1733

compiled versions of, 460

in 64-bit Access VB applications, 1580–1581

working in 64-bit environment, 1580–1581

.accdw shortcut files, using, 1320

### .ade files

compiled versions of, 460

linking tables from, 485

Make ADE button, 1732

### .adp files (Access Data Projects)

about, 6

After Del Confirm events, 1172

application architecture, 1494, 1497

Before Del Confirm events, 1172

creating, 15, 125

creating execute only copies, 1732

data validity and, 8

defining USysRibbons table two load custom ribbons, 1681

importing menus and toolbars from source databases, 461

linking tables from, 485

objects in SQL Server databases, 1850–1851

packaging and signing, 1741

support for, 161

using double quotation marks ( " ") in literal expressions, 1781

using single quotation marks ( ' ') in literal expressions, 1780

### & (ampersand)

concatenating expressions with, 563

concatenating strings, in web databases, 327

using to display available characters, 870

### ' (apostrophe)

using an alphanumeric constants, 1780

### \* (asterisk)

as LIKE wildcard, 203

as wildcard, 328

description of, 566

record indicator icon, 596

using in web queries for SharePoint, 699

### \ (backward slash)

description of, 566

using to display character immediately following, 868

### .bdcm files

importing, 1366

### [ ] (brackets)

following exclamation point (!), 1507

in expressions, 564, 575, 1209

in query parameters, 660

using in desktop application, 1776

using in lists, 328

using to display color text, 868, 871



**^ (caret), description of, 566**

**, (comma)**

as separator, 472

as thousands separator, 867

**< < (double left arrow) button, in Available Fields list, 345**

**" " (double quotation marks). *See also* " " (quotation marks)**

using with text strings, 327

**> > (double right arrow) button, in Available Fields list, 345**

**= (equal sign)**

in field validation expressions, 327

meaning of, 328

**# Error**

avoiding, 1135

correcting, 489

**! (exclamation point)**

forces left-alignment, 868

forces placeholders to fill left to right, 870

separating table and field names with, 575, 576

using, 1507–1508

using in lists, 328

**/ (forward slash), dividing numeric expressions with, 566**

**>= (greater than or equal to sign), meaning of, 328**

**> (greater than sign)**

to display all characters in uppercase, 870

to indicate positive numeric values, 327

**<= (less than or equal to sign), meaning of, 328**

**< (less than sign)**

displaying all characters lowercase, 870

meaning of, 328

**.mdb files**

about, 6, 7

Access 2010 and, 7

client databases and, 289

command-line options, 1736

compiled versions of, 460

Complex Data concept and, 8

components of, 1494–1497

converting to current Access version, 1384

creating temporary variables in macros, 1219

DAO architecture and, 1497

Database Tools tab and, 61–62

defining data, 8

for web databases, 289

opening embedded macros, 1216

opening in current Access version, 109, 1499

packaging and signing, 1741

RDBMS and, 6

saving databases, 34

storing character fields using Unicode, 197

toolbar property

forms, 896

reports, 1127

using data types in, 193

using embedded macros, 1216

**.mde files**

32-bit version compatibility, 1387

as execute-only files, 34

compiled versions of, 460

importing menus and toolbars from source databases, 461

opening in current Access version, 109, 1499

working in 64-bit environment, 1580–1581

**- (minus sign), as character in format strings, 868, 870**

**< > (not equal to sign), meaning of, 328**

**( ) (parentheses)**

adding to expressions, 572

adding to SQL expressions, 1780

as character in format strings, 868, 870

**% (percentage sign), place at last character to be multiplied by 100, 868**

**. (periods)**

separating numbers and currency data types, 867

separating table and field names with, 575, 576

using, 1507–1508

**+ (plus sign)**

as character in format strings, 868, 870

description of, 566

using in web databases, 327

**# (pound sign)**

as LIKE wildcard, 203

as placeholder character, 868

as wildcard, 328

date/time literals, 1780

designating fill characters, 868

using in date and time values, 557

**? (question mark)**

as LIKE wildcard, 203

wildcard, meaning of, 328

**" " (quotation marks)**

creating string constants using, 563

fixing errors caused by importing delimited text, 479

using with literal strings, 1780, 1781

using with text strings, 472

**<Separator> option, 45**

**@ sign, using as placeholder character, 870**

**' ' (single quotation marks)**

creating string constants, 563

creating string constants in SQL, 1780

using with text strings, 472

**> (single right arrow) button, in Available Fields list, 345**

**# Type, avoiding, 1135**

## A

### Access

about, xxxv, 161–163

about security for, 47–48. *See also* Trust Center

architecture, 125–128

as application development system, 13–15

- as object-oriented programming environment, 785–789
- as RDBMS database, 6–13
- as Windows event-driven application, 1167–1169
- backward compatibility of, 1496
- commands
  - executing, 1550–1551
  - selecting, 41, 63
- components of application architecture, 1494–1497
- data types, 190–193
- evaluating arithmetic expressions, 576
- importing secured files, 462
- interface elements, 27
- maximum text data length, 481
- .mdb files and, 7
- new look, xxxvi, 23–24
- objects. *See* objects
- opening access for the first time, 21–23
- opening existing database, 25–27
- prior versions of
  - converting from, 1383–1385
  - opening forms with Navigation controls in, 1013
  - redesigning macro windows, from, 364–365
  - using embedded macros in, 1216
  - using older macros names from, 1228
- reporting problems to Microsoft, 37
- requiring Field row field names, 564
- submitting suggestions to Microsoft, 37
- system requirements, 1375–1376
- table design in, 168
- troubleshooting ODBC file sources, 458

**Access 2010**

- converting from prior versions, 1383–1385
- installing
  - sample files from companion CD, 1387
  - system requirements, 1375–1376
- installing Office with, 1376–1383

**Access About dialog box, 38**

**Access database (.accdb files)**

- about, 6
- after publishing databases, 1295
- architecture of, 125–128
- compacting, 285–286
- components of, 1494–1497
- constructing literal expressions, 1780
- crosstab queries in, 1774
- DAO architecture and, 1497
- DAO FirstFind vs. ADO Find methods, 1638
- Database tools tab and, 61–62
- deleting, 1294–1295
- for web databases, 289
- linking, 452, 482, 1716
- making backup copies, 239–242
- packaging and signing, 1741

- photo size limit of, 1602
- saving, 34
- storing character fields using Unicode, 197
- using data types in, 193
- using SQL aggregate functions, 1780, 1837
- Visual Basic procedures and, 1500

**Access Database Engine (ACE), 459**

**Access Database Engine (DBEngine)**

- about, 1494, 1496
- DAO hierarchy and, 1498–1501

**Access Data Projects (.adp files)**

- about, 6
- After Del Confirm events and, 1172
- application architecture, 1494, 1497
- Before Del Confirm events and, 1172
- creating, 15, 125
- creating execute only copies, 1732
- data validity and, 8
- defining USysRibbons table to load custom ribbons, 1681
- importing menus and toolbars from source databases, 461
- linking tables from, 485
- objects in SQL Server databases, 1850
- packaging and signing, 1741
- support for, 161
- using double quotation marks ( " ") in literal expressions, 1781
- using single quotation marks ( ' ') in literal expressions, 1780

**Access desktop applications. *See* Access database (.accdb files)**

**Access Options dialog box**

- configuring Name AutoCorrect Options, 227–229
- customizing look of Access using, 110
- modifying global settings by way of
  - about, 112–113
- Add-ins category, 54, 121–122
- Client Settings category, 118. *See also* Client Settings category in Access Options dialog box
- Current Database category, 113–114. *See also* Current Database category in Access Options dialog box
- Customize Ribbon category, 63, 69, 119–120, 897
- Datasheet category, 114–115
- Language category, 117–118
- Object Designers category, 115–116. *See also* Object Designers category in Access Options dialog box
- Proofing category, 116, 885
- Quick Access Toolbar category, 40–41, 46–47, 120–121, 1677
- Trust Center category, 52, 55, 122
- opening to customize Ribbon, 64
- Show Property Update Options Buttons, 229, 268
- Show Table Names option, 623
- Tabbed Documents settings, 111

**Access Services**

- about, 37
- assigning web forms as displays, 1289–1290
- avoiding modifying All Site Content page, 1323
- compiling web objects on servers, 1295
- creating Access Applications folders, 1294
- displaying timeout message, 1311
- extending to SharePoint server, 1322–1327
- feedback on validation rule failures, 1302
- functions, 1838
- instantiating templates, 1342–1345
- publishing databases to, 18–19, 157. *See also* publishing web databases
- reports navigation toolbar, 1305
- saving record changes, 1308–1309
- shell
  - about, 1312–1313
  - downloading web applications, 1314
  - reviewing Modify Application Page, 1318–1320
  - setting site permissions, 1315–1318
- supporting page breaks, 993
- viewing application settings, 1312
- waiting for server processing and, 1309–1310
- Web Compatibility Checker and, 351
- working with Recycle Bin, 1321–1322

**ACE (Access Database Engine), 459****ACE/JET wildcard characters, 1522****Acrobat, viewing exported web reports, 1306****Action Catalog**

- categories displayed in, 1256
- displaying untrusted action names, 1228
- searching, 1196
- This Database node in, 1260

**action queries**

- confirming action queries, 709
- delete
  - deleting groups of rows, 725–730
  - using with Before Delete, 396
- inserting data, 721–725
- make-table
  - creating, 714–717
  - running, 719–720
- SQL
  - DELETE statement, 1815–1816
  - INSERT statement (append query), 1816–1818
  - SELECT...INTO statement, 1819–1820
  - UPDATE statement, 1820–1822
- troubleshooting, 730–732
- updating groups of rows, 704–713
  - converting select queries to update queries, 706–707
  - generic update queries, 713
  - running update queries, 707–709

- testing with select query, 704–705
- updating multiple fields, 709–711
- using multiple tables or queries, 711–713
- updating web tables, 360
- using with OpenQuery macro, 1857

**ActiveX**

- applications
  - automation, 1585, 1862
  - linking name of file using, 314
  - Microsoft Graph feature, 681
  - Paint, 755–756
- objects
  - in forms, 753–756
  - OLE, 190
  - string data type and, 1475
  - using WithEvents keyword and, 1481
- troubleshooting calendar control, 1601

**ActiveX controls**

- getting access to, 794, 799
- in data manipulation, 9–10
- options in Trust Center dialog box, 53–54
- web forms and, 916

**ActiveX Data Objects (ADO). *See* ADO (ActiveX Data Objects)****ActiveX Data Objects (ADODB), 1502–1504****ActiveX objects**

- in tables, database limitations, 235

**add-in files, restrictions on, 53****Add-In Manager, 62****Add-ins category in Access Options dialog box, 54, 121–122****Add-Ins group, 62****Add New Action box, 372****Add New Action combo box, 1197, 1205, 1228, 1256, 1867****Add New Field heading, 176, 297****Administrator group, 62****ADO (ActiveX Data Objects)**

- about, 1497–1498
- architecture, 1502–1504
- fetching and modifying information, 447
- recordsets, working with, 1520–1524

**Adobe Acrobat, viewing exported web reports, 1306****Adobe Reader, viewing exported web reports, 1306****ADODB (ActiveX Data Objects ), 1502–1504****ADO Extensions for DDL and Security (ADOX), 1502–1504****ADO Find method, 1638****ADOX (ADO Extensions for DDL and Security), 1502–1504****ADP objects. *See also* Access Data Projects (.adp files)**

- macro actions, 1850–1851

**advanced form design**

- client forms
  - conditional formatting in, 978–985
  - creating multipage, 992–996
  - PivotChart, 996–999

- displaying option group values, 976–978
- many-to-one forms
  - creating, 946–947
  - designing, 948–952
- many-to-one queries, designing, 947–948
- navigation controls
  - about, 1000–1001
  - applying Quick Styles to buttons, 1012–1013
  - as collection of controls, 1003–1004
  - creating navigation buttons, 1004–1011
  - options, 1001–1002
  - troubleshooting dragging forms in Layout view, 1006
- subforms
  - about, 952–953
  - creating Main form, 969–972
  - creating subdatasheet, 973–976
  - designing first level, 962–963
  - designing innermost, 956–962
  - editing from subform controls, 971
  - embedding, 964–968
  - sizing controls, 965
  - specifying main source, 968
  - specifying source, 954–956
- tab controls
  - formatting properties, 991–992
  - setting order of, 877–878
  - troubleshooting seeing, 992
  - working with, 985–990
- using subreports in forms, 972
- using web browser controls, 1014–1019
- advanced report design**
  - adding PivotCharts, 1160–1163
  - adding values across a group, 1135–1136
  - building queries, 1108–1109
  - calculated values
    - adding page numbers, 1130–1131
    - adding print dates, 1129–1130
    - calculating percentages, 1141–1142
    - concatenating text strings, 1140
    - performing detail line calculations, 1132–1135
    - using conditional formatting, 1146–1151
    - using running sums, 1143–1146
  - creating grand totals, 1137–1138
  - creating reports, 1109–1111
  - defining grouping and sorting criteria, 1111–1114
  - hiding redundant values, 1138–1139
  - properties
    - for reports, 1119–1128
    - for report sections, 1115–1119
  - using subreports
    - about, 1151–1152
    - building subreport queries, 1155–1156
    - challenges of, 1152–1155
    - designing subreports, 1156–1157
    - embedding, 1157–1160
- After Del Confirm events, 1172**
- After Delete, 409–411**
- After Insert events, 398–403**
- After Update events, 404–408**
- aggregate functions**
  - description of, 647
  - descriptions of SQL, 1837
  - using as calculated values, 1135
  - using in desktop applications (accdb) files, 1780
- Alias argument, 394**
- alignment**
  - buttons, 811, 909
  - commands, 843
- Allow AutoCorrect option, 885**
- Allowed Value List Edits lookup property, 280**
- Allow Layout View property, 818**
- Allow Multiple Rows Per Reply check box, 502**
- Allow Multiple Values lookup property, 279**
- All Site Content page, 1323**
- All tab, properties in the property sheet, 892–896**
- Alternate Row Color button, 1064, 1116**
- American National Standards Institute (ANSI), 1522**
  - Access support for, 674
  - ANSI-standard VALUES clause, 1818
  - control characters, 189
  - embedded blanks in, 1776
  - ODBC in Access, 445
  - portability of Access and, 1773
  - SQL
    - Access support for, 674
    - DISTINCTROW equivalent, 1801
    - referencing output-column-name, 1781
    - wildcard characters, 1522
    - zeros, 1482
- ampersand (&)**
  - concatenating expressions with, 563
  - concatenating strings, in web databases, 327
  - using to display available characters, 870
- Analyze commands, 62**
- Anchoring button, 884, 903**
- anchoring, in web forms, 901–903**
- AND operator**
  - specifying, 201
  - specifying multiple comparisons using, 327
  - vs. OR, 557–560
- ANSI (American National Standards Institute). See American National Standards Institute (ANSI)**
- ANSI-standard SQL language, 674**
- ANSI-standard VALUES clause, 1818**
- API Declarations and Constants for Visual Basic, Windows, 1578**

**APIs (application programming interfaces)**

- accessing Windows, 1575
- calling VB in 64-bit values, 1576
- common Windows calls, 1578
- functions, Type statement and, 1492
- passing pointers to, 1578
- using LongLong data type interacting with, 1579

**apostrophe ('), using an alphanumeric constants, 1780****append queries**

- creating, 721–724
- running, 725
- SQL, INSERT statement, 1816–1818

**Append To row, 724****application development system, Access as, 13–15****Application Parts feature**

- to create tables, 178–182
- to create web tables, 300–304

**Application Parts Forms, 835–839****application programming interface (API). See APIs (application programming interfaces)****applications**

- ActiveX
  - automation, 1585, 1862
  - linking name of file using, 314
  - Paint, 755–756
- creating shortcuts, 1733–1737
- hybrid, 289
- reviewing Modify Application Page, 1318–1320
- Windows event-driven, 1167–1168
- working in web browsers. *See also* publishing web databases
  - about, 1299
  - downloading web, 1314
  - hiding a repositioning Web Report Toolbar, 1305
  - resizing and moving columns, 1308
  - understanding session management, 1311–1312
  - using Datasheet forms, 1306–1309
  - using web forms, 1299–1304
  - using web reports, 1304–1306
  - viewing application settings, 1312
  - waiting for server processing, 1309–1310

**ApplyFilter command, 1177****ApplyFilter macro action, 1856****architecture**

- Access, 125–128
- ADO, 447
- DAO, 447

**arguments**

- Alias, 394
- ByVal, 1526
- column, 365, 1194
- Control Name, 1238, 1244, 1248, 1267, 1866
- Data Mode, 1202, 1244, 1275, 1866
- declaring data type of, 1485–1486, 1528
- Description, 439

displaying boxes for, 1197–1198

entering, 364–365

Error Description, 379, 380, 383–384

Error Number, 379, 383

**Expression**

- AskEdit Submacro, 1244
- in web-supported macro actions, 1867
- SetLocalVar action, 428
- SetReturnVar data action, 432
- SetStateAndZip Submacro, 1248
- SyncWeddingAndCity, 1236

Filter, 1855

Filter Name, 1202

Form Name, 1244, 1866

for OpenForm client action, 1202–1203

Go To, 1220, 1221, 1224, 1225

Item, 1236, 1244, 1248

**Logic Designer**

- adding to, 1196
- displaying, 369

Macro Name, 1220, 1224, 1241, 1260, 1867

Message, 1198, 1213, 1221, 1867

**Name**

- AskEdit Submacro, 1244
- in web-supported macro actions, 1867
- RemoveTempVar macro action, 1219
- SetLocalVar macro action, 428
- SetReturnVar data macro action, 432
- SetStateAndZip Submacro, 1248
- SyncWeddingAndCity, 1236

not trusted, 1228

**Object Name**

- AskEdit Submacro, 1244
- BrowseTo macro action, 1275, 1276
- in web-supported macro actions, 1867
- SyncWeddingAndCit, 1236

**Object Type**

- AskEdit Submacro, 1244
- BrowseTo macro action, 1275
- in web-supported macro actions, 1867
- SyncWeddingAndCit, 1236

OnError action, 1220

Options, 1227

Order By, 1866

Page, 1275

passing no, 1545

Path To Subform Control, 1275, 1276–1277

Property, 1267–1268

Report Name, 1263, 1866

Save, 1212, 1227, 1244, 1866

SetStateAndZip Submacro, 1248

subroutines accepting, 1539

SyncWeddingAndCity, 1236

TestCity Submacro, 1241

- testing data types using fixed, 1474
- Title, 1198, 1213
- Type, 1198, 1213
- using mouse to display, 386
- Value, 1244, 1267
- View, 1202, 1227, 1244, 1855, 1857
- web supported macro actions and, 1866–1867
- Where
  - BrowseTo macro action, 1275
- Where Condition
  - as web-supported argument, 1866
  - BrowseTo macro action, 1275
  - empty values in, 1237
  - LookupRecord data block, 394
  - OpenForm macro action, 1202, 1233
  - OpenReport, 1855
- Window Mode, 1202–1203, 1244, 1866
- arithmetic expressions, 566–571**
  - operator precedence, 568–569
  - operators used in, 566
  - using DateDiff function, 566–568
- arithmetic functions, 1831–1832**
- arrays**
  - conversion function, 1832
  - declaring
    - static, 1492
    - within procedures, 1489–1490
  - determining Variants as, 1476
  - dimensions, 1481
  - loops and, 1542
  - ParamArray argument, 1526
  - passing, 1486
  - two-dimensional, 1836
  - using in example code, 1565
  - vs. counters, 1541
- arrow buttons**
  - Move Up/Move Down, 85, 86
  - Up, 390
- AskEdit submacro, using, 1244–1245**
- asterisk (\*)**
  - as LIKE wildcard, 203
  - description of, 566
  - meaning of wildcard, 328
  - record indicator icon, 596
  - using in web queries for SharePoint, 699
- Attachment button, 798, 801**
- attachment controls, 751**
- Attachment data type**
  - about, 8–9, 191
  - selecting, 193
- Attachments dialog box, 751–752, 770**
- Attachment web data type**
  - about, 312
  - using, 314
- attributes, 5**

- AutoCorrect Options, 116, 885**
- AutoexecXmpl macro, 1201–1202**
- Auto Expand in combo boxes, 833**
- AutoFormats, 897**
- AutoKeys macros, creating, 1712–1713**
- AutoNumber data type**
  - about, 190
  - converting to, 264
  - duplicate data in forms and, 773–774
  - in web tables, 315
  - selecting, 192
- Auto Order, to reorder controls, 878**
- Auto Resize property, 856**
- Available Fields list**
  - copying fields from, 345
- Avg function**
  - description of, 647, 1837
  - using in reports, 1135

## B

- Background button, 816**
- Background Color button, 810**
- Backstage view**
  - about, 27–29
  - Access Options dialog box. *See* Access Options dialog box
  - attributes, 1700–1701
  - closing, 39
  - controls, 1701–1703
  - Help tab, 37–39
  - Info tab, 29
  - New tab, 31–33, 169–170, 291
  - Print tab, 33
  - Recent tab, 29–31
  - Save & Publish tab, 34–37
  - Security Warning on, 50
- Back Up Database command, 240**
- backward slash (\)**
  - description of, 566
  - using to display character immediately following, 868
- BCS (Business Connectivity Services)**
  - about, 1349–1350
  - connecting data services, 1366–1373
  - exporting BCS Entity to XML file, 1365–1366
  - mapping Finder methods, 1363–1365
- BDC (Business Data Catalog)**
  - about, 1349
  - adding AccessControllist tag to model, 1371
  - creating model, 1365–1366
  - definition files, 1352–1353
    - creating, 1357–1363
    - working with, 1352–1355
  - importing model into Access, 1366–1373
  - saving model, 1370
  - updating model, 1371

**BDE (Borland Database Engine Closed), 482****Before Change events, 370–396**

- collapsing and expanding actions, 385–388
- defining multiple actions, 381–385
- including comments in, 372–373
- moving actions, 388–392
- occurrences of, 370–371
- preventing duplicate records, 392–396
- using Group construct, 373–374
- using If Blocks, 375–377
- using RaiseError data action, 378–380

**Before Del Confirm events, 1172****Before Delete event, 396–398****Between operator, 561–562****BETWEEN predicate, 1775–1776****blank space**

- as character in format strings, 868, 870

**Bold button, 810****Book button, 1644–1646****Boolean (true or false) values**

- AND and OR operators
  - specifying, 201
  - specifying multiple comparisons using, 327
- AND vs. OR, 557–560
- choosing control for, 833–835
- control buttons that hold
  - Check Box button, 797
  - Toggle Button, 797
- data type, Visual Basic, 1475
- NOT operator, 1797
- prefix naming convention, 1480
- SharePoint vs. Access, 313, 1283
- specifying Default Value field property, 316
- Yes/No data type
  - about, 190
  - limitations of, 265
  - using, 312, 313

**borders, setting border styles in forms, 891–892****Borland Database Engine (BDE), 482****Bound Column lookup property, 279****Bound Object Frame button, 798****brackets ([ ])**

- following exclamation point (!), 1507
- in expressions, 564, 575, 1209
- in query parameters, 660
- using desktop application, 1776
- using in lists, 328
- using to display color text, 868, 871

**breakpoints (stopping points) (VB)**

- setting, 1463–1464, 1469
- using, 1466

**Browse dialog box, 771****browsers. See web browsers****BrowseTo macro actions**

- Path argument guidelines, 1277
- using to browse web forms and web reports, 1275–1279
- vs. OpenForm, 1278

**build button, 966, 968****Builder button, 572, 941, 980****building complex queries, 630–634****built-in Access commands**

- selecting, 41, 63, 1550

**built-in menu commands (VB), 1549–1551****Business Connectivity Services (BCS)**

- about, 1349–1350
- connecting data services, 1366–1373
- exporting BCS Entity to XML file, 1365–1366
- mapping Finder methods, 1363–1365

**Business Data Catalog (BDC)**

- about, 1349
- adding AccessControlList tag to model, 1371
- creating model, 1365–1366
- definition files
  - creating, 1357–1363
  - working with, 1352–1355
- importing model into Access, 1366–1373
- saving model, 1370
- updating model, 1371

**buttons**

- ACCDE, 1732
- alignment, 811, 909
- Alternate Row Color, 1064, 1116
- Anchoring, 884, 903
- arrow
  - Move Up/Move Down, 85, 86
  - UP, 390
- Attachment, 798, 801
- Background, 816
- Background Color, 810
- Bold, 810
- Book, 1644, 1645, 1646
- Bound Object Frame, 798
- build, 966, 968
- Builder, 572, 941, 980
- Calendar, 1612
- Change Shape, 812
- Check Box, 797
- Combo Box, 796, 829
- command, 756–757, 795
- Conditional Formatting, 812
- Contact List navigation, 1005, 1008–1010
- Control Wizards, Use, 798, 829, 916
- Create E-Mail, 494–495
- Crosstab, 658
- Delete Rule, 982
- Export BDC Model, 1365
- Font Color, 909

Font Group, 810  
 footers/header group, 799  
 Format Painter, 810  
 Form Design, 969  
 header/footer group, 799  
 Hyperlink, Insert, 795  
 Image, 798  
 Image, Insert, 799  
 Import/Export customizations, 69  
 Insert ActiveX Control, 794, 799  
 Insert Chart, 796  
 Insert Hyperlink, 795  
 Insert Image, 799  
 Insert Or Remove Page Break, 796  
 Italic, 810  
 Label, 795  
 Line, 796  
 List Box, 797  
 Make ACCDE, 34  
 Make ADE, 1732  
 MsgBox Function, 1243–1244  
 Navigational Control, 795  
 navigation, creating, 1004–1011  
 ODBC Database, 448  
 Option, 798  
 Option Group, 796  
 PivotTable Tools, 687–688  
 Query Design, 947  
 Query Wizard, 641  
 Quick Styles, 811  
 Rectangle, 797  
 Redo, 1865  
 Remove Or Insert Page Break, 796  
 Report Wizard, 1110  
 Select, 795  
 Set Control Defaults, 798, 897, 898, 916  
 Shape Effects, 812  
 Shape Fill, 812  
 Shape Outline, 812  
 Show Property Update Options, 229, 268  
 Shutter Bar Open/Close, 71  
 Subform/Subreport, 798  
 Tab Control, 795  
 Text Box, 795  
 Toggle, 797  
 Totals, 645  
 Unbound Object Frame, 797  
 Underline, 810  
 Use Control Wizards, 798, 829, 916  
 web browser, 32  
 Web Browser Control, 795, 1016  
**ByRef keyword, 1486**  
**Byte data types, Visual Basic, 1475**  
**ByVal keyword, 1485–1486, 1536, 1539**

## C

**caching data locally, 1341**  
**Calculated data type**  
     about, 191  
     selecting, 193  
     vs. calculated expressions, 563  
**calculated fields, sorting on, 699**  
**calculated values**  
     adding page numbers, 1130–1131  
     adding print dates, 1129–1130  
     adding values across a group, 1135–1136  
     calculating percentages, 1141–1142  
     calculating totals on expressions, 1136  
     concatenating text strings, 1140  
     creating grand totals, 1137–1138  
     hiding duplicate values, 1138–1139  
     performing detail line calculations, 1132–1135  
     using aggregate functions, 1135  
     using conditional formatting, 1146–1151  
     using running sums, 1143–1146  
**Calculated web data type**  
     about, 312  
     as display values for Lookup fields, 345  
     using, 314, 319–327  
**Calendar buttons, 1612**  
**calendars, providing graphical, 1595–1602**  
**calendar view of SharePoint list data, creating, 1322–1327**  
**Call**  
     keywords, 1539  
     statement, 1538–1539  
**calls for**  
     Visual Basic in 64-bit values, 1576  
     Windows APIs, 1578  
**Caption field property, 316, 582**  
**captions**  
     changing form, 1270  
**caret (^), description of, 566**  
**Cartesian products, 622**  
**Cascade Delete Relationships, defining, 349–350**  
**categories, custom**  
     about, 78–80  
     creating, 83–84  
     display order rules, 86  
     hiding and renaming object shortcuts, 93–96  
     unhiding object shortcuts, 97–99  
**cells**  
     web form, splitting and merging, 912–914  
     web reports, splitting and merging, 1094  
**Change Shape button, 812**  
**CHAR[ACTER] SQL data type, 459**  
**Chart Wizard**  
     Insert Chart button, 796  
**Check Box button, 797**



check boxes, 746, 833–835

Check For Truncated Number Fields check box, 1075

Choice data type, creating in SharePoint list, 341

Choose Builder dialog box, 1214

class modules (VB)

about, 1454, 1529–1530

Property Get procedure, 1530–1532

Property Let procedure, 1532–1535

Property Set procedures, 1535–1538

ClearMacroError action, 1226

CLI (Common Language Interface), 446

client databases

about, 289

preparing for the web, 357–360

referencing data types, 320

client forms

conditional formatting in, 978–985

creating multiple-page, 992–996

drop-down list for Navigation Target Name property, 1005

PivotChart

building, 996–998

embedding linked, 998–999

using multiple data bar rules, 982

client objects, on web databases, 289

client reports

adding PivotCharts, 1160–1163

laying out, 1109–1111

using subreports

building subreport queries, 1155–1156

designing subreports, 1156–1157

embedding, 1157–1160

Client Settings category in Access Options dialog box

about, 118–119

confirming action queries, 709

displaying XML ribbon errors, 1674

keyboard shortcuts for entering data, 597

setting options for performance of linked tables, 483

setting table design options, 226–227

client tables

adding indexes, 222–226

creating in design view, 186–187

creating simple, 176–178

databases in

creating, 168–175

creating default templates, 230–233

limitations on, 235

defining fields

about, 187–189

choosing field names, 189–190

data types, 190–193

defining input masks, 204–207

field properties, 193–200

defining primary key, 208

defining relationships, 215–222

printing table definitions, 233–234

setting table design options, 226–229

subdatasheet properties, 213–214

using application parts, 178–181

using data type parts, 182–186

validation rules

defining field, 201–203

defining table, 209–211

client web forms

adding scroll bars, 876

adding smart tags, 879–881

controls for, 881–886

enabling and locking controls, 876–877

establishing standard design on, 897–899

formatting properties

about, 865–866

for Currency data types, 866–869

for date/time data types, 872–874

for Number data types, 866–869

for Text data types, 869–872

setting order of, 877–878

setting properties

allowing users to change views, 887

controlling updates, 890

defining pop-up and modal forms, 888–889, 891

preventing user from opening control menu, 891

properties on All tab, 892–896

setting border styles, 891–892

setting navigation options, 888

Clipboard

commands, 58

Office, 601

using cut to move databases, 245

Clipboard group

copying and replacing data, 600–602

using Cut command to unlink tables, 492

Close Database command, 29

Close event property, 1170

CloseWindow macro action, 1227, 1866, 1867

Code Builder, creating embedded macros, 1213

Collect data commands, 61

collecting data via email. *See* emails; *See* HTML forms;

*See* InfoPath forms

collections, referencing, 1505–1509

color names and codes, 1841–1847

Color Picker dialog box, choosing theme colors, 928–929

colors

adding color to report sections in Layout view, 1092

changing themes, 933–934

customizing colors in forms, 816–817

customizing gridlines, 883

custom property, 882

special effects and, 860–862

Colors dialog box, 862

**Color setting dialog box**

- changing background color for reports, 1116

**Column Count lookup property, 279****Column Heads lookup property, 279****Column-Name, 1776–1777****column-names in SQL table or query, 1780–1781****Column Widths lookup property, 279****Combo Box button, 796, 829****combo boxes**

- about, 748–749
- allowing space for scroll bars, 280
- as controls in, 829–833
- formatting, 978
- in web forms, 917
- keyboard shortcuts for, 767
- Lookup properties, 280

**Combo Box Wizard, 798, 829–832****comma (,)**

- as separator, 472
- as thousands separator, 867

**command buttons, 756–757, 795****Command Button Wizard, 798****command-line options, 1735–1736****commands**

- Access
  - executing, 1550–1551
- ApplyFilter, 1177
- Backstage view
  - Close Database, 29
  - Open, 29
  - Save, 28
  - Save Database As, 28
  - Save Object As, 28
- Back Up Database, 240
- Compact And Repair Database, 62, 285–286
- Copy, 600–602
- Cut, 492
  - for moving records, 601
- Database Tools tab
  - Add-Ins, 62
  - Administer, 62
  - Analyze, 62
  - Macro, 62
  - Move Data, 62
  - Relationships, 62
  - Tools, 62
- Delete, 244
- External Data tab
  - Collect Data, 61
  - Export, 61
  - Import & Link, 61
  - Web Linked Lists, 61
- Filter By Form, 782

**Gridlines, 919, 1100–1101****Home tab, 58–59**

- Clipboard, 58
- Find, 59
- Records, 59
- Sort & Filter, 58
- Text Formatting, 59
- Window, 59

**Insert Rows**

- above selected fields rows, 255–257
- in Indexes window, 225

**macro, 1858–1860****Manage Attachments, 751****Open, for reports, 1036****Paste**

- in Clipboard group, 601
- undoing, 259

**Paste Append, 601****Quick Access Toolbar**

- about, 39–40
- adding, 42, 47
- changing order of, 45, 47

**Quick Create, 819–823****Replace, 600****Report**

- completing web report created by, 1083
- modifying web report, 1077–1083
- using, 1066–1068

**Save & Publish**

- Publish To Access Services, 34, 36
- Save Database As, 34
- Save Object As, 34, 35–36

**Size To Fit, 845–847****SQL View, 445–446****Sync All, 1332–1333****system, 1861–1862****Undo, 244, 245, 259****Visual Basic**

- executing, 1464–1469
- menu, 1549–1551

**WindowHide, 1708****Work Offline, 1338****Common Language Interface (CLI), 446****Compact And Repair Database command, 62, 285–286****comparison predicate, 1777–1778****comparison symbols, in validation rules, 202****Complex Data concept, 8****complex data, validating**

- checking for duplicate names, 1605–1607
- checking for overlapping data, 1611–1613
- maintaining special unique values, 1610–1611
- testing for related records when deleting records, 1607–1608
- verifying prerequisites, 1608–1610

**complex queries**

- creating for web, 695–700
- customizing query properties
  - about, 663–664
  - applying to dynamically linked tables, 673–674
  - controlling output, 664–665
  - defining subdatasheets, 669–673
  - Record Locks property, 673
  - using Orientation property, 674
  - working with unique records and values, 665–668
- designing, 630–634
- in SQL view, 674–679
- joins
  - creating a inner, 622–630
  - outer joins, 634–641
- PivotCharts
  - about, 681–682
  - designing, 690–695
- PivotTables
  - about, 681–682
  - building a query for, 682–685
  - designing, 685–690
- totals queries
  - building crosstabs, 652–660
  - selecting records from groups, 650–652
  - totals within groups, 645–650
- using query parameters, 660–663
- using Query Wizard, 641
- using select queries to update data, 680–681

**complex reports**

- adding PivotCharts, 1160–1163
- adding values across a group, 1135–1136
- building queries for, 1108–1109
- calculated values
  - adding page numbers, 1130–1131
  - adding print dates, 1129–1130
  - calculating percentages, 1141–1142
  - concatenating text strings, 1140
  - hiding redundant values, 1138–1139
  - performing detail line calculations, 1132–1135
  - using conditional formatting, 1146–1151
  - using running sums, 1143–1146
- creating, 1109–1111
- creating grand totals, 1137–1138
- defining grouping and sorting criteria, 1111–1114
- properties
  - for reports, 1119–1128
  - for report sections, 1115–1119
- using subreports
  - about, 1151–1152
  - building subreport queries, 1155–1156
  - challenges of, 1152–1155
  - designing subreports, 1156–1157
  - embedding, 1157–1160

**concatenation**

- of expressions, 563
- of Null values, 565
- of strings, in web databases, 327
- of text strings, 1140

**conditional expressions in macros, 1207–1209****Conditional Formatting button, 812****conditional formatting rules**

- in the client forms, 978–985
- opening in prior Access versions, 981
- using in calculated values, 1146–1151

**Conditional Formatting Rules Manager dialog box, 979–985, 1146–1148****constants (VB)**

- about, 1474
- data types, 1475–1476
- declaring, 1477, 1478
- statements to define
  - Const, 1479
  - Enum, 1483–1485
  - ReDim, 1489–1490

**Const statement, 1479****Contact List navigation button, 1005, 1008–1010****control anchoring, in web forms, 901–903****Control Formatting group buttons, 811–812****control layouts, in web forms**

- about, 901–903
- creating label controls, 918
- moving controls within, 905–908
- removing in client forms, 911–912

**Control Name argument, 1238, 1244, 1248, 1267, 1866****control padding, in web forms, 921–922****controls group buttons**

- description of, 795–799
- locking, 800

**controls in forms**

- about, 745
- adjusting layout of, 849–850
- aligning, 852–856
- attachment control, 751–753
- changing defaults, 897
- check boxes, 746
- combo boxes, 829–833. *See also* combo boxes
- command buttons, 756–757
- enabling and locking controls, 876–877
- enabling Snap To Grid, 850–852
- image control, 754
- list boxes, 747–748
- moving, 808–810, 855
- navigation controls, 757–758
  - about, 1000–1001
  - applying Quick Styles to buttons, 1012–1013
  - as collection of controls, 1003–1004
  - creating navigation buttons, 1004–1011

- dragging forms in Layout view, 1006
- opening in Access prior versions with, 1013
- options, 1001–1002
- option buttons, 746
- option groups, 746, 976–978
- resizing using arrows, 853
- resizing using Auto Resize property, 856
- selecting multiple controls, 844
- sizing, 808–810, 844–849
- tab controls, 749–750, 985–992
- taste line, 844–847
- toggle button, 746
- unbound object from, 754
- using multiple data bar rules, 982
- web browser controls, 759–760, 1014–1018
- controls in macros**
  - embedding macros, 1212
- controls in web forms, working with events, 1260–1261**
- controls in web reports**
  - limitations of, 1105
  - moving to different sections, 1095
  - resizing in Layout view, 1096
- Control Wizards button, Use, 798, 829, 916**
- conversion errors, dealing with, 267–268**
- conversion functions, descriptions of, 1832–1833**
- Copy command, copying and replacing data, 600–602**
- counters, 1541**
- Count function, 647, 1135, 1837**
- Create A New Data Source To SQL Server Wizard, 450–451**
- Create E-Mail button, 494–495**
- Create Relationship Wizard, 301–304**
- Create Shortcut Wizard, 1734–1735**
- Create tab, 59–60**
- creating**
  - Access Applications folder, 1294
  - .adp files, 15, 125, 1732
  - append queries, 721–724
  - Application Parts Forms, 838
  - application shortcuts, 1733–1737
  - AutoKeys macros, 1712–1713
  - BDC model
    - definition files, 1357–1363
    - models, 1365–1366
  - calculated fields, 319–327
  - calendar view of SharePoint list data, 1322–1327
  - categories, 83–84, 86
  - Choice data type, 341
  - client forms, 992–996
  - client tables, 176–178, 186–187
  - Code Builder embedded macros, 1213
  - crosstab queries, 653–658
  - custom categories, 83–84
  - custom Data Type Parts, 1731–1732
  - custom Quick Access toolbar, 1703
  - custom Ribbons, 1665–1666, 1671
  - databases, 168–175, 1727–1730
  - database templates, 230–233, 1727–1730
  - data source linking ODBC databases, 448–451
  - Data Type Parts, 1731–1732
  - embedded macros, 1212–1215
  - embedded web macros, 1258–1260
  - equi-join queries, 622, 634
  - execute-only databases, 1732–1733
  - forms
    - multi-page client, 992–995
    - using Quick Create Commands, 819–823
    - with Form Wizard, 823–826
  - grand totals, 1137–1138
  - groups, 84–87
  - HTML forms. *See* email collection wizard
  - import procedures, 539–540
  - inner joins, 622–630
  - label controls, 918
  - links
    - to ODBC databases, 448–451
    - to tables, 1769
  - lookup fields, 337–341
  - macros for web, 1254–1260
  - Main form, 969–972
  - make-table queries, 714–718
  - many-two-one forms, 946–947
  - multiple-page client forms, 992–995
  - named data macros, 412–416
  - navigation buttons, 1004–1011
  - object shortcuts in groups, 87–91
  - outer joins, 634–636
  - PivotCharts, 681–682, 690–695
  - PivotTables, 681–682, 685–690
  - primary keys, 1762–1763
  - procedures in modules, 1458–1459
  - Quick Access toolbar, 1703
  - Quick Create Commands forms, 819–823
  - relationships, using lookup fields
    - about, 341–343
    - Cascade Delete Relationships, 349–350
    - Restrict Delete Relationships, 343–349
  - report labels, 1059–1061
  - reports, 1109–1111. *See also* reports
  - reports are shot data, 1772
  - Ribbons with XML, 1665–1666, 1671
  - shortcuts
    - for applications, 1733–1737
    - object, 87–91
  - snapshot data, reports, 1772
  - string constants, 563, 1780
  - subdatasheet subform, 973–976
  - submacros within objects, 1205
  - tables, 176–178, 186–187

**creating** (*continued*)

## templates

- default database, 230–233
- for databases, 1727–1730
- for forms, 897–899
- for tables, 180
- templates for databases, 1727–1730
- temporary variables in macros, 1219
- text expressions, 563–565
- unique identifiers, 1762–1763
- update queries, 711–713
- USysRibbons tables, 1667–1671
- VBA callbacks, 1691–1692
- web databases, 291–296
- web forms. *See* simple web forms
- web macros, 1254–1258
- web queries, 695–700
- web reports, 1077–1083, 1083–1086
- web tables
  - in Datasheet view. *See* Datasheet view
  - simple, 297–300
  - using Application Parts, 300–304
  - using Data Type Parts, 304–306

**Criteria row**

- in query parameters, 660
- looking for single values, 556
- looking for values in multiple rows, 559

**Crosstab button, 658****crosstab queries**

- about, 652–653
- Access database language for, 1774
- creating simple, 653–658
- in Access databases, 1774
- partitioning data in, 658–660

**Currency data type**

- about, 190
- converting to, 264
- formatting properties, 866–869
- selecting, 192
- Visual Basic, 1475

**Currency web data type**

- about, 312
- format property settings, 925–926

**Current Database category in Access Options dialog box**

- about, 113–114
- assigning web forms for Access Services, 1289–1290
- Check For Truncated Number Fields check box, 1075
- compacting databases, 286
- creating macros to run when database opens, 1201–1202
- enabling Windows themes in forms, 951–952
- keeping users out of Design and Layout views, 887
- limiting returned records, 618
- propagating changes in tables, 239
- setting table design options, 227
- web message box title, 1257

**custom categories and groups**

- about, 78
- creating categories, 83–84
- creating groups, 84–87
- creating object shortcuts in groups, 87–91
- display order rules, 86
- dragging and dropping objects into groups, 90
- hiding and renaming object shortcuts, 93–96
- hiding groups in categories, 91–93
- unhiding object shortcuts, 97–99
- viewing categories in Navigation pane, 101

**Custom Dictionaries, 116****Customize Ribbon category in Access Options dialog box, 63–65, 69, 119–120, 897****customizing**

- Application Parts, 300–304, 1728
- colors, 816–817
  - automatically setting, 882
  - gridlines, 883
  - in themes, 933–934
  - special effects and, 860–862
- email messages, 506–507
- fonts, 862–865, 935–936
- forms
  - adding lines, 857–858
  - adding scroll bars, 876
  - adding smart tags, 879–881
  - adjusting fonts, 862–864
  - allowing different views, 887
  - changing control defaults, 897
  - controlling changes to, 890
  - defining pop-up and/or modal forms, 888–889
  - defining templates, 897–899
  - defining window controls, 891
  - drawing rectangles, 858–859
  - enabling and locking controls, 876–877
  - examining alignment and size of controls, 843–844
  - lining up controls, 852–856
  - other client form properties, 892–896
  - other control properties, 882–885
  - setting border style, 891–892
  - setting tab order, 877–878
  - sizing controls, 844–849, 853
  - “Snapping” controls to grid, 850–852
  - specifying format for Date/Time, 872–875
  - specifying format for numbers and currency, 866–869
  - specifying format for text, 869–872
  - specifying format for Yes/No fields, 874–875
  - using colors and special effects, 860–862
- help with common typing mistakes, 116
- modules (VB), 1459–1461
- Navigation pane
  - display objects, 73–74
  - display order, 102

- display view for users, 96
- lists of objects, 93
- number of databases above Info Tab, 29
- query properties
  - about, 663–664
  - applying to dynamically linked tables, 673–674
  - controlling output, 664–665
  - defining subdatasheets, 669–673
  - Record Locks property, 673
  - using Orientation property, 674
  - working with unique records and values, 665–668
- Quick Access Toolbar, 39–47
- Ribbon, 63–69
  - exporting customizations, 69
- Ribbon, using XML
  - about, 1665–1666
  - building, 1671–1679
  - creating USysRibbons table, 1667–1671
  - loading, 1679–1682
- Visual Basic Editor, 1459
- web forms
  - about, 899–900
  - adding gridlines, 918–921
  - adding spaces with control padding, 921–922
  - control layouts and control anchoring in, 901–903
  - creating titles, 922–923
  - formatting column of controls, 909–910
  - inserting rows and columns, 914–916
  - lining of controls, 903–905
  - moving controls to different sections, 923–925
  - moving controls within layouts, 905–908
  - removing control layouts, 911–912
  - resizing controls, 910–911
  - setting control properties, 925–926
  - setting form properties, 927–928
  - Shared Resources feature, 938–944
  - splitting and merging cells, 912–914
  - using themes, 928–937
  - using web-compatible controls, 916–918
- custom queries, providing by forms, 1619**
- Cut command**
  - for moving records, 601
  - using to unlink tables, 492

## D

- DAO (Data Access Objects)**
  - about, 1497–1498
  - common objects with ADODB and ADOX, 1504
  - fetching ODBC databases, 447
  - manipulating data types using, 1516–1519
  - recordsets, working with, 1512–1516
  - support for, 1385
  - using FindFirst method, 1638
  - using in Access architecture, 1497–1501
- DAPs (data access pages), 127**
- data.** *See also* **databases**
  - analysis, 1754–1756
  - changing form, 776–777
  - checking for duplicate names, 1605–1607
  - checking for overlapping data, 1611–1613
  - cutting, 601
  - deleting data and testing for related records, 1607–1608
  - deleting data in forms, 776
  - delimited, 472–473
    - importing, 472–473
  - duplicate data in forms, 773
  - event properties
    - for changing data, 1172–1173
  - exporting, 1823–1829
  - finding records across date spans, 637–638
  - fixed-width data
    - importing, 473–474
  - importing
    - about, 452–453
  - keyboard shortcuts for entering, 766–767
  - linking. *See* linking files
  - maintaining special unique values, 1610–1611
  - moving to database software, 15–17
  - normalization of. *See* normalization of data
  - randomly load data complex procedure, 1553–1568
  - replacing, 600
  - searching for and filtering in Datasheet view, 612–619
  - searching form, 777–779
  - selecting before changing, 599–600
  - selecting from single tables
    - about, 548–551
    - entering selection criteria. *See* selection criteria, single table
    - setting field properties, 553–555
    - specifying fields, 551–552
  - selection
    - automating, 1615–1619
    - providing custom queries by forms, 1619–1627
  - sorting
    - form data, 780–783
    - in Datasheet view, 608–610
    - in queries, 583–585
  - storing dates and times, 556–557
  - validating
    - using macros, 1239–1245
    - using preset values, 1247–1251
    - verifying prerequisites, 1608–1610
    - viewing on forms, 763–764
    - vs. information, 1751
- Data Access Objects (DAO), 447.** *See also* **DAO (Data Access Objects)**
  - use in Access architecture, 1497–1501
- data access pages (DAPs), 127**

**data attributes, changing**

- about, 261
- data lengths, 266–267
- data types, 262–266
- dealing with conversion errors, 267–268

**data bar rules, 981, 982, 983**

- conditional formatting rules, troubleshooting, 1150

**database applications, designing**

- analyzing tasks, 1749–1751
- application design strategy, 1747
- capturing point-in-time data, 1771–1772
- concepts for
  - about, 1757
  - efficient relationships, 1768–1769
  - problems with waste, 1757–1760
  - rules for table design, 1760–1768
- creating report snapshot data, 1772
- data analysis, 1754–1757
- fundamentals of, 1743–1747
- improving performance of critical tasks, 1770
- organizing tasks, 1753–1754
- selecting data, 1751–1753

**Database Documenter, 233–234****database files**

- importing
  - about, 452–453
  - dbase files, 453–457
  - SQL tables, 456–459
  - vs. linking, 451–452
- opening in exclusive or shared mode, 521

**database objects. *See also* objects**

- description of macro actions, 1854–1856
- Visual Basic procedures and, 1500

**Database Properties dialog box, 29****databases**

- about, 4–7
- backing up, 239–242
- compacting, 285–286
- creating execute-only, 1732–1733
- creating new, 168–175, 230–233
- creating new web, 291–296
- creating templates, 1727–1730
- enabling an untrusted database, 49–51
- encrypting, 1737–1738
- exploring desktop. *See* desktop databases
- exporting to, 1823–1824
- importing
  - Access objects, 459–462
  - secured files, 462
- in Access, 125
- ODBC, exporting to, 1827
- packaging and signing, 1739–1742
- setting startup properties, 1706–1708
- using copying to move, 245

**Database Splitter Wizard, 1717–1720****Database Tools tab, 61–62****Database window, 71, 73. *See also* Navigation pane****data control**

- about, 6
- data sharing and, 12–13

**data definition**

- about, 6
- and storage, 7–9

**data entry operations macro actions, 1851****Data Execution Prevention (DEP) mode, settings for, 54–55****data import/export macro actions, 1852–1854****data lengths, changing, 266–267****data macros, 361–442**

- debugging, 438–441
- design facility, overview, 364–369
- sharing logic, 442–443
- understanding recursion, 441–442
- uses of, 362–364
- working with after events, 398–410
  - After Delete, 409–411
  - After Insert, 398–403
  - After Update, 404–408
- working with before
  - Before Change events. *See* Before Change events
- working with before events, 369–398
  - Before Delete, 396–398

**data manipulation, 6, 9–12****Data Mode argument, 1202, 1244, 1275, 1866****Data Services option, troubleshooting, 1367****Datasheet category in Access Options dialog box, 114–115****Datasheet forms**

- resizing and moving columns, 1308
- using in web browsers, 1306–1309

**Datasheet view****changing data**

- adding new records, 596–599
- copying and replacing data, 600–602
- deleting rows, 602–603
- replacing data, 600
- selecting before, 599–600
- understanding record indicators, 596
- creating calculated fields, 319–327
- creating web tables in
  - about, 306–308
  - creating calculated fields, 319–327
  - defining field validation rules, 327–331
  - defining web fields, 307–311
  - setting field properties, 315–318
  - understanding web data types, 311–315

**filtering data, 613–619****keyboard options, setting, 597–599****keyboard shortcuts**

- entering data, 597
- for scrolling, 591

- selecting data, 592
  - setting keyboard options, 597
- opening tables, 133
- Query window in
  - about, 141
  - checking field properties, 554
  - setting field properties, 553
  - working with subdatasheets, 592–595
- saving Filter By property in, 337
- searching for data, 612–613
- setting field properties in, 553
- sorting data, 608–611
- table windows in, 137–138
- using subforms in, 962
- working with hyperlinks, 603–607

**Data Source dialog box, creating data source using, 448**

**data sources linking ODBC databases, creating, 448–451**

**Data Type Parts, 182–186, 1731–1732**

#### data types

- Attachment
  - about, 8–9, 191
  - selecting, 193
- Attachment web
  - about, 312
  - using, 314
- AutoNumber
  - about, 190
  - converting to, 264
  - duplicate data in forms and, 774
  - in web tables, 315
  - selecting, 192
- Boolean, Visual Basic, 1475
- Byte, Visual Basic, 1475
- Calculated
  - about, 191
  - selecting, 193
  - vs. calculated expressions, 563
- Calculated web
  - about, 312
  - as display values for Lookup fields, 345
  - using, 314, 319–327
- changing, 262–266
- CHAR[ACTER] SQL, 459
- Choice, creating in SharePoint list, 341
- Currency
  - about, 190
  - converting to, 264
  - formatting properties, 866–869
  - selecting, 192
- Currency web
  - about, 312
  - format property settings, 925–926
- DATE SQL, 459

- Date/Time
  - about, 190
  - converting to, 264
  - formatting properties, 872–874
  - troubleshooting setting defined default value for, 1596
  - understanding, 192
- Date/Time SharePoint, 313
- Date/Time web
  - about, 312
  - format property settings, 926
  - using, 313
- Date, Visual Basic, 1476
- Decimal, Visual Basic, 1475
- description of VB, 1475–1476
- DOUBLE SQL, 459
- Double, Visual Basic, 1475
- field, 188, 190–193, 266
- FLOAT SQL, 459
- Hyperlink
  - about, 191, 603–604
  - activating hyperlinks, 604–605
  - converting to, 263
  - editing hyperlinks, 607
  - fixing email, 1594
  - inserting new hyperlinks, 605–606
  - selecting, 192
- Hyperlink web
  - about, 312
  - using, 314
- IMAGE SQL, 459
- Integer, Visual Basic, 1475
- INT SQL, 459
- LongLong, Visual Basic, 1475, 1579–1580
- LongPtr, Visual Basic, 1475, 1576–1577
- Long, Visual Basic, 1475
- Lookup & Relationship web, 312
- manipulating, using DAO, 1516–1519
- Memo
  - about, 190
  - converting to, 262
- Memo web, 312
- Number
  - about, 190
  - converting to, 263
  - formatting properties, 866–869
  - selecting, 191
- Number web
  - about, 312
  - format property settings, 925–926
- Object, Visual Basic, 1476
- OLE Object
  - about, 190
  - selecting, 192
- REAL SQL, 459



**data types** (*continued*)

Single, Visual Basic, 1475

SMALLINT SQL, 459

String, Visual Basic, 1475

testing, 1474

Text

about, 190

converting to, 262

selecting, 191

TEXT SQL, 459

Text web

about, 312

using, 313

TIME SQL, 459

TIMESTAMP SQL, 459

TINYINT SQL, 459

User-defined, Visual Basic, 1476

VARCHAR SQL, 459

Variant, Visual Basic, 1476

web, 311–315

Yes/No

about, 190

limitations of, 265

using, 312, 313

**DateAdd** (interval, amount, date) function, 581**Date And Time** dialog box, 799, 1102–1103, 1130**date and time** functions, 581**Date** data type, Visual Basic, 1476**DateDiff** function

in arithmetic expressions, 566–568

in Expression Builder, 574–580

**Date()** function, 581, 1129**DatePart** (interval, date) function, 581**DATE** SQL data type, 459**Date/Time** data type

about, 190

converting to, 264

formatting properties, 872–874

troubleshooting setting defined default value for, 1596

understanding, 192

**Date/Time** functions, 1833**Date/Time** SharePoint data type, 313**Date/Time** web data type

about, 312

format property settings, 926

using, 313

**date** values, formatting, 201**Day(dates)** function, 581**dBase** files

accessing, 9, 454

as IN clause source, 1787–1788

exporting long field names to, 1825

exporting spreadsheets to, 1824–1825

importing, 453–456

linking, 481–482

linking to, 487–488

troubleshooting ODBC file sources, 458

**DBEngine** (Access Database Engine)

about, 1494, 1496

DAO hierarchy and, 1498–1501

**debugging tools** (Visual Basic)

breakpoints (stopping points)

setting, 1463–1464, 1469

using, 1466

examining procedure call sequences, 1473–1474

using Immediate window, 1464–1469

using Watch window, 1469–1472

**Decimal** data types, Visual Basic, 1475**Declare** statements, 1575–1576, 1578**declaring**

arrays within procedures, 1489

constants and variables, 1477–1478

data type for arguments, 1485–1486, 1528

new subroutines, 1527–1528

new VB functions, 1525–1526

static arrays, 1492

**Default Database Folder** box, 113**Default Value** field property, 316**delete** action queries

deleting groups of rows, 725–730

using with Before Delete, 396

**Delete** command, 244**Delete** Rule button, 982**DELETE** statement, 1815–1816**deleting**

data in forms, 776

embedded macros, 1215–1216

fields, 260–261

groups of rows, 725–730

in update queries, 704–710

named data macros, 418–419

records in InfoPath, 526

rows in Datasheet view, 602

tables, 244–245

**delimited** data

fixing errors caused by, 479

setting up the import, 472–473

**DEP** (Data Execution Prevention) mode, 54–55**Description** argument, 439**Description** property

for table fields, 188

for web fields, 309

**Design** view

creating tables in, 186–187

customizing forms in

adjusting layout of controls, 849–850

enabling Snap To Grid, 850–852

- examining alignment and size of controls, 843–844
- lining up controls, 852–856
- sizing controls, 844–849, 853
- fixing errors from importing text files, 479
- keeping users of, 887
- Query window in
  - about, 139–141
  - changing size of fonts for, 678
  - clearing design grid, 552
  - creating web queries, 697, 698
  - designing expressions, 699
  - opening new window, 947
  - opening query property sheet, 663, 670
  - selecting all fields, 805
  - selecting data for single table, 549–550
  - selecting fields for single table, 551–552
  - specifying field names, 582–583
  - using inner joins, 624–625
  - using keyboard for moving between Windows, 257
- report windows in, 149–152
- setting field properties in, 193–199
- table properties in, 212–215
- table windows in, 134–136
- desktop applications**
  - creating shortcuts, 1733–1737
  - designing for client-server, 1716–1717
- desktop databases.** *See also* **accdb files**
  - opening, 128–132
  - using Database Splitter Wizard, 1717–1720
  - using linked tables, 1716, 1719–1724
- detail sections, in reports, 1028–1030**
- dialog boxes**
  - Access About, 38
  - Access Options
    - configuring Name AutoCorrect options, 227–229
    - customizing look using, 110
    - modifying global settings by way of. *See* Access Options dialog box
    - opening to customize Ribbon, 64
    - Show Property Update Options Buttons, 229
    - Show Table Names option, 623
    - Tabbed Documents settings, 111
  - Attachments, 751–752, 770
  - Browse, 771
  - Choose Builder, 1214
  - Color Picker, choosing theme colors, 928
  - Colors, 862
  - Color setting
    - changing background color for reports, 1116
  - Conditional Formatting Rules Manager, 979–985, 1146–1148
  - Database Properties, 29
  - Data Source, 448
  - Date And Time, 799, 1102–1103, 1130
  - Documenter, 233–234
  - Edit Relationships, 221–222
  - Enter Table Properties, 336
  - Enter Validation Message, 334–335
  - Export - dBase File, 1824
  - Export - Excel Spreadsheet, 1824
  - Expression Builder, 321–325
  - File Download, 1314
  - File New Database, 171–172
  - File Open
    - importing dBase files, 454
    - linking dBase files, 488
  - Find And Replace
    - FindNextRecord macro actions and, 1856
    - FindRecord macro actions and, 1857
    - replacing data, 600
    - Search box vs., 779
    - searching for data, 612–613, 777–779
  - Get External Data - Access Database, 484–486
  - Get External Data - dBASE File
    - importing dBase files, 453
    - linking dBase files, 487–488
  - Get External Data - Excel spreadsheet, 488–489
  - Get External Data - ODBC Database, 448, 457, 490
  - Insert Picture, 799
  - Join Properties
    - changing properties for queries, 635
    - defining links on multiple tables, 625
  - Manage Attachments, 798
  - Microsoft Access, 771
  - Microsoft Office Genuine Advantage confirmation, 172–173
  - Microsoft Office Security Options, 52
  - Modify Button, 43–44
  - Navigation Options
    - exploring, 80–82
    - hiding custom groups, 91–92
  - New Formatting Rule, 982
  - Open
    - database files in exclusive mode, 521, 1737
    - database files in shared mode, 521
    - finding and opening existing database files, 25
  - Options
    - customizing VBE, 1459
    - modifying settings for debugging VB code, 1461
    - setting custom colors for code elements, 1460
    - using Option Explicit statements, 1477
  - Page Numbers, 1131
  - Page Setup, 1118, 1148
  - Personalization
    - setting color for 3-D objects, 791
  - Print, 33
  - Privacy Options, 21–23

**dialog boxes** (*continued*)

## Properties

- changing Hidden property, 98
- changing PivotTable field captions, 689–690
- customizing query properties, 664
- defining properties for printers, 1042
- in Navigation pane, 94
- modifying shortcut targets, 1734
- running programs in compatibility mode, 1737

## Publish Failed, 1298

## References

- selecting Access database engine object library, 1497
- selecting ActiveX Data Objects Library, 1504

## Rename

- changing name of custom tabs, 65

## Resolve Conflicts, 1340

## Save As, 178

## server login, 1827

## Show Table

- adding fields lists, 954
- building report queries, 1046
- creating web queries, 697
- defining Subdatasheets, 669
- making SQL default option, 677
- opening database for first time, 216
- seeing defining relationships, 222
- selecting tables or queries, 548–550

## Single Step, 1861

## SQL Server Login

- displaying for SQL data source, 490
- using Trusted Connection in, 1827

## Subform Linker, 966–967

## Trust Center, 52–55

VBE Options. *See* Options dialog box

## WCF Connection, 1359

## Windows Choose File, 752

## Windows Color and Appearance, 791

## Workflows, 1861

## Workflow Tasks, 1861

## Zoom, 803, 1055–1056, 1298

**Dialog Box Launchers, 57****Dictionaries, Custom, 116****Dim statement, 1480–1483****Display control lookup property, 278****DISTINCTROW function, 1801****DoCmd object, 1549–1550****Documenter dialog box, 233–234****documents**

## word processing

- defining contents, 7
- import/export data, 9
- RDBMS vs., 12

**Document Window Options section in Access Options dialog box, 110–111****dollar sign (\$), as character in format strings, 868, 870****Do...Loop statement, 1539–1540****Double data types, Visual Basic, 1475****double left arrow (<<) button, in Available Fields list, 345****double quotation marks (" ")**

- creating string constants using, 563
- fixing errors caused by importing delimited text, 479
- using in literal strings, 1780, 1781
- using with text strings, 472

**double right arrow (>>) button, in Available Fields list, 345****DOUBLE SQL data type, 459****drag and drop fields in Query window, 551****Draw application**

- OLE Object data type and, 192

**drawing tools, 857–859****duplicate**

- contact table names, 1605–1607
- rows in databases, 1801, 1808
- rows in queries
  - eliminating, 680
  - in append queries, 730
  - preventing, 665

**E****Edit Relationships dialog box, 221–222****E-, e-, generating scientific notation, 868****email collection wizard**

## for HTML forms

- choosing recipients, 508–510
- choosing users type of form, 497
- creating HTML form, 495–496
- customizing email messages, 506–507
- identifying recipients' email addresses, 504–505
- instructing recipient to click Reply, 507
- methods of entering email addresses, 503–504
- previewing messages, 510
- processing replies, 500–503
- seeing all the fields in the email collection wizard, 500
- selecting fields on the form, 498–500

## using InfoPath forms

- choosing fields in the email form, 517–518
- choosing recipients, 522–523
- choosing users type of form, 516–517
- processing replies, 518
- selecting recipients' email addresses, 519
- specifying subject line, 520–521

**EmailDatabaseObject, 1852****E-mail Messages Wizard, 1852****emails. *See also* email collection wizard**

- collecting data via, 493

**embedded blanks, 1776**

**embedding**

- linked pivot chart, 998–999
- macros. *See also* Invoice Audit Web form
  - creating, 1212–1215
  - deleting, 1215–1216
  - editing, 1209–1212
- objects and reports, 1033–1035
- PivotCharts to client reports, 1161–1163
- subforms, 964–968
- subforms in client reports, 1157–1160

**Enable AutoJoin, 624****encrypting databases, 1737–1738****endless loops, causes of, 1175****enterprise web services, 1348****Enter Table Properties dialog box, 336****Enter Validation Message dialog box, 334–335****Enum statement, 1483–1485****equal sign (=)**

- in field validation expressions, 327
- meaning of, 328

**equi-join queries, 622, 634****Error Description argument, 379, 380, 383–384****Error Number argument, 379, 383****errors**

- analyzing
  - publish errors, 1297–1298
  - using named data macros, 420–425
- clearing MacroError object, 1226
- conversion errors, dealing with, 267–268
- # Error
  - avoiding, 1135
  - correcting, 489
- fixing
  - caused by importing delimited data, 479
  - using imported spreadsheets, 468–470
  - using imported text files, 479–480
- trapping
  - event properties for, 1185
  - in Invoice Audit Web form, 1278–1279
  - in macros, 1219–1225
  - using RaiseError data action, 378–380

**Error Trapping (Visual Basic), 1461****event processing**

- about events in Windows, 1167–1168
- about form and report events, 1169–1170

**event properties**

- for changing data, 1172–1174
- for detecting changes in PivotTables and PivotCharts, 1181–1184
- for detecting filters applied to forms and reports, 1177
- for detecting focus changes, 1175–1177
- for detecting timer expiration, 1185
- for opening and closing forms and reports, 1170–1171
- for printing, 1184–1185

for trapping errors, 1185

for trapping keyboard and mouse events, 1178–1181

**event sequencing and form editing, 1185–1188****Event statement, 1485–1486****examine all error codes complex procedure (VB), 1568–1574****Excel**

- 64-bit versions of Office 2010 and, 1387
- as output option, 153
- defining smart tags, 879
- exporting data to Access
  - about, 462–463
  - as temporary table, 464
  - fixing errors, 468–471
  - preparing spreadsheet, 463–464
- exporting to
  - data, 9, 1319, 1824–1825
  - PivotTables, 688
- importing
  - web reports, 1306
- keeping prior versions, 1383
- linking to, 488–489
- list of user interface control identifiers, 1676
- OLE Object data type and, 192
- organizing data in, 168
- programming language for, 1452
- similarities to Pivot Tables, 681
- switching to databases from, 16
- themes in, 930
- troubleshooting ODBC file sources, 458
- viewing exported web reports, 1306

**exclamation point (!)**

- forces left-alignment, 868
- forces placeholders to fill left to right, 870
- separating table and field names with, 575, 576
- using, 1507–1508
- using in lists, 328

**EXISTS predicate, 1778–1779****Export BDC Model button, 1365****Export commands, 61****Export - dBase File dialog box, 1824****Export - Excel Spreadsheet dialog box, 1824****Export Text Wizard, 1826****Expression arguments**

- AskEdit Submacro, 1244
- in web-supported macro actions, 1867
- SetLocalVar action, 428
- SetReturnVar data action, 432
- SetStateAndZip Submacro, 1248
- SyncWeddingAndCity, 1236

**Expression Builder**

- about, 572–573
- adding DateDiff function, 574–580
- creating embedded macros, 1213
- moving ScreenTips, 574
- using exclamation mark by, 576

**Expression Builder dialog box, 321–325****expressions**

- about using, 562–563
- adding parentheses (), 572
- arithmetic, 566–571
  - operator precedence, 568–569
  - operators used in, 566
  - using DateDiff function, 566–568
- complex, 572–580
- conditional expressions in macros, 1207–1209
- creating text expressions, 563–565
- date and time functions, 581
- IsClient
  - limitations of functionality, 1282
  - using web, 1281
- referencing table fields, 711
- SQL
  - using parentheses () in, 1780
  - using brackets, 1209
  - using date, 722

**External Data tab, 60–61****F****fetching**

- information, using ADO, 447
- Number and Description properties, 1221

**field data types, 188, 190–193, 266****field lists**

- in queries, assigning alias names, 715–716
- selecting all fields in, 805
- using with Controls group, 800–801

**field properties**

- checking, 554
- on general tab, 193–199
- setting, 553–555

**Field row field names, 564****fields**

- adding to web reports, 1091–1095
- changing default data types, 266
- changing names of, 247–251
- copying, 257–260
- deleting, 260–261
- editing joined, 628
- importance of sequence of field definitions, 252
- inserting, 255–257
- in web tables
  - choosing names, 311
  - defining, 307–311
  - renaming, 299
- moving, 251–255
- referring to names of, 210
- selecting, 551–552
- selecting multiple using keyboard, 807

- specifying names of, 582–583
- true/false fields
  - choosing controls, 833–835
- updating with select queries, 680–681
- yes/no fields
  - choosing types of controls, 833–835

**Field Size field property, 316****field validation expressions, 327****field validation rules**

- checking, 586–587
- defining, 201–203
- feedback on failures, 1302
- in web databases, 327–331
- Record Not Saved dialog displayed, 1303
- violations of, 731

**File Download dialog box, 1314****File New Database dialog box, 171–172****File Open dialog box**

- importing dBase files, 454
- linking dBase files, 488

**file system functions, 1836–1837****File tab on Backstage view**

- changing default data types for fields, 266

**Filter argument, 1855****Filter By**

- Group, commands, 74
- property, 336–337

**Filter By Form command, 782****filtering**

- and searching for data in Datasheet view, 612–619
- form data, 780–782

**filter macro actions, 1856–1858****Filter Name argument, 1202****Filter table property, 212****Find And Replace dialog box**

- FindNextRecord macro actions and, 1856
- FindRecord macro actions and, 1857
- replacing data, 600
- Search box vs., 779
- searching for data, 612–613, 777–779

**Find commands, 59****Finder methods, 1363****FindNextRecord macro action, 1856****FindRecord macro action, 1857****Find Unmatched Query Wizard, 641–642, 643****First function, 647, 1135****fixed-width**

- data, importing, 473–474
- file, defining import specification for, 480

**fixing errors**

- importing spreadsheets, 468–471
- importing text files, 479–480

**FLOAT SQL data type, 459****Font Color button, 909**

**Font group buttons, 810–811****fonts**

- changing font size for Query windows, 678
- customizing, 862–865, 935–936
- in code editing, 1460
- size, increasing in Expression Builder, 324

**footer/header group buttons, 799****footers, in reports, 1028–1030****For Each...Next statement, 1541–1542****foreign keys, 1768**

- joining relationships, 624

**Format field property, 316****Format Painter button, 810****formatting**

- adding scroll bars, 876
- Application Parts Forms, 836–837
- columns of controls on web forms, 909–910
- combo boxes, 978
- conditional client forms, 978–985
- controls in forms, 811–812
- date values, 201
- fonts, 810–811
- options, quick access to web forms, 919
- property settings, 991–992
  - about, 865–866
    - for Currency data types, 866–869
    - for date/time data types, 872–874
    - for Number data types, 866–869
    - for Text data types, 869–872
  - specifying both input mask setting and, 875
- text boxes, 978
- text strings values, 201
- web reports, 1102–1105
- year values, 866

**Form Design button, 969****form events, 1169–1170****form modules (Visual Basic)**

- about, 1454–1455
- editing, 1458

**Form Name argument, 1202, 1244, 1866****Form/Report Design View, 844, 898****forms**

- about, 126, 142–143
- about key design terms using, 792
- ActiveX objects, 753–756
- adding filters, 780–782
- adding records, 768–775
- adding scroll bars, 876
- adding smart tags, 879–881
- advanced form design. *See* advanced form design
- building
  - Allow Layout View property, 818
  - checking design results, 817–818
  - formatting controls, 811–812
  - formatting fonts, 810–811

## input form, 806–818

- modifying fields, 826–828
- moving and sizing controls, 808–810
- property sheets and form sections, 801–806
- record sources, 792–793
- setting form properties, 815–816
- setting label properties, 814
- setting text box properties, 813–814
- using Application Parts Forms, 835–839
- using design tools, 789–793
- using field lists with Control group, 800–801, 805
- using Form Wizard, 823–826
- using Quick Create commands, 819–823
- building custom ribbons, 1671–1679
- changing captions, 1270
- changing data, 776–777
- changing name of default form, 898
- check boxes in, 746, 833–835
- choosing width and height, 791
- colors and special effects, 860
- combo boxes. *See also* combo boxes
  - about, 748–749
  - as controls in, 829–833
  - formatting, 978
- conditional formatting in, 978–985
- continuous, 741
- controls for client, 881–886
- controls group buttons, 795–799
  - formatting control using, 811–812
  - locking, 800
- controls in
  - about, 745
  - adjusting layout of, 849–850
  - aligning, 852–856
  - attachment control, 751–753
  - changing defaults, 897
  - check boxes, 746, 833–835
  - command buttons, 756–757
  - enabling Snap To Grid, 850–852
  - image, 754
  - list boxes, 829–830
  - moving, 855
  - moving and sizing controls, 808–810
  - navigation controls. *See* navigation controls
  - option buttons, 746, 833
  - option groups, 746, 976–978
  - resizing using arrows, 853
  - resizing using Auto Resize property, 856
  - selecting multiple controls, 844
  - Size To Fit command, 845–847
  - sizing, 808–810
  - tab controls, 749–750, 985
  - toggle buttons, 746, 833–835
  - using multiple data bar rules, 982
  - web browser controls, 759–760, 1014–1018

**forms** (*continued*)

- customizing
  - colors, 816–817
  - fonts, 862–865
- customizing in Design view
  - adjusting layout of controls, 849–850
  - aligning controls, 852–856
  - enabling Snap To Grid, 850–852
  - sizing controls, 844–849
- deleting data, 776
- displaying images, 938–944
- drawing tools in, 857–859
- duplicate data, 773–774
- editing and event sequencing, 1185–1188
- enabling and locking controls, 876–877
- establishing standard design on, 897–899
- event properties
  - for detecting filters applied to forms, 1177
  - for opening and closing forms, 1170–1171
- footers, 739
- formatting properties
  - about, 865–866
  - for Currency data types, 866–869
  - for date/time data types, 872–874
  - for Number data type, 866–869
  - for Text data types, 869–872
- group, 60
- header/footer group buttons, 799
- headers, 739–740
- linking using filters, 1631–1632
- list boxes in, 747–748
- macro actions in, 156–158
- modal, 744–745, 888–889, 891
- moving around on, 763–764
- multiple-page client forms, 992–996
- multiple-page forms, 740–741
- multiple-table query, 946–952
- object-oriented programming and, 785–789
- opening secondary forms, 1231–1235
- option buttons, 746, 833–835
- PivotCharts
  - as client forms, 996–1000
  - as forms, 761–763
- PivotTables as, 761–763
- pop-up, 743–744, 888–889
- printing, 783–784
- referencing objects in macros, 1229–1230
- searching for data, 777–779
- section details, 739
  - property sheets and, 801–806
- setting color for 3-D objects, 791

## setting properties

- allowing users to change views, 887
- controlling updates, 890
- preventing user from opening control menu, 891
- properties on All tab, 892–896
- setting border styles, 891–892
- setting navigation options, 888

## setting tab order, 877–878

## sorting data, 780–783

## special effects and colors, 860–862

## Split, 742

## subforms

- about, 742–743, 952–953
- creating Main Form, 969–972
- designing first level, 962–963
- designing innermost, 956–962
- embedding, 964–968
- rules for referencing, 1231
- sizing controls, 965
- specifying main source, 968
- specifying source, 954–956
- using subdatasheets, 973–976

## synchronizing two related forms, 1235–1239

## synchronizing using class events, 1635–1638

## tabbing on multiple page, 1613–1615

## testing status information between linked forms, 1245–1247

## toggle buttons, 746, 833–835

## triggering data tasks, 1639–1643

## typing data into forms using keyboard shortcuts, 766–767

## unbound object frame, 754

## uses of, 737–738, 1023

## using, 972

## using themes, 928–937, 951–952

## viewing data on, 763–764

## web

- building. *See* web forms
- limitations of, 763
- working in layout view, 899–900

## windows

- in Design view, 143–145
- in Form view, 147–148
- in Layout view, 146–147

**Forms/Report Design View, 1214****Form view, 147–148****Form Wizard**

- building many-to-one form, 948–952
- creating forms, 823–826
- designing subforms, 957–961
- modifying forms created by, 826–828

**For...Next statement, 1540–1541****forward slash (/), dividing numeric expressions with, 566**

**FoxPro**

- databases, 9, 13, 14
- indexes, 223

**FROM clause, 675, 1782–1785****FullName index, 224–225****functions**

- Access Services, 1838
- aggregate
  - description of, 647
  - description of SQL, 1837
  - using in calculated values, 1135
  - using in desktop applications (accdb) files, 1780
- API, Type statement and, 1492
- arithmetic, 1831–1832
- Avg
  - description of, 647, 1837
  - using in reports, 1135
- conversion, 1832–1833
- Count, 647, 1135, 1837
- Date(), 581, 1129
- DateAdd(interval, amount, date), 581
- DateDiff
  - in arithmetic expressions, 566–568
  - in Expression Builder, 574–580
- DatePart(interval, date), 581
- date/time, 1833
- Day(date), 581
- file system, 1836–1837
- First, 647, 1135
- GetAllSettings, 1836
- Hour(date), 581
- IIF, 211
- IsArray, 1476, 1834
- IsCurrentWebUserInGroup, 1325
- keyboard shortcuts to jump to, 1597
- Last, 647, 1135
- logic, 1834
- Max, 647, 1135, 1837
- MessageBox
  - about, 1197–1198
  - displaying when creating macros, 1213–1214
  - vs. MsgBox, 1198, 1243–1244
- Min, 647, 1135, 1837
- Month(date), 581
- MsgBox
  - options settings, 1243
  - return values, 1244
  - vs. MessageBox, 1198, 1243–1244
- ObjPtr, 1578–1579
- pointer valued, 1578–1579
- SQL aggregate, 1780, 1837

- StDev, 1135, 1837
- StDevP, 1837
- string, 1834–1836
- StrPtr, 1578–1579
- Sum, 647, 1135, 1837
- Total, 647
- user interface system, 1836–1837
- Var, 647, 1135
- VarP, 1837
- VarPtr, 1578–1579
- Visual Basic
  - declaring new functions, 1525
  - declaring new subroutines, 1527
  - queries using, 127
  - web-compatible, 1838–1840
- Weekday(date), 581
- Year(date), 581

**Function statements**

- in middle of procedures, 1459
- using, 1525–1527

**G****GetAllSettings function, 1836****Get External Data - Access Database dialog box, 484–486****Get External Data - dBASE File dialog box**

- importing dBase files, 453–454
- linking dBase files, 487–488

**Get External Data - Excel Spreadsheet dialog box, 488–489****Get External Data - ODBC Database dialog box, 448, 457, 490****Get External Data - SharePoint Site wizard, 532–534, 536–537****global settings in Access Options dialog box, modifying**

- about, 112–113
- Add-Ins category, 121–122
- Client Settings category, 118
- Current Database category, 113–114
- Customize Ribbon category, 119–120
- Datasheet category, 114–115
- Language category, 117–118
- Object Designers category, 115–116
- Proofing category, 116
- Quick Access Toolbar category, 120
- Trust Center category, 122

**Go To**

- argument, 1220, 1221, 1224, 1225
- statement, 1543

**grand totals, creating, 1137–1138****greater than or equal to sign (>=), meaning of, 328****greater than sign (>)**

- to display all characters in uppercase, 870
- to indicate positive numeric values, 327



**gridlines**

- adding in web forms, 918–921
- adding to web reports, 1100–1102

**Gridlines command, 919, 1100–1101****GROUP BY clause, 675, 1785–1786****Group constructs, using, 373–374****grouping criteria, defining in complex reports, 1111–1114****grouping information, in reports, 1050–1059****grouping options, understanding, 1113–1114****groups, custom**

- about, 78–80
- creating groups, 84–87
- creating object shortcuts, 87–91
- display order rules for, 86
- dragging and dropping objects into groups, 90
- hiding and renaming object shortcuts, 93–96
- hiding groups in categories, 91–93
- unhiding object shortcuts, 97–99

**groups, in reports, 1028–1030****Group, Sort, And Total pane, 1051–1059****Group & Sort button, 1089****H****HasModule property, setting to no, 1455****HAVING clause, 675, 1786–1787****header/footer group buttons, 799****headers, in reports, 1028–1030****Help tab, in Backstage view, 37–39****hiding duplicate values, 1138–1141****Home tab, exploring, 58–59****Hotmail, Outlook automatically processing replies from, 513****Hour(date) function, 581****HTML forms**

- automatic processing replies by Outlook, 513–514
- collecting data via. *See* email collection wizard
- filling out, 510–513
- look up values in, 513
- managing data collection messages, 529
- replying to data collection messages, 530
- resending data collection messages, 530–531

**HTML (Hypertext Markup Language)**

- linking name of file using, 314
- vs. XML, 1350

**HTML tags, formatting the Rich Text, 512****hybrid applications, 289****Hyperlink button, Insert, 795****Hyperlink data type**

- about, 191, 603–604
- activating hyperlinks, 604–605
- converting to, 263

**editing hyperlinks, 607****fixing email, 1594–1595****inserting new hyperlinks, 605–606****selecting, 192****Hyperlink web data type****about, 312****checking before publishing on SharePoint server, 314****using, 314****I****ID field automatically created, 315****ID numbers**

- changing after publishing web databases, 1296–1297
- internal unique, 227

**If...Then...Else statement, 1543–1544****IIF function, 211****Image button, 798****Image button, Insert, 799****image control, 754****images**

- sharing across forms and reports, 938–944
- working with linked photos, 1602–1604

**IMAGE SQL data type, 459****Immediate window, using, 1464–1469****Import/Export customizations button, 69****importing**

- Access object, 459–462
- database files
  - about, 452–453
  - dBase files, 453–456
  - modifying tables, 481
  - SQL tables, 456–459
  - vs. linking database files, 451–452
- spreadsheet data, 464–468
  - about, 462–463
  - preparing, 463–464
  - to temporary table, 464
- text files, 474–479
  - about, 471
  - delimited data, 472–473
  - fixed-width data, 473–474
  - fixing errors, 479–480

**Import & Link group**

- commands, 61
- importing text data using, 474–475
- linking to SharePoint lists using, 536

**import procedures**

- creating, 539–540
- saving, 540–542

**Import Spreadsheet Wizard**

- fixing errors, 469–471
- importing spreadsheets, 465–468

**Import Text Wizard**

- defining import specification, 480
- using, 475–479

**Import Wizard, 542, 1853****IN clause, 1787–1788****Increase/Decrease Decimals field property, 316****Indexed field property, 317****indexes, adding table**

- about, 222–223
- multiple-field, 224–226
- number of indexes supported by SharePoint lists, 349
- single-field, 223–224

**InfoPath forms**

- collecting data using. *See* email collection wizard
- filling out, 523–527
- managing data collection messages, 529
- manually processing replies, 527–529
- replying to data collection messages, 530
- resending data collection messages, 530–531

**information vs. data, 1751****Info tab, 285**

- of backstage view, 27–29

**inner joins, 622–630**

- about, 622–623
- creating queries, 623–626
- matching relationships, 624
- running queries, 626–627
- troubleshooting using correct tables, 623
- using query data, 628–629

**IN operator**

- description of, 561–562
- meaning of, 328

**IN predicate, 1788–1789****input masks**

- defining, 204–207
- inputting both settings for formatting and, 875

**Input Mask Wizard, 205–207****Insert ActiveX Control button, 794, 799****Insert Chart button, 796****Insert Hyperlink button, 795****Insert Image button, 799****Insert Or Remove Page Break button, 796****Insert Rows command**

- above selected field rows, 255–256
- in Indexes window, 225

**INSERT statement (append query), 1816–1818****instances, 5****Integer data type, Visual Basic, 1475****IntelliSense options, displaying or hiding, 323****internal ID numbers, 227****International Organization for Standardization (ISO), 445, 1350****Internet. *See* web****interval settings for DateDiff function, 567****INT SQL data type, 459****Invoice Audit Web form from macros**

- about, 1266
- calling named data macros, 1271–1275
- tracking error messages, 1278–1279
- using BrowseTo for browsing web forms and web reports, 1275–1279
- using return variables, 1271–1275

**IsArray function, 1476, 1834****IsClient expression**

- limitations of functionality, 1282
- using web, 1281

**IsCurrentWebUserinGroup function, 1325****IS NOT NULL operator, 328****ISO (International Organization for Standardization), 445, 1350****Italic button, 810****Item argument, 1236, 1244, 1248****J****Join Properties dialog box**

- changing properties for queries, 635
- defining links on multiple tables, 625

**joins**

- about, 5
- editing fields in tables, 628
- equi-join, 622, 634
- inner joins, 622–630
  - about, 622–623
  - creating queries, 623
  - matching relationships, 624
  - running queries, 626–627
  - using query data, 628–629
- matching relationships, 624
- outer joins, 634–641
  - between tables, 220
  - building simple, 634–636
  - solving complex to unmatched problems, 636–641
- self join relationships in web databases, 345
- troubleshooting using correct tables, 623
- unions, 674
- using tables related by more than one field, 625

## K

### keyboard

- commands
  - display or hide IntelliSense options, 323
  - for moving data in Datasheet view, 590–592
  - for moving fields, 253
  - moving between windows, 257
  - setting keyboard options in Datasheet view, 597
- options
  - setting, 597–599
- shortcuts
  - duplicating logic on macro design surface, 390
  - ensuring lines are straight, 858
  - entering data in Datasheet view, 597
  - for combo boxes, 767
  - for deleting rows, 602
  - for list boxes, 767
  - for moving rows, 601
  - for opening Immediate window, 1464
  - for selecting fields, 551
  - jumping to functions and procedures, 1597
  - moving controls in web forms, 908
  - moving controls in web reports, 1082
  - selecting multiple fields, 807
  - typing data into forms, 766–767
- trapping, 1178–1180
- keywords, avoiding in selection criterion, 556

## L

- Label button, 795
- label properties, setting, 814
- Label Wizard, 1077
- Language category in Access Options dialog box, 117–118
- LAN, linking name of file to, 314
- Last function, 647, 1135
- Layout view
  - building web reports
    - adding color to report sections, 1092
    - adding fields, 1091–1095
    - adding grouping and sorting, 1089–1091
    - adding totals to records, 1098–1100
    - counting pages, 1100
    - formatting report, 1102–1105
    - using gridlines, 1100–1102
    - working with controls. *See* controls in web form
  - creating web reports
    - completing reports, 1083–1086
    - modifying reports, 1077–1083
  - disabling, 1705–1706
  - form windows in, 146–147
  - keeping users out of, 887
  - opening forms and seeing controls, 1331
  - report windows in, 154–155
  - web forms in, 899–900

less than or equal to sign (<=), meaning of, 328

less than sign (<)

- displaying all characters lowercase, 870
- meaning of, 328

### LIKE operator

- description of, 561–562
- in validation rules, 203
- meaning of, 328
- testing validation rules, 329
- wildcard characters, 203, 328

LIKE predicate, 1790–1791

Line button, 796

line tool, 857–859

Link Child Fields properties, 214, 972

Linked Table Manager, 492–493

### linking files

- about, 481–482
- from Access databases, 484–487
- from dBase, 487–488
- from SQL tables, 490–491
- modifying linked tables, 491–492
- performance considerations, 482–484
- relinking tables, 492
- restrictions on types of objects, 481
- security considerations, 482
- to spreadsheet files, 488–489
- to text files, 488–489
- unlinking tables, 492
- using .accdb files, 452
- vs. importing database files, 451–452

linking SharePoint lists into Access, 535–539

### linking tables

- Access showing links, 625
- joins. *See* joins

Link Master Fields properties, 214, 972

Link Spreadsheet Wizard, 489

Link Text Wizard, 489

List Box button, 797

### list boxes

- about, 747–748
- in forms, 829

list boxes, working with multiple-selection, 1615–1619

List Box Wizard, 798

list of values, cycling through, 671

List Rows lookup property, 280

### lists

- filtering with other, 1628–1631
- selecting from summary, 1627–1628

List Settings page, assigning permissions on, 1325

List Width lookup property, 280

### literals (SQL)

- using single quotes ( ' '), 1780

### live preview

- disabling, 930

**LOBSystem in XML, 1352**

**Logic Designer**

- arguments
  - adding to, 1196
  - displaying, 369
- creating web macros, 1255
- Keyboard shortcuts for, 391
- macros
  - creating embedded, 1213
  - editing embedded, 1210–1212
- working with, 1194–1199

**logic functions, 1834**

**Long data type, Visual Basic, 1475**

**Long Date, format property**

- in Report Wizard, 1129

**LongLong data type, Visual Basic, 1475, 1579–1580**

**LongPtr data type, Visual Basic, 1475, 1576–1577**

**lookup fields**

- creating in web databases, 337–341
- creating relationships using
  - about, 341–343
  - Cascade Delete Relationships, 349–350
  - Restrict Delete Relationships, 343–349
- working with multi-value, 280–286

**lookup properties**

- about, 275–277
- overview of settings, 278–280
- sorting issues, 586
- using query designer to define, 553

**LookupRecord data block, arguments for, 394**

**Lookup & Relationship web data type, 312**

**Lookup Tab settings, 278**

**lookup values, in data collection forms, 513**

**Lookup Wizard**

- about, 191, 275–276
- creating Lookup fields in web database using, 338–341
- creating relationships using Lookup fields, 342–343
- defining Restrict Delete relationships using, 344–349

**loops, 1175, 1542**

**Lotus and Paradox file support, 453**

## M

**machine data sources, 448**

**Machine Data Sources tab, troubleshooting ODBC file sources, 458**

**macro actions. *See also* macros**

- ADP objects, 1850–1851
- ApplyFilter, 1856
- BrowseTo, 1275, 1276, 1278
- CloseWindow, 1227, 1866, 1867
- database objects, 1854–1856
- data entry operations, 1851
- data import/export, 1852
- filters, 1856–1858

FindNextRecord, 1856

FindRecord, 1857

GoToRecord, 1854

in controls, 156–158

in forms and reports, 156–158

macro commands, 1858–1860

Macro Name, 1220, 1224, 1241, 1867

MessageBox, 1867

not trusted, 1227

OnError, 1220–1221

OpenForm, 1202–1203, 1227, 1233, 1278, 1855, 1866

OpenQuery, 1227, 1857

OpenReport, 1855

query, 1856–1858

RaiseError, 378–380

RemoveTempVar, 1219

renamed from prior Access versions, 1867–1868

Requery, 1172, 1238, 1550, 1857

RunMacro, 1205, 1262

running, 1549–1551

search, 1856–1858

SetLocalVar, 428–429

SetProperty

- in form contexts, 1282

- using web form controls, 1266–1270

SetReturnVar, 432

system commands, 1861–1862

understanding not trusted, 1226–1228

unsafe, 1227

user interface, 1864–1865

using BrowseTo for browsing, 1275–1279

using older names, 1228

VB equivalents for, 1551

web-supported, 1255, 1866–1868

windows management, 1866

**Macro Builder option, 1213, 1214**

**Macro commands, 62**

**macro commands macro actions, 1858–1860**

**Macro design surface, dragging web forms and records to, 1263**

**MacroError object, clearing, 1226**

**Macro Name action arguments, 1220, 1224, 1241, 1867**

**Macro Name argument, 1260**

**macros. *See also* macro actions**

- about, 126
- adding to Quick Access Toolbar, 43
- avoiding type coercion issues, 1282–1284
- calling named data, 1658–1661
- conditional expressions
  - understanding, 1207–1209
- controls in
  - embedding, 1212
  - creating AutoKeys, 1712–1713
  - defining multiple actions, 1201–1203

**macros** (*continued*)

- embedded. *See also* Invoice Audit Web form
  - creating, 1212–1215
  - deleting, 1215–1216
  - editing, 1209–1212
  - using prior versions of Access, 1216
  - using temporary variables, 1216–1219
- event properties running
  - for changing data, 1172–1174
  - for detecting changes in PivotTables and PivotCharts, 1181–1184
  - for detecting filters applied to forms and reports, 1177
  - for detecting focus changes, 1175–1177
  - for detecting timer expiration, 1185
  - for opening and closing forms and reports, 1170–1171
  - for printing, 1184–1185
  - for trapping errors, 1185
  - for trapping keyboard and mouse events, 1178–1181
- finish running, 1203
- group, 62
- names from prior versions of Access, 1228
- opening
  - in Design view, 156
- opening secondary forms, 1231–1235
- referencing form and report objects, 1229–1231
- referencing in VB, 1509
- saving, 1199
- seeing complete list, 1198
- settings in databases
  - not trusted locations, 54
- SharePoint
  - checking user permission group levels for, 1279–1280
- starting and stopping applications, 1708–1711
- submacros
  - RunMacro action, 1205, 1262
  - using AskEdit, 1244–1245
  - using OnError action referencing, 1221
  - using RunMacro action to call message, 1262
  - working with, 1204–1207
- synchronizing two related forms, 1235–1239
- testing, 1199–1200
- testing status information between linked forms, 1245–1247
- trapping errors, 1219–1225
- user interface macros
  - referencing local variables, 1274
- uses of, 1193–1194
- using, 1584
- using Logic Designer, 1194–1199
- using Visual Basic procedures, 127–128
- validating data, 1239–1245
  - using preset values, 1247–1251

- waiting for server processing, 1309–1310
- web

- creating, 1254–1258
  - creating embedded, 1258
  - linking web macro objects, 1258
  - testing application, 1284

**Macros & Code group, 60**

- macro windows, redesigning, from prior Access Versions, 364–365

- mail merge, exporting to, 1826

- Make ACCDE button, 34

- Make ADE button, 1732

- make-table queries

- creating, 714–718

- running, 719–720

- make-table query, SELECT... INTO statement, 1819–1820

- Manage Attachments command, 751

- Manage Attachments dialog box, 798

- many-to-many relationships

- about, 5

- complex, 283

- concept of, 1748, 1764, 1769

- linking tables to create, 215

- maintaining special unique values, 1610–1611

- many-to-one forms

- creating, 946–947

- designing, 948–952

- many-to-one queries, designing, 947–948

- MAPI (Messaging Application Programming Interface), 1852

- mask characters, input, 204–205

- master/child links, setting, 966, 968, 972

- Max function, 647, 1135, 1837

- maximize/minimize buttons, troubleshooting, 136

- Max Records query property, 673

- MDI (multiple-document interface ) mode

- closing objects with one click, 109

- vs. single-document interface (SDI), 107–111

- Me in code, 1508

- Memo data type

- about, 190

- converting to, 262

- Memo Settings field property, 317

- Memo web data type, 312

- menu commands (VB), built-in, 1549–1551

- Message argument, 1198, 1213, 1221, 1867

- MessageBox function

- about, 1197–1198

- displaying when creating macros, 1213–1214

- vs. MsgBox, 1198, 1243–1244

**MessageBox** macro action, 1867  
**Messaging Application Programming Interface (MAPI)**, 1852  
**Microsoft Access** dialog box, 771  
**Microsoft ActiveX Data Objects Library**, 1504, 1522  
**Microsoft Excel**. *See* Excel  
**Microsoft FoxPro**  
     databases, 9, 13, 14  
     indexes, 223  
**Microsoft Graph** feature, 681  
**Microsoft newsgroups**, help from, 17  
**Microsoft Office 14.0 Access Database Engine Object Library**, loading, 1497  
**Microsoft Office 2010**. *See* Office 2010  
**Microsoft Office Backstage View**. *See* Backstage view  
**Microsoft Office Genuine Advantage** confirmation dialog box, 172–173  
**Microsoft Office Security Options** dialog box, 51  
**Microsoft Outlook**. *See* Outlook  
**Microsoft PowerPoint**  
     keeping prior versions, 1383  
     OLE Object data type and, 192  
     programming language for, 1452  
     themes in, 930  
**Microsoft Project**, 1387  
**Microsoft SharePoint Server**. *See* SharePoint server  
**Microsoft SQL Server Desktop Engine**. *See* SQL Server Desktop Engine (MSDE)  
**Microsoft Visual Studio**  
     building Smart tags, 879  
     creating BDC model definition files, 1352–1353  
**Microsoft Word**  
     as output option, 153  
     keeping prior versions, 1383  
     OLE Object data type and, 192  
     programming language for, 1452  
     themes in, 930  
     viewing exported web reports, 1306  
**Microsoft Word Mail Merge Wizard**, 1826, 1854  
**Min** function, 647, 1135, 1837  
 minus sign (–), as character in format strings, 868, 870  
**modal forms**, 744–745, 888–889, 891  
**Modified Expression** field property, 317  
**Modify Application Page**, reviewing, 1318–1320  
**Modify Button** dialog box, 43  
**Mod** operator, description of, 566  
**modules**  
     about, 126  
     coded in Visual Basic, 158–160  
**modules (VB)**  
     about, 1452–1455  
     creating new procedures, 1458–1459

    customizing, 1459–1461  
     declaring private or public variables in, 1477  
     objects, 1453–1454  
     Property Get procedure, 1530–1532  
     Property Let procedure, 1532–1535  
     Property Set procedures, 1535–1538  
     understanding class, 1529–1530  
**Month(date)** function, 581  
**mouse**  
     displaying super tooltip, 386  
**mouse events**, trapping, 1178–1181  
**Move Data** wizards group, 62  
**Move Down** arrow, 85  
**Move Up** arrow, 85  
 moving to database software, 15–17  
**MSDE (SQL Server desktop engine)**, 6  
**MsgBox** function  
     in web macros, 1257  
     options settings, 1243  
     return values, 1244  
     vs. **MessageBox**, 1198, 1243–1244  
**MSM, Outlook** automatically processing replies from, 513  
**multiple-document interface (MDI)**  
     closing objects with one click, 109  
     vs. single-document interface (SDI), 107–111

## N

**Name** argument  
     AskEdit Submacro, 1244  
     in web-supported macro actions, 1867  
     RemoveTempVar action, 1219  
     SetLocalVar action, 428  
     SetReturnVar data action, 432  
     SetStateAndZip Submacro, 1248  
     SyncWeddingAndCity, 1236  
**Name AutoCorrect** options, configuring, 227–229  
**named data macros**, 411–438  
     about, 411–412  
     analyzing errors, 420–425  
     calling, 416–417, 1271–1275  
     creating, 412–416  
     deleting, 418–419  
     renaming, 418–419  
     saving, 415  
     using local variables, 427–429  
     using parameters, 424–426  
     using return variables, 429–437  
**names**, checking for duplicate, 1605–1607  
**naming** what tables, 300  
**NARA (U.S. National Archives and Records Administration)**, 1605  
**Navigational Control** button, 795

**navigation controls, 757–758**

- about, 1000–1001
- applying Quick Styles to buttons, 1012–1013
- as collection of controls, 1003–1004
- creating navigation buttons, 1004–1011
- opening in Access prior versions with, 1013
- options, 1001–1002
- troubleshooting dragging forms in Layout view, 1006

**navigation forms. *See* navigation controls****Navigation Options dialog box**

- exploring, 80–82
- hiding custom groups, 91–92

**navigation options, setting options in forms, 888****Navigation pane**

- adjusting width of, 71–72
- categories, custom
  - about, 78–80
  - creating, 83–84
  - hiding and renaming object shortcuts, 93–96
  - unhiding object shortcuts, 97–99
- categories in
  - viewing, 101
- collapsing a group, 74
- exploring, 72–78
- groups, custom
  - about, 78–80
  - creating groups, 84–87
  - creating object shortcuts, 87–91
  - display order rules for, 86
  - dragging and dropping objects into, 90
  - hiding and renaming object shortcuts, 93–96
  - hiding groups in categories, 91–93
  - unhiding object shortcuts, 97–99
- jumping to object, 71
- objects
  - creating shortcuts, 87
  - dragging and dropping into groups, 90
  - hiding and renaming shortcuts, 93–96
  - manually sorting, 101–102
  - opening, 133
  - searching for, 102–106
  - unhiding shortcuts, 97–99
- sorting Views in, 99–101

**Navigation Target Name property, 1005****network shares, Trusted Documents on, 53****New Formatting Rule dialog box, 982****normalization of data**

- calculated values and, 320
- multi-values lookup fields and, 283
- rules for
  - creating unique identifiers (primary keys), 1762–1763
  - field independence, 1765–1767
  - field uniqueness, 1760–1762
  - functional dependence, 1763–1765
- table design and, 622

**not equal to (< >), meaning of, 328****NotInList event, handling, 1590–1594****NotInList event, using, 1242****NOT operator, 1797**

- meaning of, 328

**NULL predicate, 1791–1792****Null values**

- about, 199
- concatenating, 565
- in text field, 871
- properties of, 200

**Number data type**

- about, 190
- converting to, 263
- formatting properties, 866–869
- selecting, 191

**Number web data type**

- about, 312
- format property settings, 925–926

**O****Object data type, Visual Basic, 1476****Object Dependencies feature, 242–243****Object Designers category in Access Options dialog box**

- about, 115–116
- changing default data types for fields, 266
- changing default name for forms, 898
- changing default name for report templates, 898
- changing defaults Output All Fields property, 664
- changing font size for Query window, 678
- Enable AutoJoin, 624
- selecting Always Use Event Procedures check box, 1214
- setting options for table design, 228, 266
- setting selection options for controls, 844
- Show Table Names option, 623

**object methods**

- DAO
  - manipulating data types using, 1516–1519
  - recordsets, 1512–1516
- working with ADO recordsets, 1520–1524

**Object Name argument**

- AskEdit Submacro, 1244
- BrowseTo macro action, 1275, 1276
- in web-supported macro actions, 1867
- SyncWeddingAndCit, 1236

**object-oriented programming, 785–789****objects**

- checking dependencies, 242–244
- clearing MacroError object, 1226
- creating shortcuts in custom groups, 87–91
- dragging and dropping into custom groups, 90
- hiding and renaming shortcuts, 93–96
- importing Access, 459–462

- in desktop databases, 125–126
- in reports, 1033–1035
- linking
  - restrictions on, 481
- linking web macro, 1258
- manually sorting in Navigation pane, 101–102
- module, 1453–1454
- opening, 133
- organizing, 745
- printing reports about, 62
- referencing, 1505–1509
- referencing in forms and reports using macros, 1229–1230
- renaming, 246
- RunMacro action creating submacros within, 1205
- searching for, 102–106
- unhiding shortcuts, 97–99
- using Web Compatibility Checker with, 352
- Object Type argument**
  - AskEdit Submacro, 1244
  - BrowseTo macro action, 1275
  - in web-supported macro actions, 1867
  - SyncWeddingAndCit, 1236
- object variables, assigning, 1509**
- ObjPtr function, 1578–1579**
- ODBC Database button, 448**
- ODBC databases, exporting to, 1827**
- ODBC Driver Manager, 446–447**
- ODBC drivers**
  - importing databases using, 456
  - list of, 449
- ODBC (Open Database Connectivity)**
  - about, 446–447
  - databases, creating data source linking to, 448–451
  - file types
    - troubleshooting, 458
- ODBC Timeout query property, 673**
- Office 2010**
  - installing
    - 64-bit version, 1385–1387
    - about, 1376–1377
    - system requirements, 1375–1376
    - upgrading prior versions, 1381–1383
    - with no prior versions, 1377–1381
  - Security Options dialog box, 51
  - User Interface Control Identifiers, 1676
- Office Clipboard task pane, 601**
- Office downloads, diagnostic tool, 21–22**
- Office Fluent Ribbon. See Ribbon**
- OLE Object data type**
  - about, 190
  - selecting, 192
- OLE object fields**
  - entering data into, 771
- OnError macro action, 1220–1221**
- On Error statement, 1552–1553**
- one-to-many relationships, 220, 1769, 1816, 1821**
- one-to-one relationships, 1769**
- Open command, 29**
- Open command for reports, 1036**
- Open Database Connectivity (ODBC). See ODBC (Open Database Connectivity)**
- Open dialog box**
  - database files in exclusive mode, 521, 1737
  - database files in shared mode, 521
  - finding and opening existing database files, 25
- OpenForm macro action, 1202–1203, 1227, 1233, 1855, 1866**
- OpenForm vs. BrowseTo macro actions, 1278**
- opening**
  - Access for the first time, 21–23
  - existing database, 25–27
  - macros in Design view, 156
  - objects, 133
  - Visual Basic Editor, 159
- OpenQuery macro action, 1227, 1857**
- OpenRecordset Parameter Setting, 1513–1514**
- OpenReport macro action, 1855**
- operator precedence, arithmetic, 568–569**
- Option Button, 798**
- option buttons, 746, 833–835**
- Option Explicit statements, using (Visual Basic), 1477**
- Option Group button, 796**
- option groups, 746, 976–978**
- Options argument, 1227**
- Options dialog box**
  - customizing VBE, 1459
  - modifying settings for debugging VB code, 1461
  - setting custom colors for code elements, 1460
  - using Option Explicit statements, 1477
- Oracle databases**
  - importing, 456
  - SQL in, 674
- Order By argument, 1866**
- ORDER BY clause, 1792–1794**
- order of precedence rules, 568**
- Orientation query property, 674**
- OR operator**
  - specifying, 201
  - specifying multiple comparisons using, 327
  - vs. AND, 557–560
- outer joins, 634–641**
  - between tables, 220
  - building simple, 634–636
  - solving complex unmatched problem, 636–641
- Outlook**
  - automatic processing replies, 513–514
  - manually processing replies to InfoPath forms, 527–529



**Outlook** (*continued*)

- using email collection wizard for HTML forms with
  - choosing to have replies automatically processed, 507
  - opening up address book, 509
  - selecting as address method, 504
  - validating email address of sender, 507
- using email collection wizard for InfoPath forms with
  - choosing message field for addresses, 520–521
  - storing replies, 518

**Output All Fields Property, 664**

**output-column-name, referencing, 1781**

**P**

**Package Solution Wizard, 1725**

**packaging and signing databases, 1739–1742**

**Page argument, 1275**

**page numbers, adding, 1129–1132**

**Page Numbers dialog box, 1131**

**Page Setup dialog box, 1118, 1148**

**Page size form property, adjusting, 1304**

**Paint application**

- OLE Object data type and, 192

**Paint as ActiveX application, 755**

**Paradox and Lotus file support, 453**

**ParamArray argument, 1526**

**Parameter boxes, 1262**

**PARAMETERS declaration, 1794–1795**

**parameters, passing to web forms and reports, 1261–1265**

**parentheses ( )**

- adding to expressions, 572
- adding to SQL expressions, 1780
- as character in format strings, 868, 870

**partitioning data in crosstab queries, 658–660**

**passing**

- no arguments, 1545
- pointers to APIs, 1578

**passing arrays, 1486**

**pass-through queries, 1781**

**Paste Append command, 601**

**Paste command, undoing, 259**

**Path To Subform Control argument, 1275, 1276–1277**

**pencil icon, record indicator, 596**

**percentages, calculating, 1141–1142**

**percentage sign (%), place at last character to be multiplied by 100, 868**

**periods (.)**

- separating numbers and currency data types, 867
- separating table and field names with, 575
- using, 1507

**Personalization dialog box, setting color for 3-D objects, 791**

**photos, working with linked, 1602–1604**

**Pin To List option, 30–31**

**PivotCharts**

- about, 681
- adding to client reports, 1160–1163
- as client forms, 996–1000
- as forms, 761–763
- designing, 690–695
- embedding linked, 998–999
- event properties for detecting changes in, 1181–1184

**PivotTable design window, 686**

**PivotTables**

- as forms, 761–763
- building, 682–685
- designing, 685
- event properties
  - for detecting changes in PivotTables, 1181–1184
- tools on the ribbon, 687–688

**plus sign (+)**

- as character in format strings, 868, 870
- description of, 566
- using in web databases, 327

**pointer value functions, in 64-bit Access VB applications, 1578–1579**

**point-in-time data, capturing, 1771–1772**

**pop-up forms, 743–744, 888–889**

**pound sign (#)**

- as LIKE wildcard, 203
- as placeholder character, 868
- as wildcard, 328
- date/time literals, 1780
- designating fill characters, 868
- using in date and time values, 557

**PowerPoint**

- keeping prior versions, 1383
- OLE Object data type and, 192
- programming language for, 1452
- themes in, 930

**prerequisites, verifying, 1608–1610**

**primary keys**

- as unique identifiers, 1762
- causing duplicate data in forms, 773
- changing, 283–285
- defining, 208, 481
- joining relationships, 624
- on web databases, 335

**print dates, adding, 1129–1132**

**Print dialog box, 33**

**printing**

- event properties for, 1184
- forms, 783–784
- reports, 1039–1044
- reports about objects, 62
- table definitions, 233–234
- web reports, 1306

**Print Preview**

- in Print tab, 33
- in reports, 1026–1027
- report windows in, 152–153
- seeing two pages, 153
- Zoom control in, 153

**Print tab, 33****Privacy Options dialog box, 21–23****Privacy options, in Trust Center dialog box, 55****Private statement, 1486–1488****procedures**

- declaring private variables in, 1477
- examining procedure call sequences, 1473–1474
- running public, 1466

**process-driven database design, 1747****project files. *See* Access Data Projects (.adp files)****Project, Microsoft, 1387****Projects web database, 294–295****Proofing category in Access Options dialog box, 116–117, 885****Properties dialog box**

- changing Hidden property, 98
- changing PivotTable field captions, 689–690
- customizing query properties
  - about, 663–664
  - applying to dynamically linked tables, 673–674
  - controlling output, 664–665
  - defining subdatasheets, 669–673
  - Record Locks property, 673
  - using Orientation property, 674
  - working with unique records and values, 665–668
- defining properties for printers, 1042
- in Navigation pane, 94
- modifying shortcut targets, 1734
- running programs in compatibility mode, 1737

**properties, referencing, 1505–1509****Property argument, 1267–1268****property box values, cycling through list of values, 671****Property Get procedure, 1530–1532****Property Let procedure, 1532–1535****Property Set procedure, 1535–1538****property sheets and form sections, 801–806****Property statements, in middle of procedures, 1459****PtrSafe attributes, 1577****Public statement, 1488–1489****public web services, 1348****Publish Failed dialog box, 1298****publishing web databases. *See also* applications**

- about, 1290–1296
- analyzing publish errors, 1297–1298
- backing up original web database, 1293
- compiling web objects on servers, 1295
- converting embedded images in client objects to shared images, 1292
- creating Access Applications folder, 1294

**using web databases in Access**

- changing web applications, 1330–1335
- instantiating Access Services template, 1342–1345
- re-enabling prompt, 1328
- resolving synchronization conflicts, 1335–1337
- saving application as local database, 1341–1342
- working offline, 1337–1341

**Publish To Access Services command, 34, 36****Q****quantified predicate, 1796****queries**

- about, 126
- action. *See* action queries
- append
  - creating, 721–724
  - running, 725
  - SQL, INSERT statement, 1816–1818
- building for complex reports, 1108–1109
- building report, 1046–1047
- building subreport, 1155–1156
- changing data
  - adding new records, 596–597
  - copying and pasting, 600–602
  - deleting rows, 602–603
  - replacing, 600
  - selecting and, 599–600
  - understanding record indicators, 596
- changing default Output All Fields property, 664
- complex
  - building query for PivotTables, 682–685
  - creating PivotCharts, 681–682, 690–695
  - creating PivotTables, 681–682, 685–690
  - creating queries for web, 695–701
  - customizing query properties. *See* query properties
  - designing, 630–634
  - inner joins, 622–630
  - in SQL view, 674–679
  - outer joins, 634–641
  - totals queries. *See* totals
  - using query parameters, 660–663
- creating PivotCharts from
  - about, 681–682
  - designing, 690–695
- creating PivotTables from
  - building, 682–685
  - designing, 685–690
  - tools on the ribbon, 687–688
- creating web, 695–700
- crosstab
  - about, 652–653
  - Access database language for, 1774
  - creating simple, 653–658
  - partitioning data in, 658–660

**queries** (*continued*)

- defining by SQL, 626
- defining relationships between, 220
- delete
  - deleting groups of rows, 725–730
  - using with Before Delete, 396
- expressions in. *See* expressions
- filtering data, 613–619
- group, 60
- hyperlinks, working with, 603–607
- in SQL view
  - building a query, 675–679
  - learning, 675
  - types of queries, 674
  - updating data, 680
- joins. *See* joins
- keyboard options, setting, 597–599
- keyboard shortcuts, 590–592, 597
- limiting returned records, 618
- links between tables, 625
- make-table
  - creating, 714–718
  - running, 719–720
  - SELECT...INTO statement, 1819–1820
- many-to-one, 947–948
- parameters in
  - updating, 1263
  - using, 660–663
- pass-through, 1781
- searching data, 612–613
- select
  - about, 545
  - converting to update queries, 706
  - testing for deleted group of rows, 725–728
  - testing updated groups of rows, 704–706
- SELECT SQL
  - about, 1774
  - aggregate functions, 647, 1837
  - BETWEEN predicate, 1775–1776
  - clauses, 675
  - Column-Name, 1776–1777
  - comparison predicates, 1777–1778
  - discretions, 1779–1781
  - EXISTS predicate, 1778–1779
  - FROM clause, 675, 1782–1785
  - GROUP BY clause, 675, 1785–1786
  - HAVING A clause, 1786–1787
  - HAVING clause, 675
  - IN clause, 1788–1789
  - IN predicate, 1787–1788
  - LIKE predicate, 1790–1791
  - limitations to updating data, 680–681
  - NULL predicate, 1791–1792
  - ORDER BY clause, 1792–1794

- PARAMETERS declaration, 1794–1795
- quantified predicate, 1796
- search condition, 1797–1799
- SELECT clause, 675
- SELECT statements, 1799–1806
- subquery, 1806–1810
- TRANSFORM statement, 1810–1811
- UNION query operator, 1811–1813
- WHERE clause, 675, 1813–1814
- wildcard characters or strings, 1790
- setting properties in
  - controlling query output, 664–665
  - cycling through property box values, 671
  - defining subdatasheet, 669–673
  - Max Records, 673–674
  - ODBC Timeout, 673–674
  - Orientation property, 674
  - Source Connect Str, 673–674
  - Source Database, 673–674
  - Unique Records settings, 665–668
  - Unique Values settings, 665–668
  - using Record Locks property, 673
- single table
  - entering selection criteria, 555–562
  - selecting data from, 548–551
  - setting field properties, 553–555
  - sorting data in, 583–585
  - specifying field names, 582–583
  - specifying fields, 551–552
  - using expressions. *See* expressions
- sorting data, 608–611
- specifying correct column headings, 657
- specifying sort criteria, 585
- SQL action
  - DELETE statement, 1815–1816
  - INSERT statement (append query), 1816–1818
  - SELECT... INTO statement (make-table query), 1819–1820
  - UPDATE statement, 1820–1822
- SQL commands, storing as, 1773
- subdatasheets, working with, 592–596
- testing validation rule changes, 586–589
- totals queries
  - building crosstab queries, 652–660
  - selecting records from groups, 650–652
  - summarizing information with, 644
  - totals within groups, 645–650
- unions, 674. *See also* joins
- update
  - converting select queries to, 706–707
  - creating, using multiple tables or queries, 711–713
  - running, 707–709
  - updating multiple fields, 709–711
  - using Query Wizard, 641–644
  - using Visual Basic functions, 127

- windows
  - in Datasheet view, 141
  - in Design view, 139–141
- queries (custom), providing by forms, 1619–1627
- Query Builder, with record source property in forms, 804–805
- Query Design button, 947
- query designer
  - choosing correct table in query designer, 623
  - defining lookup properties, 553
  - SQL in, 674
- query macro actions, 1856–1858
- query properties
  - customizing
    - about, 663–664
    - applying to dynamically linked tables, 673–674
    - controlling query output, 664–665
    - defining subdatasheets, 669–673
    - Record Locks property, 673
    - using Orientation property, 674
    - working with unique records and values, 665–668
- Max Records, 673
- ODBC Timeout, 673
- Orientation, 674
- Record Locks property, 673
- Source Connect Str, 673
- Source Database, 673
- Query window
  - in Datasheet view
    - changing data, 596–603
    - filtering data, 613–619
    - keyboard options, setting, 597–599
    - keyboard shortcuts, 590–592, 597
    - searching data, 612–613
    - setting field properties, 553
    - sorting data, 608–611
    - working with hyperlinks, 603–607
    - working with subdatasheets, 592–596
  - in Design view
    - about, 139–141
    - changing size of fonts for, 678
    - clearing design grid, 552
    - creating web queries, 697, 698
    - designing expressions, 699
    - entering query parameters, 660
    - opening new window, 947
    - opening query property sheet, 663, 670
    - selecting all fields, 805
    - selecting data for single table, 549–550
    - selecting fields for single table, 551–552
    - using inner joins, 624–625
    - using keyboard to move between Windows, 257
- Query Wizard
  - creating queries for web, 697
  - using, 641–644
- question mark (?)

- as LIKE wildcard, 203
- wildcard, meaning of, 328
- Quick Access Toolbar
  - adding groups, 59
  - adding macros to, 43
  - adding Sync All command, 1333
  - as category in Access Options dialog box, 40–41, 46–47, 120–121, 1677
  - changing button face, 44
  - commands on
    - about, 39–40
    - adding, 42, 47
    - changing order of, 45
  - creating custom, 1703
  - customizing, 40–47
  - customizing ribbon, 63
  - in Access Options dialog box, 120–121
  - removing items from, 46
- Quick Create commands, creating forms using, 819–823
- Quick Print, 33
- Quick Styles button, 811

## R

- RaiseError macro action, 379
- RaiseEvent statement, 1545
- RDBMS (relational database management system)
  - about, 4–5
  - Access as, 6–13
  - capabilities of, 6
  - data control in, 12–13
  - data manipulation in, 9–12
  - defining rules, 8–9
  - SQL and, 446
- Reader, viewing exported web reports, 1306
- Read List operations, 1363
- REAL SQL data type, 459
- record indicators
  - asterisk icon, 596
  - pencil, 596
- Record Locks query property, 673
- records
  - adding totals to, 1098–1100
  - testing for related, 1607–1608
- Records commands, 59
- RecordSetObject.Open Parameter Settings, 1520–1521
- recordsets
  - about, 545
  - selecting fields in, 551
  - working with DAO, 1512–1516
- Recordset Type property, 665
- Record Source property, creating a query for, 804–805
- record sources, in forms, 792–793
- Record Validation Rule option, 332–334
- Rectangle button, 797

- rectangle tool, 857–859
- Recycle Bin in Access Services, 1321–1322
- ReDim statement, 1481, 1489–1490
- Redo button, 1865
- References dialog box
  - selecting Access database engine object library, 1497
  - selecting ActiveX Data Objects Library, 1504
- referential integrity rules, 602, 665, 731, 829
- RegOpenKeyA API, Windows, 1575
- relation, 5
- relational databases. *See also* RDBMS (relational database management system)
  - about, 4–5
  - benefits of, 1768–1769
  - changing sequence of rows, 602
  - primary key on, 208
- relationships
  - creating, using lookup fields
    - about, 341–343
    - Cascade Delete Relationships, 349–350
    - Restrict Delete Relationships, 343–349
  - definition of, 5
- Relationships commands, 62
- Relationships window, 221–222
- Remove Or Insert Page Breaks button, 796
- RemoveTempVar action arguments, 1219
- Rename dialog box
  - changing names of custom tabs, 65–66
- renaming
  - fields in web tables, 299
  - named data macros, 418–419
  - objects, 246
  - renaming object shortcuts, 93–96
  - tables, 245–247
- renaming object shortcuts
  - object shortcuts, 136
- Repair Database command, Compact And, 62
- Replace command, 600
- Replication Options, 62
- Report command
  - completing web report created by, 1083–1086
  - modifying web report, 1077–1083
  - using, 1066–1068
- reporting Access problems, 37
- reporting spreadsheet data
  - fixing errors, 468–471
- report modules (Visual Basic)
  - about, 1454–1455
  - editing, 1458
- Report Name argument, 1263, 1866
- Report properties, 1120–1128
- reports
  - about, 126, 149, 1024–1025
  - adding PivotCharts to client, 1160–1163
  - automating, 1649–1658
  - building
    - about, 1046
    - adding text boxes, 1062
    - creating labels, 1059–1061
    - defining grouping criteria, 1111–1114
    - defining sorting criteria, 1111–1113
    - designing, 1048–1050
    - setting properties for, 1119–1128
    - setting properties for Report section, 1115–1119
    - using calculated values. *See* calculated values
    - using Group, Sort, And Total pane, 1051–1059
    - using Report command, 1066–1068
    - using Report Wizard, 1069–1075, 1109–1111, 1129
  - building custom ribbons, 1671–1679
  - changing name of templates, 898
  - complex reports
    - building queries for, 1108
  - detail sections, 1028–1030
  - displaying images, 938–944
  - event properties
    - for detecting filters applied to reports, 1177
    - for opening and closing reports, 1170–1171
  - footers, 1028–1030
  - group, 60
  - grouping information, 1050–1059
  - groups, 1028–1030
  - headers, 1028–1030
  - linking using filters, 1632–1635
  - macro actions in, 156–158
  - objects embedded in, 1033–1035
  - printing, 1039–1044
  - Print Preview, 1026–1027
  - referencing objects in macros, 1229–1230
  - report events, 1169–1170
  - sorting information, 1050–1059
  - subreports, 1030–1033
    - building client report with, 1155–1160
    - embedding, 1151
    - rules for referencing, 1231
    - understanding, 1152–1155
  - totaling information, 1050–1059
  - understanding Open command, 1036
  - uses of, 1024
  - web. *See* web reports
  - windows
    - in Design view, 149–152
    - in Layout view, 154–155
    - in Print Preview, 152–153
    - in Report view, 155–156
- Report section
  - adding color to, 1092
  - setting properties for, 1115–1119

- Report view
  - about, 1035–1039
  - report windows in, 155–156
- Report Wizard
  - about, 1069
  - laying out client reports, 1109–1111
  - limitations for web reports, 1077
  - setting Long Date, 1129
  - specifying options, 1069–1074
  - viewing results, 1074–1075
- Report Wizard button, 1110
- RequeryFlag, 1218, 1246
- Requery macro actions, 1172, 1238, 1550, 1857
- Required field property, 317
- Resend This E-Mail Message button, 530
- Resolve Conflicts dialog box, 1340
- Restrict Delete Relationships, defining, 343–349
- return variables, using, 1271–1276
- reversing changes, 269
- Ribbon
  - collapsing, 62
  - creating custom, with XML
    - about, 1665–1666
    - building, 1671–1679
    - creating USysRibbons table, 1667–1671
    - loading, 1679–1682
  - customizing, 63–69
  - loading images into custom controls, 1695–1696
  - PivotTable tools on, 687
  - scrolling through, 58
  - troubleshooting data services option on, 1367
  - understanding, 57
  - using attributes
    - adding options to Backstage view, 1697–1703
    - creating Quick Access Toolbar, 1703
    - creating VBA callbacks, 1691–1692
    - hiding options and Backstage view, 1696–1697
    - RibbonX attribute tables, 1683–1685
    - RibbonX controls, 1685–1687
    - setting tab focus, 1704–1705
    - updating elements, 1692–1694
- Ribbon Name property of forms, 1692
- RibbonX
  - Attributes, 1683–1685
  - Controls, 1685–1687
- rich client, 289–290
- Rich Text format, formatting HTML tags, 512
- rooms, Append To, 724
- Row Source lookup property, 279
- Row Source Type lookup property, 279
- rows
  - alternating row color, 1064, 1116
  - deleting, 602–603
  - deleting groups of, 725–730
  - duplicate database, 1801, 1808
  - duplicate query
    - eliminating, 680
    - in append queries, 730
    - preventing, 665
  - errors in action queries, 732
  - filtering Datasheet view, 614
  - inserting
    - above selected fields rows, 255–256
    - in Indexes window, 225
    - in web forms, 914–916
  - naming field rows, 564
  - sorting, 550
  - updating groups of, 704–713
    - converting select queries to update queries, 706–707
    - creating update query, 711–713
    - running an update query, 707–709
    - updating multiple fields, 709–711
- rules. *See also* validation rules
  - conditional formatting
    - in client forms, 978–985
    - opening in prior Access versions, 981
    - using in calculated values, 1146–1151
  - data bar, 981, 982, 983
  - data normalization, 283, 320, 1760–1768
  - Delete Rule button, 982
  - field validation
    - checking, 586–587
    - defining, 201–203
    - in web databases, 327–331
    - receiving feedback on failures, 1302
    - violations of, 731
  - in designing database applications
    - breaking rules, 1770–1772
    - creating unique identifiers (primary keys), 1762–1763
    - field independence, 1765–1767
    - fields uniqueness, 1760–1762
    - functional dependence, 1763–1765
  - macro
    - for referencing form and report controls, 1230
    - for referencing form and report properties, 1230
    - for referencing forms and reports, 1229–1230
    - for referencing subforms in subreports, 1231
  - referential integrity, 602, 665, 731, 829
  - table validation
    - checking, 588–589
    - defining, 209–211
    - in Web bases, 332–335
    - violations of, 731
- RunCommand
  - GoToRecord macro action and, 1854
  - in Access 2010, 1868
  - method, 1550–1551
  - parameters, 1551
- Run command dialogue box, 707
- RunMacro action

- calling message submacros, 1262
- creating submacros within objects, 1205

running sums, 1143–1146

runtime mode, understanding, 1724–1727

Rushmore (Microsoft FoxPro), 223

## S

Save argument, 1212, 1227, 1244, 1866

Save As dialog box, 178

Save command, 28

Save Database As command, 28, 34

Save Object As command, 28, 34, 35–36

Save & Publish tab, 34–37

schema of databases, 289

ScreenTips, 132, 574

scroll bars

- adding to forms, 876
- allowing space for, 280

scrolling

- keyboard shortcuts in Datasheet view, 591
- reducing in web reports, 1099

SDI (single-document interface) mode

- closing objects with one click, 109
- vs. multiple-document interface (MDI), 107–111

SDK (Smart Tag Software Development Kit), 879

Search Bar

- in Navigation pane, 102–106
- moving focus to, 104

search condition, 1797–1799

searching and filtering data in Datasheet view, 612–619

searching in forms for data, 777–779

security. *See also* Trust Center

- about Access, 47–48
- assigning permissions on List Settings page, 1325
- importing secured files, 462
- linking file considerations, 482

Security Warning, on Backstage view, 50

Select button, 795

Select Case statement, 1545–1547

SELECT clause, 675

Selected Fields list, copying fields to, 345

selecting data, keyboard shortcuts in Datasheet view, 592

SELECT... INTO statement (make-table query), 1819–1820

selection criteria, single table

- about, 555–556
- AND vs. OR operators, 557–560
- entering, 555–556
- using Between, In, and Like operators, 561–562
- working with dates and times, 556–557, 561

select queries

- about, 545
- converting to update queries, 706
- testing for deleted group of rows, 725–728
- testing updated group of rows, 704–706

SELECT SQL queries

- about, 1774
- aggregate functions, 647, 1837
- BETWEEN predicate, 1775–1776
- building, 674–679
- clauses, 675
- Column-Name, 1776–1777
- comparison predicates, 1777–1778
- EXISTS predicate, 1778–1779
- expressions, 1779–1781
- FROM clause, 675, 1782–1785
- GROUP BY clause, 675
- GROUP BY THE clause, 1785–1786
- HAVING clause, 1786–1787
- IN clause, 1787–1788
- IN predicate, 1788–1789
- LIKE predicate, 1790–1791
- limitations to updating data, 680–681
- NULL predicate, 1791–1792
- ORDER BY clause, 1792–1794
- PARAMETERS declaration, 1794–1795
- quantified predicate, 1796
- search condition, 1797–1799
- SELECT clause, 675
- SELECT statement, 1799–1806
- subquery, 1806–1810
- TRANSFORM statement, 1810–1811
- UNION query operator, 1811–1813
- WHERE clause, 675, 1813–1814
- wildcard characters for strings, 1790

self join relationships, in web databases, 345

Self-signing certificate, using, 1740

server login dialog box, 1827

Set Control Defaults, 798

Set Control Defaults button, 798, 897, 898, 916

SetLocalVar action arguments, 428–429

SetProperty macro action

- behavior for web controls, 1270
- in form contexts, 1282
- using web form controls, 1266–1270

SetReturnVar data action arguments, 432

Set statements, 1509–1512

Shape Effects button, 812

Shape Fill button, 812

Shape Outline button, 812

Shared Resources feature, 938–944

SharePoint data

- about, 531
- exporting to, 1828–1829
- imported SharePoint list doesn't include all records, 535
- importing a list from data site, 532–535
- linking SharePoint list, 535–539
- SharePoint Designer 2010
  - creating BDC definition files, 1357–1363
  - downloading, 1355
  - exporting BCS Entity to XML file, 1365–1366
  - mapping Read List operations, 1363–1365
- SharePoint ID field, 335
- SharePoint lists
  - Calculated column created in, 314
  - creating calendar view of data in, 1322–1327
  - creating Choice data type in, 341
  - creating relationships between, 342
  - linking into Access, 535–539
  - number of indexes supported, 349
  - sharing data with, 13
  - tables in Web databases and, 289
- SharePoint server
  - analyzing publish errors, 1297–1298
  - Business Connectivity Services (BCS)
    - about, 1349–1350
    - definition files, 1352–1355
  - Business Data Catalog (BDC) and, 1349
  - checking user permission group levels for, 1279–1280
  - connecting to data through, 1349
  - creating calendar view of SharePoint list data, 1322–1327
  - instantiating Access Services template, 1342–1345
  - publishing databases with, 18, 34, 289
  - publishing web databases. *See also* publishing web databases
    - about, 1288
    - creating Access Applications folder, 1294
  - Recycle Bin in, 1321
  - settings for web queries, 700–701
  - treatment of Boolean values, 313, 1283
  - using macros to publish to web, 157
  - using select asterisk (\*) in web queries, 699
  - viewing application settings, 1312
- sharing images, 938–944
- Shortcut Command-Line Options, 1735–1736
- shortcuts, creating application, 1733–1737
- Shortcut Wizard, Create, 1734–1735
- Show Property Update Options Buttons, 229, 268
- Show Table dialog box
  - adding field lists, 954
  - building report queries, 1046
  - creating web query, 697
  - defining relationships, 219
  - defining Subdatasheets, 669
  - making SQL default option, 677
  - opening database for first time, 216–217
  - seeing defining relationships, 222
  - selecting data from multiple tables, 623
  - selecting tables or queries, 548–550
- Show Table Names option, 623
- Shutter Bar Open/Close button, 71
- signing and packaging databases, 1739–1742
- simple crosstab queries, 653–658
- simple forms, using Form Wizard, 823–828
- simple queries
  - changing data
    - adding new records, 596–597
    - copying and pasting, 600–602
    - deleting rows, 602–603
    - replacing, 600
    - selecting and, 599–600
    - understanding record indicators, 596
  - filtering data, 613–619
  - keyboard options, setting, 597
  - keyboard shortcuts, 590–592, 597
  - searching data, 612–613
  - sorting data, 608–611
  - subdatasheets, working with, 592–595
  - testing validation rule changes, 586–589
  - working with hyperlinks, 603–607
- simple queries (single table)
  - about, 548
  - arithmetic expressions, 566–571
    - operator precedence, 568–569
    - operator used in, 566
    - using DateDiff function, 566–568
  - entering selection criteria
    - about, 555–556
    - AND vs. OR operators, 557–560
    - using Between, In, and Like operators, 561–562
    - working with dates and times, 556–557, 561
  - expressions
    - adding parentheses (), 572
    - complex, 572–580
    - date and time functions, 581
  - selecting data from, 548–551
  - setting field properties, 553–555
  - sorting data, 583–585
  - specifying field names, 582–583
  - specifying fields, 551–552
- simple reports
  - about, 1045–1046
  - building queries for, 1046–1047
  - designing, 1048–1050
  - finishing, 1059–1065
  - grouping, sorting and totaling information, 1050–1059



**simple web forms**

- about, 900–901
- adding gridlines, 918–921
- control layouts
  - about, 901–903
  - lining up, 903–905
  - moving controls within, 905–908
  - removing, 911–912
- creating titles, 922–923
- formatting column of controls, 909–910
- inserting rows and columns, 914–916
- moving controls, 923–925
- resizing controls, 910–911
- splitting and merging cells, 912–914
- using control padding, 921–922
- using web-compatible controls, 916–918

**Single data type, Visual Basic, 1475****single-document interface (SDI)**

- closing objects with one click, 109
- vs. multiple-document interface (MDI), 107–111

**single quotation marks ( ' ' )**

- creating string constants, 563
- creating string constants in SQL, 1780
- using with text strings, 472

**single right arrow (>) button, in Available Fields list, 345****Single Step dialog box, 1861****Size To Fit command, 845–847****SMALLINT SQL data type, 459****smart tags**

- adding, 879–881
- assigning to control with data, 880

**Smart Tag Software Development Kit (SDK), 879****snapshot data, creating report, 1772****Snap To Grid**

- enabling, 850–852
- vs. To Grid, 851

**Sort & filter commands, 58****sorting criteria, defining in complex reports, 1111–1113****sorting data**

- applying multiple sorts in reverse order, 609
- importance of specifying sort criteria, 585
- in Datasheet view, 608–611
- in queries, 583–585
- on forms, 780–783
- troubleshooting sorting criteria, 648

**sorting information, in reports, 1050–1059****Soundex codes, 1605–1606****Source Connect Str query property, 673****Source Database query property, 673****Special Effect options, 812****special effects, colors and, 860–862****Split Form view, 742****spreadsheet data. *See also* Excel**

- defining data, 7
- exporting data, 9, 1824–1825
- exporting data to Access
  - about, 462–463
  - fixing errors, 468–471
  - preparing, 463–464
  - to temporary table, 464
- linking to, 488–489
- organizing, 168
- switching to databases from, 16

**SQL aggregate functions, 1780, 1837****SQL Server**

- creating data source, 449–451
- importing from, 456–459
- linking files to, 490
- moving tables to, 62
- security for linking databases, 482
- settings for web queries, 700–701
- SQL and, 446, 674

**SQL Server 2008, downloading, 161****SQL Server Configuration Manager, 490****SQL Server Desktop Engine (MSDE), 6****SQL Server Express Edition**

- downloading, 161
- installing, 456

**SQL Server Login dialog box**

- displaying for SQL data source, 490
- using Trusted Connection in, 1827

**SQL Server Reporting Services 2008, installing, 1306****SQL Server version 7.0, connecting project files to, 161****SQL Server wizard, Create A New Data Source To, 450–451****SQL (Structured Query Language)**

- about, 445–446
- action queries
  - DELETE statement, 1815–1816
  - INSERT statement (append query), 1816–1818
  - SELECT... INTO statement (make-table query), 1819–1820
  - UPDATE statement, 1820–1822
- aggregate functions, 1780, 1837

**ANSI SQL**

- Access support for, 674
- DISTINCTROW equipment, 1801
- referencing output-column-name, 1781

**defining a query in, 626****SELECT queries**

- about, 1774
- BETWEEN predicate, 1775–1776
- building, 675–679
- clauses, 675
- Column-Name, 1776–1777
- comparison predicate, 1777–1778

- EXISTS predicate, 1778–1779
  - expressions, 1779–1782
  - FROM clause, 675, 1782–1785
  - GROUP BY clause, 675, 1785–1786
  - HAVING clause, 675, 1786–1787
  - IN clause, 1787–1788
  - IN predicate, 1788–1789
  - LIKE predicate, 1790–1791
  - limitations to updating data, 680
  - literal strings in, 1780
  - NULL predicate, 1791–1792
  - ORDER BY clause, 1792–1794
  - PARAMETERS declaration, 1794–1795
  - quantified predicate, 1796
  - search condition, 1797–1799
  - SELECT clause, 675
  - SELECT statements, 1799–1806
  - subquery, 1806–1810
  - TRANSFORM statement, 1810–1811
  - UNION query operator, 1811–1813
  - WHERE clause, 675, 1813–1814
  - wildcard characters for strings, 1790
  - specifying column-names, 1780–1781
  - tables
    - importing, 456–459
    - linking files to, 490
    - naming columns, 1780–1781
  - SQL View command, 445–446
  - Static statement, 1491–1492
  - StDevP function, 1837
  - StDev (standard deviation) function, 647, 1135, 1837
  - string concatenation, in web databases, 327
  - String data type, Visual Basic, 1475
  - string functions, 1834–1836
  - strings
    - creating constants, 563
    - formatting text strings values, 201
    - wildcard characters for, 1790
    - zero-length strings, 199–200
  - StrPtr function, 1578–1579
  - Subdatasheet Expanded table property, 214
  - Subdatasheet Height table property, 214
  - Subdatasheet Name table property, 213
  - subdatasheets
    - defining, 669–673
    - working in Datasheet view with, 592–595
  - Subform Linker dialog box, 966–967
  - subforms
    - about, 742–743, 952–953
    - creating Main Form, 969–972
    - creating subdatasheet, 973–976
    - designing first level, 962–963
    - designing innermost, 956–962
    - editing from subform controls, 971
    - embedding, 964–968
    - rules for referencing, 1231
    - sizing controls, 965
    - specifying main source, 968
    - specifying source, 954–956
  - Subform/Subreport button, 798
  - submacros
    - RunMacro action within objects creating, 1205
    - using AskEdit, 1244–1245
    - using OnError macro action, 1220–1221
    - using RunMacro action to call message, 1262
    - working with, 1204–1207
  - subquery, 1806–1810
  - subreports
    - rules for referencing, 1231
    - using in forms, 972
    - using in reports, 1030–1033
      - building client reports from, 1155–1160
      - embedding, 1151
      - understanding, 1152–1155
  - Subreport/Subform button, 798
  - subroutines, 1539
  - Sub statements
    - in middle of procedures, 1459
    - using, 1527–1529
  - suggestions to improve Access, submitting, 37
  - Sum function, 647, 1135, 1837
  - Super Video Graphics Array (SVGA) for Print Preview, 1027
  - SVGA (Super Video Graphics Array) for Print Preview, 1027
  - Sync All command, 1332–1333
  - synchronization conflicts, resolving, 1335–1337
  - SyncWeddingAndCity arguments, 1236
  - System Button Face, 861
  - system commands macro actions, 1861–1862
- ## T
- Tabbed Documents setting, 111, 136
  - tabbing on multiple page forms, 1613–1615
  - tab character, using between fields, 472–473
  - Tab Control button, 795
  - tab controls
    - about, 749–750
    - formatting properties, 991–992
    - setting order of, 877–878
    - troubleshooting seeing multiple rows of tabs, 992
    - working with, 985–990
  - Table Analyzer Wizard
    - looking at lookup properties, 275–280
    - using, 270–275

**table design**

## changing

- about, 237–238
  - backing up databases, 239–242
  - changing data attributes. *See* data attributes
  - checking object dependencies, 242–244
  - copying fields, 257–260
  - deleting fields, 260–261
  - deleting tables, 244–245
  - field names, 247–251
  - field properties, 268–269
  - inserting fields, 255–257
  - looking at lookup properties, 275–280
  - moving fields, 251–255
  - renaming tables, 245–247
  - reversing changes, 269–270
  - working with multi-value lookup fields, 280–286
- compacting databases, 285–286
- for the web. *See* web tables, designing
- in Access, 168
- setting options, 226–229, 266
- using Table Analyzer Wizard, 270–275

**tables**

- about, 125, 531
- ActiveX objects in, database limitations, 235
- adding indexes
  - about, 222–223
  - multiple-field, 224–226
  - single-field, 223–224
- append queries, inserting data using, 721–725
- backing up, 240–242
- building queries from single
  - entering selection criteria, 555–562
  - selecting data, 548–551
  - setting field properties, 553–555
  - sorting data, 583, 583–585
  - specifying field names, 582–583
  - specifying fields, 551–552
  - specifying sort criteria, 585
  - using Expression Builder, 572–580
  - using expressions, 562–572
- changing names on, 177–178
- creating
  - in Design view, 186–187
  - simple, 176–178
  - using Application Parts feature, 178–182
  - using Data Type Parts feature, 182–186
- creating links, 1769
- Datasheet view
  - opening in, 133
- defining fields
  - about, 187–190
  - choosing names, 189–190
  - data types, 190–193
  - setting field properties, 193–199
  - validation rules for, 201–203
- defining relationships
  - about, 215–217
  - between two tables, 217–220
  - on multiple fields, 220–222
- deleting, 244–245
- design options, 226–229
- editing fields in joined, 628
- fields
  - changing names, 247–251
  - copying, 257–260
  - deleting, 260–261
  - importance of sequence of field definitions, 252
  - inserting, 255–257
  - moving, 251–255
  - properties, 268–269
- finding records across date spans, 637–638
- foreign keys, 1768
- group, 60
- input masks
  - defining, 204–207
- joins
  - outer, 220
- limiting returned records, 618
- linking files to. *See* linking files
- make-table queries
  - creating, 714–718
  - running, 719–720
- move to SQL Server wizards, 62
- one-to-many relationships, 1769
- one-to-one relationships, 1769
- other properties in Design view
  - Filter property, 212
  - Link Child Fields property, 214
  - Link Master Fields property, 214
  - setting Subdatasheet table properties in applications, 214–215
  - Subdatasheet properties, 213
- primary keys, 1762–1763
  - changing, 283–285
  - defining, 208
  - on web databases, 335
- printing
  - definitions, 233–234
- propagating changes in tables, 239
- properties of dynamically linked, 673–674
- renaming, 245–247
- reversing changes, 269–270
- rules of good table design, 1768
- selecting all fields, 552

- selecting data from multiple
  - choosing correct table in query designer, 623
  - creating inner joins, 622–630
  - solving complex problems with queries, 630–634
- SQL
  - linking files to, 490
  - naming columns, 1780–1781
- SQL importing, 456–459
- troubleshooting
  - extra rows in lower half of screen, 138
  - maximize/minimize buttons, 136
- unlinking linked tables, 492
- viewing Tables collection, 1469
- windows
  - in Datasheet view, 137–138
  - in Design view, 134–136
- yes/no fields, 833–835
- Tables And Related Views category, 75–78, 82**
- Table Templates, 180**
- table validation rules**
  - checking, 588–589
  - defining, 209–211
  - feedback on failures, 1302
  - in web bases, 332–335
  - Record Not Saved dialog displayed, 1303
  - violations of, 731
- tabs. *See also* tab controls**
  - adding built-in groups to, 1676–1679
  - All, in the property sheet, 892–896
  - Backstage view
    - File, changing default data types for fields, 266
    - Help, 37–39
    - Info, 29
    - New, 31–33
    - Print, 33
    - Recent, 29–31
    - Save & Publish, 34–37
  - Create, 59–60
  - Database Tools, 61–62
  - External Data, 60–61
  - Home, 58–59
  - Info, 285
- templates**
  - changing name of report templates, 898
  - creating in forms, 897–899
  - creating new database default, 230–233
  - creating new databases using, 169–173, 180, 1727–1730
  - creating new web databases, 291–294
  - group, 60
  - instantiating Access Services, 1342
  - web, 1019
- temporary variables, using, 1216–1219**
- TestCity Submacro arguments, 1241**
- testing**
  - data types, 1474
  - for related records one deleting records, 1607
  - validation rule changes, 268
  - validation rules, 586–589
- Text Box button, 795**
- text boxes, formatting, 978**
- text box properties, setting, 813–814**
- Text data type, 869–872**
  - about, 190
  - converting to, 262
  - selecting, 191
- text expressions, 563–565**
- text files**
  - exporting to, 1825–1826
  - importing
    - about, 471–472
    - delimited data, 472–473
    - fixed-width data, 473–474
    - fixing errors, 479–480
    - maximum text data length, 481
  - linking to, 488–489
- Text Formatting commands, 59**
- TEXT SQL data type, 459**
- text strings values**
  - concatenating, 1140
  - enclosing in quotation marks (" "), 327
  - formatting, 201
- Text web data type**
  - about, 312
  - using, 313
- themes, using, 928–937**
- This Database node, 1260**
- timeout messages, 1311**
- TIME SQL data type, 459**
- TIMESTAMP SQL data type, 459**
- TINYINT SQL data type, 459**
- Title argument, 1198, 1213**
- toggle button, 746, 797, 833–835**
- top-down database design, 1747**
- total functions, 647**
- totaling information, in reports, 1050–1059**
- totals**
  - building crosstab, 652–660
  - selecting records from groups, 650–652
  - Total functions, 647
  - within groups, 645–650
- Totals button, 645**
- TRANSFORM statement, 1810–1811**
- trapping errors**
  - event properties for, 1185
  - in Invoice Audit Web form, 1278–1279
  - in macros, 1219–1225

**trapping errors (VB), 1551–1553****troubleshooting**

- action queries, 730–732
- calendar control, 1601
- choosing correct table in query designer, 623
- connecting to server using trusted authentication, 491
- data services option on ribbon, 1367
- displaying more than one value in crosstabs, 655
- dragging forms in Layout view, 1006
- extra rows in lower half of tables, 138
- ID values of lookup fields in web reports, 1086
- imported SharePoint list doesn't include all records, 535
- ODBC file sources, 458
- restoring records from Recycle Bin in Access Services, 1322
- seeing all the fields in the email collection wizard, 500
- seeing complete list of macros, 1198
- seeing controls in Layout view, 1331
- seeing events and property sheet window, 1261
- seeing multiple rows of tabs, 992
- seeing web browser control listed in controls group, 1016
- setting defined default value for date/time field, 1596
- sorting criteria, 648
- viewing web forms or web reports in browsers, 1265
- web browser control display, 760

**true/false fields, choosing controls, 833–835****Trust Center**

- about, 48
- dialog box, 52–55
- enabling a not trust database, 49–51
- macros that are not trusted and, 1226–1227

**Trust Center category in Access Options dialog box, 52, 55, 122–123****Trust Center dialog box, 52–55****trusted authentication, troubleshooting connection to server using, 491****Trusted Documents, 53****Trusted Locations**

- about, 53
- defining, 55–57
- in corporate network environment, 55

**Trusted Publishers, 52–53****Trustworthy Computing initiative, 47–48****Type argument, 1198, 1213****type coercion issues, avoiding, 1282–1284****Type statement, 1492–1493****typing mistakes, help with, 116, 885****U****Unassigned Objects group, 84, 86–87****unbound controls, using in reports, 1135****unbound object frame, 754****Unbound Object Frame button, 797****Underline button, 810****Undo command, 244, 245, 259****Unicode Compression property, 197****union queries, 674****UNION query operator, 1811–1813****Unique field property, 317****Universal Naming Convention (UNC), linking name of file to LAN, 314****unlinking linked tables, 492****Unrelated Objects category, 76, 82****unsafe macro actions, 1227****Up arrow button, 390****Update Parameters link, updating query using, 1263****update queries**

- converting select query to, 706–707
- creating, using multiple tables or queries, 711–713
- deleting groups of rows, 725–730
- generic queries, 713
- running, 707–709
- updating multiple fields, 709–711

**UPDATE statement, 1820–1822****Update To row, 707****URLs, finding, 1300****Use Control Wizards button, 798, 829, 916****user-defined data type, Visual Basic, 1476****user interface macros**

- description of macro actions, 1864–1865
- referencing local variables, 1274

**user interface system functions, 1836–1837****user permission levels, checking SharePoint, 1279–1280****U.S. National Archives and Records Administration (NARA), 1605–1606****USysRibbons table**

- creating, 1667–1671
- defining for .adp files, 1681

**V****validating data**

- by presetting values, 1247–1251
- using macros, 1239–1245

**Validation Rule field property, 317, 318****validation rules**

- about, 8
- checking changes to, 268, 770, 1817
- comparison symbols in, 202
- feedback on failures, 1302
- field
  - checking, 586–587
  - defining, 196, 201–203
- in web databases
  - defining field validation rules, 327–331
  - defining table validation rules, 332–335
- Record Not Saved dialog displayed, 1303
- setting control, 886
- specifying, 199

- table
  - defining, 209–211
  - testing rule changes, 586–589
- testing
  - changes, 586–589
  - for characters, 203
- using LIKE operator to test, 329
- violations of, 731
- Validation Text field property, 317, 318**
- Value argument, 1244, 1267–1268**
- value list lookup field, 341**
- VARCHAR SQL data type, 459**
- Var function, 647, 1135, 1837**
- variables**
  - referencing local, 1274
- variables (VB)**
  - about, 1474
  - data types, 1475
  - declaring public or private, 1477–1478
  - naming conventions for, 1480
  - statements to define
    - Dim, 1480–1483
    - Event, 1485–1486
    - Private, 1486–1488
    - Public, 1488–1489
    - ReDim, 1489–1490
    - Static, 1491–1492
    - Type, 1492–1493
  - using temporary, 1216–1219
- Variant data type, Visual Basic, 1476**
- VarP function, 1837**
- VarPtr function, 1578–1579**
- VBA7 language updates, 1579–1580**
- VBA (Visual Basic for Applications)**
  - changing ribbon tab focus, 1704–1705
  - creating callbacks, 1691–1692
- VBE Options dialog box. *See* Options dialog box**
- View argument, 1202, 1227, 1244, 1855, 1857**
- View commands, 58**
- Views**
  - allowing users to change, 887
  - Backstage view
    - about, 27–29
    - closing, 39
    - Help tab, 37–39
    - Info tab, 29
    - New tab, 31–33
    - Print tab, 33
    - Recent tab, 29–31
    - Save & Publish tab, 34–37
    - Security Warning on, 50
  - calendar view of SharePoint list data, creating, 1322
  - Datasheet view. *See* Datasheet view
  - Design view. *See* Design view
  - Layout view. *See* Layout view
  - Navigation pane
    - sorting, 99–101
  - Report view
    - about, 1035–1039
    - report windows in, 155–156
  - Split Form, 742
  - SQL view
    - building a query, 675–679
    - learning, 675
    - types of queries, 674
    - updating data, 680
- Vista. *See* Windows Vista**
- Visual Basic Editor (VBE)**
  - about, 1455–1461
  - applying options to new databases, 232
  - building Smart tags, 879
  - object list box, 1458
  - opening, 62, 159
  - procedure list box, 1458
  - viewing Visual Basic code, 1456
- Visual Basic for Applications (VBA)**
  - changing ribbon tab focus, 1704–1705
  - creating callbacks, 1691–1692
- Visual Basic functions**
  - declaring new functions, 1525–1526
  - declaring new subroutines, 1527–1528
  - event properties running
    - for changing data, 1172–1174
    - for detecting changes in PivotTables and PivotCharts, 1181–1184
    - for detecting filters applied to forms and reports, 1177
    - for detecting focus changes, 1175–1177
    - for detecting timer expiration, 1185
    - for opening and closing forms and reports, 1170–1171
    - for printing, 1184–1185
    - for trapping errors, 1185
    - for trapping keyboard and mouse events, 1178–1181
  - queries using, 127
- Visual Basic (VB)**
  - 64-bit Access applications
    - about, 1574–1575
    - LongLong data types, 1475, 1579–1580
    - LongPtr data types, 1475, 1576–1577
    - LongPtr type coercion, 1578–1579
    - PtrSafe attributes, 1577
    - setting registry key to test upper memory, 1576
    - supporting previous versions of Access, 1577–1578
    - understanding pointer value functions, 1578–1579
    - using Declare statements, 1575–1576
    - VBA7 language updates, 1579–1580
  - ActiveX in. *See* individual headings under ActiveX

**Visual Basic (VB)** *(continued)*

## architecture

ADO (ActiveX Data Objects), 1502–1504

DAO (Data Access Objects) model, 1497–1501

## arrays

conversion function, 1832

declaring static, 1492

declaring within procedures, 1489

determining Variants as, 1476

dimensions, 1481

ParamArray argument, 1526

passing, 1486

two-dimensional, 1836

using in example code, 1565

assigning object variables, 1509–1512

## assisting data entry

filling in data, 1585–1590

fixing email hyperlinks, 1594–1595

handling NotInList event, 1590–1594

providing graphical calendars, 1595–1602

working with linked photos, 1602–1604

## automating complex tasks

linking related tasks, 1643–1649

triggering data tasks, 1639–1643

## automating data selection

filtering list with another, 1628–1631

providing custom queries by forms, 1619–1627

selecting from summary lists, 1627–1628

working with multiple-selection list boxes, 1615–1619

automating reports, 1649–1658

ByRef keyword, 1486

calling named data macros, 1658–1661

## commands

executing, 1464–1469

menu, 1549–1551

compiling, 1713–1714

## constants

about, 1474

Const statement, 1479

data types, 1475–1476

declaring, 1477, 1478

Enum statement, 1483–1485

ReDim statement, 1489–1490

## controlling flow of statements

Call statement, 1538–1539

For Each...Next statement, 1541–1542

For...Next statement, 1540–1541

Go To statement, 1543

If...Then...Else statement, 1543–1544

RaiseEvent statement, 1545

Select Case statement, 1545–1547

While...Wend statement, 1547–1548

With...End statement, 1548–1549

counters, 1541

data types, 1475–1476

## debugging tools

breakpoints (stopping points), setting and using, 1466, 1469

examining procedure call sequences, 1473–1474

setting a breakpoint (stopping point), 1463–1464

using Immediate window, 1464–1469

using Watch window, 1469–1472

Declare statements, 1575–1576, 1578

Do...Loop statement, 1539–1540

form modules, 1454–1455

functions. *See* Visual Basic functions

Function statements, 1525–1527

## linking related data

linking forms and reports using filters, 1631–1635

synchronizing forms using class events, 1635–1638

looking for, 1188

loops, 1542

macro actions. *See* macro actions

macros, referencing, 1509

## modules

about, 1452–1455

creating new procedures, 1458–1459

customizing, 1459–1461

declaring private or public variables in, 1477

understanding class. *See* class modules (VB)

## object methods

manipulating data types using DAO, 1516–1519

working with ADO recordsets, 1520–1524

working with DAO recordsets, 1512–1516

Option Explicit statements, using, 1477

procedures, 128, 1500

declaring private variables in, 1477

running public, 1466

randomly load data complex procedure, 1553–1568

referencing collections, objects, and properties, 1505–1509

report modules, 1454–1455

routines, coding in modules, 158–160

Sub statements, 1527–1528

tabbing on multiple page forms, 1613–1615

trapping errors, 1551–1553

troubleshooting window, 1464

using, 1584–1585

using with .accde files, 1580–1581

## validating complex data

checking for duplicate names, 1605–1607

checking for overlapping data, 1611–1613

maintaining unique values, 1610–1611

testing for related records, 1607–1608

verifying prerequisites, 1608–1610

## variables

about, 1474

declaring public or private, 1477–1478

Dim statement, 1480–1483

- Event statement, 1485–1486
- naming conventions for, 1480
- Private statements, 1486–1488
- Public statements, 1488–1489
- ReDim statement, 1489–1490
- Static statement, 1491–1492
- Type statement, 1492–1493
- using temporary, 1216–1218
- viewing code, 1456

## Visual Studio

- building Smart tags, 879
- creating BDC model definition files, 1352–1353

## W

**Watch window (Visual Basic), using, 1469–1472**

**WCF Connection dialog box, 1359**

## web

- compatible functions, 1838–1840
- creating Application Parts Forms for, 838
- creating queries for, 695–700
- display forms
  - assigning startup form, 1289–1290
- macro actions, 1255, 1866–1868
- macros
  - creating, 1254–1258
  - creating embedded, 1258–1260
  - linking web macro objects, 1258
- message box title, 1257
- publishing Access Services applications to, 157
- publishing databases to, 17–19, 18

**web browser buttons, 32**

**Web Browser Control button, 795, 1016**

**web browser controls, 759–760**

- using, 1014–1019

## web browsers

- downloading web applications, 1314
- viewing web forms or web reports using, 1265
- working with applications in
  - about, 1299
  - hiding a repositioning Web Report Toolbar, 1305
  - understanding session management, 1311–1312
  - using Datasheet forms, 1306–1309
  - using web forms, 1299–1304
  - using web reports, 1304–1306
  - viewing application settings, 1312
  - waiting for server processing, 1309–1310
  - working with columns in Datasheet forms, 1308

**Web Compatibility Checker tool, 351–359, 1292, 1333**

## web databases

- about, 289
- creating lookup fields in web databases, 337
- defining field validation rules in, 327–331
- defining primary key, 335
- defining table validation rules in, 332–335

- publishing to web
  - about, 1290–1296
  - analyzing publish errors, 1297–1298
  - backing up original database, 1293
  - compiling web objects on servers, 1295
  - converting embedded images to shared images, 1292
  - creating Access Applications folder, 1294
- self join relationships, 345
- setting field properties for, 315–318
- using in Access published
  - changing web applications, 1330–1335
  - instantiating Access Services template, 1342–1345
  - re-enabling prompt, 1328
  - resolving synchronization conflicts, 1335–1337
  - saving application as local database, 1341–1342
  - working offline, 1337–1341
- using Web Compatibility Checker, 351–359
- viewing relationships, 351

## web forms

- adding gridlines, 918–921
- allow space on right edge for controls, 916
- changing caption property, 1270
- changing control padding, 921–922
- combo boxes, 917
- control anchoring
  - about, 901–903
- control layouts
  - about, 901–903
  - creating label controls, 918
  - lining up, 903–905
  - moving controls within, 905–908
  - removing in client forms, 911–912
- controls
  - formatting columns of, 909–910
  - moving to different sections, 923–925
  - resizing, 910–911
  - selecting all controls in column, 910
  - using SetProperty action, 1266–1270, 1282
  - using web-compatible, 916–918
  - working with events, 1260–1261
- creating titles, 922–923
- filtering drop-down list in Navigation Target Name property, 1005
- format property settings
  - for Date/Time data type, 926
  - for Number and Currency data types, 925–926
- inserting columns, 914–915
- inserting rows, 915
- limitations of, 763
- page break controls in, 993
- passing parameters, 1261–1265
- seeing events and property sheet window, 1261
- setting properties, 927–928
- splitting and merging cells, 912–914, 1094



**web forms** (*continued*)

- using BrowseTo for browsing, 1275–1279
- using in web browsers, 1299–1304
- using navigation controls, 1003–1014
- using subreports, 1031
- Visual Basic and, 1453
- web forms
  - aligning, 903–905
  - working in layout view, 899–900

**Web Linked Lists commands, 61****web reports**

- building in Layout view
  - adding color to report sections, 1092
  - adding fields, 1091–1095
  - adding grouping, 1089–1091
  - adding sorting, 1089–1091
  - adding totals to records, 1098–1100
  - counting pages, 1100
  - formatting report, 1102–1105
  - using gridlines, 1100–1102
- controls in
  - moving to different sections, 1095
  - resizing in Layout view, 1096
- creating using Report command in Layout view
  - completing report, 1083–1086
  - modifying Report command report, 1077–1083
- dragging forms and records to Macro design surface, 1263
- dragging to Macro design surface, 1263
- limitations of, 1105–1106
- passing parameters, 1261–1265
- printing exported, 1306
- supporting subreports, 1151
- troubleshooting ID values of lookup fields, 1086
- using BrowseTo for browsing, 1275–1279
- using in web browsers, 1304–1306
- viewing
  - exported web reports, 1306
  - using web browsers, 1265
- yes/no fields in, 1065

**Web Report Toolbar**

- hiding a repositioning, 1305

**web services, 1347–1349****websites**

- mashup, 1347
- setting permissions, 1315–1318

**web-supported macro actions, 1866–1868****web tables, designing**

- about, 287
- choosing table names, 300
- creating new databases
  - using templates, 291–296
- creating tables
  - simple, 297–300
  - using Application Parts, 300–304

- using Datasheet view. *See* Datasheet view
- using Data Type Parts, 304–306
- data types, 311–315
- defining primary key, 335
- defining table validation rules, 332–335
- fields
  - choosing names, 311
  - renaming, 299
- lookup fields in. *See* lookup fields

- number of fields allowed, 315
- using Create Relationship Wizard, 301–304
- using Web Compatibility Checker, 351–360

**web templates, 1019****Weekday(date) function, 581****WHERE clause, 675, 1813–1815****Where Condition argument**

- as web-supported argument, 1866
- BrowseTo macro action, 1275
- empty values in, 1237
- LookupRecord data block, 394
- OpenForm macro action, 1202, 1233
- OpenReport macro action, 1855

**While...Wend statement, 1547–1548****wildcard characters**

- ACE/JET, 1522
- ANSI, 1522
- for strings, 1790
- LIKE, 203

**Window Color and Appearance dialog box, 791****WindowHide command, 1708****Window Mode argument, 1202–1203, 1244, 1866****Windows 7**

- forms with themes like, 951–952
- opening access for the first time in, 22
- User Account Control, 232

**Windows API Declarations and Constants for Visual Basic, 1578****Windows authentication, 1827****Windows Choose File dialog box, 752****Windows commands, 59****Windows event-driven applications, 1167–1169****windows management macro actions, 1866****Windows RegOpenKeyA API, 1575****Windows Vista**

- forms with themes like, 951–952
- User Account Control, 232

**With...End statement, 1548–1549****WithEvents keyword, using, 1481****Wizard Decide option, 272****Wizards**

- Chart
  - Insert Chart button, 796
- Combo Box, 798, 829–832

- Command Button, 798
- Create A New Data Source To SQL Server, 450–451
- Create Relationship, 301–304
- Create Shortcut, 1734–1735
- Database Splitter, 1717–1720
- email collection, for HTML forms
  - choosing recipients, 508–510
  - choosing users type of form, 497
  - creating HTML form, 495–496
  - customizing email messages, 506–507
  - identifying recipients email addresses, 504–505
  - instructing recipient to click Reply, 507
  - methods of entering email addresses, 503–504
  - previewing messages, 510
  - processing replies, 500–502
  - seeing all fields in the email collection wizard, 500
  - selecting fields on the form, 498–500
- email collection, for InfoPath forms
  - choosing fields in the email form, 517–518
  - choosing recipients, 522–523
  - choosing users type of form, 516–517
  - processing replies, 518
  - selecting recipients email addresses, 519
  - specifying subject line, 520–521
- E-mail Messages, 1852
- Export, 1853
- Export Text, 1826
- Find Unmatched Query, 641–642, 643
- Form
  - building many-to-one form, 948–952
  - creating forms, 823–826
  - designing subforms, 957–961
  - modifying forms created by, 826–828
- Get External Data - SharePoint Site, 532–534, 536–537
- Import, 542, 1853
- Import Spreadsheet
  - fixing errors, 469–471
  - importing spreadsheets, 465–468
- Import Text
  - defining import specification, 480
  - using, 475–479
- Input Mask, 205–207
- Label, 1077
- Link Spreadsheet, 489
- Link Text, 489
- List Box, 798
- Lookup
  - about, 275–276
  - creating Lookup fields in web database using, 338–341
  - creating relationships using Lookup fields, 342–343
  - defining Restrict Delete relationships using, 344–349
- Microsoft Word Mail Merge, 1826, 1854
- Package Solution, 1725

- Query
  - creating queries for Web, 697
  - using, 641–644
- Report
  - about, 1069
  - laying out client reports, 1109–1111
  - limitations for web reports, 1077
  - setting Long Data, 1129
  - specifying options, 1069–1074
  - viewing results, 1074–1075
- Table Analyzer
  - looking at Lookup properties, 275–280
  - using, 270–275

## Word

- as output option, 153
- keeping prior versions, 1383
- OLE Object data type and, 192
- programming language for, 1452
- themes in, 930
- viewing exported web reports, 1306

## word processing documents

- defining contents, 7
- export/import data, 9
- RDBMS vs., 12

## Workflows dialog box, 1861

## Workflows Tasks dialog box, 1861

## Work Offline command, 1338

## World Wide Web. *See also* web

- linking URL on, 314

# X

## XML

- Backstage view attributes, 1700–1701
- Backstage view controls, 1701–1703
- creating custom ribbons with
  - about, 1665–1666
  - building, 1671–1679
  - creating USysRibbons table, 1667–1671
  - loading, 1679–1682
- displaying ribbon errors, 1674
- exporting BCS Entity to, 1365–1366
- in BDC definition files, 1352–1355
- using, 1350–1351
- using ribbon attributes
  - adding options to Backstage view, 1697–1703
  - creating Quick Access Toolbar, 1703
  - hiding options and Backstage view, 1696–1697
  - loading images into custom controls, 1695–1696
- RibbonX attributes, 1683–1685
- RibbonX controls, 1685–1687
- setting ribbon tab focus, 1704–1705
- updating elements, 1692–1694

## Y

Year(date) function, 581

year values, formatting, 866

Yes/No data type

- about, 190

- limitations of, 265

- using, 312, 313

yes/no fields

- choosing types of controls, 833–835

- in web reports, 1065

- specifying format for, 874–875

## Z

zero-length strings, 199–200

Zoom control, in Print Preview, 153

Zoom dialog box, 803, 1055–1056, 1298

Zoom window

- opening, 563

- using to enter expressions, 564