



## **VBA AND MACROS:**

## Microsoft Excel 2010

AUTOMATE REPORTS BUILD FUNCTIONS VISUALIZE DATA WRITE FAST, RELIABLE SCRIPTS



#### FREE SAMPLE CHAPTER



in



# VBA and Macros: Microsoft® Excel® 2010

Bill Jelen Tracy Syrstad



800 E. 96th Street Indianapolis, Indiana 46240

#### Contents at a Glance

	Introduction 1
1	Unleash the Power of Excel with VBA 7
2	This Sounds Like BASIC, So Why Doesn't It Look Familiar?
3	Referring to Ranges
4	User-Defined Functions 79
5	Looping and Flow Control 107
6	R1C1-Style Formulas 127
7	What Is New in Excel 2010 and What Has Changed 139
8	Create and Manipulate Names in VBA 147
9	Event Programming 159
10	Userforms—An Introduction 183
11	Creating Charts 203
12	Data Mining with Advanced Filter 249
13	Using VBA to Create Pivot Tables 287
14	Excel Power 329
15	Data Visualizations and Conditional Formatting
16	Reading from and Writing to the Web
17	Dashboarding with Sparklines in Excel 2010 411
18	Automating Word 433
19	Arrays 453
20	Text File Processing 463
21	Using Access as a Back End to Enhance Multiuser Access to Data 475
22	Creating Classes, Records, and Collections
23	Advanced Userform Techniques 511
24	Windows API 535
25	Handling Errors 549
26	Customizing the Ribbon to Run Macros
27	Creating Add-Ins
	Index

### VBA and Macros: Microsoft<sup>®</sup> Excel<sup>®</sup> 2010

#### Copyright © 2010 by Que Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-7897-4314-5 ISBN-10: 0-7897-4314-0

Library of Congress Cataloging-in-Publication Data: Jelen, Bill. VBA and macros : Microsoft Excel 2010 / Bill Jelen, Tracy Syrstad. p. cm. Includes index. ISBN-13: 978-0-7897-4314-5 ISBN-10: 0-7897-4314-0

1. Microsoft Excel (Computer file) 2. Microsoft Visual Basic for applications. 3. Business—Computer programs. 4. Electronic spreadsheets. I. Syrstad, Tracy. II. Title.

HF5548.4.M523J46 2010 005.54—dc22

2010018831

Printed in the United States of America

Eighth Printing: August 2014

#### Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Microsoft and Excel are a registered trademarks of Microsoft Corporation.

#### Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

#### **Bulk Sales**

Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales 1-800-382-3419 corpsales@pearsontechgroup.com

For sales outside the United States, please contact

International Sales international@pearson.com Associate Publisher Greg Wiegand

Acquisitions Editor Loretta Yates

Development Editor Sondra Scott

Managing Editor Sandra Schroeder

Senior Project Editor Tonya Simpson

Copy Editor Keith Kline

Indexer Erika Millen

Proofreader Language Logistics

Technical Editor Bob Umlas

Publishing Coordinator Cindy Teeters

#### Book Designer Anne Jones

**Compositor** Bronkella Publishing

## Contents

In	troduction	1
	Getting Results with VBA	1
	What Is in This Book?	1
	Reduce the Learning Curve	1
	Excel VBA Power	2
	Techie Stuff Needed to Produce Applications	2
	Does This Book Teach Excel?	2
	The Future of VBA and Windows Versions of Excel	4
	Versions of Excel	4
	Special Elements and Typographical Conventions	5
	Code Files	6
	Next Steps	6
1	Unleash the Power of Excel with VBA	7
	The Power of Excel	7
	Rarriare to Entry	7
	The Macro Recorder Doesn't Work!	/
	Visual Basic Is Not Like BASIC	8
	Good News: Climbing the Learning Curve Is Easy	8
	Great News: Excel with VBA Is Worth the Effort	8
	Knowing Your Tools: The Developer Tab	9
	Macro Security	10
	Adding a Trusted Location	10
	Using Macro Settings to Enable Macros in Workbooks Outside of Trusted Locations	11
	Using Disable All Macros with Notification	12
	Overview of Recording, Storing, and Running a Macro	12
	Filling Out the Record Macro Dialog	13
	Running a Macro	14
	Creating a Macro Button on the Ribbon	14
	Creating a Macro Button on the Quick Access Toolbar	15
	Assigning a Macro to a Form Control, Text Box, or Shape	16
	Using New File Types in Excel 2010	18
	Understanding the VB Editor	19
	VB Editor Settings	19
	The Project Explorer	20
	The Truperness Willow	
	Understanding ShortComings of the Macro Kecorder	ו∠ ככ
	Examining Coue in the FloyIdining Window Running the Macro on Another Day Produces Undesired Results	25
	המוחווות נוכ וומנוס טו הוטנוכו שמיד וטמעכים טומכיווכע הכינונים הכינונים איירי איירי איירי איירי איירי איירי איי	zJ

	Possible Solution: Use Relative References When Recording	
	Never Use the AutoSum Button While Recording a Macro	
	Three Tips When Using the Macro Recorder	
	Next Steps	
2	This Sounds Like BASIC, So Why Doesn't It Look Familiar?	
	I Can't Understand This Code	
	Understanding the Parts of VBA "Speech"	34
	VRA Is Not Really Hard	37
	VBA Help Files: Using F1 to Find Anything	
	Using Help Topics	
	Examining Recorded Macro Code: Using the VB Editor and Help	
	Optional Parameters	41
	Defined Constants	41
	Properties Can Return Objects	46
	Using Debugging Tools to Figure Out Recorded Code	46
	Stepping Through Code	
	More Debugging Options: Breakpoints	
	Backing Up or Moving Forward in Code	
	Not Stepping Inrough Each Line of Code	
	Using a Watch to Set a Breaknoint	
	Using a Watch on an Object	
	Object Browser: The Ultimate Reference	
	Seven Tips for Cleaning Up Recorded Code	
	Tip 1: Don't Select Anything	
	Tip 2: Cells(2,5) Is More Convenient Than Range("E2")	59
	Tip 3: Ride the Range from the Bottom to Find Last Row	59
	Tip 4: Use Variables to Avoid Hard-Coding Rows and Formulas	60
	Tip 5: R1C1 Formulas That Make Your Life Easier	61
	Tip 6: Learn to Copy and Paste In a Single Statement Tip 7: Use With End With to Deform Multiple Actions	
	Next Stens	01 64
3	Referring to Kanges	65
	The Range Object	
	Syntax to Specify a Range	
	Named Kanges	
	Shortcut for Referencing Ranges	66
	Referencing Ranges in Other Sheets	67
	Referencing a Range Relative to Another Range	68

	Use the Cells Property to Select a Range	
	Using the Office at Droperty to Defer to a Dange	
	Use the Density to Kelel to a Kalige	09
		۱ /
	Using the Columns and Kows Properties to Specify a Kange	
	Use the Union Method to Join Multiple Ranges	72
	Use the Intersect Method to Create a New Range from Overlapping Ranges	73
	Use the ISEMPTY Function to Check Whether a Cell Is Empty	73
	Use the CurrentRegion Property to Select a Data Range	74
	Use the Areas Collection to Return a Noncontiguous Range	77
	Referencing Tables	77
	Next Steps	
4	User-Defined Functions	79
7	Creating User-Defined Functions	70
	Sharing IDFs	
	Jianing Obi S	01 07
	Set the Current Workbook's Name in a Cell	
	Set the Current Workbook's Name and File Path in a Cell	
	Check Whether a Workbook Is Open	
	Check Whether a Sheet in an Open Workbook Exists	
	Count the Number of Workbooks in a Directory	84
	Retrieve USERID	85
	Retrieve Date and Time of Last Save	86
	Retrieve Permanent Date and Time	87
	Validate an E-mail Address	
	Sum Cells Based on Interior Color	
	Count Unique Values	
	Kemove Duplicates from a Kange	الا
	Fillu tile Filst Nollzero-Leligui Cell III a Nalige Substitute Multinle Characters	95 04
	Retrieve Numbers from Mixed Text	
	Convert Week Number into Date	
	Separate Delimited String	96
	Sort and Concatenate	97
	Sort Numeric and Alpha Characters	99
	Search for a String Within Text	100
	Reverse the Contents of a Cell	101
	Multiple Max	101
	Return Hyperlink Address	
	Keturn the Column Letter of a Cell Address	

	Static Random	103
	Using Select Case on a Worksheet	104
	Next Steps	105
5	Looping and Flow Control	
	ForNext Loops	
	Using Variables in the For Statement	110
	Variations on the For Next Loop	110
	Exiting a Loop Early After a Condition Is Met	111
	Nesting One Loop Inside Another Loop	112
	Do Loops	
	Using the While or Until Clause in Do Loops	115
	WhileWend Loops	117
	VBA Loop: For Each	117
	Object Variables	117
	Flow Control: Using IfThenElse and Select Case	120
	Basic Flow Control: If Then Else	121
	Conditions	121
	IfThenEnd If	
	Either/Or Decisions: IfThenElseEnd If	
	Using IfElse IfEnd If for Multiple Conditions	
	Using Select CaseEnd Select for Multiple Conditions	
	Complex Expressions in Case Statements	124 124
	Next Steps	126
6	R1C1-Style Formulas	127
	Referring to Cells: A1 Versus R1C1 References	
	Switching Excel to Display R1C1-Style References	
	The Miracle of Excel Formulas	129
	Enter a Formula Once and Copy 1,000 Times	129
	The Secret: It's Not That Amazing	130
	Explanation of R1C1 Reference Style	132
	Using R1C1 with Relative References	132
	Using R1C1 with Absolute References	133
	Using R1C1 with Mixed References	133
	Referring to Entire Columns or Rows with R1C1 Style	134
	Replacing Many A1 Formulas with a Single R1C1 Formula	134
	Remembering Column Numbers Associated with Column Letters	136
	Array Formulas Require R1C1 Formulas	137
	Next Steps	

7	What Is New in Excel 2010 and What Has Changed	
	If It Has Changed in the Front End, It Has Changed in VBA	
	The Ribbon	
	Charts	
	Pivot Tables	140
	Slicers	140
	Conditional Formatting	140
	Tables	141
	Sorting	
	SmartArt	142
	Learning the New Objects and Methods	143
	Compatibility Mode	144
	Version	144
	Excel8CompatibilityMode	145
	Next Steps	146
8	Create and Manipulate Names in VBA	147
	Excel Names	147
	Global Versus Local Names	147
	Adding Names	
	Deleting Names	
	Adding Comments	150
	Types of Names	
	Formulas	151
	Strings	
	Numbers	152
	Tables	153
	Using Arrays in Names	
	Reserved Names	154
	Hiding Names	155
	Checking for the Existence of a Name	155
	Next Steps	158
9	Event Programming	
	Levels of Events	
	Using Events	
	Event Parameters	
	Enabling Events	161
	Workbook Events	
	Workbook Level Sheet and Chart Events	
	Worksheet Events	

Chart Sheet Events	172
Embedded Charts	172
Application-Level Events	176
Next Steps	
10 Userforms: An Introduction	
User Interaction Methods	
Input Boxes	
Message Boxes	
Creating a Userform	
Calling and Hiding a Userform	
Programming the Userform Userform Events	
Programming Controls	188
lising Basic Form Controls	189
Using Labels. Text Boxes, and Command Buttons	
Deciding Whether to Use List Boxes or Combo Boxes in Forms	
Adding Option Buttons to a Userform	
Adding Graphics to a Userform	195
Using a Spin Button on a Userform	196
Using the MultiPage Control to Combine Forms	198
Verifying Field Entry	
Illegal Window Closing	
Getting a Filename	201
Next Steps	
11 Creating Charts	203
Charting in Excel 2010	
Referencing Charts and Chart Objects in VBA Code	
Creating a Chart	
Specifying the Size and Location of a Chart	
Later Referring to a Specific Chart	
Recording Commands from the Layout or Design Tabs	
Specifying a Built-in Chart Type	
Specifying a Template Chart Type	210
Changing a Chart's Layout or Style	211
Using SetElement to Emulate Changes on the Layout Tab	213
Changing a Chart Title Using VBA	
Emulating Changes on the Format Tab	218
Using the Format Method to Access Formatting Options	218
Creating Advanced Charts	234
Creating True Open-High-Low-Close Stock Charts	235

Creating Bins for a Frequency Chart	236
Creating a Stacked Area Chart	239
Exporting a Chart as a Graphic	244
Creating a Dynamic Chart in a Userform	244
Creating Pivot Charts	246
Next Steps	
12 Data Mining with Advanced Filter	249
Replacing a Loop with AutoFilter	249
Using New AutoFilter Techniques	251
Selecting Visible Cells Only	255
Advanced Filter Is Easier in VBA Than in Excel	257
Using the Excel Interface to Build an Advanced Filter	258
Using Advanced Filter to Extract a Unique List of Values	258
Extracting a Unique List of Values with the User Interface	259
Extracting a Unique List of Values with VBA Code	
Getting Unique Combinations of Two or More Fields	
Using Advanced Filter with Criteria Ranges	
Joining Multiple Criteria with a Logical UK	
Joining Two Criteria Williad Logical AND	
The Most Complex Criteria: Replacing the List of Values with a Condition Created as the Result of a Formula	
Ilsing Filter in Place in Advanced Filter	275
Catching No Records When Using Filter in Place	
Showing All Records After Filter in Place	
The Real Workhorse: xlFilterCopy with All Records Rather Than Unique Records Only	
Copying All Columns	277
Copying a Subset of Columns and Reordering	278
Using Filter in Place with Unique Records Only	
Excel in Practice: Turning Off a Few Drop-Downs in the AutoFilter	
Next Steps	
13 Using VBA to Create Pivot Tables	287
Introducing Pivot Tables	
Understanding Versions	
New in Excel 2010	
New Beginning with Excel 2007	
Creating a Vanilla Pivot Table in the Excel Interface	290
Understanding Compact Layout	293
Building a Pivot Table in Excel VBA	
Defining the Pivot Cache	
Creating and Configuring the Pivot Table	
Adding Fields to the Data Area	

Economy why rou cannot wove of change fait of a fivot nepott	
Determining Size of a Finished Pivot Table to Convert the Pivot Table to Values	
Using Advanced Pivot Table Features	
Using Multiple Value Fields	
Counting the Number of Records	
Grouping Daily Dates to Months, Quarters, or Years	
Changing the Calculation to Show Percentages	
Eliminating Blank Cells in the Values Area	
Controlling the Sort Order with AutoSort	
Replicating the Report for Every Product	
Filtering a Data Set	
Manually Filtering Two or More Items in a Pivot Field	
Using the Conceptual Filters	
Using the Search Filter	
Setting Up Slicers to Filter a Pivot Table	
Filtering an OLAP Pivot Table Using Named Sets	
Using Other Pivot Table Features	
Calculated Data Fields	
Calculated Items	
Using ShowDetail to Filter a Recordset	
Changing the Layout from the Design Tab	
Suppressing Subtotals for Multiple Row Fields	
Next Steps	
14 Excel Power	
File Operations	
File Operations List Files in a Directory	
File Operations List Files in a Directory Import CSV	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Word	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments with Centering	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments with Centering Place a Chart in a Comment	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments with Centering Place a Chart in a Comment Utillities to Wow Your Clients	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments with Centering Place a Chart in a Comment Utilities to Wow Your Clients Using Conditional Formatting to Highlight Selected Cell	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments with Centering Place a Chart in a Comment Utilities to Wow Your Clients Using Conditional Formatting to Highlight Selected Cell Highlight Selected Cell Without Using Conditional Formatting	
File Operations List Files in a Directory Import CSV Read Entire TXT to Memory and Parse Combining and Separating Workbooks Separate Worksheets into Workbooks Combine Workbooks Filter and Copy Data to Separate Worksheets Export Data to Word Working with Cell Comments List Comments Resize Comments Resize Comments with Centering Place a Chart in a Comment Utilities to Wow Your Clients Using Conditional Formatting to Highlight Selected Cell Highlight Selected Cell Without Using Conditional Formatting Custom Transpose Data	

Techniques for VBA Pros	
Pivot Table Drill-Down	
Speedy Page Setup	
Calculating Time to Execute Code	
Custom Sort Order	
Cell Progress Indicator	
Protected Password Box	
Change Case	
Selecting with SpecialCells	
ActiveX Right-Click Menu	
Cool Applications	
Historical Stock/Fund Quotes	
USING VBA Extensibility to Add Code to New Workbooks	
Next Steps	
15 Data Visualizations and Conditional Formatting	
Introduction to Data Visualizations	
VBA Methods and Properties for Data Visualizations	
Adding Data Bars to a Range	
Adding Color Scales to a Range	
Adding Icon Sets to a Range	
Specifying an Icon Set	
Specifying Ranges for Each Icon	
Using Visualization Tricks	
Creating an Icon Set for a Subset of a Range	
Using Two Colors of Data Bars in a Range	
Using Other Conditional Formatting Methods	
Formatting Cells That Are Above or Below Average	
Formatting Cells in the Top 10 or Bottom 5	
Formatting Unique or Duplicate Cells	
Formatting Cells Based on Their Value	
Formatting Cells That Contain Text	
Formatting Cells That Contain Dates	
Ilsing a Formula to Determine Which Cells to Format	387
Using the New Number Format Property	388
Next Stens	389
16 Daading from and Writing to the Web	201
To reading 110111 allu Willing to the Web	
Ucting vala 110111 life Web Manually Creating a Web Auery and Refrecting with VRA	וענ רמכ
Ilsing VRA to Undate an Existing Web Overv	
Building Many Web Oueries with VBA	
· · · · · · · · · · · · · · · · · · ·	

Using Application . On Time to Periodically Analyze Data	
Scheduled Procedures Require Ready Mode	
Specifying a Window of Time for an Update	
Canceling a Previously Scheduled Macro	
Closing Excel Cancels All Pending Scheduled Macros	
Scheduling a Macro to Run x Minutes in the Future	
Scheduling a Verbal Keminder	
Scheduling a Macro to Run Every 2 Minutes	
Publishing Data to a Web Page	
Using VBA to Create Custom web Pages	
USHIY EXCEL AS A CONCENT MANAGEMENT SYSTEM	
DOILUS, FTF HOUL EXCEL	
Next Steps	
17 Dashboarding with Sparklines in Excel 2010	411
Creating Sparklines	412
Scaling the Sparklines	414
Formatting Sparklines	418
Using Theme Colors	418
Using RGB Colors	
Formatting Sparkline Elements	
Formatting Win/Loss Charts	
Creating a Dashboard	
Observations About Sparklines	
Creating 100's of Individual Sparklines in a Dashboard	
Next Steps	
18 Automating Word	
Early Binding	
Compile Error: Can't Find Object or Library	
Late Binding	
Creating and Referencing Objects	
The New Keyword	
CreateObject Function	
GetObject Function	438
Using Constant Values	439
Using the Watch Window to Retrieve the Real Value of a Constant	
Using the Object Browser to Retrieve the Real Value of a Constant	
Understanding Word's Objects	
Document Object	
Selection Object	
Range Object	
Bookmarks	

Controlling Form Fields in Word	450
Next Steps	452
19 Arrays	
Declare an Array	
Multidimensional Arrays	454
Fill an Array	455
Empty an Array	456
Arrays Make It Easier to Manipulate Data, but Is That All?	
Dynamic Arrays	459
Passing an Array	
Next Steps	
20 Text File Processing	
Importing from Text Files	
Importing Text Files with Fewer Than 1,048,576 Rows	
Reading Text Files with More Than 1,048,576 Rows	
Writing Text Files	
Next Steps	
21 Using Access as a Back End to Enhance Multiuser Access to I	Data475
ADO Versus DAO	
The Tools of ADO	
Adding a Record to the Database	
Retrieving Records from the Database	
Updating an Existing Record	
Deleting Records via ADO	
Summarizing Records via ADO	
Other Utilities via ADO	
Checking for the Existence of Tables	
Checking for the Existence of a Field	
Adding a Table On the Fly	
Adding a Field On the Fly	
SQL Server Examples	
22 Creating Classes, Records, and Collections	
Inserting a Class Module	
Irapping Application and Embedded Chart Events	
Application Events	
Croating a Custom Object	
Creating a Custoffi Object	

Using a Custom Object	
Using Property Let and Property Get to Control How Users Utilize Custom Objects	
Collections	501
Creating a Collection in a Standard Module	501
Creating a Collection in a Class Module	502
User-Defined Types	506
Next Steps	509
23 Advanced Userform Techniques	511
Using the UserForm Toolbar in the Design of Controls on Userforms	511
More Userform Controls	
Check Boxes	512
Tab Strips	513
RefEdit	515
Toggle Buttons	517
Using a Scrollbar As a Slider to Select Values	517
Controls and Collections	519
Modeless Userforms	521
Using Hyperlinks in Userforms	522
Adding Controls at Runtime	523
Resizing the Userform On-the-fly	524
Adding a Control On-the-fly	525
Sizing On-the-fly	
Adding Other Controls	
Adding an Image Un-the-fly	
Adding Help to the Userform	
Showing Accelerator Reys	
Creating the Tah Order	
Coloring the Active Control	
Transparent Forms	
Next Steps	534
24 Windows API	
— What Is the Windows API?	
Understanding an API Declaration	
Using an API Declaration	
API Examples	
Retrieve the Computer Name	
Check Whether an Excel File Is Open on a Network	539
Retrieve Display-Resolution Information	540

Custom About Dialog	541
Disable the X for Closing a Userform	541
Running Timer	542
Playing Sounds	543
Retrieving a File Path	543
Finding More API Declarations	547
Next Steps	547
25 Handling Errors	549
What Happens When an Error Occurs?	549
Debug Error Inside Userform Code Is Misleading	551
Basic Error Handling with the On Error Goto Syntax	552
Generic Error Handlers	554
Handling Errors by Choosing to Ignore Them	554
Suppressing Excel Warnings	556
Encountering Errors on Purpose	556
Train Your Clients	557
Errors While Developing Versus Errors Months Later	557
Runtime Error 9: Subscript Out of Range	557
RunTime Error 1004: Method Range of Object Global Failed	558
The Ills of Protecting Code	559
More Problems with Passwords	560
Errors Caused by Different Versions	561
Next Steps	562
26 Customizing the Ribbon to Run Macros	563
Out with the Old, In with the New	563
Where to Add Your Code: customui Folder and File	564
Creating the Tab and Group	565
Adding a Control to Your Ribbon	566
Accessing the File Structure	571
Understanding the RELS File	571
Renaming the Excel File and Opening the Workbook	572
Custom UI Editor Tool	572
Using Images on Buttons	572
Microsoft Office Icons	573
Custom Icon Images	574
Troubleshooting Error Messages	577
The Attribute "Attribute Name" on the Element "customui Ribbon" Is Not Defined in the DTD/Schema	577
Illegal Qualified Name Character	578

Element "customui Tag Name" Is Unexpected According to Content Model of Parent Element "customui	
Tag Name"	578
Excel Found Unreadable Content	579
Wrong Number of Arguments or Invalid Property Assignment	580
Nothing Happens	580
Other Ways to Run a Macro	580
Keyboard Shortcut	580
Attach a Macro to a Command Button	581
Attach a Macro to a Shape	582
Attach a Macro to an ActiveX Control	
Running a Macro from a Hyperlink	584
Next Steps	585
27 Creating Add-Ins	
Characteristics of Standard Add-Ins	587
Converting an Excel Workbook to an Add-In	
Using Save As to Convert a File to an Add-In	
Using the VB Editor to Convert a File to an Add-In	590
Having Your Client Install the Add-In	591
Standard Add-Ins Are Not Secure	592
Closing Add-Ins	593
Removing Add-Ins	593
Using a Hidden Workbook as an Alternative to an Add-In	593
Next Steps	595
Index	

## **About the Authors**

**Bill Jelen**, Excel MVP and the host of MrExcel.com, has been using spreadsheets since 1985, and he launched the MrExcel.com website in 1998. Bill was a regular guest on Call for Help with Leo Laporte and has produced more than 1,200 episodes of his daily video podcast, Learn Excel from MrExcel. He is the author of 30 books about Microsoft Excel and writes the monthly Excel column for *Strategic Finance* magazine. You will most frequently find Bill taking his show on the road, doing half-day Power Excel seminars wherever he can find a room full of accountants or Excellers. Before founding MrExcel.com, Jelen spent 12 years in the trenches—working as a financial analyst for finance, marketing, accounting, and operations departments of a \$500 million public company. He lives near Akron, Ohio, with his wife, Mary Ellen, and his sons, Josh and Zeke.

**Tracy Syrstad** is the project manager for the MrExcel consulting team. She was introduced to Excel VBA by a co-worker who encouraged her to learn VBA by recording steps and then modifying the code as needed. Her first macro was a simple lookup and highlight for a parts index, although it hardly seemed simple then. But she was encouraged by this success and others to follow. She'll never forget the day when it all clicked. She hopes this book will bring that click to its readers sooner and with less frustration. She lives near Sioux Falls, South Dakota, with her husband, John.

## Dedication

To everyone in the MrExcel.com message board community. —Bill Jelen

To John, who would only accept perfection, even if it took four coats of paint. —Tracy Syrstad

## Acknowledgments

Thanks to Tracy Syrstad for being a great co-author and for doing a great job of managing all the consulting projects at MrExcel.com.

Bob Umlas is the smartest Excel guy I know and is an awesome technical editor. At Pearson, Loretta Yates is an excellent acquisitions editor.

Along the way, I've learned a lot about VBA programming from the awesome community at the MrExcel.com message board. VoG and Richard Schollar and Jon von der Heyden all stand out as having contributed posts that lead to ideas in this book. Thanks to Pam Gensel for Excel macro lesson #1. Mala Singh taught me about creating charts in VBA, and Oliver Holloway brought me up to speed with accessing SQL Server.

At MrExcel.com, thanks to Barb Jelen, Wei Jiang, Tracy Syrstad, Schar Oswald, and Scott Pierson. Thanks also to Josh and Zeke Jelen, who have been picking up hours after school learning how to edit and produce the MrExcel podcast.

Finishing five Excel books for Excel 2010 simultaneously has been a monumental task. My family was incredibly supportive during this time. Thanks for Josh, Zeke, and Mary Ellen Jelen.

—Bill

Thanks to Bill Jelen, whose trust in me to run the consulting side of his business has done so much in building my self-confidence. And to LKH, whose blog I've learned so much from about writing and balancing working in the home and still having a personal life.

Richard Schollar and Joe Miskey: You've both been invaluable managing member issues at the forum and I feel I don't say thank you often enough. Thank you! And thanks to all the moderators who keep the board organized, despite the best efforts of the spammers.

There have been so many MrExcel.com clients whose projects have shown myriad ways that Excel can be used. Your excitement and appreciation over the solution we provide you has brightened my day as often as your unique projects have kept this job interesting.

—Tracy

## We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an associate publisher for Que Publishing, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.

When you write, please be sure to include this book's title and author as well as your name, email address, and phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Greg Wiegand Associate Publisher Que Publishing 800 East 96th Street Indianapolis, IN 46240 USA

## **Reader Services**

Visit our website and register this book at quepublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

## **Getting Results with VBA**

As corporate IT departments have found themselves with long backlogs of requests, Excel users have discovered they can produce the reports needed to run their business themselves using the macro language *Visual Basic for Applications* (VBA). VBA enables you to achieve tremendous efficiencies in your day-today use of Excel. This is both a good and bad thing. On the good side, without waiting for resources from IT, VBA helps you figure out how to import data and produce reports in Excel. On the bad side, you are now stuck importing data and producing reports in Excel.

## What Is in This Book?

You have taken the right step by purchasing this book. I can help you reduce the learning curve so that you can write your own VBA macros and put an end to the burden of generating reports manually.

#### **Reduce the Learning Curve**

This Introduction provides a brief history of spreadsheets. Chapter 1 introduces the tools and confirms what you probably already know: The macro recorder does not work. Chapter 2 helps you understand the crazy syntax of VBA. Chapter 3 breaks the code on how to work efficiently with ranges and cells. By the time you get to Chapter 4, you will know enough to begin using the 25 sample user-defined functions in that chapter.

Chapter 5 covers the power of looping using VBA. The case study in this chapter creates a program to produce a department report, and then wraps that report routine in a loop to produce 46 reports.

## INTRODUCTION

#### IN THIS INTRODUCTION

Getting Results with VBA1
What Is in This Book?1
The Future of VBA and Windows Versions of Excel
Special Elements and Typographical Conventions5
Code Files6

Chapter 6 covers R1C1-style formulas. Chapter 7 looks at what changed in Excel VBA from Excel 2003 to Excel 2010. In the past, it was fairly straightforward to create VBA code that would run on any of the recent versions of Excel. Unfortunately, with the sweeping changes in Excel 2007 and Excel 2010, it became significantly more difficult to create this VBA code. Chapter 8 covers names. Chapter 9 includes some great tricks that use event programming. Chapter 10 introduces custom dialog boxes that you can use to collect information from the human using Excel.

#### **Excel VBA Power**

Chapters 11 through 13 provide an in-depth look at charting, Advanced Filter, and pivot tables. Any report automation tool will rely heavily on these concepts. Chapter 14 includes 25 code samples designed to exhibit the power of Excel VBA.

Chapters 15 through 18 handle data visualizations, web queries, sparklines, and automating another Office program such as Word.

#### **Techie Stuff Needed to Produce Applications**

Chapter 19 shows how to use arrays to build fast applications. Chapters 20 and 21 handle reading and writing to text files and Access databases. The techniques for using Access databases enable you to build an application with the multi-user features of Access while keeping the friendly front end of Excel.

Chapter 22, as it examines classes and collections, covers VBA from a Visual Basic programmer's point of view. Chapter 23 discusses advanced userform topics. Chapter 24 teaches some tricky ways to achieve tasks using the Windows application programming interface. Chapters 25 through 27 deal with error handling, custom menus, and add-ins.

#### **Does This Book Teach Excel?**

Microsoft believes the average Office user touches only 10 percent of the features in Office. I realize everyone reading this book is above average, and I have a pretty smart audience at MrExcel.com. Even so, a poll of 8,000 MrExcel.com readers shows that only 42 percent of smarter-than-average users are using any one of the top 10 power features in Excel.

I regularly present a Power Excel seminar for accountants. These are hard-core Excelers who use Excel 30 to 40 hours every week. Even so, two things come out in every seminar. First, half the audience gasps when they see how quickly you can do tasks with a particular feature such as automatic subtotals or pivot tables. Second, someone in the audience routinely trumps me. For example, someone asks a question, I answer, and someone in the second row raises a hand to give a better answer.

The point? You and I both know a lot about Excel. However, I will assume that in any given chapter, maybe 58 percent of the people have not used pivot tables before and maybe even fewer have used the "Top 10 Filter" feature of pivot tables. With this in mind, before I show how to automate something in VBA, I briefly cover how to do the same task in the

Excel interface. This book does not teach you how to do pivot tables, but it does alert you when you might need to explore a topic and learn more about it elsewhere.

#### CASE STUDY: MONTHLY ACCOUNTING REPORTS

This is a true story. Valerie is a business analyst in the accounting department of a medium-size corporation. Her company recently installed an overbudget \$16 million ERP system. As the project ground to a close, there were no resources left in the IT budget to produce the monthly report that this corporation used to summarize each department.

However, Valerie had been close enough to the implementation process to think of a way to produce the report herself. She understood that she could export General Ledger data from the ERP system to a text file with comma-separated values. Using Excel, Valerie was able to import the G/L data from the ERP system into Excel.

Creating the report was not easy. Like many companies, there were exceptions in the data. Valerie knew that certain accounts in one particular cost center needed to be reclassed as an expense. She knew that other accounts needed to be excluded from the report entirely. Working carefully in Excel, Valerie made these adjustments. She created one pivot table to produce the first summary section of the report. She cut the pivot table results and pasted them into a blank worksheet. Then she created a new pivot table report for the second section of the summary. After about 3 hours, she had imported the data, produced five pivot tables, arranged them in a summary, and neatly formatted the report in color.

#### **Becoming the Hero**

Valerie handed the report to her manager. The manager had just heard from the IT department that it would be months before they could get around to producing "that convoluted report." When Valerie created the Excel report, she became the instant hero of the day. In 3 hours, Valerie had managed to do the impossible. Valerie was on cloud nine after a well-deserved "atta-girl."

#### **More Cheers**

The next day, Valerie's manager attended the monthly department meeting. When the department managers started complaining that they could not get the report from the ERP system, this manager pulled out his department report and placed it on the table. The other managers were amazed. How was he able to produce this report? Everyone was relieved to hear that someone had cracked the code. The company president asked Valerie's manager if he could have the report produced for each department.

#### **Cheers Turn to Dread**

You can probably see this coming. This particular company had 46 departments. That means 46 one-page summaries had to be produced once a month. Each report required importing data from the ERP system, backing out certain accounts, producing five pivot tables, and then formatting the reports in color. Even though it had taken Valerie 3 hours to produce the first report, after she got into the swing of things, she could produce the 46 reports in 40 hours. This is horrible. Valerie had a job to do before she became responsible of spending 40 hours a month producing these reports in Excel.

#### **VBA to the Rescue**

Valerie found my company, MrExcel Consulting, and explained her situation. In the course of about a week, I was able to produce a series of macros in Visual Basic that did all the mundane tasks. For example, it imported the data, backed out certain accounts, did five pivot tables, and applied the color formatting. From start to finish, the entire 40-hour manual process was reduced to two button clicks and about 4 minutes.

Right now, either you or someone in your company is probably stuck doing manual tasks in Excel that can be automated with VBA. I am confident that I can walk into any company with 20 or more Excel users and find a case equally amazing as Valerie's.

## The Future of VBA and Windows Versions of Excel

Seven years ago, there were many rumblings that Microsoft might stop supporting VBA. There is now plenty of evidence to indicate that VBA will be around in Windows versions of Excel through 2025. When VBA was removed from the Mac version of Excel 2008, a huge outcry from customers led to it being included in the next Mac version of Excel.

Microsoft has stated that in Excel 15, which is the next version of Excel, it will stop providing support for XLM macros. These macros were replaced by VBA in 1993, and 17 years later, they are still supported. There is a chance that Microsoft will introduce a new programming language for the macro recorder in Excel 15. Assuming Microsoft continues to support VBA for 17 years after Excel 2012, you should be good through the mid-to-late 2020s.

However, you can see Microsoft's lack of commitment to VBA. Office 2003 offered a few features, such as the Research Pane and SmartTags, which could only be automated with Visual Basic.Net. The charting macro recorder, which was not finished in time to ship with Excel 2007, is included in Excel 2010.

The tools that you learn today will be good for the next 15 years. Even if Microsoft decides to scrap VBA in favor of another language, your coding skills will most likely transfer to the new platform.

#### **Versions of Excel**

This Third Edition of *VBA and Macros* is designed to work with Excel 2010. The previous editions of this book covered code for Excel 97 through Excel 2007. In 80 percent of the chapters, the code for Excel 2010 will be identical to code in previous versions. However, there are exceptions. For example, Microsoft offers new sorting logic, and charts have changed completely. In addition, the conditional formatting and data visualization tools in Chapter 15 are brand new. With Excel 2010, pivot tables offer new calculation options and slicers. The XML examples in Chapter 17 will work only with Excel 2003 or newer.

#### **Differences for Mac Users**

Although Excel for Windows and Excel for the Mac are similar in their user interface, there are a number of differences when you compare the VBA environment. Certainly, nothing in Chapter 24 that uses the Windows API will work on the Mac. The overall concepts discussed in the book apply to the Mac, but differences will exist. You can find a general list of differences as they apply to the Mac at http://www.mrexcel.com/macvba.html.

## **Special Elements and Typographical Conventions**

The following typographical conventions are used in this book:

- *Italic*—Indicates new terms when they are defined, special emphasis, non-English words or phrases, and letters or words used as words
- Monospace—Indicates parts of VBA code such as object or method names, and filenames
- Italic monospace—Indicates placeholder text in code syntax

**Bold monospace**—Indicates user input

In addition to these typographical conventions, there are several special elements. Each chapter has at least one case study that presents a real-world solution to common problems. The case study also demonstrates practical applications of topics discussed in the chapter.

In addition to the case studies, you will see New icons, Notes, Tips, and Cautions.

Notes provide additional information outside the main thread of the chapter discussion that might be useful for you to know.



Tips provide quick workarounds and time-saving techniques to help you work more efficiently.

CAUTION -

Cautions warn about potential pitfalls you might encounter. Pay attention to the Cautions; they alert you to problems that may otherwise cause you hours of frustration.

## **Code Files**

As a thank you for buying this book, the authors have put together a set of 50 Excel workbooks that demonstrate the concepts included in this book. This set of files includes all the code from the book, sample data, additional notes from the authors, and 25 additional bonus macros. To download the code files, visit this book's web page at http://www.quepublishing.com or http://www.mrexcel.com/getcode2010.html.

## **Next Steps**

Chapter 1 introduces the editing tools of the Visual Basic environment and shows why using the macro recorder is not an effective way to write VBA macro code.

# Unleash the Power of Excel with VBA

## **The Power of Excel**

*Visual Basic for Applications* (VBA) combined with Microsoft Excel is probably the most powerful tool available to you. VBA is sitting on the desktops of 500 million users of Microsoft Office, and most have never figured out how to harness the power of VBA in Excel. Using VBA, you can speed the production of any task in Excel. If you regularly use Excel to produce a series of monthly charts, you can have VBA do the same task for you in a matter of seconds.

## **Barriers to Entry**

There are two barriers to learning successful VBA programming. First, Excel's macro recorder is flawed and does not produce workable code for you to use as a model. Second, for many who learned a programming language such as BASIC, the syntax of VBA is horribly frustrating.

#### The Macro Recorder Doesn't Work!

Microsoft began to dominate the spreadsheet market in the mid-1990s. Although it was wildly successful in building a powerful spreadsheet program to which any Lotus 1-2-3 user could easily transition, the macro language was just too different. Anyone proficient in recording Lotus 1-2-3 macros who tried recording a few macros in Excel most likely failed. Although the Microsoft VBA programming language is much more powerful than the Lotus 1-2-3 macro language, the fundamental flaw is that the macro recorder does not work.

With Lotus 1-2-3, you could record a macro today, play it back tomorrow, and it would faithfully work.

## IN THIS CHAPTER

The Power of Excel7	1
Barriers to Entry7	,
Knowing Your Tools: The Developer Tab	9
Macro Security10	9
Overview of Recording, Storing, and Running a Macro12	2
Running a Macro14	ł
Using New File Types in Excel 201018	8
Understanding the VB Editor19	9
Understanding Shortcomings of the Macro Recorder	



When you attempt the same feat in Microsoft Excel, the macro might work today but not tomorrow. In 1995, when I tried to record my first Excel macro, I was horribly frustrated by this.

#### Visual Basic Is Not Like BASIC

The code generated by the macro recorder was unlike anything I had ever seen. It said this was "Visual Basic" (VB). I had the pleasure of learning half a dozen programming languages at various times; this bizarre-looking language was horribly unintuitive and did not resemble the BASIC language I had learned in high school.

To make matters worse, even in 1995 I was the spreadsheet wizard in my office. My company had forced everyone to convert from Lotus 1-2-3 to Excel, which meant I was faced with a macro recorder that didn't work and a language that I couldn't understand. This was not a good combination of events.

My assumption in writing this book is that you are pretty talented with a spreadsheet. You probably know more than 90 percent of the people in your office. I also assume that even though you are not a programmer, you might have taken a class in BASIC at some point. However, knowing BASIC is not a requirement—it actually is a barrier to entry into the ranks of being a successful VBA programmer. There is a good chance that you have recorded a macro in Excel and a similar chance that you were not happy with the results.

#### Good News: Climbing the Learning Curve Is Easy

Even if you've been frustrated with the macro recorder, it is really just a small speed bump on your road to writing powerful programs in Excel. This book will not only teach you why the macro recorder fails, but also how to change the recorded code into something useful. For all the former BASIC programmers in the audience, I will decode VBA so that you can easily pick through recorded macro code and understand what is happening.

#### Great News: Excel with VBA Is Worth the Effort

Although you probably have been frustrated with Microsoft over the inability to record macros in Excel, the great news is that Excel VBA is powerful. Absolutely anything you can do in the Excel interface can be duplicated with stunning speed in Excel VBA. If you find yourself routinely creating the same reports manually day after day or week after week, Excel VBA will greatly streamline those tasks.

The authors of this book work for MrExcel Consulting. In this role, we have automated reports for hundreds of clients. The stories are often similar: The MIS department has a several-month backlog of requests. Someone in accounting or engineering discovers that he or she can import some data into Excel and get the reports necessary to run the business. This is a liberating event—you no longer need to wait months for the IT department to write a program. However, the problem is that after you import the data into Excel and win accolades from your manager for producing the report, you will likely be asked to produce the same report every month or every week. This becomes very tedious.

8

Again, the great news is that with a few hours of VBA programming, you can automate the reporting process and turn it into a few button clicks. The reward is great. So, hang with me as we cover a few of the basics.

This chapter exposes why the macro recorder does not work. It also walks through an example of recorded code and demonstrates why it will work today but fail tomorrow. I realize that the code you see in this chapter might not be familiar to you, but that's okay. The point of this chapter is to demonstrate the fundamental problem with the macro recorder. You also learn the fundamentals of the Visual Basic environment.

## **Knowing Your Tools: The Developer Tab**

Let's start with a basic overview of the tools needed to use VBA. By default, Microsoft hides the VBA tools. You need to complete the following steps to change a setting in Excel options to access the Developer tab.

- 1. Open the File menu to get to the new Backstage view.
- 2. Along the left navigation bar, select Options under Excel.
- 3. In the Excel Options dialog, select Customize Ribbon from the left navigation.
- **4.** In the Right list box, the Developer tab is third from the bottom. Select the check box next to this item.
- 5. Click OK to return to Excel.

Excel displays the Developer tab shown in Figure 1.1.

Figure 1.1	- EL -	Home	Insert	Page Layout	Formula	s Data	a Rev	iew	View	Developer		
The Developer tab	1		Record Mac	ro		Prop	perties	3	🐨 Maj	p Properties	📑 Import	0
provides an interface for	Visual I Basic	Macros	Jse Relative Macro Secur	References rity	Insert Desig	□ Q⊒ Viev □ 1 Run	v Code Dialog	Source	Exp	ansion Packs resh Data	避 Export	Document
running and recording	Dasic	C	ode		C	ontrols			1	XML		Modify
macros.												

The Code group on the Developer tab contains the icons used for recording and playing back VBA macros, as listed here:



- Macros icon—Displays the Macro dialog, where you can choose to run or edit a macro from the list of macros.
- **Record Macro icon**—Begins the process of recording a macro.
- Use Relative Reference icon—Toggles between using relative or absolute recording. With relative recording, Excel will record that you move down three cells. With absolute recording, Excel will record that you selected cell A4.
- Macro Security icon—Accesses the Trust Center, where you can choose to allow or disallow macros to run on this computer.

The Controls group of the Developer tab contains an Insert menu where you can access a variety of programming controls that can be placed on the worksheet. See "Assigning a Macro to a Form Control, Text Box, or Shape," later in this chapter. Other icons in this group enable you to work with the on-sheet controls. The Run Dialog button enables you to display a custom dialog box or userform that you designed in VBA. For more on userforms, see Chapter 10, "Userforms: An Introduction."

The XML group of the Developer ribbon contains tools for importing and exporting XML documents.

## **Macro Security**

After VBA macros were used as the delivery method for some high-profile viruses, Microsoft changed the default security settings to prevent macros from running. Therefore, before we can begin discussing the recording of a macro, we need to show you how to adjust the default settings.

In Excel 2010, you can either globally adjust the security settings or control macro settings for certain workbooks by saving the workbooks in a trusted location. Any workbooks stored in a folder that is marked as a trusted location will automatically have its macros enabled.

You can find the macro security settings under the Macro Security icon on the Developer tab. When you click this icon, the Macro Settings category of the Trust Center is displayed. You can use the left navigation bar in the dialog to access the Trusted Locations list.

#### **Adding a Trusted Location**

You can choose to store your macro workbooks in a folder that is marked as a trusted location. Any workbook stored in a trusted folder will have its macros enabled. Microsoft suggests that a trusted location should be on your hard drive. The default setting is that you cannot trust a location on a network drive.

To specify a trusted location, follow these steps:

- 1. Click Macro Security in the Developer tab.
- 2. Click Trusted Locations in the left navigation pane of the Trust Center.
- **3.** If you want to trust a location on a network drive, select Allow Trusted Locations on My Network.
- Click the Add New Location button. Excel displays the Microsoft Office Trusted Locations dialog (see Figure 1.2).
- 5. Click the Browse button. Excel displays the Browse dialog.
- **6.** Browse to the parent folder of the folder you want to be a trusted location. Click the trusted folder. Although the folder name does not appear in the Folder Name box, click OK. The correct folder name will appear in the Browse dialog.

- **7.** If you want to trust subfolders of the selected folder, select Subfolders of This Location Will Be Trusted.
- 8. Click OK to add the folder to the Trusted Locations list.





Although trusted locations are not new in Excel 2010, Microsoft has made the process of adding trusted locations more discoverable in Excel 2010.

#### Using Macro Settings to Enable Macros in Workbooks Outside of Trusted Locations

For all macros not stored in a trusted location, Excel relies on the macro settings. The Low, Medium, High, and Very High settings that were familiar in Excel 2003 have been renamed.

To access the macro settings, click Macro Security in the Developer tab. Excel displays the Macro Settings category of the Trust Center dialog. Select the second option, Disable All Macros with Notification. A description of each option follows:

■ Disable All Macros Without Notification—This setting prevents all macros from running. This setting is for people who never intend to run macros. Because you are currently holding a book that teaches you how to use macros, it is assumed that this setting is not you. This setting is roughly equivalent to the old Very High Security setting in Excel 2003. With this setting, only macros in the Trusted Locations folders can run.

■ Disable All Macros with Notification—This setting is similar to Medium security in Excel 2003 and is the recommended setting. In Excel 2003, a Medium setting caused a box to be displayed when you opened a file containing macros. This box forced the

1

person to choose either Enable or Disable. Many novice Excel users randomly choose from this box. In Excel 2010, the message is displayed in the Message Area that macros have been disabled. You can choose to enable the content by clicking that option, as shown in Figure 1.3.

- Disable All Macros Except Digitally Signed Macros—This setting requires you to obtain a digital signing tool from VeriSign or another provider. This might be appropriate if you are going to be selling add-ins to others, but a bit of a hassle if you just want to write macros for your own use.
- Enable All Macros (Not Recommended: Potentially Dangerous Code Can Run)—This setting is similar to Low macro security in Excel 2003. Although it requires the least amount of hassle, it also opens your computer up to attacks from malicious Melissa-like viruses. Microsoft suggests that you do not use this setting.

Figure	1	.3
--------	---	----

Open a macro workbook using the Disable All Macros with Notification setting to enable the macros.

💮 Security Warning			Macros have b	oeen disable	i. Enab	Enable Content			
	A3		• ()	$f_{x}$					
	A	В	С	D	E	F	G		

#### **Using Disable All Macros with Notification**

It is recommended that you set your macro settings to Disable All Content with Notification. If you use this setting and open a workbook that contains macros, you will see a Security Warning in the area just above the formula bar. Assuming you were expecting macros in this workbook, click Enable Content.

If you do not want to enable macros for the current workbook, dismiss the Security Warning by clicking the X at the far right of the message bar.

If you forget to enable the macros and attempt to run a macro, Excel indicates that you cannot run the macro because all macros have been disabled. If this occurs, close the workbook and reopen it to access the message bar again.

CAUTION -

After you enable macros in a workbook stored on a local hard drive and then save the workbook, Excel will remember that you previously enabled macros in this workbook. The next time you open this workbook, macros will be automatically enabled.

## **Overview of Recording, Storing, and Running a Macro**

Recording a macro is useful when you do not have experience in writing lines of code in a macro. As you gain more knowledge and experience, you will begin to record lines of code less frequently.

To begin recording a macro, select Record Macro from the Developer tab. Before recording begins, Excel displays the Record Macro dialog box, as shown in Figure 1.4.

Record Macro	? ×
Macro name:	
Macro 1	
Shortcut <u>k</u> ey: Ctrl+	
Store macro in:	
This Workbook	•
Description:	
	OK Cancel

#### Filling Out the Record Macro Dialog

Figure 1.4

Use the Record Macro dialog box to assign a name and a shortcut key to the macro being recorded.

> In the Macro Name field, type a name for the macro. Be sure to type continuous characters. For example, type Macro1 without a space, not Macro 1 with a space. Assuming you will soon be creating many macros, use a meaningful name for the macro. A name such as FormatReport is more useful than Macro1.

The second field in the Record Macro dialog box is a shortcut key. If you type J in this field, and then press Ctrl+J, this macro runs. Note that most of the lowercase shortcuts from Ctrl+a through Ctrl+z already have a use in Excel. Rather than being limited to the unassigned Ctrl+j, you can hold down the Shift key and type Shift+A through Shift+Z in the shortcut box. This will assign the macro to Ctrl+Shift+A.

C A U T I O N You can reuse a shortcut key for a macro. If you assign a macro to Ctrl+c, Excel will run your macro instead of doing the normal action of copy.

In the Record Macro dialog box, choose where you want to save a macro when it is recorded: Personal Macro Workbook, New Workbook, This Workbook. It is recommended that you store macros related to a particular workbook in This Workbook.

The Personal Macro Workbook (Personal.x1sb) is not a visible workbook; it is created if you choose to save the recording in the Personal Macro Workbook. This workbook is used to save a macro in a workbook that will open automatically when you start Excel, thereby enabling you to use the macro. After Excel is started, the workbook is hidden. If you want to display it, select Unhide from the View tab.

Lt is not recommended you use the personal workbook for every macro you save. Save only those macros that assist you in general tasks—not in tasks that are performed in a specific sheet or workbook.

The fourth box in the Record Macro dialog is for a description. This description is added as a comment to the beginning of your macro. Note that legacy versions of Excel automatically noted the date and username of the person recording the macro. Excel 2010 no longer automatically inserts this information in the Description field.

After you select the location where you want to store the macro, click OK. Record your macro. When you are finished recording the macro, click the Stop Recording icon in the Developer tab.

You can also access a Stop Recording icon in the lower-left corner of the Excel window. Look for a small blue square to the right of the word *Ready* in the status bar. Using this Stop button might be more convenient than returning to the Developer tab. After you record your first macro, this area will usually have a Record Macro icon, which is a small red dot on an Excel worksheet.

## **Running a Macro**

If you assigned a shortcut key to your macro, you can play it by pressing the key combination. Macros can also be assigned to toolbar buttons, forms controls, drawing objects, or you can run them from the Visual Basic toolbar.

#### Creating a Macro Button on the Ribbon

You can add an icon to a new group on the Ribbon to run your macro. This is appropriate for macros stored in the Personal Macro Workbook. Follow these steps to add a macro button to the Ribbon:

- 1. Click the File menu and select Excel Options to open the Excel Options dialog.
- **2.** In the Excel Options dialog, select the Customize Ribbon category from the left-side navigation.

Note that a shortcut to replace steps 1 and 2 is to right-click the Ribbon and select Customize Ribbon.

- **3.** In the list box on the right, choose the tab name where you want to add an icon.
- **4.** Click the New Group button below the right list box. Excel adds a new entry called New Group (Custom) to the end of the groups in that ribbon tab.

1

- **5.** To move the group to the left in the ribbon tab, click the up-arrow icon on the right side of the dialog several times.
- 6. To rename the group, click the Rename button. Type a new name, such as Report Macros. Click OK. Excel will show the group in the list box as Report Macros (Custom). Note that the word *Custom* will not appear in the Ribbon.
- **7.** Open the upper-left drop-down and choose Macros from the list. The Macros category is fourth in the list. Excel displays a list of available macros in the left list box.
- 8. Choose a macro from the left list box. Click the Add button in the center of the dialog. Excel moves the macro to the right list box in the selected group. Excel uses a generic VBA icon for all macros. You can change the icon by following steps 9 and 10.
- **9.** Click the macro in the right list box. Click the Rename button at the bottom of the right list box. Excel displays a list of 180 possible icons. Choose an icon. Alternatively, type a friendly label for the icon, such as Format Report.
- 10. Click OK to close Excel options. The new button appears on the selected Ribbon tab.

#### Creating a Macro Button on the Quick Access Toolbar

You can add an icon to the Quick Access toolbar to run your macro. If your macro is stored in the Personal Macro Workbook, you can have the button permanently displayed in the Quick Access toolbar. If the macro is stored in the current workbook, you can specify that the icon should appear only when the workbook is open. Follow these steps to add a macro button to the Quick Access toolbar:

- 1. Click the File menu and select Options to open the Excel Options dialog.
- **2.** In the Excel Options dialog select the Quick Access Toolbar category from the left-side navigation.

Note that a shortcut to replace steps 1 and 2 is to right-click the Quick Access toolbar and select

- Customize Quick Access Toolbar.
- 3. If your macro should be available only when the current workbook is open, open the upper-right drop-down and change For All Documents (Default) to For <FileName. xlsm>. Any icons associated with the current workbook are displayed at the end of the Quick Access toolbar.
- **4.** Open the upper-left drop-down and select Macros from the list. The Macros category is fourth in the list. Excel displays a list of available macros in the left list box.
- Choose a macro from the left list box. Click the Add button in the center of the dialog. Excel moves the macro to the right list box. Excel uses a generic VBA icon for all macros. You can change the icon by following steps 6 through 8.
- 6. Click the macro in the right list box. Click the Modify button at the bottom of the right list box. Excel displays a list of 180 possible icons (see Figure 1.5).
Figure 1.5

toolbar.

Attach a macro to a but-

ton on the Ouick Access

Considering Excel 2003 offered 4,096 possible icons and an icon editor, the list of 180 is a major disappointment.



Enter the ToolTip Here

- 7. Choose an icon from the list. In the Display Name box, replace the macro name with a short name that will appear in the ToolTip for the icon.
- 8. Click OK to close the Modify Button dialog.
- 9. Click OK to close Excel options. The new button appears on the Quick Access toolbar.

#### Assigning a Macro to a Form Control, Text Box, or Shape

If you want to create a macro specific to a workbook, store the macro in the workbook and attach it to a form control or any object on the sheet.

Follow these steps to attach a macro to a form control on the sheet:

- On the Developer tab, click the Insert button to open its drop-down list. Excel offers 12 form controls and 12 ActiveX controls. Many icons look similar in this drop-down. Click the Button Form Control icon at the upper-left icon in the drop-down.
- 2. Move your cursor over the worksheet; the cursor changes to a plus sign.
- **3.** Draw a button on the sheet by clicking and holding the left mouse button while drawing a box shape. Release the button when you have finished.
- **4.** Choose a macro from the Assign Macro dialog box and click OK. The button is created with generic text such as Button 1. To customize the text or the button appearance, follow steps 5 through 7.

- 5. Type a new label for the button. Note that while you are typing, the selection border around the button changes from dots to diagonal lines to indicate that you are in Text Edit mode. You cannot change the button color while in Text Edit mode. To exit Text Edit mode, either click the diagonal lines to change them to dots or Ctrl-click the button again. Note that if you accidentally click away from the button, you should Ctrl+click the button to select it. Then drag the cursor over the text on the button to select the text.
- **6.** Right-click the dots surrounding the button and select Format Control. Excel displays the Format Control dialog with seven tabs across the top. If your Format Control dialog has only a Font tab, you failed to exit Text Edit mode. If this occurred, close the dialog, Ctrl-click the button, and repeat this step.
- 7. Use the settings in the Format Control dialog to change the font size, font color, margins, and similar settings for the control. Click OK to close the Format Control dialog when you have finished. Click on a cell to unselect the button.
- 8. Click the button to run the macro.

Macros can be assigned to any worksheet object such as clip art, a shape, SmartArt graphics, or text box. In Figure 1.6, the top button is a traditional button form control. The other images are clip art, a shape with WordArt, and a SmartArt graphic. To assign a macro to any object, right-click the object, and select Assign Macro.



#### Figure 1.6

Assigning a macro to a form control or an object appropriate for macros stored in the same workbook as the control. You can assign a macro to any of these objects.

# Using New File Types in Excel 2010

Excel 2010 offers support for four file types. Macros are not allowed to be stored in the default file type. You have to use the Save As setting for all of your macro workbooks, or you can change the default file type used by Excel 2010.

The available files types are as follows:

- Excel Workbook (.xlsx)—Files are stored as a series of XML objects and then zipped into a single file. This new file-saving paradigm in Excel 2010 allows for significantly smaller file sizes. It also allows other applications (even Notepad!) to edit or create Excel workbooks. Unfortunately, macros cannot be stored in files with an .xlsx extension.
- Excel Macro-Enabled Workbook (.xlsm)—This is similar to the default .xlsx format, except macros are allowed. The basic concept is that if someone has an .xlsx file, he or she will not need to worry about malicious macros. However, if they see an .xlsm file, they should be concerned that there might be macros attached.
- **Excel Binary Workbook (.xlsb)**—This is a binary format designed to handle the larger 1.1-million-row grid size in Excel 2010. Legacy versions of Excel stored their files in a proprietary binary format. Although binary formats might load quicker, they are more prone to corruption, and a few lost bits can destroy the whole file. Macros are allowed in this format.
- Excel 97-2003 Workbook (.xls)—This format produces files that can be read by anyone using legacy versions of Excel. Macros are allowed in this binary format; however, when you save in this format, you lose access to any cells outside of A1:IV65536. In addition, if someone opens the file in Excel 2003, he or she will lose access to anything that used features introduced in Excel 2007 or later.

To avoid having to choose a macro-enabled workbook in the Save As dialog, you can customize your copy of Excel to always save new files in the .xlsm format by following these steps:

- 1. Click the File menu and select Excel Options.
- 2. In the Excel Options dialog, select the Save category from the left navigation pane.
- **3.** The first drop-down is Save Files in This Format. Open the drop-down and select Excel Macro-Enabled Workbook (\*.xlsm). Click OK.
  - Although you and I are not afraid to use macros, I have encountered some people who seem to freak
  - out when they see the .xlsm file type. They actually seem angry that I sent them an .xlsm file that did not have any macros. Their reaction seemed reminiscent of King Arthur's "You got me all worked up!" line in *Monty Python and the Holy Grail*.

If you encounter someone who seems to have a fear of the .xlsm file type, remind them of these points:

- Every workbook created in the past 20 years could have had macros, but in fact, most did not.
- If someone is trying to avoid macros, they should use the security settings to prevent macros from running anyway (refer to Figure 1.3). They can still open the .xlsm file to get the data in the spreadsheet.

With these arguments, I hope you can overcome any fears of the .xlsm file type so that it can be your default file type.

# Understanding the VB Editor

Figure 1.7 shows an example of the typical VB Editor screen. You can see three windows: Project Explorer, the Properties window, and the Programming window. Don't worry if your window doesn't look exactly like this because you will see how to display the windows you need in this review of the editor.



# **VB Editor Settings**

Several settings in the VB Editor enable you to customize this editor. The following subsection covers the setting that will help with your programming.

#### Customizing VB Editor Options Settings

Under Tools, Options, Editor, you will find several useful settings. All settings except for one are set correctly by default. The remaining setting requires some consideration on your part. This setting is Require Variable Declaration. By default, Excel does not require you to declare variables. I prefer this setting because it can save time when you create a program. My coauthor prefers to change this setting to require variable declaration. This change forces the compiler to stop if it finds a variable that it does not recognize, which reduces misspelled variable names. It is a matter of your personal preference if you turn this setting on or keep it off.

#### **The Project Explorer**

The Project Explorer lists any open workbooks and add-ins that are loaded. If you click the + icon next to the VBA Project, you will see that there is a folder with Microsoft Excel objects. There can also be folders for forms, class modules, and standard modules. Each folder includes one or more individual components.

Right-clicking a component and selecting View Code or just double-clicking the components brings up any code in the Programming window. The exception is userforms, where double-clicking displays the userform in Design view.

To display the Project Explorer window, select View, Project Explorer from the menu, and then press Ctrl+R or click the Project Explorer icon on the toolbar.

Figure 1.8 shows the Project Explorer pane. This pane can show Microsoft Excel objects, userforms, modules, and class modules.

×



🖧 Module 1

To insert a module, right-click your project, select Insert, and then choose the type of module you want. The available modules are as follows:

- Microsoft Excel objects—By default, a project consists of sheet modules for each sheet in the workbook and a single ThisWorkbook module. Code specific to a sheet such as controls or sheet events is placed on the corresponding sheet. Workbook events are placed in the ThisWorkbook module. You learn more about events in Chapter 9, "Event Programming."
- Forms—Excel allows you to design your own forms to interact with the user. You learn more about these forms in Chapter 10.
- **Modules**—When you record a macro, Excel automatically creates a module in which to place the code. Most of your code will reside in these types of modules.
- Class modules—Class modules are Excel's way of letting you create your own objects. They also allow pieces of code to be shared among programmers without the programmer needing to understand how it works. You will learn more about class modules in Chapter 22, "Creating Classes, Records, and Collections."

### **The Properties Window**

Figure 1.9 Invoice.txt file.

The Properties window enables you to edit the properties of various components such as sheets, workbooks, modules, and form controls. The Property list varies according to what component is selected. To display this window, select View, Properties Window from the menu, press F4, or click the Project Properties icon on the toolbar.

# **Understanding Shortcomings of the Macro Recorder**

Suppose you work in an accounting department. Each day you receive a text file from the company system showing all the invoices produced the prior day. This text file has commas separating each field. The columns in the file are InvoiceDate, InvoiceNumber, SalesRepNumber, CustomerNumber, ProductRevenue, ServiceRevenue, and ProductCost (see Figure 1.9).

File Edit Format View Help	
InvoiceDate, InvoiceNumber, SalesRepNumber, Customero 06/06/2011,123829,521,62754,53400,0,299807 06/06/2011,123830,545,46323,88200,0,307563 06/06/2011,123831,554,46323,88200,0,521726 06/06/2011,123832,521,46326,830900,0,494831 06/06/2011,123833,554,54325,673600,0,374953 06/06/2011,123833,552,41508,467100,0,257342 06/06/2011,123837,552,47366,58500,10000,308719 06/06/2011,123837,554,46533,191700,0,109534	Number,ProductRevenue,ServiceRevenue,ProductCos

Each morning, you manually import this file into Excel. You add a total row to the data, bold the headings, and then print the report for distribution to a few managers.

1

This seems like a simple process that would be ideally suited to using the macro recorder. However, due to some problems with the macro recorder, your first few attempts might not be successful. The following case study explains how to overcome these problems.

22

# CASE STUDY: PREPARING TO RECORD THE MACRO

The task mentioned in the previous section is perfect for a macro. However, before you record a macro, think about the steps you will use. In this case, the steps you will use are as follows:

- **1.** Click the File menu and select Open.
- 2. Navigate to the folder where Invoice.txt is stored.
- 3. Select All Files(\*.\*) from the Files of Type drop-down list.
- 4. Select Invoice.txt.
- 5. Click Open.
- 6. In the Text Import Wizard—Step 1 of 3, select Delimited from the Original Data Type section.
- 7. Click Next.
- 8. In the Text Import Wizard—Step 2 of 3, clear the Tab key and select Comma in the Delimiters section.
- 9. Click Next.
- 10. In the Text Import Wizard—Step 3 of 3, select General in the Column Data Format section and change it to Date: MDY.
- **11.** Click Finish to import the file.
- 12. Press the End key followed by the down arrow to move to the last row of data.
- 13. Press the down arrow one more time to move to the total row.
- 14. Type the word Total.
- **15.** Press the right-arrow key four times to move to Column E of the total row.
- 16. Click the Autosum button and press Ctrl+Enter to add a total to the Product Revenue column while remaining in that cell.
- 17. Click the AutoFill handle and drag it from Column E to Column G to copy the total formula to Columns F and G.
- 18. Highlight Row 1 and click the Bold icon on the Home tab to set the headings in bold.
- 19. Highlight the Total row and click the Bold icon on the Home tab to set the totals in bold.
- 20. Press Ctrl+A to select all cells.
- 21. From the Home tab, select Format, AutoFit Column Width.

After you have rehearsed these steps in your head, you are ready to record your first macro. Open a blank workbook and save it with a name such as MacroToImportInvoices.xlsm. Click the Record Macro button on the Developer tab.

In the Record Macro dialog, the default macro name is Macro1. Change this to something descriptive like ImportInvoice. Make sure that the macros will be stored in This Workbook. You might want an easy way to run this macro later, so enter the letter i in the Shortcut Key field. In the Description field, add a little descriptive text to tell what the macro is doing (see Figure 1.10). Click OK when you are ready.

1

Mag	ro name:
	ImportInvoice
Sho	rtcut key:
	Ctrl+ i
Sto	re macro in:
	This Workbook
Des	cription:
	Imports invoice.txt, adds a total row. Does some formatting

# **Recording the Macro**

Figure 1.10 Before recording your macro, complete the

Record Macro dialog box.

The macro recorder is now recording your every move, but don't be nervous. For this reason, perform your steps in exact order without extraneous actions. If you accidentally move to Column F, type a value, clear the value, and then move back to E to enter the first total, the recorded macro blindly makes that same mistake day after day after day. Recorded macros move fast, but there is nothing like watching the macro recorder play out your mistakes repeatedly.

Carefully, execute all the actions necessary to produce the report. After you have performed the final step, click the Stop button in the lower-left corner of the Excel window or click Stop Recording in the Developer tab.

Now it is time to look at your code. Switch to the VB Editor by selecting Visual Basic from the Developer tab or pressing Alt+F11.

#### **Examining Code in the Programming Window**

Let's look at the code you just recorded from the case study. Don't worry if it doesn't make sense yet.

To open the VB Editor, press Alt+F11. In your VBA Project (MacroToImportInvoices.x1s), find the component Module1, right-click the module, and select View Code. Notice that some lines start with an apostrophe—these are comments and are ignored by the program. The macro recorder starts your macros with a few comments, using the description you entered in the Record Macro dialog. The comment for the Keyboard Shortcut is there to remind you of the shortcut.

The comment does *not* assign the shortcut. If you change the comment to be Ctrl+J, it does not change the shortcut. You must change the setting in the Macro dialog box in Excel or run this line of code:

Application.MacroOptions Macro:="ImportInvoice", \_ Description:="", ShortcutKey:="j" Recorded macro code is usually pretty neat (see Figure 1.11). Each noncomment line of code is indented four characters. If a line is longer than 100 characters, the recorder breaks it into multiple lines and indents the lines an additional four characters. To continue a line of code, type a space and an underscore at the end of the line.

Note that the physical limitations of this book do not allow 100 characters on a single line. Therefore,

- the lines will be broken at 80 characters so that they fit on a page. For this reason, your recorded macro
  - might look slightly different from the ones that appear in this book.

Figure 1.11 The recorded macro is neat looking and nicely indented.

eneral)	▼ Importinvoice
Sub	ImportInvoice()
' Im	portInvoice Macro
' Im	ports invoice.txt, adds a total row. Does some formatting.
' Ke	yboard Shortcut: Ctrl+i
	<pre>Workbooks.OpenText Filename:="C:\Users\Owner\Documents\invoice.txt", Origin _ :=437, StartKow:=1, DataType:=xlDelimited, TextQualifier:=xlDoubleQuote _ , ConsecutiveDelimiter:=False, Tab:=False, Semicolon:=False, Comma:= _ True, Space:=False, Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _ Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), TrailingMinusNumbers</pre>
	:-Irue
	Selection.End(Aldown).Select
	Addig((Add)).coloci
	Selection AutoFill Destination:=Dange("F1:G11") Time:=vFillDefault
	Bange (Filishin) Select
	Rows(11:11) Select
	Selection Fond Bold = True
	Dowe ("11-11") Select
	Selection Font Bold = True
	Selection.Columns.AutoFit
End	Sub

Consider that the following seven lines of recorded code is actually only one line of code that has been broken into seven lines for readability:

```
Workbooks.OpenText Filename:= _
    "C:\invoice.txt", Origin:=437, StartRow:=1, DataType:=xlDelimited, _
    TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
    Tab:=True, Semicolon:=False, Comma:=True, Space:=False, _
    Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), Array(3, 1), _
    Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), _
    TrailingMinusNumbers:=True
```

Counting this as one line, the macro recorder was able to record our 21-step process in 14 lines of code, which is pretty impressive.

Each action you perform in the Excel user interface might equate to one or more lines of recorded code.

Some actions might generate a dozen lines of code.

#### Test Each Macro

It is always a good idea to test macros. To test your new macro, return to the regular Excel interface by pressing Alt+F11. Close Invoice.txt without saving any changes. MacroToImportInvoices.xls is still open.

Press Ctrl+I to run the recorded macro. It should work beautifully if you completed the steps correctly. The data is imported, totals are added, bold formatting is applied, and the columns are made wider. This seems like a perfect solution (see Figure 1.12).

Figure 1.12		A	В	С	D	E	F	G
The macro formate the	1	InvoiceDate	InvoiceNumber	SalesRepNumber	CustomerNumber	ProductRevenue	ServiceRevenue	ProductCost
	2	6/6/2011	123829	S21	C8754	538400	0	299897
data in the sheet.	3	6/6/2011	123830	S45	C4056	588600	0	307563
	4	6/6/2011	123831	S54	C8323	882200	0	521726
	5	6/6/2011	123832	S21	C6026	830900	0	494831
	6	6/6/2011	123833	S45	C3025	673600	0	374953
	7	6/6/2011	123834	S54	C8663	966300	0	528575
	8	6/6/2011	123835	S21	C1508	467100	0	257942
	9	6/6/2011	123836	S45	C7366	658500	10000	308719
	10	6/6/2011	123837	S54	C4533	191700	0	109534
	11	Total				5797300	10000	3203740
	12							

# **Running the Macro on Another Day Produces Undesired Results**

After testing the macro, be sure to save your macro file to use on another day. The next day, after receiving a new Invoice.txt file from the system, you open the macro, press Ctrl+I to run it, and disaster strikes. The data for June 6 happened to have 9 invoices, while the data for the June 7 has 17 invoices. However, the recorded macro blindly added the totals in Row 12 because this was where you put the totals when the macro was recorded (see Figure 1.13).

#### Figure 1.13

The intent of the recorded macro was to add a total at the end of the data. but the recorder made a macro that always adds totals at Row 11.

	A	В	С	D	E	F	G
1	InvoiceDate	InvoiceNumber	SalesRepNumber	CustomerNumber	ProductRevenue	ServiceRevenue	ProductCost
2	6/7/2011	123813	S82	C8754	716100	12000	423986
3	6/7/2011	123814		C4894	224200	0	131243
4	6/7/2011	123815	S43	C7278	277000	0	139208
5	6/7/2011	123816	S54	C6425	746100	15000	350683
6	6/7/2011	123817	S43	C6291	928300	0	488988
7	6/7/2011	123818	S43	C1000	723200	0	383069
8	6/7/2011	123819	S82	C6025	982600	0	544025
9	6/7/2011	123820	S17	C8026	490100	45000	243808
10	6/7/2011	123821	S43	C4244	615800	0	300579
11	Total	123822	S45	C1007	5703400	72000	3005589
12	6/7/2011	123823	S87	C1878	338100	0	165666
13	6/7/2011	123824	S43	C3068	567900	0	265775
14	6/7/2011	123825	S43	C7571	123456	0	55555
15	6/7/2011	123826	S55	C7181	37900	0	19811
16	6/7/2011	123827	S43	C7570	582700	0	292000
17	6/7/2011	123828	S87	C5302	495000	0	241504
18	6/7/2011	123828	S87	C5302	495000	0	241504
19							

This problem arises because the macro recorder is recording all your actions in absolute mode by default. Instead of using the default state of the macro recorder, the next section discusses relative recording and how this might get you closer to a final solution.

#### **Possible Solution: Use Relative References When Recording**

By default, the macro recorder records all actions as *absolute* actions. If you navigate to Row 11 when you record the macro on Monday, the macro will always go to Row 11 when the macro is run. This is rarely appropriate when dealing with variable numbers of rows of data. The better option is to use relative references when recording.

Macros recorded with absolute references note the actual address of the cell pointer, such as A11. Macros recorded with relative references note that the cell pointer should move a certain number of rows and columns from its current position. For example, if the cell pointer starts in cell A1, the code ActiveCell.Offset(16, 1).Select would move the cell pointer to B17, which is the cell 16 rows down and 1 column to the right.

Let's try the same case study again, this time using relative references. The solution will be much closer to working correctly.

# CASE STUDY: RECORDING THE MACRO WITH RELATIVE REFERENCES

Let's try to record the macro again, but this time you will use relative references. Close Invoice.txt without saving changes. In the workbook MacroToImportInvoices.xls, record a new macro by selecting Record Macro from the Developer tab. Give the new macro a name of ImportInvoicesRelative and assign a different shortcut key such as Ctrl+Shift+J (see Figure 1.14).

.14	Record Macro					
ady to record a	Macro name: ImportInvoicesRelative					
	Shortcut key: Ctrl+Shift+ J Store macro in: This Workbook					
	Use relative references for some of the steps of the macro to format the invoice.txt file.] OK Cancel					

As you start to record the macro, go through the process of opening the Invoice.txt file. Before navigating to the last row of data by pressing the End key + then the down-arrow key, click the Use Relative Reference button on the Developer tab (refer to Figure 1.1).

Continue through the actions in the script from the case study:

- 1. Press the End key followed by the down-arrow key to move to the last row of data.
- 2. Press the down arrow one more time to move to the total row.
- 3. Type the word Total.

Figure 1 Getting re second try

- 4. Press the right-arrow key four times to move to Column E of the Total row.
- 5. Click the Autosum button, and then press Ctrl+Enter to add a total to the Product Revenue column while remaining in that cell.
- 6. Click the AutoFill handle and drag from Column E to Column G to copy the total formula to Columns F and G.
- 7. Press Shift+spacebar to select the entire row. Type Ctrl+b to apply bold formatting to it. At this point, you need to move to Cell A1 to apply bold to the headings. You do not want the macro recorder to record the movement from Row 11 to Row 1 because it would record this as moving 10 rows up, which might not be correct tomorrow. Before moving to A1, toggle the Use Relative Recording button off, and then continue recording the rest of the macro.
- 8. Highlight Row 1 and click the Bold icon to set the headings in bold.
- 9. Press Ctrl+A to select all cells.
- **10.** From the Home tab, select Format, AutoFit Column Width.
- **11.** Select cell A1.
- 12. Stop recording.

Press Alt+F11 to go to the VB Editor to review your code. The new macro appears in Module1 below the previous macro.

If you close Excel between recording the first and second macro, Excel inserts a new module called Module2 for the newly recorded macro.

The following code has been edited with two comments that will help you remember where you turned the relative recording on and then off:

```
Sub ImportInvoicesRelative()
' ImportInvoicesRelative Macro
' Use relative references for some of the steps of the macro
 to format the invoice.txt file
   Workbooks.OpenText Filename:=
        "C:\invoice.txt", Origin:=437, StartRow:=1, DataType:=xlDelimited,
       TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False,
       Tab:=True, Semicolon:=False, Comma:=True, Space:=False,
       Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), Array(3, 1),
       Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), _
      TrailingMinusNumbers:=True
' Turned on relative recording here
   Selection.End(xlDown).Select
   ActiveCell.Offset(1, 0).Range("A1").Select
   ActiveCell.FormulaR1C1 = "'Total"
   ActiveCell.Offset(0, 4).Range("A1").Select
   Selection.FormulaR1C1 = "=SUM(R[-16]C:R[-1]C)"
   Selection.AutoFill Destination:=ActiveCell.Range("A1:C1"), Type:= _
       xlFillDefault
```

1

```
' Turned off relative recording here
    ActiveCell.Range("A1:C1").Select
    ActiveCell.Rows("1:1").EntireRow.Select
    ActiveCell.Activate
    Selection.Font.Bold = True
    Rows("1:1").Select
    Selection.Font.Bold = True
    Cells.Select
    Selection.Columns.AutoFit
    Range("A1").Select
End Sub
```

To test the macro, close Invoice.txt without saving, and then run the macro with Ctrl+J. Everything should look good, and you should get the same results.

The next test is to see whether the program works on the next day when you might have more rows. Figure 1.15 shows the data for June 7.



Open MacroToImportInvoices.xls and run the new macro with Ctrl+J. This time, everything should look good with the totals in the correct places. Look at Figure 1.16—see anything out of the ordinary?

Fig	ure	1.	16

The result of running the Relative macro.

		A B		С	D	E	F	G
	1	InvoiceDate	InvoiceNumber	SalesRepNumber	CustomerNumber	ProductRevenue	ServiceRevenue	ProductCos
	2	6/7/2011	123813	S82	C8754	716100	12000	42398
	3	6/7/2011	123814		C4894	224200	0	13124
	4	6/7/2011	123815	S43	C7278	277000	0	13920
	5	6/7/2011	123816	S54	C6425	746100	15000	35068
	6	6/7/2011	123817	S43	C6291	928300	0	48898
	7	6/7/2011	123818	S43	C1000	723200	0	38306
	8	6/7/2011	123819	S82	C6025	982600	0	54402
	9	6/7/2011	123820	S17	C8026	490100	45000	24380
1	10	6/7/2011	123821	S43	C4244	615800	0	30057
1	11	6/7/2011	123822	S45	C1007	271300	0	15325
1	12	6/7/2011	123823	S87	C1878	338100	0	16566
1	13	6/7/2011	123824	S43	C3068	567900	0	26577
1	14	6/7/2011	123825	S43	C7571	123456	0	5555
1	15	6/7/2011	123826	S55	C7181	37900	0	1981
1	16	6/7/2011	123827	S43	C7570	582700	0	29200
1	17	6/7/2011	123828	S87	C5302	495000	0	24150
1	18	6/7/2011	123828	S87	C5302	495000	0	24150
-	19	Total				3527156	0	173564

If you aren't careful, you might print these reports for your manager. If you did, you would be in trouble. When you look in cell E19, Excel has inserted a green triangle to tell you to look at the cell. If you happened to try this back in Excel 95 or Excel 97 before SmartTags, there would not have been an indicator that anything was wrong.

When you move the cell pointer to E19, an alert indicator pops up near the cell. This indicator tells you the formula fails to include adjacent cells. If you look in the formula bar, you will see that the macro totaled only from Row 10 to Row 18. Neither the relative recording nor the nonrelative recording is smart enough to replicate the logic of the AutoSum button.

At this point, some people would give up. However, imagine that you might have had fewer invoice records on this particular day. Excel would have rewarded you with the illogical formula of =SUM(E6:E1048574) and a circular reference, as shown in Figure 1.17.

		E7	<b>-</b> (9	<i>f<sub>sc</sub></i> =SUM(E6:E1	.048574)			
		A	В	С	D	E	F	G
+ h	1	InvoiceDate	InvoiceNumber	SalesRepNumber	CustomerNumber	ProductRevenue	ServiceRevenue	ProductCost
IN	2	6/9/2011	123850		C1654	161000	0	90761
	3	6/9/2011	123851		C6460	275500	10000	146341
	4	6/9/2011	123852		C5143	925400	0	473515
	5	6/9/2011	123853		C7868	148200	0	75700
	6	6/9/2011	123854		C3310	890200	0	468333
	7	Total				0	0	0

Figure 1.17 The result of running the Relative macro w fewer invoice records

If you have tried using the macro recorder, most likely you would run into similar problems as the ones produced in the last two case studies. Although this is frustrating, you should be happy to know that the macro recorder actually gets you 95 percent of the way to a useful macro.

Your job is to recognize where the macro recorder is likely to fail and then to be able to dive into the VBA code to fix the one or two lines that require adjusting to have a perfect macro. With some added human intelligence, you can produce awesome macros to speed up your daily work.

If you are like me, you are cursing Microsoft about now. We have wasted a good deal of time over a couple of days, and neither macro works. What makes it worse is that this sort of procedure would have been handled perfectly by the old Lotus 1-2-3 macro recorder introduced in 1983. Mitch Kapor solved this problem 24 years ago, and Microsoft still can't get it right.

Did you know that up through Excel 97, Microsoft Excel secretly ran Lotus command-line macros? I found this out right after Microsoft quit supporting Excel 97. At that time, a number of companies upgraded to Excel XP, which no longer supported the Lotus 1-2-3 macros. Many of these companies hired us to convert the old Lotus 1-2-3 macros to Excel VBA. It is interesting that from Excel 5, Excel 95, and Excel 97, Microsoft offered an interpreter that could handle the Lotus macros that solved this problem correctly, yet their own macro recorder couldn't (and still can't!) solve the problem.

#### Never Use the AutoSum Button While Recording a Macro

There actually is a macro-recorder solution to the current problem. It is important to recognize that the macro recorder will never correctly record the intent of the AutoSum button.

If you are in cell E99 and click the AutoSum button, Excel starts scanning from cell E98 upward until it locates a text cell, a blank cell, or a formula. It then proposes a formula that sums everything between the current cell and the found cell.

However, the macro recorder records the particular result of that search on the day that the macro was recorded. Rather than record something along the lines of "do the normal AutoSum logic," the macro recorder inserts a single line of code to add up the previous 98 cells.

The somewhat bizarre workaround is to type a SUM function that uses a mix of relative and absolute row references. If you type =SUM(E\$2:E10) while the macro recorder is running, Excel correctly adds code that will always sum from a fixed row two down to the relative reference that is just above the current cell.

Here is the resulting code with a few comments:

```
Sub FormatInvoice3()
' FormatInvoice2 Macro
' Third try. Use relative. Don't touch AutoSum
' Keyboard Shortcut: Ctrl+Shift+K
   Workbooks.OpenText Filename:="C:\Users\Owner\Documents\invoice.txt", Ori-
gin
        :=437, StartRow:=1, DataType:=xlDelimited,
TextQualifier:=xlDoubleQuote
        , ConsecutiveDelimiter:=False, Tab:=False, Semicolon:=False, Comma:=
        True, Space:=False, Other:=False, FieldInfo:=Array(Array(1, 3), Ar-
ray(2, 1),
       Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)),
TrailingMinusNumbers _
        :=True
    ' Relative turned on here
    Selection.End(xlDown).Select
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Total"
    ActiveCell.Offset(0, 4).Range("A1").Select
    ' Don't use AutoSum. Type this formula:
    Selection.FormulaR1C1 = "=SUM(R2C:R[-1]C)"
    Selection.AutoFill Destination:=ActiveCell.Range("A1:C1"), Type:= _
        xlFillDefault
    ActiveCell.Range("A1:C1").Select
     Relative turned off here
    ActiveCell.Rows("1:1").EntireRow.Select
    ActiveCell.Activate
```

```
Selection.Font.Bold = True
Cells.Select
Selection.Columns.AutoFit
Range("A1").Select
End Sub
```

This third macro will consistently work with any size dataset.

To see a demo of recording this macro, search for Excel VBA 1 at YouTube.

# **Three Tips When Using the Macro Recorder**

П

You will rarely be able to record 100 percent of your macros and have them work. However, you will get much closer by using these three tips demonstrated in the following subsections.

#### Tip 1: Use Relative References Setting Usually Needs to Be On

Microsoft should have made this setting be the default. Unless you specifically need to move to Row 1 from the bottom of a dataset, you should usually leave the Use Relative References button in the Developer tab turned on.

#### Tip 2: Use Special Navigation Keys to Move to Bottom of a Dataset

If you are at the top of a dataset and need to move to the last cell with data, you can press Ctrl+down arrow or press the End key and then the down-arrow key.

Similarly, to move to the last column in the current row of the dataset, press Ctrl+right arrow or press End and then press the right-arrow key.

By using these navigation keys, you can jump to the end of the dataset, no matter how many rows or columns you have today.

#### Tip 3: Never Touch the AutoSum Icon While Recording a Macro

The macro recorder will not record the "essence" of the AutoSum button. Instead, it will hard-code the formula that resulted from pressing the AutoSum button. This formula does not work any time you have more or fewer records in the dataset.

Instead, type a formula with a single dollar sign, such as =SUM(E\$2:E10). When this is done, the macro recorder records the first E\$2 as a fixed reference and starts the SUM range directly below the Row 1 headings. Provided the active cell is E11, the macro recorder recognizes E10 as a relative reference pointing directly above the current cell.

# **Next Steps**

Chapter 2, "This Sounds Like BASIC, So Why Doesn't It Look Familiar?" examines the three macros you recorded in this chapter to make more sense out of them. After you know how to decode the VBA code, it will feel natural to either correct the recorded code or simply write code from scratch. Hang on through one more chapter. You'll soon learn that VBA is the solution, and you'll be writing useful code that works consistently.

This page intentionally left blank

# Index

# Symbols

3-D format, changing, 230-2343-D rotation settings, 224-22932-bit API declarations, changing to 64-bit, 538

64-bit API declarations, changing 32-bit declarations to, 538

#### A

A1-style references, 127-128 About dialog, customizing, 541 AboutMrExcel() procedure, 541 above/below average cells, formatting, 383 absolute mode, 25 absolute references, 133 accelerator keys, displaying, 529 Access databases. See databases Activate event, 187 active control, coloring, 530-532 ActiveFilters property, 289 ActiveX controls attaching macros to, 583-584 right-click menu for, 360-362 ActiveX data objects. See ADO Add method, 148-149, 442 Add3ColorScale() procedure, 375 AddAboveAverage method, 383 AddChart method, 203 AddControl event, 187, 195, 199 AddCrazyIcons() procedure, 382

# AddGlowToTitle() procedure, 223 add-ins

Add-Ins dialog, 588 characteristics of, 587-588 closing, 593 converting workbooks to, 588-590 hidden workbooks as alternative to add-ins, 593-594 installing, 591 removing, 593 security, 592 Add-Ins dialog, 588 Addition procedure, 81 AddTransfer() procedure, 480-481 AddTwoDataBars() procedure, 381 ADO (ActiveX data objects) compared to DOA (data access objects), 477 connections, 478 cursors, 478 fields adding on-the-fly, 489-490 checking existence of, 488 lock type, 479 overview, 478-480 records adding, 480-481 deleting, 485 retrieving, 481-483 summarizing, 485-486 updating, 483-485 recordsets, 478 tables adding on-the-fly, 489 checking existence of, 487-488 ADOAddField() procedure, 489-490 ADOCreateReplenish() procedure, 489

#### ADOWipeOutAttribute() procedure, 485 Advanced Filter building with Excel interface, 258 case study: creating reports for each customer, 280-283 criteria ranges case study, 268 explained, 265-266 formula-based conditions, 268-275 logical AND criteria, 267 logical OR criteria, 267 extracting unique list of values, 258-264 getting unique combinations of two or more fields, 263-264 with user interface, 259 with VBA code, 260-263 Filter in Place, 275-276, 283-285 overview, 257 xlFilterCopy with all records, 276-280 copying all columns, 277 copying a subset of columns and reordering, 278-280 AdvancedFilter method, 260 AfterUpdate event, 190, 193-197 ahtAddFilterItem API function, 546 aht apiGetOpenFileName API function, 544-546 aht apiGetSaveFileName API function, 544-546 AllColumnsOneCustomer() procedure, 277 AllowMultipleFilters property, 289 **API declarations** 32-bit versus 64-bit, 538 ahtAddFilterItem, 546 aht\_apiGetOpenFileName, 544-546 aht\_apiGetSaveFileName, 544-546

calling, 537 DisplaySize, 540 explained, 535-536 finding, 547 FindWindow, 541-543 GetComputerName, 538-539 GetSystemMenu, 541-542 KillTimer, 542-543 10pen, 539 PlayWavSound, 543 SetTimer, 542-543 ShellAbout, 541 AppEvent\_AfterCalculate() event, 176 AppEvent\_NewWorkbook() event, 177 AppEvent ProtectedViewWindowActivate() event, 177 AppEvent ProtectedViewWindowBeforeClose() event, 177 AppEvent ProtectedViewWindowDeactivate() event, 177 AppEvent ProtectedViewWindowOpen() event, 177 AppEvent\_ProtectedViewWindowResize() event, 177 AppEvent\_SheetActivate() event, 177 AppEvent\_SheetBeforeDoubleClick() event, 178 AppEvent\_SheetBeforeRightClick() event, 178 AppEvent\_SheetCalculate() event, 178 AppEvent\_SheetChange() event, 178 AppEvent\_SheetDeactivate() event, 178 AppEvent\_SheetFollowHyperlink() event, 178 AppEvent\_SheetPivotTableUpdate() event, 178

AppEvent\_SheetSelectionChange() event, 178 AppEvent WindowActivate() event, 179 AppEvent\_WindowDeactivate() event, 179 AppEvent WindowResize() event, 179 AppEvent WorkbookActivate() event, 179 AppEvent\_WorkbookAddinInstall() event, 179 AppEvent\_WorkbookAddinUninstall() event, 179 AppEvent\_WorkbookAfterXmlExport() event, 181 AppEvent\_WorkbookAfterXmlImport() event, 181 AppEvent WorkbookBeforeClose() event, 179 AppEvent\_WorkbookBeforePrint() event, 180 AppEvent\_WorkbookBeforeSave() event, 180 AppEvent WorkbookBeforeXmlExport() event, 181 AppEvent\_WorkbookBeforeXmlImport() event, 181 AppEvent\_WorkbookNewSheet() event, 180 AppEvent WorkbookOpen() event, 180 AppEvent WorkbookPivotTableCloseConnection() event, 180 AppEvent WorkbookPivotTableOpenConnection() event, 180 AppEvent WorkbookRowsetComplete() event, 181 AppEvent WorkbookSync() event, 181 application-level events, 176-181 trapping, 494-495

Application.OnTime, 399-400 scheduling macros to run every two minutes, 403-404 macros to run x minutes in the future, 401-402 scheduled procedures with ready mode, 400 verbal reminders, 402 specifying a window of time for updates, 400 applications checking version of, 144-145 compatibility issues Compatibility mode, 145 explained, 144 historical stock/fund quotes, 362-363 ApplyLayout method, 203 ApplyTexture() procedure, 220 ApplyThemeColor() procedure, 220 Areas collection, 77 arrays advantages of, 457-458 array formulas, 137-138 declaring, 453-454 defined, 453 dynamic arrays, 459-460 emptying, 456-457 filling, 455-456 multidimensional arrays, 454 names, 153-154 one-dimensional arrays, 454 passing, 460 art, SmartArt, 142-144 Assign3DPreset() procedure, 224 AssignBevel() procedure, 230 asterisks (\*), 356-358

asymmetric pivot tables, named sets for, 322-323 attaching macros to ActiveX controls, 583-584 to command buttons, 581-582 to shapes, 582-583 The Attribute "Attribute Name" on the Element "customui Ribbon" Is Not Defined in the DTD/Schema (error message), 577 AutoFilter filtering by color, 253 filtering by icon, 254 replacing loops with, 249-251 selecting dynamic data range with, 254-255 selecting multiple items, 252 selecting visible cells only, 255-256 selecting with Search box, 252-255 turning off drop-downs in, 285 AutoFilterCustom() procedure, 285 automation (Word) bookmarks, 448-449 constant values, 439-441 controlling form fields, 450-452 creating and referencing objects, 437-439 Document object, 442-443 early binding, 433-436 explained, 433 late binding, 436-437 macro recorder, 441 Range object, 444-447 Selection object, 443-444 AutoSort, 308 AutoSum button, 30-31

#### В

BASIC, 8 BeforeDragOver event, 187, 190, 193-199 BeforeDropOrPaste event, 187, 190, 193-199 BeforeUpdate event, 190, 193-197 below/above average cells, formatting, 383 bevel format, changing, 230-234 binding early binding, 433-436 late binding, 436-437 bins, creating for frequency charts, 236-239 blank cells eliminating from pivot tables, 308 formatting cells that contain blanks or errors, 387 bookmarks, 448-449 BookOpen() function, 83 Bottom 5 cells, formatting, 383-384 breakpoints, 49, 55 btnClose\_Click() procedure, 512 BubbleSort() procedure, 98 built-in chart types, 208-210 buttons. See also specific buttons attaching macros to, 581-582 custom icon images, 574-575 help buttons, 505-506 Microsoft Office icons, 573-574

#### C

.Calculation options, 306-307 calculations calculated data fields, 324-325 calculated items, 325

changing to show percentages, 305-308 elapsed time, 353-354 calling API declarations, 537 userforms, 186 Can't find object or library (error message), 435-436 case of text, changing, 359-360 Case statements, 124 case studies cleaning up recorded code, 62-64 converting Excel 2003 custom toolbar to Excel 2010, 575-577 criteria ranges, 268 custom functions, 80 data visualization, 327 entering A1 versus R1C1 references, 131 entering military time into cell, 171 filtering to top five or top 10, 319 formula-based conditions, 270 Go To Special instead of looping, 256-257 help buttons, 505-506 hidden workbook to hold macros and forms, 594 looping through directory files, 119-120 multicolumn list boxes, 532 named ranges for VLOOKUP, 156-157 page setup errors, 555 password cracking, 560 recording macros, 22-23 relative references, 26-28

cells A1-style references, 127-128 blank cells, eliminating from pivot tables, 308 checking for empty cells, 73-74 comments charts in, 341-342 listing, 337-339 resizing, 339-341 conditional formatting. See conditional formatting entering military time into, 171 noncontiguous cells, selecting/ deselecting, 347-349 progress indicators, creating, 355-356 R1C1-style references absolute references, 133 array formulas, 137-138 case study: entering A1 versus R1C1 references, 131 explained, 127-128 formulas, 129-132 mixed references, 133 multiplication table example, 134-135 referring to entire columns/rows, 134 relative references, 132-133 remembering column numbers associated with column letters, 136 switching to, 128 returning column letter of cell address, 103 reversing contents of, 101 selected cells, highlighting, 342-344, 344-345 selecting with SpecialCells, 360

setting workbook name in, 82 summing based on interior color, 89-90 Cells(), 59 Cells property as parameters in Range property, 69 selecting ranges with, 68-69 centering cell comments, 340-341 Change event, 190, 193-199 ChangeFormat() procedure, 446 ChangeStyle() procedure, 447 ChangeTheChartLater() procedure, 207 changing range size, 71-72 text case, 359-360 Chart\_Activate() event, 173 Chart BeforeDoubleClick() event, 173 Chart\_BeforeRightClick() event, 173 Chart\_Calculate() event, 173 Chart\_Deactivate() event, 173 Chart DragOver() event, 175 Chart\_DragPlot() event, 175 chart events, 166-167, 172-175, 495-497 ChartFormat method, 203 ChartFormat object, 218 Chart Layout gallery, 211-213 Chart\_MouseDown() event, 174 Chart\_MouseMove() event, 174 Chart\_MouseUp() event, 174 Chart\_Resize() event, 174 charts built-in chart types, 208-210 in cell comments, 341-342 chart events, 166-167, 172-175, 495-497 trapping, 495-497 creating, 204-207

dynamic charts, creating in userforms, 244-245 embedded charts, 172 exporting as graphics, 244-245 formatting 3-D rotation settings, 224-229 bevel and 3D format, 230-234 chart elements to which formatting applies, 218-234 Format method, 218-234 glow settings, 222-223 line settings, 222 object fill, 219-222 reflection settings, 223 shadow settings, 223 soft edges, 223-224 frequency charts, 236-239 Layout tab, 213-218 layouts, 211-213 new features (Excel 2010), 139-140 Open-High-Low-Close (OHLC) charts, 235-236 overview, 203 pivot charts, 246-247 referencing, 203-207 SetElement method, 213-218 sparklines. See sparklines specifying size and location of, 204-205 stacked area charts, 239-243 styles, 211-213 template chart types, 210-211 Win/Loss charts, 426-427 Chart\_Select() event, 174-175 Chart\_SeriesChange() event, 175 ChartStyle property, 213 ChartType property, 208

CheckBox control, 512-513 check boxes, 512-513 CheckDisplayRes() procedure, 540 CheckForSheet() procedure, 84 checking existence of names, 155-156 for open files, 539 whether workbook is open, 83 CheckUserRights() procedure, 86 class modules creating collections in, 502-504 inserting, 493 cleaning up recorded code case study, 62-64 tips for, 58-61 ClearAllFilters method, 289 ClearTable method, 289 Click event, 187, 190, 193-196, 200 clients, training about error handling, 557 Close method, 443 closing add-ins, 593 documents, 443 Excel, 401 userform windows, 200-201 code protection, 559 collections Areas, 77 creating in class modules, 502-504 in standard module, 501-502 defined, 501 explained, 35 grouping controls into, 519-521 ColName() function, 103

color color scales adding to ranges, 374-375 explained, 367 coloring active control, 530-532 filtering by, 253 RGB colors in sparklines, 421-423 summing cells based on interior color, 89-90 theme colors for sparklines, 418-421 using two colors of data bars in range, 380-382 ColorFord() procedure, 251 ColorFruitRedBold() procedure, 121-122 ColumnExists() procedure, 488 ColumnHeaders() procedure, 455 columns copying all columns, 277 remembering column numbers associated with column letters, 136 subset of columns, copying, 278-280 Columns property, 72 combining worksheets into workbooks, 334-335 combo boxes, 191-193 command buttons attaching macros to, 581-582 events for, 189 CommandButton event, 191 comments adding to names, 150 in cells charts in, 341-342 listing, 337-339 resizing, 339-341 compact layout, 293-294 CompactLayoutColumnHeader property, 289

CompactLayoutRowHeader property, 289 CompactRowIndent property, 290 compatibility issues checking application version with Version property, 144-145 Compatibility mode, 145 explained, 144 Compatibility mode, 145 complex expressions, 124 ComplexIf() procedure, 124-126 computer names, retrieving, 538-539 concatenation, 97-98 conceptual filters (pivot tables), 313-316 conditional formatting color scales adding to ranges, 374-375 explained, 367 data bars adding to ranges, 369-374 explained, 367 determining which cells to format, 387-388 formatting cells based on value, 385 formatting cells in top 10 or bottom 5, 383-384 formatting cells that are above/below average, 383 formatting cells that contain blanks or errors. 387 formatting cells that contain dates, 386 formatting cells that contain text, 386 formatting unique or duplicate cells, 384-385 highlighting selected cell, 342-344 icon sets adding to ranges, 375-378 explained, 368

new features (Excel 2010), 140-141 NumberFormat property, 388-389 VBA methods and properties, 368-369 conditions (If statement), 121 configuring pivot tables, 295-296 connections (ADO), 478 constant values defined constants, 41-45 explained, 439 retrieving with Object Browser, 440-441 retrieving with Watch window, 440 ContainsText() function, 100-101 content management system, 407-409 controls. See also userforms active control, coloring, 530-532 ActiveX controls attaching macros to, 583-584 right-click menu for, 360-362 adding at runtime, 523-529 adding on-the-fly, 525 CheckBox, 512-513 grouping into collections, 519-521 programming, 188 RefEdit. 515 renaming, 188 Ribbon control arguments, 569-571 Ribbon control attributes, 566 running macros from, 16-17 ScrollBar, 517-519 TabStrip, 513-515 tip text, adding to userforms, 530 ToggleButton, 517 troubleshooting, 189

converting Excel 2003 custom toolbar to Excel 2010, 575-577 pivot tables to values, 299-301 week numbers into dates, 96 workbooks to add-ins, 588-590 ConvertToFormulas method, 289 CopyFromRecordSet method, 481 copying data into worksheets, 335-336 formulas, 129-130 macros into workbooks, 363-365 ranges, 61 subset of columns, 278-280 CopyToNewFolder() procedure, 120 counting records, 303 unique values, 90-91 workbooks in directory, 84-85 CountMyWkbks() procedure, 85 cracking passwords, 560 CreatedStackedChart() procedure, 242-243 CreateFrequencyChart() procedure, 238-239 CreateMemo() procedure, 448-449 CreateObject() function, 438 CreateOHCLChart() procedure, 236 CreatePivot() procedure, 298-299 CreatePivotTable method, 295 CreateSummaryReportUsingPivot() procedure, 246-247, 300-301 criteria ranges case study, 268 explained, 265-266 formula-based conditions, 268-275

logical AND criteria, 267 logical OR criteria, 267 Criteria reserved name, 155 CSV files, importing, 331-332 CurrentRegion property, 74-76 cursors, 478 custom About dialog, 541 custom functions. See UDFs (user-defined functions) Custom UI Editor, 572 CustomerByProductReport() procedure, 309-312 customizina data transposition, 345-347 icon images, 574-575 objects creating custom objects, 497-498 Property Let/Property Get procedures, 499-501 referencing, 498-499 Ribbon to run macros control arguments, 569-571 control attributes, 566 custom icon images, 574-575 Custom UI Editor tool, 572 customui folder, 564-565 error messages, 577-580 explained, 563-565 file structure, accessing, 571 Microsoft Office icons, 573-574 RELS file, 571-572 tab and group, 565-566 sort orders, 354-355 web pages, 406 customui folder, 564-565

#### D

dashboards creating, 427-432 sparklines creating, 412-413 creating 100's of individual sparklines in a dashboard, 428-432 formatting, 418-421 observations about, 428 scaling, 414-418 types of sparklines, 411 data getting from the Web, 391-392 publishing to web pages, 404-406 data access objects (DAO), 477 data bars adding to ranges, 369-374 explained, 367 using two colors of data bars in range, 380-382 data transposition, customizing, 345-347 data visualizations applying, 327 color scales, adding to ranges, 374-375 conditional formatting determining which cells to format, 387-388 formatting cells based on value, 385 formatting cells in top 10 or bottom 5, 383-384 formatting cells that are above/ below average, 383 formatting cells that contain blanks or errors, 387 formatting cells that contain dates, 386

formatting cells that contain text, 386 formatting unique or duplicate cells, 384-385 NumberFormat property, 388-389 data bars adding to ranges, 369-374 using two colors of data bars in range, 380-382 explained, 368 icon sets adding to ranges, 375-378 creating for subset of range, 378-380 VBA methods and properties for, 368-369 DataBar2() procedure, 372-373 DataBar3() procedure, 373 Database reserved name, 155 databases ADO connections, 478 cursors, 478 lock type, 479 overview, 478-480 recordsets, 478 fields adding on-the-fly, 489-490 checking existence of, 488 Multidimensional Database (MDB) format, 475 records adding, 480-481 deleting, 485 retrieving, 481-483 summarizing, 485-486 updating, 483-485

shared access databases, creating, 477-478 SOL Server, 490-491 tables adding on-the-fly, 489 checking existence of, 487-488 DataExtract() procedure, 490-491 DataSets variable, 473 dates converting week numbers into, 96 formatting cells that contain dates, 386 grouping to months, quarters, or years, 303-305 retrieving permanent date/time, 87 retrieving saved date/time, 86-87 DateTime() function, 87 DblClick event, 187, 190, 193-197, 200 Deactivate event, 187 Debug button, 551 Debug errors, 551-552 debugging tools breakpoints, 49 jumping forward/backward in code, 49-50 querying variable values, 50-54 Run to Cursor, 50 stepping through code, 46-48 watches, 55 declaring arrays, 453-454 variables, 20 defined constants, 41-45 defining pivot cache, 295 ranges, 444-446 Delete method, 149-150

DeleteFord() procedure, 251 deleting names, 149-150 records, 485 selections from recorded code, 58 delimited files, opening, 467-470 delimited strings, separating, 96-97 deselecting noncontiguous cells, 347-349 Design tab, changing layout from, 325-326 Developer tab, viewing, 9-10 directories counting workbooks in, 84-85 listing files in, 329-331 looping through directory files, 119-120 **Disable All Macros Except Digitally Signed** Macros setting, 12 Disable All Macros with Notification settina, 11-12 **Disable All Macros Without Notification** setting, 11 disabling X button for closing userforms, 541-542 **DisplayAllMember method**, 289 DisplayContextTooltips property, 290 DisplayFieldCaptions property, 290 displaying R1C1-style references, 128 DisplayMemberPropertyTooltips property, 290 display-resolution information, retrieving, 540 **DisplaySize API function, 540** dll (dynamic link libraries), 535 Do loops explained, 113-115 Until clause, 115-117 While clause, 115-117

DOA (data access objects), 477 Document object closing documents, 443 explained, 442 opening documents, 442 printing documents, 443 saving documents, 442-443 documents closing, 443 creating, 442 exporting to, 336-337 opening, 442 printing, 443 saving, 442-443 drilling down pivot tables, 349-350 DropButtonClick event, 190, 193 duplicate cells, formatting, 384-385 duplicates, removing from ranges, 91-92 dynamic arrays, 459-460 dynamic charts, creating in userforms, 244-245 dynamic data ranges, selecting with AutoFilter, 254-255 dynamic link libraries (dll), 535 DynamicAutoFilter() procedure, 255

# Е

early binding, 433-436 elapsed time, calculating, 353-354 Element "customui Tag Name" Is Unexpected (error message), 578 e-mail addresses, validating, 88-89 embedded chart events, trapping, 495-497 embedded charts, 172 EmpAddCollection() procedure, 504 EmpPayCollection() procedure, 501-502 empty cells, checking for, 73-74 emptying arrays, 456-457 **Enable All Macros (Not Recommended:** Potentially Dangerous Code Can Run) setting, 12 enable/disable macro settings, 11-12 enabling events, 161 macros, 12 encountering errors on purpose, 556 EndKey method, 443 Enter event, 190, 193-197, 200 Err object, 554 Error event, 187, 190, 193-197, 200 error handling debug errors inside userform code, 551-552 encountering errors on purpose, 556 Err object, 554 error messages The Attribute "Attribute Name" on the Element "customui Ribbon" Is Not Defined in the DTD/Schema, 577 Can't find object or library, 435-436 Element "customui Tag Name" Is Unexpected, 578 Excel Found Unreadable Content, 579 Illegal Qualified Name Character, 578 runtime error 9: Subscript Out of Range, 557 runtime error 1004: Method Range of Object Global Failed, 558-559 Wrong Number of Arguments or Invalid Property Assignment, 580

errors caused by different versions, 561 errors while developing versus errors months later, 557 explained, 549-552 formatting cells that contain blanks or errors, 387 generic error handlers, 554 ignoring errors, 554 On Error GoTo syntax, 552-554 On Error Resume Next statement, 554-555 page setup errors, 555 problems with passwords, 560-561 protecting code, 559 suppressing Excel warnings, 556 training clients, 557 error messages The Attribute "Attribute Name" on the Element "customui Ribbon" Is Not Defined in the DTD/Schema, 577 Can't find object or library, 435-436 Element "customui Tag Name" Is Unexpected, 578 Excel Found Unreadable Content, 579 Illegal Qualified Name Character, 578 runtime error 9: Subscript Out of Range, 557 runtime error 1004: Method Range of Object Global Failed, 558-559 Wrong Number of Arguments or Invalid Property Assignment, 580 Evaluate method, 153 events. See also specific events application-level events, 176-181, 494-495 chart events, 172-175

CheckBox control events, 513 for combo boxes, 191-193 for command buttons, 189 embedded chart events, trapping, 495-497 enabling, 161 explained, 160 for graphics, 195-202 for labels, 189 levels of events, 159-160 for list boxes, 191-193 for MultiPage control, 198-200 for option buttons, 194-195 parameters, 160 RefEdit control events, 516 Scrollbar control events, 519 for spin buttons, 196-202 TabStrip control events, 515 for text boxes, 189 ToggleButton control events, 517 userform events, 186-187 workbook events, 161-167 worksheet events, 168-172 EveryOtherRow() procedure, 455 Excel 97-2003 Workbook file type, 18 Excel 2003 custom toolbar, converting to Excel 2010, 575-577 Excel 2007 pivot table features, 288-290 Excel 2010 file types, 18-19 pivot table features, 288 Excel Binary Workbook file type, 18 **Excel Found Unreadable Content (error** message), 579 Excel Macro-Enabled Workbook file type, 18 Excel Workbook file type, 18

Excel8CompatibilityMode property, 145 Execute method, 485 Exit event, 191-197, 200 exiting For...Next loop after condition is met, 111-112 ExportChart() procedure, 244 exporting charts as graphics, 244-245 to Word document, 336-337 expressions in Case statements, 124 Extract reserved name, 155

### F

FieldListSortAscending property, 290 fields adding on-the-fly, 489-490 adding to pivot tables, 296-299 calculated data fields, 324-325 checking existence of, 488 field entry in userforms, verifying, 200 form fields, controlling in Word, 450-452 multiple value fields (pivot tables), 302-303 File menu, Save As command, 589 files checking for open files, 539 CSV files, importing, 331-332 file structure, accessing, 571 file types in Excel 2010, 18-19 filenames, retrieving, 201-202 listing, 329-331 looping through directory files, 119-120 paths, retrieving, 543-546 RELS file, 571-572

text files fixed-width files, 463-467 importing files with fewer than 1,048,576 rows, 463-470 importing files with more than 1,048,576 rows, 470-473 reading and parsing, 332-333 writing, 473-474 filling arrays, 455-456 FillOutWordForm() procedure, 451-452 Filter in Place, 275-276, 283-285 FilterByFontColor() procedure, 253 FilterBylcon() procedure, 254 filtering data into worksheets, 335-336 pivot tables conceptual filters, 313-316 filtering to top five or top 10, 319 manual filters, 312-313 with named sets, 321-323 Search filter, 316-317 slicers, 319-321 FilterNoFontColor() procedure, 253 filters Advanced Filter building with Excel interface, 258 case study: creating reports for each customer, 280-283 extracting unique list of values, 258-264 Filter in Place, 275-276 overview, 257 xlFilterCopy with all records, 276-280 AutoFilter filtering by color, 253 filtering by icon, 254 replacing loops with, 249-251

selecting dynamic data range with, 254-255 selecting multiple items, 252 selecting visible cells only, 255-256 selecting with Search box, 252-255 turning off drop-downs in, 285 finding API declarations, 547 first nonzero-length cell, 93 FindJPGFilesInAFolder() procedure, 119-120 FindWindow API function, 541-543 first nonzero-length cell, finding in range, 93 FirstNonZeroLength() function, 93 fixed-width files, opening, 463-467 flow control complex expressions in Case statements, 124 If statement conditions, 115, 121-124 If...Then...Else, 121 If...Then...Else...End If, 122-123 If...Then...End If, 121-122 nested If statements, 124-126 Select Case...End Select statement, 123 folders, customui, 564-565 For...Next loops exiting early after condition is met, 111-112 explained, 107-109 nesting, 112 Step clause, 110-111 variables, 110 Format method, 218-234 Format Shape dialog, 230 Format tab. See formatting

FormatAboveAverage() procedure, 383 FormatBelowAverage() procedure, 383 FormatBetween10And20() procedure, 386 FormatBorder() method, 222 FormatBottom5Items() procedure, 383-384 FormatConditions object, 368 FormatContainsA() procedure, 386 FormatDatesLastWeek() procedure, 386 FormatDuplicate() procedure, 385 FormatLessThan15() procedure, 386 FormatLineOrBorders() procedure, 222 FormatShadow() procedure, 223 FormatSoftEdgesWithLoop() procedure, 224

#### formatting

charts 3-D rotation settings, 224-229 bevel and 3D format, 230-234 chart elements to which formatting applies, 218-234 Format method, 218-234 glow settings, 222-223 line settings, 222 object fill, 219-222 reflection settings, 223 shadow settings, 223 soft edges, 223-224 conditional. See conditional formatting ranges, 446-447 sparklines RGB colors, 421-423 sparkline elements, 423-426 theme colors, 418-421 Win/Loss charts, 426-427 FormatTop10Items() procedure, 383

FormatTop12Percent() procedure, 384 FormatUnique() procedure, 385 FormatWithPicture() procedure, 221 forms. See userforms formulas array formulas, 137-138 determining which cells to format, 387-388 entering once and copying down the column, 129-130 formula-based conditions, 268-275 names, 151 R1C1 formulas, 61 frequency charts, 236-239 FruitRedVegGreen() procedure, 122 FTP, 409-410 functions. See specific functions

#### G

generic error handlers, 554 GetAddress() function, 102-103 GetComputerName API function, 538-539 GetFileName() function, 546 GetObject() function, 438-439 GetSettings() procedure, 558 GetSystemMenu API function, 541-542 GetUniqueCustomers() procedure, 260 GetUnsentTransfers() procedure, 481-482 alobal names, 147-148 glow settings, 222-223 Go To Special dialog, 256-257 graphics. See also icons adding on-the-fly, 526-527 events for, 195-202 exporting charts as, 244-245 SmartArt, 142-144 groups, creating for Ribbon, 565-566

#### Н

HandleAnError() procedure, 553 handling errors. See error handling hard-coding, 60-61 help adding to userforms, 529-532 accelerator keys, 529 coloring active control, 530-532 control tip text, 530 help buttons, 505-506 help files, 143 installing, 37-38 selecting libraries in, 45 help topics, 39 hidden workbooks as alternative to add-ins, 593-594 case study: hidden workbook to hold macros and forms, 594 Hide method, 186 hidina hidden workbooks as alternative to add-ins, 593-594 case study: hidden workbook to hold macros and forms, 594 names, 155 userforms, 186 HighlightFirstUnique() procedure, 385-388 highlighting selected cells, 342-345 HighlightWholeRow() procedure, 388 historical stock/fund quotes application, 362-363 HomeKey method, 443 hovering, 53 hyperlink addresses, returning, 102-103 hyperlinks in userforms, 522 running macros from, 584

#### 

icons custom icon images, 574-575 filtering by, 254 icon sets adding to ranges, 375-378 creating for subset of range, 378-380 explained, 368 Microsoft Office icons, 573-574 If statements conditions, 121 If...Then...Else, 121 If...Then...Else...End If, 122-123 If...Then...End If, 121-122 nesting, 124-126 ignoring errors, 554 Illegal Qualified Name Character (error message), 578 images. See graphics; icons Immediate window, 50-53 Import10() procedure, 470 ImportAll() procedure, 470-471 ImportData function, 156-157 importing CSV files. 331-332 text files files with fewer than 1,048,576 rows, 463-470 files with more than 1,048,576 rows, 470-473 IncrementRotationHorizontal property, 229 IncrementRotationVertical property, 229 IncrementRotationX property, 229 IncrementRotationY property, 229 IncrementRotationZ property, 229

InGridDropZones property, 290 Initialize event, 187 input boxes, 183-184 InputBox function, 183-184 inserting class modules, 493 InsertText() procedure, 444 installing add-ins, 591 help files, 37-38 Intersect method, 73 IsEmailValid() function, 88-89 ISEMPTY function, 73-74 IsWordOpen() procedure, 438

#### J

jet engine, 476 joining multiple ranges, 72-73 jumping forward/backward in code, 49-50

#### Κ

keyboard shortcuts, running macros with, 580-581 KeyDown event, 187, 191-197, 200 KeyPress event, 187, 191-197, 200 KeyUp event, 187, 191-200 keywords, New, 437 KillTimer API function, 542-543

#### L

Label event, 191, 194-195 labels, 189 last row, determining, 59-60 LastSaved() function, 86-87 late binding, 436-437 Layout event, 187, 195 Layout tab, 213-218 LavoutRowDefault property, 290 layouts charts, 211-213 compact lavout, 293-294 pivot table layout, 325-327 Ibl Email Click() procedure, 522 Ibl\_SelectAll\_Click() procedure, 520 lbl\_unSelectAll\_Click() procedure, 520 Ibl Website Click() procedure, 522 learning curve for VBA, 8 levels of events, 159-160 libraries dynamic link libraries (dll), 535 selecting in help files, 45 lighting, VBA constants for, 233-234 Line Input method, 472 line settings, 222 LineFormat object, 222 list boxes combo boxes versus, 191-193 multicolumn list boxes, 532 listing cell comments, 337-339 files in directories, 329-331 lists, sorting, 354-355 Load method, 186 local names, 147-148 location of charts, specifying, 204-205 lock type (ADO), 479 logical AND criteria, 267 logical OR criteria, 267 loops Do explained, 113-115 Until clause, 115-117 While clause, 115-117 For Each, 117-119
For...Next exiting early after condition is met, 111-112 explained, 107-109 nesting, 112 Step clause, 110-111 variables, 110 Go To Special instead of looping, 256-257 looping through directory files, 119-120 replacing with AutoFilter, 249-251 While...Wend, 117 **IOpen API function, 539** Lotus 1-2-3 macros, 29

### Μ

macro recorder, 441 cleaning up recorded code case study, 62-64 tips for, 58-61 examining code from, 39-46 flaws in, 7-8, 21-31 absolute mode, 25 AutoSum button, 30-31 examining code in Programming window, 23-25 recording macros case study, 22-23 relative references, 26-29 relative references case study, 26-28 tips for, 31 Macro Security icon (Developer tab), 9 macros. See also specific procedures attaching to ActiveX controls, 583-584 to command buttons, 581-582 to shapes, 582-583

canceling previously scheduled, 400-401 closing, 401 copying into workbooks, 363-365 holding in hidden workbooks, 594 recording, 12-14, 22-23 running, 14-17 from form controls, 16-17 from hyperlinks, 584 with keyboard shortcuts, 580-581 from Quick Access toolbar, 15-16 from Ribbon. See Ribbon scheduling to run every two minutes, 403-404 to run x minutes in the future, 401-402 security, 10-12 Disable All Macros with Notification setting, 12 enable/disable settings, 11-12 trusted locations, 10-11 testing, 25 Macros icon (Developer tab), 9 manual filters (pivot tables), 312-313 manually creating web queries, 392-395 material types, 232 maximum values in range, returning addresses of, 101-102 MaxPoint property, 371 MDB (Multidimensional Database) format, 475 Me keyword, 186 message boxes, 184 methods. See specific methods Microsoft Office icons, adding to buttons, 573-574 military time, entering into cells, 171

MinPoint property, 371 mixed references, 133 mixed text retrieving numbers from, 95 sorting numeric and alpha characters, 99-100 modeless userforms, 521 Modify method, 371 modules, 21 MouseDown event, 187, 191-196, 200 MouseMove event, 187, 191, 194-196, 200 MouseUp event, 187, 191, 194-196, 200 MoveAfterTheFact() procedure, 205 MoveAndFormatSlicer() procedure, 321 MsqBox function, 184 MSubstitute() function, 94-95 multicolumn list boxes, 532 multidimensional arrays, 454 Multidimensional Database (MDB) format, 475 MultiPage control, 198-200 multiple actions in With...End With blocks, 61 multiple characters, substituting, 94-95 multiple items, selecting, 252 multiple row fields, suppressing subtotals for, 326-327 multiple value fields (pivot tables), 302-303 Multiplelf() procedure, 122 multiplication table, building with R1C1style references, 134-135 MultiSelect property, 192-193 MyFullName() function, 82-83 MyName() function, 82

#### Ν

Name property, 149 named ranges, 66 named sets, 321-323 NameExists function, 155-157 names adding comments about, 150 array names, 153-154 checking existence of, 155-156 computer names, retrieving, 538-539 creating, 148-149 deleting, 149-150 explained, 147 formula names, 151 global versus local names, 147-148 hiding, 155 named ranges for VLOOKUP, 156-157 number names, 152-153 reserved names, 154-155 storing values in, 152 string names, 151-152 table names, 153 workbook names, setting in cell, 82 NASDAQMacro() procedure, 416-418 navigation keys, 31 nesting If statements, 124-126 loops, 112 NetTransfers() procedure, 486 new features (Excel 2010) charts. 139-140 conditional formatting, 140-141 objects/methods, 143 pivot tables, 140 Ribbon, 139

slicers. 140 SmartArt, 142 sorting, 141-142 tables. 141 New keyword, 437 NewDocument() procedure, 442 noncontiguous cells, selecting/ deselecting, 347-349 noncontiguous ranges, returning, 77 NumberFormat() procedure, 388-389 NumberFormat property, 388-389 numbers names, 152-153 retrieving from mixed text, 95 static random numbers, generating, 103 week numbers, converting into dates, 96 NumFilesInCurDir() function, 84-85 NumUniqueValues() function, 90-91

# 0

Object Browser, 56-57, 440-441 object-oriented languages, 33-34 object variables, 117-119 objects. See also specific objects ActiveX data objects. See ADO bookmarks, 448-449 in collections, 35 creating and referencing CreateObject() function, 438 GetObject() function, 438-439 New keyword, 437 custom objects creating, 497-498 Property Let/Property Get procedures, 499-501 referencing, 498-499

explained, 34 fill, 219-222 new features (Excel 2010), 143 properties, 36, 37 returned by properties, 46 watches on, 55 ObjectThemeColor property, 219 objForm LostFocus() procedure, 532 Offset property, 69-70, 251 OHLC (Open-High-Low-Close) charts, 235-236 OldLoop() procedure, 250 OldLoopToDelete() procedure, 250 OneColorGradient method, 221, 222 one-dimensional arrays, 454 On Error GoTo syntax, 552-554 On Error Resume Next statement, 554-555 open files, checking for, 539 Open-High-Low-Close (OHLC) charts, 235-236 Open method, 442 opening delimited files, 467-470 documents, 442 fixed-width files, 463-467 OpenSchema method, 487 OpenText method, 40, 42, 463 optimizing calculating elapsed time, 353-354 Page Setup, 350-353 option buttons, 194-195 optional parameters, 41 Origin parameter, 41 overlapping ranges, creating new ranges from, 73

#### Ρ

Page Setup, 350-353, 555 parameters event parameters, 160 explained, 35-36 optional parameters, 41 parsing text files, 332-333 PassAnArray() procedure, 460 passing arrays, 460 passwords cracking, 560 password box protection, 356-358 problems with, 560-561 pasting ranges, 61 .Patterned method, 221 Peltier, Jon, 243 percentages, showing, 305-308 permanent date/time, retrieving, 87 Personal Macro Workbook, 13 pivot cache, 295 pivot charts, 246-247 pivot tables building in Excel interface, 290-294 building in VBA, 294-301 adding fields to data area, 296-299 creating and configuring pivot table, 295-296 defining pivot cache, 295 calculated data fields, 324-325 calculated items, 325 changing calulations to show percentages, 305-308 changing layout of, 325-327 compact layout, 293-294 controlling sort order with AutoSort, 308 counting number of records, 303

data visualization, applying, 327 determining size of and converting pivot table to values, 299-301 drilling down, 349-350 eliminating blank cells in values area, 308 Excel 2007 new features, 288-290 Excel 2010 new features, 288 explained, 287 filtering data sets conceptual filters, 313-316 filtering to top five or top 10, 319 manual filters, 312-313 Search filter, 316-317 slicers, 319-321 with named sets, 321-323 with ShowDetail, 325 grouping daily dates to months, quarters, or years, 303-305 limitations, 299 multiple value fields, 302-303 new features (Excel 2010), 140 replicating reports for every product, 309-312 supressing subtotals for multiple row fields, 326-327 PivotColumnAxis property, 290 PivotRowAxis property, 290 playing sounds, 543 PlayWavSound API function, 543 Pope, Andy, 243 PresetGradient method, 222 PresetTextured method, 220 Print\_Area reserved name, 155 Print Titles reserved name, 155 PrintDrillIndicators property, 290 printing documents, 443 PrintOut method, 443

Priority property, 369 private properties, 497 procedural languages, 33-34 procedures. See specific procedures Programming window, examining macro recorder code in, 23-25 progress indicators, 355-356 Project Explorer, 20-21 properties. See also specific properties explained, 36-37 return values, 46 Properties window, 21 Property Get procedure, 499-501 Property Let procedure, 499-501 protecting code, 559 password boxes, 356-358 public properties, 497 publishing data to web pages, 404-406

# Q

queries, 391-395 QueryClose event, 187, 201 querying variable values, 50-54 Quick Access toolbar, 15-16 QuickFillMax() procedure, 456 QuickSort() procedure, 99-100

## R

R1C1-style references, 61 absolute references, 133 array formulas, 137-138 case study: entering A1 versus R1C1 references, 131 explained, 127-128 formulas, 129-132 mixed references, 133

multiplication table example, 134-135 referring to entire columns/rows, 134 relative references, 132-133 remembering column numbers associated with column letters, 136 switching to, 128 random numbers, generating, 103 Range object, 65-66, 444-447 defining ranges, 444-446 formatting ranges, 446-447 Range property, 69 ranges color scales, adding, 374-375 copying/pasting in one statement, 61 creating from overlapping ranges, 73 criteria ranges case study, 268 explained, 265-266 formula-based conditions, 268-275 logical AND criteria, 267 data bars, adding, 369-374 defining, 444-446 first nonzero-length cell, finding, 93 formatting, 446-447 icon sets, adding, 375-378 specifying icon set, 376 specifying ranges for each icon, 377-378 joining multiple ranges, 72-73 named ranges, 66, 156-157 names adding comments about, 150 creating, 148-149 deleting, 149-150 Range object, 65-66, 444-447

referencing, 59 with Offset property, 69-70 in other sheets, 67 relative to another range, 68 shortcuts, 66-67 removing duplicates from, 91-92 resizing, 71-72 returning addresses of maximum values in range, 101-102 returning noncontiguous ranges, 77 selecting with AutoFilter, 254-255 with Cells property, 68-69 with CurrentRegion property, 74-76 specifying syntax, 66 with Columns/Rows properties, 72 Ranges(), 59 RangeText() procedure, 444 reading text files, 332-333 files with fewer than 1,048,576 rows, 463-470 importing files with more than 1,048,576 rows, 470-473 ReadLargeFile() procedure, 472-473 Record Macro dialog box, 13 Record Macro icon (Developer tab), 9 recorded code, cleaning up, 58-64 recording macros, 12-14, 22-23. See also macro recorder records adding to databases, 480-481 counting number of, 303 deleting, 485 retrieving from databases, 481-483 showing after Filter in Place, 276

summarizing, 485-486 updating, 483-485 recordsets, 325, 478 RefEdit control, 515 references A1-style references, 127-128 case study: entering A1 versus R1C1 references, 131 R1C1-style references absolute references, 133 array formulas, 137-138 explained, 127-128 formulas, 129-132 mixed references, 133 multiplication table example, 134-135 referring to entire columns/rows, 134 relative references, 132-133 remembering column numbers associated with column letters, 136 switching to, 128 referencing charts, 203-207 custom objects, 498-499 objects CreateObject() function, 438 GetObject() function, 438-439 New keyword, 437 ranges, 59 with Offset property, 69-70 in other sheets, 67 relative to another range, 68 shortcuts, 66-67 tables, 77-78 reflection settings, 223

refreshing web gueries, 392-395 relative references, 26-31 case study, 26-28 R1C1-style references, 132-133 **RELS file, 571-572** RememberTheName() procedure, 206 Remove Duplicates command, 384-385 RemoveControl event, 187, 195, 200 removing add-ins. 593 duplicates from ranges, 91-92 renaming controls, 188 replacing loops with AutoFilter, 249-251 replicating reports for every product, 309-312 reports creating with Advanced Filter, 280-283 replicating for every product, 309-312 reserved names, 154-155 Reset button, 549-550 ResetRotation method, 229 Resize event, 187 Resize property, 71-72 resizing cell comments, 339-341 ranges, 71-72 userforms, 524 resolution, 540 RetrieveNumbers() function, 95 retrieving file paths, 543-546 filenames, 201-202 records, 481-483 return values of properties, 46 ReturnsMaxs() function, 101-102

RevenueByCustomers() procedure, 261 ReverseContents() function, 101 reversing cell contents, 101 RGB colors, 421-423 Ribbon changes in Excel 2010, 139 customizing to run macros control arguments, 569-571 control attributes, 566 custom icon images, 574-575 Custom UI Editor tool, 572 customui folder, 564-565 error messages, 577-580 explained, 563-565 file structure, accessing, 571 Microsoft Office icons, 573-574 RELS file, 571-572 tab and group, 565-566 macro buttons, creating, 14-15 rotation, 224-229 RotationX property, 228 RotationY property, 229 RotationZ property, 229 RowAxisLayout method, 289 rows, determining last row, 59-60 Rows property, 72 Run to Cursor debugging tool, 50 RunCustReport() procedure, 278-279 running macros, 14-17 from form controls, 16-17 from Quick Access toolbar, 15-16 from Ribbon, 14-15 timers, 542-543 RunReportForEachCustomer() procedure, 281-283

runtime adding controls at, 523-529 errors runtime error 9: Subscript Out of Range, 557 runtime error 1004: Method Range of Object Global Failed, 558-559

### S

Save As command (File menu), 589 Save method, 442 saved date/time, retrieving, 86-87 saving documents, 442-443 sbX\_Change() procedure, 245 sbY\_Change() procedure, 245 scaling sparklines, 414-418 scheduling macros to run every two minutes, 403-404 to run x minutes in the future, 401-402 verbal reminders, 402 Scroll event, 187, 195, 200 ScrollBar control, 517-519 Search box, 252-255 Search filter (pivot tables), 316-317 searching for strings within text, 100-101 security add-ins, 592 macro security Disable All Macros with Notification setting, 12 enable/disable settings, 11-12 trusted locations, 10-11 password box protection, 356-358 Select Case...End Select statement, 123

Select...Case statement, 104 Select statements, 123 SelectCase() procedure, 123 selected cells, highlighting, 342-345 selectina cells, 360 libraries, 45 multiple items, 252 noncontiguous cells, 347-349 ranges with Cells property, 68-69 with CurrentRegion property, 74-76 Selection object, 443-444 SelectSentence() procedure, 445 separating delimited strings, 96-97 worksheets into workbooks, 333-334 SetElement method, 203, 213-218 SetPresetCamera values, 225-229 SetReportInItalics() procedure, 559 SetTimer API function, 542-543 SetValuesToTabStrip() procedure, 514 shadow settings, 223 shapes, attaching macros to, 582-583 shared access databases, creating, 477-478 sharing UDFs (user-defined functions), 81-82 sheet events (workbook level), 166-167 SheetExists() function, 83-84 sheets, verifying existence of, 83-84 ShellAbout API function, 541 Show method, 186 ShowAllData method, 276 ShowCustForm() procedure, 263 ShowDetail, 325

ShowDrillIndicators property, 290 ShowTableStyleColumnHeaders property, 290 ShowTableStyleColumnStripes property, 290 ShowTableStyleLastColumn property, 290 ShowTableStyleRowHeaders property, 290 ShowTableStyleRowStripes property, 290 SimpleFilter() procedure, 285 size of charts, 204-205 of pivot tables, 299-301 slicers, 319-321 SmartArt, 142 soft edges, formatting, 223-224 SortConcat() function, 97-98 sorter() function, 99-100 sorting AutoSort, 308 with custom sort orders, 354-355 new features (Excel 2010), 141-142 numeric and alpha characters, 99-100 with SortConcat() function, 97-98 SortUsingCustomLists property, 290 sounds, playing, 543 sparklines creating, 412-413, 428-432 formatting RGB colors, 421-423 sparkline elements, 423-426 theme colors, 418-421 Win/Loss charts, 426-427 observations about, 428 scaling, 414-418 types of sparklines, 411

SpecialCells method, 276, 360 SpecifyExactLocation() procedure, 205 SpecifyLocation() procedure, 205 speed testing, 350-353 spin button events, 196-202 SpinDown event, 198 SpinUp event, 198 SQL Server, 490-491 stacked area charts, 239-243 standard modules, creating collections in, 501-502 StartRow parameter, 41 statements. See also loops Case, 124 If conditions, 115, 121-124 If...Then...Else, 121 If...Then...Else...End If, 122-123 If...Then...End If, 121-122 nesting, 124-126 On Error GoTo, 552-554 On Error Resume Next, 554-555 Select...Case, 104 Select Case...End Select, 123 Type..End Type, 506 state\_period() function, 103 static random numbers, generating, 103 StaticRAND() function, 103 Step clause (For statement), 110-111 stepping through code, 46-48 stock quotes, historical stock/fund quotes application, 362-363 StoplfTrue property, 369 StoreDashboard() procedure, 430-431 StoreTheName() procedure, 207 storing values in names, 152 StringElement() function, 96-97

strings delimited strings, separating, 96-97 finding within text, 100-101 names, 151-152 Styles gallery, 212-213 Sub cbConfirm Click() procedure, 484-485 subsets of ranges, creating icon sets for, 378-380 substituting multiple characters, 94-95 SubtotalLocation method, 289 subtotals, suppressing for multiple row fields, 326-327 SumColor() function, 89-90 summarizing records, 485-486 summing cells based on interior color, 89-90 suppressing Excel warnings, 556 subtotals for multiple row fields, 326-327 SwapElements() procedure, 100 switching to R1C1-style references, 128

#### Т

tab strips, 513-515 TableExists() procedure, 487-488 tables adding on-the-fly, 489 checking existence of, 487-488 exporting to, 336-337 names, 153 new features (Excel 2010), 141 pivot tables. *See* pivot tables referencing, 77-78 TableStyle2 property, 290 tabs creating for Ribbon, 565-566 tab order for userforms, 530 TabStrip control, 513-515 template chart types, 210-211 Terminate event, 187 testing macros, 25 speed testing, 350-353 text case, changing, 359-360 control tip text, 530 formatting cells that contain text, 386 mixed text, sorting numeric and alpha characters, 99-100 retrieving numbers from mixed text, 95 searching for strings within, 100-101 text boxes, 189 text files delimited files, opening, 467-470 fixed-width files, opening, 463-467 importing, 463-473 reading and parsing, 332-333 writing, 473-474 text files delimited files, opening, 467-470 fixed-width files, opening, 463-467 importing files with fewer than 1,048,576 rows, 463-470 files with more than 1,048,576

rows, 470-473

Text Import Wizard, 42, 464-467 Text to Columns Wizard, 43

writing, 473-474

reading and parsing, 332-333

TextBox event, 191, 195 TextToColumns method, 471 theme colors for sparklines, 418-421 time elapsed time, calculating, 353-354 military time, entering into cells, 171 permanent date/time, retrieving, 87 saved date/time, retrieving, 86-87 timers, 542-543 ToggleButton control, 517 toolbars converting Excel 2003 custom toolbar to Excel 2010, 575-577 UserForm toolbar, 511 ToolTips, 53 Top 10 cells filtering to, 319 formatting, 383-384 Top5Customers() procedure, 317-319 Top10Filter() procedure, 252 Top/Bottom Rules, 383-384 TrailingMinusNumbers parameter, 42, 561 training clients about error handling, 557 transparent forms, 533-534 TransposeArray() procedure, 458 transposing data, 345-347 TrapChartEvent() procedure, 497 trapping application events, 494-495 embedded chart events, 495-497 TrickyFormatting() procedure, 380 troubleshooting. See error handling; error messages trusted locations, 10-11 TwoColorGradient() procedure, 221

Type..End Type statement, 506 types, user-defined types (UDTs), 506-509 TypeText method, 444

### U

UDFs (user-defined functions) BookOpen(), 83 case study, 80 ColName(), 103 ContainsText(), 100-101 creating, 79-81 DateTime(), 87 FirstNonZeroLength(), 93 GetAddress(), 102-103 IsEmailValid(), 88-89 LastSaved(), 86-87 MSubstitute(), 94-95 MyFullName(), 82-83 MyName(), 82 NumFilesInCurDir(), 84-85 NumUniqueValues(), 90-91 RetrieveNumbers(), 95 ReturnsMaxs(), 101-102 ReverseContents(), 101 sharing, 81-82 SheetExists(), 83-84 SortConcat(), 97-98 sorter(), 99-100 state\_period(), 103 StaticRAND(), 103 StringElement(), 96-97 SumColor(), 89-90 UniqueValues(), 91-92 Weekday(), 96 WinUserName(), 85-86 UDTs (user-defined types), 506-509 Union method, 72-73 unique cells, formatting, 384-385 Unique Records Only, 283-285 unique values counting, 90-91 extracting with Advanced Filter, 258-264 getting unique combinations of two or more fields, 263-264 with user interface, 259 with VBA code, 260-263 UniqueCustomerProduct() procedure, 263-264 UniqueCustomerRedux() procedure, 261 UniqueProductsOneCustomer() procedure, 266 UniqueValues() function, 91-92 Unload method, 186 Until clause (Do loops), 115-117 updating records, 483-485 web queries, 395 Use Relative Reference icon (Developer tab), 9 UseBookmarks() procedure, 448 UseGetObject() procedure, 438 user-defined functions. See UDFs user-defined types (UDTs), 506-509 UserForm toolbar, 511 UserForm Initialize() procedure, 527-528 UserForm QueryClose() procedure, 532 userforms, 183-202 calling, 186 command buttons, 189 controls adding at runtime, 523-529 adding on-the-fly, 525 CheckBox, 512-513

grouping into collections, 519-521 programming, 188 RefEdit, 515 ScrollBar, 517-519 TabStrip, 513-515 ToggleButton, 517 troubleshooting, 189 creating, 184-185 Debug errors inside userform code, 551-552 disabling X button for closing userforms, 541-542 dynamic charts, creating, 244-245 field entry, verifying, 200 filenames, retrieving, 201-202 help, adding, 529-532 accelerator keys, 529 coloring active control, 530-532 control tip text, 530 hiding, 186 hyperlinks in, 522 images adding on-the-fly, 526-527 graphics events, 195-202 input boxes, 183-184 labels, 189 list boxes, 191-193 message boxes, 184 modeless userforms, 521 MultiPage control, 198-200 option buttons, 194-195 resizing on-the-fly, 524 spin buttons, 196-202 tab order, 530 text boxes, 189 transparent forms, 533-534 UserForm toolbar, 511

viewing code, 186 windows, closing, 200-201 **USERID** function, 85-86 UserIDs, retrieving, 85-86

### V

validating e-mail addresses, 88-89 values constant values explained, 439 retrieving with Object Browser, 440-441 retrieving with Watch window, 440 converting pivot tables to, 299-301 duplicates, removing from ranges, 91-92 formatting cells based on, 385 maximum values in range, returning addresses of, 101-102 storing in names, 152 unique values counting, 90-91 extracting with Advanced Filter, 258-264

#### variables

DataSets, 473 hard-coding versus, 60-61 in For statements, 110 object variables, 117-119 querving values of, 50-54 requiring declaration, 20 wdApp, 435 wdDoc, 435

VB Editor, 19-21 converting files to add-ins, 590-591 debugging tools breakpoints, 49 jumping forward/backward in code, 49-50 querying variable values, 50-54 Run to Cursor, 50 stepping through code, 46-48 watches, 55 Object Browser, 56-57 Programming window, 23-25 Project Explorer, 20-21 Properties window, 21 settings, 19-20 VBA (Visual Basic for Applications) advantages of, 8-9 learning curve, 8 syntax, 34-37 VBA Extensibility, 363-365 verbal reminders, scheduling, 402 verifying field entry, 200 Version property, 144-145 versions, errors caused by different versions, 561 viewing Developer tab, 9-10 Project Explorer, 20 Properties window, 21 userform code, 186 visible cells, selecting with AutoFilter, 255-256 Visual Basic for Applications. See VBA (Visual Basic for Applications) Visual Basic icon (Developer tab), 9 visualizations. See data visualizations VLOOKUP function, 156-157

#### W

warnings, suppressing, 556 Watch window, 440 watches querying variable values with, 53-54 setting breakpoints, 55 wdApp variable, 435 wdDoc variable, 435 web pages creating custom, 406 publishing data to, 404-406 web queries, 391-392 building, 396-399 creating manually and refreshing with VBA, 392-395 scraping, 399 updating, 395 week numbers, converting into dates, 96 Weekday() function, 96 While clause (Do loops), 115-117 While...Wend loops, 117 Window API declarations Windows API declarations 32-bit versus 64-bit, 538 ahtAddFilterItem, 546 aht\_apiGetOpenFileName, 544-546 aht\_apiGetSaveFileName, 544-546 calling, 537 DisplaySize, 540 explained, 535-536 finding, 547 FindWindow, 541-543 GetComputerName, 538-539 GetSystemMenu, 541-542 KillTimer, 542-543 lOpen, 539

PlayWavSound, 543 SetTimer, 542-543 ShellAbout, 541 windows for userforms, closing, 200-201 Win/Loss charts, 426-427 WinUserName() function, 85-86 With...End With blocks, 61 wizards, Text Import Wizard, 464-467 Word automation bookmarks, 448-449 constant values explained, 439 retrieving with Watch window, 440 controlling form fields, 450-452 creating and referencing objects CreateObject() function, 438 GetObject() function, 438-439 New keyword, 437 Document object closing documents, 443 creating documents, 442 explained, 442 opening documents, 442 printing documents, 443 saving documents, 442-443 early binding, 433-436 explained, 433 late binding, 436-437 macro recorder, 441 Range object, 444-447 defining ranges, 444-446 formatting ranges, 446-447 Selection object, 443-444 Word documents, exporting to, 336-337 WordEarlyBinding() procedure, 435 WordLateBinding() procedure, 437

Workbook Activate() event, 161 Workbook AddInInstall() event, 165 Workbook AddInUninstall event, 165 Workbook AfterXmlExport() event, 166 Workbook AfterXmlImport() event, 166 Workbook BeforeClose() event, 163-164 Workbook\_BeforePrint() event, 163, 494 Workbook\_BeforeSave() event, 162 Workbook BeforeXmlExport() event, 166 Workbook\_BeforeXmlImport() event, 166 Workbook\_Deactivate() event, 161 Workbook NewSheet() event, 164 Workbook\_Open() event, 161 Workbook\_Open() procedure, 594 Workbook PivotTableCloseConnection() event, 165 Workbook\_PivotTableOpenConnection() event, 165 Workbook\_RowsetComplete() event, 165 Workbook SheetActivate() event, 166 Workbook SheetBeforeDoubleClick() event, 167 Workbook\_SheetBeforeRightClick() event, 167 Workbook SheetCalculate() event, 167 Workbook\_SheetChange () event, 167 Workbook\_SheetDeactivate() event, 167 Workbook\_SheetFollowHyperlink() event, 167 Workbook SheetPivotTableUpdate() event, 167 Workbook\_SheetSelectionChange() event, 167 Workbook Sync() event, 165 Workbook\_WindowActivate() event, 165 Workbook\_WindowDeactivate() event, 165 Workbook\_WindowResize() event, 164

workbooks checking whether open, 83 combining worksheets into, 334-335 converting to add-ins, 588-590 copying macros into, 363-365 counting number of workbooks in directory, 84-85 events Workbook Activate(), 161 Workbook AddInInstall(), 165 Workbook AddInUninstall, 165 Workbook AfterXmlExport(), 166 Workbook AfterXmlImport(), 166 Workbook\_BeforeClose(), 163-164 Workbook\_BeforePrint(), 163 Workbook\_BeforeSave(), 162 Workbook\_BeforeXmlExport(), 166 Workbook BeforeXmlImport(), 166 Workbook Deactivate(), 161 Workbook NewSheet(), 164 Workbook Open(), 161 Workbook PivotTableCloseConnection(), 165 Workbook PivotTableOpenConnection(), 165 Workbook\_RowsetComplete(), 165 Workbook\_Sync(), 165 Workbook WindowActivate(), 165 Workbook\_WindowDeactivate(), 165 Workbook\_WindowResize(), 164

hidden workbooks as alternative to add-ins, 593-594 case study: hidden workbook to hold macros and forms, 594 permanent date/time, retrieving, 87 saved date/time, retrieving, 86-87 separating worksheets into, 333-334 Workbooks object, 40 Worksheet Activate() event, 168 Worksheet BeforeDoubleClick() event, 168 Worksheet BeforeRightClick() event, 169 Worksheet\_BeforeRightClick() procedure, 160 Worksheet Calculate() event, 169 Worksheet\_Change() event, 170 Worksheet\_Change() procedure, 161 Worksheet Deactivate() event, 168 Worksheet\_FollowHyperlink() event, 171 Worksheet\_PivotTableUpdate() event, 172 Worksheet\_SelectionChange() event, 170 worksheets combining into workbooks, 334-335 events, 168-172 filtering/copying data into, 335-336 referencing ranges in other sheets, 67 Select...Case statements on, 104 separating into workbooks, 333-334 WriteFile() procedure, 474 WriteHTML() procedure, 554 writing text files, 473-474 Wrong Number of Arguments or Invalid Property Assignment (error message), 580

# X-Y-Z

X button, disabling, 541-542 xlApp\_NewWorkbook() procedure, 495 XLFilterInPlace constant, 275 .xls file type, 18 .xlsb file type, 18 .xlsm file type, 18 .xlsx file type, 18

Zoom event, 187, 195, 200