Microsoft

developer

// Step by step

Microsoft Visual Basic 2013

Intermediate

Michael Halvorson

// Step by step

Your hands-on guide to Visual Basic fundamentals

Expand your expertise—and teach yourself the fundamentals of Microsoft Visual Basic 2013. If you have previous programming experience but are new to Visual Basic 2013, this tutorial delivers the step-by-step guidance and coding exercises you need to master core topics and techniques.

Discover how to:

- Master essential Visual Basic programming techniques
- Begin building apps for the Windows Store, Windows Phone 8, and ASP.NET
- Design apps using XAML markup, touch input, and live tiles
- Tackle advanced language concepts, such as polymorphism
- Manage data sources, including XML documents and web data
- Create a Windows Phone 8 app that manages key lifecycle events

Technologies Covered

- Windows 8.1
- Microsoft Visual Basic 2013
- Microsoft .NET Framework 4.5.1
- ASP.NET 4.5.1
- Windows Phone 8

About the Author

Michael Halvorson, a former Visual Basic localization manager at Microsoft, is the award-winning author of more than 35 books, including *Microsoft Visual Basic 2010 Step by Step* and *Start Here! Learn Microsoft Visual Basic 2012*.

Practice Files + Code

Available at:
http://aka.ms/VB2013_SbS/files

Microsoft Visual Basic Express 2013 is available as a free download at Microsoft.com/express. See the Introduction.

Companion eBook

See the instruction page at the back of the book.

microsoft.com/mspress

ISBN: 978-0-7356-6704-4

9 780735 667044 90000

U.S.A. $44.99
Canada $47.99
[*Recommended*]

Programming/Microsoft Visual Basic

Microsoft Press

Celebrating 30 years!

Microsoft

Microsoft Visual Basic 2013 Step by Step

Michael Halvorson

ISBN: 978-0-7356-6704-4

Third Printing: December 2014

Printed and bound in the United States of America.

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at *http://www.microsoft.com/learning/booksurvey*.

Acquisitions and Developmental Editor: Russell Jones

Production Editor: Kristen Brown

Editorial Production: Zyg Group, LLC

Technical Reviewer: Tim Patrick

Copyeditor: Richard Carey

Indexer: Bob Pfahler

Cover Design: Twist Creative • Seattle

Cover Composition: Randy Comer

Illustrator: Rebecca Demarest

Contents at a glance

Introduction *xvii*

PART I INTRODUCTION TO VISUAL STUDIO DEVELOPMENT

CHAPTER 1 Visual Basic 2013 development opportunities and the Windows Store 3

CHAPTER 2 The Visual Studio Integrated Development Environment 17

CHAPTER 3 Creating your first Windows Store application 43

CHAPTER 4 Windows desktop apps: A walkthrough using Windows Forms 79

PART II DESIGNING THE USER INTERFACE

CHAPTER 5 Working with Windows Store app controls 111

CHAPTER 6 Working with Windows Forms controls 147

CHAPTER 7 XAML markup step by step 191

CHAPTER 8 Using XAML styles 215

CHAPTER 9 Exploring Windows 8.1 design features: Command bar, flyout, tiles, and touch 235

CHAPTER 10 Creating console applications 267

PART III VISUAL BASIC PROGRAMMING TECHNIQUES

CHAPTER 11 Mastering data types, operators, and string processing 291

CHAPTER 12 Creative decision structures and loops 341

CHAPTER 13 Trapping errors by using structured error handling 375

CHAPTER 14 Using arrays, collections, and generics to manage data 397

CHAPTER 15 Innovative data management with LINQ 435

CHAPTER 16 Object-oriented programming techniques 459

PART IV DATABASE AND WEB PROGRAMMING

CHAPTER 17 Database controls for Windows desktop apps 489

CHAPTER 18 Data access for Windows Store apps 515

CHAPTER 19 Visual Studio web development with ASP.NET 543

PART V MICROSOFT WINDOWS PHONE PROGRAMMING

CHAPTER 20 **Introduction to Windows Phone 8 development** **587**

CHAPTER 21 **Creating your first Windows Phone 8 application** **607**

Index *641*

About the author *671*

Contents

Introduction . . . . . *xvii*

PART I INTRODUCTION TO VISUAL STUDIO DEVELOPMENT

Chapter 1 Visual Basic 2013 development opportunities and the Windows Store 3

Visual Basic 2013 products and opportunities 4
- An impressive range of development opportunities and platforms 5
- Taking a multiplatform approach to learning Visual Basic 7

Evaluating the Windows Store 8
- What is the Windows Store? 8
- Accessing the Windows Store 9
- Sales information and price tiers 10
- Or your application could be free... 11
- Planning ahead for certification 12
- Windows Store requirements checklist 12
- It's all in the details 15

Summary 16

Chapter 2 The Visual Studio Integrated Development Environment 17

Getting started 18

The Visual Studio development environment 19

Important tools in the IDE 22
- Organizing tools in the IDE 24
- The Designer and XAML markup 25
- Running and testing Windows Store apps 30

Working with the Properties window 33

Organizing the programming tools 36
Moving and docking tools 37
Hiding tool windows 38
Configuring the IDE for step-by-step exercises 39
Exiting Visual Studio 42
Summary 42

Chapter 3 Creating your first Windows Store application 43

Lucky Seven: A Visual Basic app for the Windows Store 44
Programming step by step 44
Designing the user interface 45
Final property settings and adjustments 61
Writing the code 63
A look at the *SpinButton_Click* event handler 67
Running Windows Store apps 68
Creating a splash screen for your app 70
Building an executable file 74
Summary 78

Chapter 4 Windows desktop apps: A walkthrough using Windows Forms 79

Inside Windows desktop apps 80
Visual Basic and Windows desktop apps 81
Creating a Windows desktop app 83
Setting properties 93
The picture box properties 97
Naming objects for clarity 98
Writing the code 99
Behind the scenes in the *SpinButton_Click* event handler 101
Running the Lucky Seven desktop app 103
Building an executable file 104

Publishing a Windows desktop app 105
Summary. 107

PART II DESIGNING THE USER INTERFACE

Chapter 5 Working with Windows Store app controls 111

Understanding Windows Store app controls 112
Roots in Windows Presentation Foundation and XAML 112
Designing for Windows 8.1 113
Using the *TextBox* control to receive input 113
Assigning *TextBox* contents to a variable. 118
Multiline *TextBox* controls. 120
Check spelling in a *TextBox* control 124
Using the *FlipView* control to display a series of images 127
Using the *MediaElement* control to play entertainment media 133
Use the *WebView* control to display live web content 141
Summary. 146

Chapter 6 Working with Windows Forms controls 147

Using the *DateTimePicker* control 148
Controls for gathering input. 154
Using the *CheckBox* control 155
Using group boxes and radio buttons 159
Processing input with list boxes. 164
Adding menus by using the *MenuStrip* control. 169
Menu features. 170
Adding access keys to menu commands 172
Processing menu choices 175
Adding toolbars with the *ToolStrip* control 180

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

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

microsoft.com/learning/booksurvey

Using dialog box controls 183
Event handlers that manage common dialog boxes 185
Summary 190

Chapter 7 XAML markup step by step 191

Introduction to XAML 192
XAML in the Visual Studio IDE 192
XAML in Blend for Visual Studio 193
XAML elements 194
Namespaces in XAML markup 196
Examining XAML project files 196
Adding XAML elements using the Code Editor 202
Summary 213

Chapter 8 Using XAML styles 215

Introduction to XAML styles 215
Where did StandardStyles.xaml go? 216
Creating new XAML styles 217
Considering the scope of a style 218
Sample markup for a new XAML style 219
Referencing a style 220
Using explicit and implicit styles 220
Practicing XAML styles 221
Building new styles from existing styles 228
IDE shortcuts for applying styles 231
Summary 233

Chapter 9 Exploring Windows 8.1 design features: Command bar, flyout, tiles, and touch 235

Creating a command bar to manage common tasks 236
Command bar features 237
Designing your command bar 238
Command bar practice step by step 240

Using the *Flyout* control to collect input and display information. 243
Designing custom tiles for your app 249
The Assets folder 249
Required tiles and uses 249
Programming live tiles 257
Planning for touch input 259
XAML controls handle touch automatically 259
Common gestures 260
Usability considerations 262
Security and permissions settings 263
Summary. 266

Chapter 10 Creating console applications 267

Console applications in Visual Studio 268
Creating a console application 268
Modules and procedures 270
The *Sub Main()* procedure 271
Interactive math games 275
Find the number. 275
Simulating dice 280
Building, publishing, and running console apps 284
Summary. 288

PART III VISUAL BASIC PROGRAMMING TECHNIQUES

Chapter 11 Mastering data types, operators, and string processing 291

Strategies for declaring variables and constants. 292
The *Dim* statement 292
Defining constants 295
Guidelines for naming variables and constants 296
Data types and the *ListBox* control 297

Operators and formulas 304
Arithmetic operators 305
Advanced arithmetic operators 308
Shortcut operators 313
How Visual Basic calculates formulas 314
Converting data types 315
The *ToString* method 316
The *Parse* method 316
The *Convert* class 318
Older type conversion functions and their uses 319
Processing strings with the *String* class 320
Common tasks 320
Sorting text 322
Working with ASCII codes 323
Sorting strings in a text box 325
Examining the Sort Text program code 328
Protecting text with basic encryption 331
Using the *Xor* operator 334
Examining the encryption program code 336
Summary 339

Chapter 12 Creative decision structures and loops 341

Event-driven programming 342
Using conditional expressions 343
If...Then decision structures 344
Testing several conditions in an *If...Then* decision structure 344
Using logical operators in conditional expressions 349
Short-circuiting by using *AndAlso* and *OrElse* *352*
Mastering *Select Case* decision structures 353
Using comparison operators with a *Select Case* structure 355
Mastering *For...Next* loops 361
Using a loop to fill a *TextBox* with string data 362

Complex *For...Next* loops 363
The *Exit For* statement 367
Writing *Do* loops 368
Avoiding an endless loop 369
Converting temperatures 370
Using the *Until* keyword in *Do* loops 372
Summary 373

Chapter 13 Trapping errors by using structured error handling 375

Processing errors by using the *Try...Catch* statement 376
When to use error handlers 376
Setting the trap: the *Try...Catch* code block 377
Path name and drive errors 378
Windows Store apps and built-in exception handling 383
Writing a flash drive error handler 384
Using the *Finally* clause to perform cleanup tasks 385
More complex *Try...Catch* error handlers 387
The *Exception* object 387
Specifying a retry period 390
Using nested *Try...Catch* blocks 392
Comparing error handlers with defensive programming techniques . . . 393
The *Exit Try* statement 394
Summary 395

Chapter 14 Using arrays, collections, and generics to manage data 397

Working with arrays of variables 398
Creating an array 398
Declaring an array with set elements 399
Setting aside memory 400
Working with array elements 401
Declaring an array and assigning initial values 402
Creating an array to hold temperatures 404
The *GetUpperBound* and *GetLowerBound* methods 404

Setting an array's size at runtime 409
Preserving array contents by using *ReDim Preserve* 414
Using *ReDim* for three-dimensional arrays 415
Processing large arrays by using methods in the *Array* class 416
The *Array* class 416
Get your sort on 422
Working with collections 422
Creating collections and generic lists 423
Declaring generic collections 424
Sample app with generic list and background image 425
Summary 433

Chapter 15 Innovative data management with LINQ 435

LINQ tools and techniques 435
Fundamental query syntax 436
Extracting information from arrays 437
Using LINQ with collections 450
Using LINQ with XML documents 454
Summary 458

Chapter 16 Object-oriented programming techniques 459

Inheriting a form by using the Inheritance Picker 460
Creating your own base classes 466
Adding a new class to your project 467
Inheriting a base class 476
Polymorphism 480
Syntax for overriding methods and properties 480
Referring to the base class with *MyBase* *481*
Experimenting with polymorphism 481
Summary 486

PART IV DATABASE AND WEB PROGRAMMING

Chapter 17 Database controls for Windows desktop apps 489

Database programming with ADO.NET 490
Database terminology 490
Working with an Access database 492
The Data Sources window 501
Using toolbox controls to display database information 506
SQL statements and filtering data 509
Summary 514

Chapter 18 Data access for Windows Store apps 515

Data binding in XAML 516
A variety of data sources 516
Binding elements 516
Binding a control to a class 517
Using a collection as a source of data 522
Accessing data in XML documents 526
Reading an XML file 526
Searching for items in an XML file 533
Writing to an XML file 536
A user interface for data entry 540
Summary 541

Chapter 19 Visual Studio web development with ASP.NET 543

Inside ASP.NET 544
Web Forms 545
ASP.NET MVC 546
Web Pages (with Razor) 547
HTML5 and JavaScript 548
Building a Web Forms website with ASP.NET 550
Software requirements for ASP.NET development 550
Essential steps 551
Webpages vs. Windows Forms 552

Using the Web Designer 557
Adding server controls to a website 561
Writing event handlers for webpage controls 563
Customizing the website template 570
Displaying database records on a webpage 573
Editing document and site master properties 581
Summary 584

PART V MICROSOFT WINDOWS PHONE PROGRAMMING

Chapter 20 Introduction to Windows Phone 8 development 587

Opportunities in the Windows Phone 8 platform 588
Key Windows Phone 8 features 589
Hardware requirements 590
Integration and collaboration 590
The Windows Phone Store 591
What is the Windows Phone Store? 591
Accessing the Windows Phone Store 591
How much money do developers make? 595
Planning ahead for certification 595
Working with Windows Phone SDK 8.0 596
Downloading the SDK 598
Comparing Windows Phone 8 and Windows Store platforms 600
Differences 601
Similarities 603
Summary 605

Chapter 21 Creating your first Windows Phone 8 application 607

Creating a Windows Phone project 608
Designing the Golf Caddy user interface 614
Writing the code 617
Testing Windows Phone apps 620

Application life cycle considerations 626
Closing or deactivating? 626
The *PhoneApplicationSerivce* class 628
Life cycle management with the *IsolatedStorageSettings* class . . . 636
Setting options in the Window Phone manifest file 637
Summary 639

Index *641*

About the author *671*

Introduction

Microsoft Visual Basic 2013 is an important upgrade and enhancement of the popular Visual Basic programming language and compiler, a technology that enjoys an installed base of millions of programmers worldwide. Visual Basic 2013 is not a standalone product but a key component of Microsoft Visual Studio 2013—a comprehensive development system that allows you to create powerful applications for Microsoft Windows 8.1, the Windows desktop, the web, Windows Phone 8, and a host of other environments.

Whether you purchase one of the commercial editions of Visual Studio 2013 or you download Visual Basic Express 2013 for a free test-drive of the software, you are in for an exciting experience. The latest features of Visual Basic will increase your productivity and programming prowess, especially if you enjoy using and integrating information from databases, entertainment media, webpages, and websites. In addition, an important benefit of learning Visual Basic and the Visual Studio Integrated Development Environment (IDE) is that you can use many of the same tools to write programs for Microsoft Visual C# 2013, Microsoft Visual C++ 2013, HTML5 and JavaScript, and other popular languages.

Microsoft Visual Basic 2013 Step by Step is a comprehensive introduction to Visual Basic programming using the Visual Basic 2013 software and Windows 8.1. I've designed this practical, hands-on tutorial with a variety of skill levels in mind. In my opinion, the best way to master a complex technology like Visual Basic is to follow the premise that programmers learn by doing. Therefore, by reading this book and working through the examples, you'll learn essential programming techniques through carefully prepared tutorials that you can complete on your own schedule and at your own pace.

Although I have significant experience with college teaching and corporate project management, this book is not a dry textbook or an "A to Z" programmer's reference; instead, it is a practical hands-on programming tutorial that puts *you* in charge of your learning, developmental milestones, and achievements. By using this book, programmers who are new to this topic will learn Visual Basic software development fundamentals in the context of useful, real-world applications; and intermediate Visual Basic programmers can quickly master the essential tools and techniques offered in the Visual Basic 2013 and Windows 8.1 upgrades.

I've taken a multiplatform approach in this book, so in addition to learning Visual Basic programming skills you'll learn to create a wide variety of applications, including Windows Store apps, Windows Forms (Windows desktop) apps, console apps, web apps

(ASP.NET), and Windows Phone 8 apps. Each of these application types has a place and a purpose in real-world development.

To complement this comprehensive approach, the book is structured into 5 topically organized parts, 21 chapters, and dozens of step-by-step exercises and sample programs. By using this book, you'll quickly learn how to create professional-quality Visual Basic 2013 applications for the Windows operating system, Windows Phone 8 platform, and a variety of web browsers. You'll also have fun!

Who should read this book

This is a step-by-step programming tutorial for readers who enjoy learning to do new things by doing them. My assumption is that you already have some experience with programming, possibly even an earlier version of Visual Basic, and that you are ready to learn about the Visual Studio 2013 product in the context of building applications that you can market in the Windows Store, Windows Forms (Windows desktop) for personal and enterprise purposes, web (ASP.NET) applications that run in browsers, and apps for the Windows Phone 8 platforms.

This book's content will supply you with concrete Visual Basic coding techniques as well as a broad overview of programming strategies suitable for Visual Basic development. The book's extensive collection of step-by-step exercises has a broad focus; they are written for technical people who understand programming and are not simply targeted toward hobbyists or absolute beginners. In addition, you will learn about the capabilities of the Windows 8.1 operating system and the specific design guidelines that Microsoft recommends for Windows 8.1 and Windows Phone 8 applications.

Assumptions

This book is designed to teach readers how to use the Visual Basic programming language. You will also learn how to use the Visual Studio 2013 IDE and development tools. This book assumes no previous experience with Visual Studio 2013, but it is written for readers who understand programming and are not absolute beginners. I assume that you are familiar with programming basics or have studied some version of BASIC or Visual Basic in the past and are now ready to move beyond elementary skills to platform-specific techniques.

If you have no prior knowledge of programming or Visual Basic, you might want to fill in some of the gaps with my introduction to Visual Basic 2012 and Windows Store development, *Start Here! Learn Visual Basic 2012* (Microsoft Press, 2012). From time

to time, I will refer to the exercises in that book to give you additional resources for your learning.

Microsoft Visual Basic 2013 Step by Step also assumes that you have acquired and are running the Windows 8.1 operating system and that you want to learn how to create applications for the Windows Store platform and other environments. To make the most of your programming practice, you will need to know a little about how to perform common tasks in Windows 8.1, how to customize the Start page and user interface, how to work with information on the web, and how to adjust basic system settings. If you also have Windows 8.1 installed on a tablet or touchpad device, all the better, because a fundamental design emphasis of Windows 8.1 is to make touch and gestures a natural way to manipulate content. You can build your applications on a laptop or desktop running Visual Studio 2013 and Windows 8.1 and then test out the applications on your tablet or touchpad.

In terms of the Visual Studio software, I assume that you are using one of the full, retail versions of Visual Studio 2013, such as Visual Studio Professional, Premium, or Ultimate. This will enable you to create the full range of application types that I describe in this book, including Windows Store apps, Windows Forms (Windows desktop) apps, console apps, Web Forms (ASP.NET) apps, and Windows Phone 8 apps.

If you don't have access to a full, retail version of Visual Studio 2013, you can experiment with the Visual Studio 2013 software by downloading free versions of the suite designed for specific platforms. These limited-feature or "Express" versions of Visual Studio 2013 are called Express for Windows, Express for Windows Desktop, Express for Windows Phone, and Express for Web. The Visual Studio website (*http://www.microsoft.com/visualstudio*) provides access to the retail and Express versions of Visual Studio, and it explains the differences among all of the available versions.

Who should not read this book

You might be disappointed with this book if you are already a knowledgeable Visual Basic programmer and are just looking to explore the new features of Visual Studio 2013. The *Step By Step* series is targeted toward readers who are professional developers but who have little to no previous experience with the topic at hand. If you are an advanced Visual Basic developer, you are likely to grow weary of the step by step exercises that introduce essential features such as decision structures, XAML markup, data access strategies, or using the .NET Framework.

Developers who have a lot of experience will feel that I'm exploring the obvious—but what is obvious to experienced programmers often isn't obvious at all to someone who is learning to use a new development platform. If Windows Store or Windows Phone programming with Visual Basic is a new concept for you, this is the place to start.

Organization of this book

This book is divided into five sections, each of which focuses on a different aspect or technology within the Visual Studio software and the Visual Basic programming language. Part I, "Introduction to Visual Studio development," provides an overview of the Visual Studio 2013 IDE and its fundamental role in .NET application creation and then moves into step-by-step development walkthroughs on the Windows Store and Windows Forms (Windows desktop) platforms.

Part II, "Designing the user interface," continues the focus on application creation in the Visual Studio IDE, emphasizing the construction of Windows Store apps, Windows Forms (Windows desktop) apps, and console apps. In particular, you'll learn how to work with XAML markup, XAML styles, important controls, and new Windows 8.1 design features, including command bar, flyout, tiles on the Windows Start page, and touch input.

Part III, "Visual Basic programming techniques," covers core Visual Basic programming skills, including managing data types, using the .NET Framework, structured error handling, working with collections and generics, data management with LINQ, and fundamental object-oriented programming skills.

Part IV, "Database and web programming," introduces data management techniques in Windows desktop and Windows Store applications, including binding data to controls and working with XML documents and Microsoft Access data sources. You'll also get an overview of ASP.NET web development strategies, along with a complete walkthrough of web development on the Web Forms (ASP.NET) platform.

Finally, Part V, "Microsoft Windows Phone programming," provides an overview of the features and capabilities presented by the Windows Phone 8 platform. You'll identify key hardware characteristics in the Windows Phone ecosystem, the marketing opportunities tendered by the Windows Phone Store, and you'll create a complete Windows Phone 8 app step by step.

Finding your best starting point in this book

This book is designed to help you build skills in a number of essential areas. You can use it if you're new to programming, switching from another programming language, or upgrading from Visual Studio 2010 or Visual Basic 2012. Use the following table to find your best starting point in this book.

If you are ...	Follow these steps
New to Visual Basic programming	**1.** Install the sample projects as described in the section "Installing the code samples," later in this Introduction. **2.** Learn essential skills for using Visual Studio and Visual Basic by working sequentially from Chapter 1 through Chapter 21. **3.** Use the companion book *Start Here! Learn Microsoft Visual Basic 2012* for additional instruction as your level of experience dictates.
Upgrading from Visual Basic 2010 or 2012	**1.** Install the sample projects as described in the section "Installing the code samples." **2.** Read Chapter 1, skim Chapters 2 through 4, and complete Chapters 5 through 21.
Interested primarily in creating Windows Store apps for Windows 8.1	**1.** Install the sample projects as described in the section "Installing the code samples." **2.** Complete Chapters 1 through 3, Chapter 5, Chapters 7 through 16, and Chapter 18.
Interested primarily in creating Windows Forms (Windows desktop) apps for Windows 8.1, Windows 8, or Windows 7	**1.** Install the sample projects as described in the section "Installing the code samples." **2.** Complete Chapters 1 through 2, Chapter 4, Chapter 6, Chapter 10, and Chapters 11 through 17.

Conventions and features in this book

This book presents information using the following conventions designed to make the information readable and easy to follow:

- Each exercise consists of a series of tasks, presented as numbered steps (1, 2, and so on) listing each action you must take to complete the exercise.
- The names of all program elements—controls, objects, methods, functions, properties, classes, variable names, and so on—appear in *italics*.
- As you work through steps, you'll occasionally see tables with lists of properties that you'll set in Visual Studio. Text properties appear within quotes, but you don't need to type the quotes.
- Boxed elements with labels such as "Note" provide additional information or alternative methods for completing a step successfully.
- Text that you type (including some code blocks) appears in **bold**.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press the Tab key.
- A vertical bar between two or more menu items (for example, File | Close) means that you should select the first menu or menu item, then the next, and so on.

System requirements

You will need the following hardware and software to work through the examples in this book:

- The Windows 8.1 operating system. (Depending on your Windows configuration, you might also require Local Administrator rights to install or configure Visual Studio 2013.) Note that while the full versions of Visual Studio 2013 do support earlier versions of Windows, such as Windows 8 and Windows 7 SP1, the features described in this book require Windows 8.1, and the screen shots will all show this environment.

- A full retail edition of Visual Studio 2013, required for completing all of the exercises in this book (Visual Studio 2013 Professional, Premium, or Ultimate). The Visual Studio website (*http://www.microsoft.com/visualstudio*) explains the differences among these versions. Alternatively, you can experiment with the Visual Studio 2013 software by downloading free versions of the suite designed for specific platforms. The limited-feature versions of Visual Studio 2013 are called Express for Windows, Express for Windows Desktop, Express for Windows Phone, and Express for Web. You will need to download all four of these Express versions to get the necessary software to complete the book's exercises. (However, even with these Express editions, there will be a few gaps; for example, you will be unable to complete Chapter 10, "Creating console applications.")
- An Internet connection to view Visual Studio help files, try out the Windows Store and Windows Phone Store, and download this book's sample files.
- A computer with 1.6 GHz or faster processor.
- 1 GB RAM (32-bit) or 2 GB RAM (64-bit).
- 16 GB available hard disk space (32-bit) or 20 GB (64-bit) for Windows 8.1.
- DirectX 9 graphics device with WDDM 1.0 or higher driver.
- 1024 × 768 minimum screen resolution.

If you want to use touch for user input, you'll need a multitouch-capable laptop, tablet, or display. A multitouch-capable device is optional for the exercises in this book, although one is useful if you want to understand what such devices are capable of. Typically, a programmer will develop software on a desktop or laptop computer and then test multitouch functionality on a multitouch-capable device.

Although this book develops applications for Windows Phone 8, a Windows Phone is not required to complete the book's step-by-step exercises.

Code samples

Most of the chapters in this book include step-by-step exercises that let you interactively try out new material learned in the main text. All sample projects can be downloaded from the following page:

http://aka.ms/VB2013_SbS/files

Follow the instructions to download the Visual_Basic_2013_SBS_Sample_Code.zip file.

Installing the code samples

Follow these steps to install the code samples on your computer so that you can use them with the exercises in this book:

1. Unzip the Visual_Basic_2013_SBS_Sample_Code.zip file that you downloaded from the book's website. (Name a specific directory along with directions to create it, if necessary.) I recommend My Documents\Visual Basic 2013 SBS for the files.
2. If prompted, review the displayed end user license agreement. If you accept the terms, select the accept option, and then click Next.

Using the code samples

The code samples .zip file for this book creates a folder named Visual Basic 2013 SBS that contains 19 subfolders—one for each of the chapters in the book that have exercises. To find the examples associated with a particular chapter, open the appropriate chapter folder. You'll find the examples for that chapter in separate subfolders. The subfolder names have the same names as the examples in the book. For example, you'll find an example called Music Trivia in the My Documents\Visual Basic 2013 SBS\Chapter 02 folder on your hard drive. If your system is configured to display file extensions of the Visual Basic project files, look for .sln as the file extension. Depending on how your system is configured, you might see a Documents folder rather than a My Documents folder.

Acknowledgments

This book is a very substantial revision of an earlier Visual Basic Step by Step book published by Microsoft Press. In fact, in almost every way, it is an entirely new book, and it is the first programming title that I have written specifically to be a multiplatform guidebook, covering Visual Basic development on the Windows Store, Windows Forms, Web Forms, and Windows Phone platforms. I am very grateful to the many talented programmers and editors who offered their ideas and contributions to this volume.

At Microsoft Press, I would like to thank Devon Musgrave for his early enthusiasm for the project and for connecting me to team members in the Visual Studio product group. At O'Reilly Media, I would like to thank again Russell Jones, who discussed many of the topics in this book with me and offered technical and practical suggestions for completing the work on schedule. I am also grateful to Tim Patrick, a technical reviewer and experienced author and developer, who worked on both this Step by Step volume and the companion book, *Start Here! Learn Microsoft Visual Basic 2012*. (Perhaps we will work on a history book someday as well, Tim!)

Within the editorial group at O'Reilly Media, I would like to thank Kristen Brown, for scheduling the editorial review and answering questions about the design; and Richard Carey, for his skillful copy editing and managing all style and localization issues that arose. (It is good to work with you again, Richard!) I would also like to thank Rebecca Demarest, Kim Burton-Weisman, and Linda Weidemann for their important artistic, editorial, and technical contributions.

I am also most grateful to the Microsoft Visual Studio 2013 development team for providing me with the preview and release candidate software to work with. In addition, I would like to thank the Microsoft Windows 8.1 team for their support and offer my special thanks to the many MSDN forum contributors who asked and answered questions about Visual Basic and Windows programming.

Finally, I offer thanks and admiration to my immediate family for their continued support of my writing projects and various academic pursuits. Once again I was able to involve my son, Henry Halvorson, with the creation of electronic music and artwork, and his contributions appear in Chapters 3, 4, 5, and 9.

Errata & book support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site:

http://aka.ms/VB2013_SbS/errata

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback is our most valuable asset. Please tell us what you think of this book at:

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

You can also learn more about Michael Halvorson's books and ideas at *http://michaelhalvorsonbooks.com*.

CHAPTER 3

Creating your first Windows Store application

After completing this chapter, you will be able to

- Design the user interface for a Windows Store app.
- Use XAML controls in the Toolbox.
- Work with random numbers, digital photos, and sound effects.
- Write Visual Basic program code for an event handler.
- Create a splash screen for your Windows Store app.
- Save, test, and build a Windows Store app.

As you learned in Chapter 2, "The Visual Studio Integrated Development Environment," the Microsoft Visual Studio 2013 IDE is ready to help you build your Visual Basic applications. In this chapter, you'll dive right in and create a Visual Basic program for the Windows Store. As a complete walkthrough exercise, this chapter describes the essential steps that you will complete each time that you create a Visual Basic application in the Visual Studio 2013 IDE. In future chapters, you'll learn more about the diversity of application types that you can create with Visual Studio, including apps for the Windows Store, the Windows desktop, the console, the web, and Windows Phone. After you learn the core Visual Basic programming skills, you'll find that all of these application types have much in common.

In this chapter, you'll learn how to create a Las Vegas-style slot machine for the Windows Store. You'll design the user interface for the program with XAML controls in the Toolbox, and you'll adjust property settings and resize objects on the page with tools in the IDE. As part of the process, you'll use the *TextBlock* control to display random numbers, the *Image* control to insert a digital photograph, and the *MediaElement* control to play a sound effect when the user spins the number 7. To create the core functionality of the Windows Store app, you'll write Visual Basic program code for an event handler. Finally, you'll create a splash screen for the app, save and test the app in the IDE, and build an executable file that can be launched from the Windows Start page.

Lucky Seven: A Visual Basic app for the Windows Store

The Windows Store app that you're going to construct is Lucky Seven, a game program that simulates a lucky number slot machine. Lucky Seven has a simple user interface and can be created and compiled in just a few minutes by using Visual Studio 2013. Here's what your program will look like when it's finished:

Programming step by step

The Lucky Seven user interface contains one button, three text block objects to display lucky numbers, a digital photo depicting cash winnings, and a text block containing the title "Lucky Seven." I produced these elements by creating five visible objects on the Lucky Seven page and then changing several properties for each object. I also added a *MediaElement* control to the page, which is not visible at runtime, to play a special sound effect when the user wins the game.

After I designed the basic user interface, I added program code for the Spin button to process the user's button clicks and to display random numbers on the page. Finally, I created a splash screen for the app and prepared it for distribution by using tools in the Visual Studio IDE.

To re-create Lucky Seven, you'll follow five essential programming steps that will be the same for most of the projects that you create with Visual Studio. You'll design the user interface with Toolbox

controls, adjust important property settings, write Visual Basic code, prepare a splash screen and other required elements, test the program, and build an executable file.

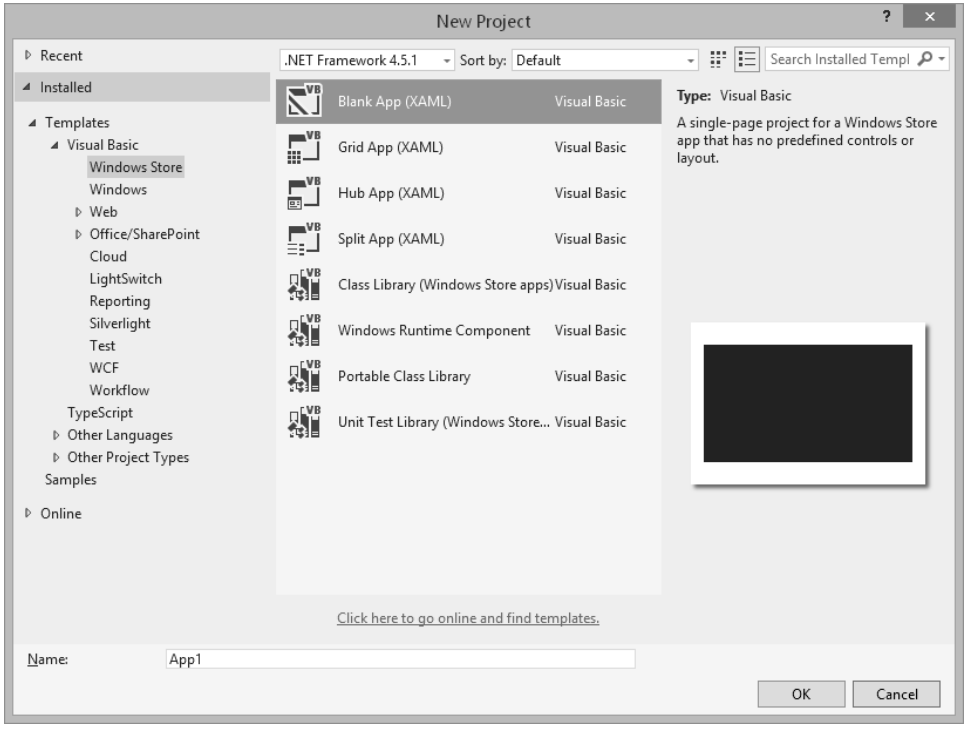
Designing the user interface

In this exercise, you'll start building Lucky Seven by first creating a new project and then using XAML controls for Windows Store apps to construct the user interface.

Create a new project

1. Start Visual Studio 2013.
2. On the Visual Studio File menu, click New Project.

   The New Project dialog box opens, as shown here:



The New Project dialog box provides access to the major template types available for creating applications with Visual Studio. On the left side of the dialog box is a list of the many template types available. Because the most recent language selection I made in this dialog box was Visual Basic, the Visual Basic templates are currently visible, but other programming templates and resources are also offered, including those for Visual C#, Visual C++, and JavaScript.

3. In the Visual Basic template group, click the Blank App (XAML) project if it is not already selected.

   When you use the Blank App template, Visual Studio will create a basic Windows Store app project with default tiles, splash screen, manifest, and startup code, but no predefined controls or layout. Note that other app types are available (which we'll get to later), including Windows (that is, Windows desktop), Web, and Windows Phone.

4. In the Name text box, type **My Lucky Seven**.

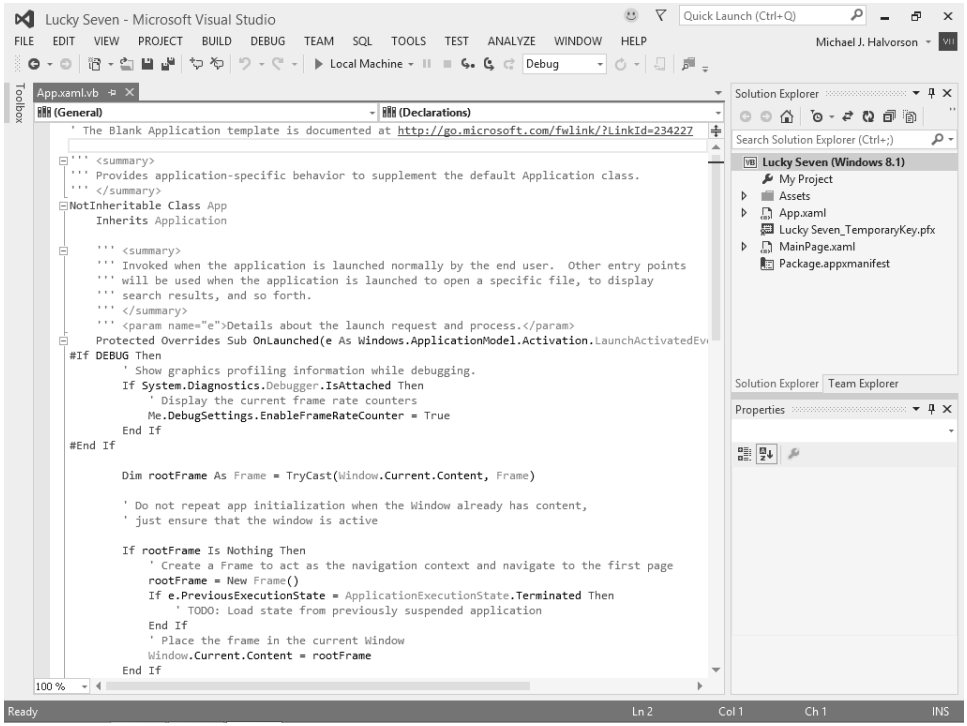
   Visual Studio assigns the name My Lucky Seven to your project. (You'll specify a folder location for the project later.)

Important I'm recommending the "My" prefix here so that you don't confuse your new app with the Lucky Seven project I've created for you on disk. However, you'll see that I don't use the "My" prefix myself in the instructions, sample projects, or screen shots in the book—I am leaving that for your use.

   If the New Project dialog box contains Location and Solution Name text boxes, you need to specify a folder location and solution name for your new programming project now. Refer to Chapter 2, in the section "Configuring the IDE for step-by-step exercises," to learn how to adjust when these text boxes appear. As I noted in Chapter 2, I will be asking you to specify a location when you first save your project—a step that is typically near the end of each exercise.

5. Click OK to create the new project in Visual Studio.

   Visual Studio prepares the IDE for a new programming project and displays Visual Basic code associated with the blank application template. Your screen will look like this:



What you see here is standard startup code for a Windows Store app created in Visual Studio 2013, and the code is stored in the file App.xaml.vb within the project. Although each project contains an App.xaml file, your work in this chapter will begin with the app's user interface, which is stored in the MainPage.xaml file.

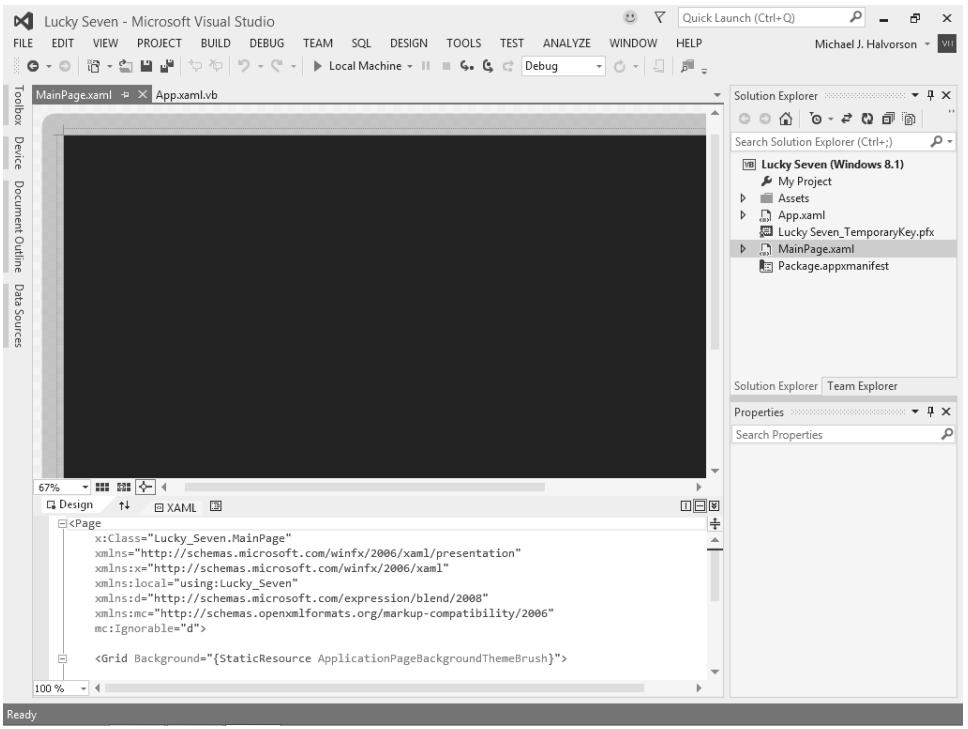
Note The section beginning *#If DEBUG Then* near the center of this illustration displays debugging information on the screen when the Windows Store app is executed in debugging mode, and it is designed for testing purposes. This code was present in the final Visual Studio 2013 software release and displays information about how long various tasks are taking during the execution of the Visual Studio app, including the frame rate for the user interface thread and how long it took (in milliseconds) to load the user interface. If you want to suppress the debugging information, remove the code between the *#If DEBUG* and *#End If* statements. For more information about the meaning of the debugging counters that appear at the top of the screen during testing, see *EnableFrameRateCounter* on *http://msdn.microsoft.com*.

You'll display that user interface now in the Designer and enhance it with Toolbox controls.

Navigate the Designer

1. Open Solution Explorer if it is not currently visible, and then double-click the file MainPage.xaml.

   Visual Studio opens MainPage.xaml in a Designer window and shows the upper-left corner of the app's main page. Below this page, you'll see the Code Editor with several lines of XAML markup associated with the user interface page in the Designer. As you add controls to the app page in the Designer, the Code Editor reflects the changes by displaying the XAML statements that will create the user interface. Your screen should look like this:



   Each time that you create a Windows Store app with Visual Basic and Visual Studio, you'll use Toolbox controls and XAML markup to design the user interface. This technique will be new to Visual Basic programmers who have primarily created Windows applications by using the technology known as Windows Forms. (You will have used the Toolbox but not XAML markup.) However, XAML will be somewhat familiar to programmers who have created Windows applications using Windows Presentation Foundation (WPF) or Windows Phone.

   Now let's review how the Designer works.

2. Click the scroll box in the Designer's vertical scroll bar, and drag it down.

   When you drag a scroll box in the Designer window, you can see more of the user interface you are working on.

3. Click the scroll box in the Designer's horizontal scroll bar, and drag it right. (Likewise, when you drag a horizontal scroll box, you can see hidden parts of the user interface.)

   Near the lower-left corner of the Designer, you'll see a Zoom tool, which allows you to zoom in on the current application page (to see more detail) or zoom out (to see more of the page). The current value of the Zoom tool is 67%. You can select a different value by clicking the Zoom tool's drop-down button.

4. Click the Zoom drop-down button, and then click Fit All.

   The entire application page now fits within the Designer. Depending on your screen resolution and the amount of screen space you have designated for the other IDE tools, you'll see a somewhat smaller version of the page.

Tip If your mouse has a mouse wheel, you can move quickly from one zoom setting to the next by holding down the Ctrl key and rotating the mouse wheel. This feature works whenever the Designer is active.

It is important to be able to quickly view different parts of the application page in different sizes while you build it. Sometimes you want to see the entire page to consider the layout of controls or other elements, and sometimes you need to view portions of the page up close. It's up to you to adjust the Designer window so that you can see the user interface clearly as you work with it.

Now set the Designer to its full-size setting.

5. Click the Zoom drop-down button, and then click 100%.

6. Adjust the Designer's vertical and horizontal scroll bars so that you can see the upper-left edge of the page.

   Seeing the edge of the page will help you orient yourself to the application window that the user sees.

   Now you'll add a Toolbox control to the page.

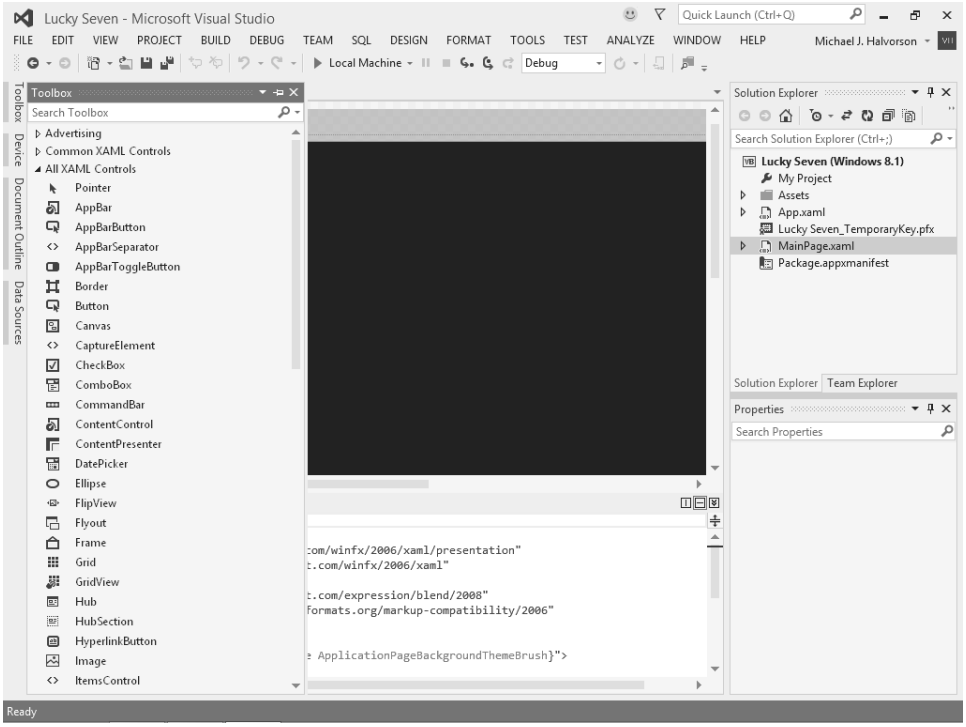
Open the Toolbox and use the *TextBlock* control

1. If the Toolbox is not currently visible, click the Toolbox tab or click the Toolbox command on the View menu.

   The Toolbox window contains a large collection of user interface controls that you can add to your application. Because you are building a Windows Store app for Windows 8.1, the types of controls that are displayed in the Toolbox are so-called XAML controls—that is, structured elements that control the look and feel of an application and can be successfully organized on a page by the XAML parser within Visual Studio.

There are also other collections of Toolbox controls for other types of applications (Windows Forms controls, HTML controls for web applications, Windows Phone controls, and so on), but you don't have to worry about that now—Visual Studio automatically loads the proper controls into the Toolbox when you open a new solution.

Your screen should look like this:



For convenience, the Toolbox controls have been organized into several groups: Advertising, Common XAML Controls (those controls that appear in many applications), and All XAML Controls (a list of all the XAML controls for Windows Store apps that are configured for use with Visual Studio).

Remember that the Toolbox window is like any other tool window in the Visual Studio IDE. You can move it, resize it, or pin it as needed. You can choose to keep the Toolbox open while you add controls to a new page (pinning it to the side of the IDE), or you can choose to use the Toolbox window's autohide feature so that the Toolbox collapses after each control has been selected.

2. Click the *TextBlock* control in the Toolbox, and move the mouse pointer to the Designer window.

   The mouse pointer changes to crosshairs. The crosshairs are designed to help you draw the rectangular shape of the *TextBlock* control on the page. You can also create a *TextBlock* with the default size by double-clicking the control in the Toolbox.

3. Click and drag to create a large rectangle-sized text block object that fills the top-left corner of the page.

   When you release the mouse button, Visual Studio creates a XAML text block object. *TextBlock* is designed to display text on your page and, in this case, can create a welcoming banner for your Windows Store app. You can update the text stored in the *TextBlock* object on your page by setting the *Text* property, either with the Properties window, XAML markup, or program code.

4. In the Properties window, change the *Text* property of the text block object to **Lucky Seven** and press Enter.

   Visual Studio displays "Lucky Seven" in the Properties window and in the Designer window. Now you'll increase the point size of the title and apply other formatting effects.

5. In the Properties window, in the *Text* category, click the Font Size text box, type **98**, and press Enter.

   The Font Size text box offers a variety of font sizes up to 72, but in this case, you're typing a larger number to create a big impact on the screen.

Tip At any time, you can delete an object and start over again by selecting the object on the page and then pressing Delete. Feel free to create and delete objects to practice creating your user interface.

6. In the Properties window, in the *Brush* category, click the *Foreground* property, if it is not already selected.

   The *Foreground* property controls the color of the text in the text block.

7. Click the Solid Color Brush button.

   The Solid Color Brush button is the second tile from the left near the top of the dialog box. (This button might also be the default selection, but it will cause no harm if you click it again.)

   When the Solid Color Brush button is selected, you'll see the Color Resources editor.

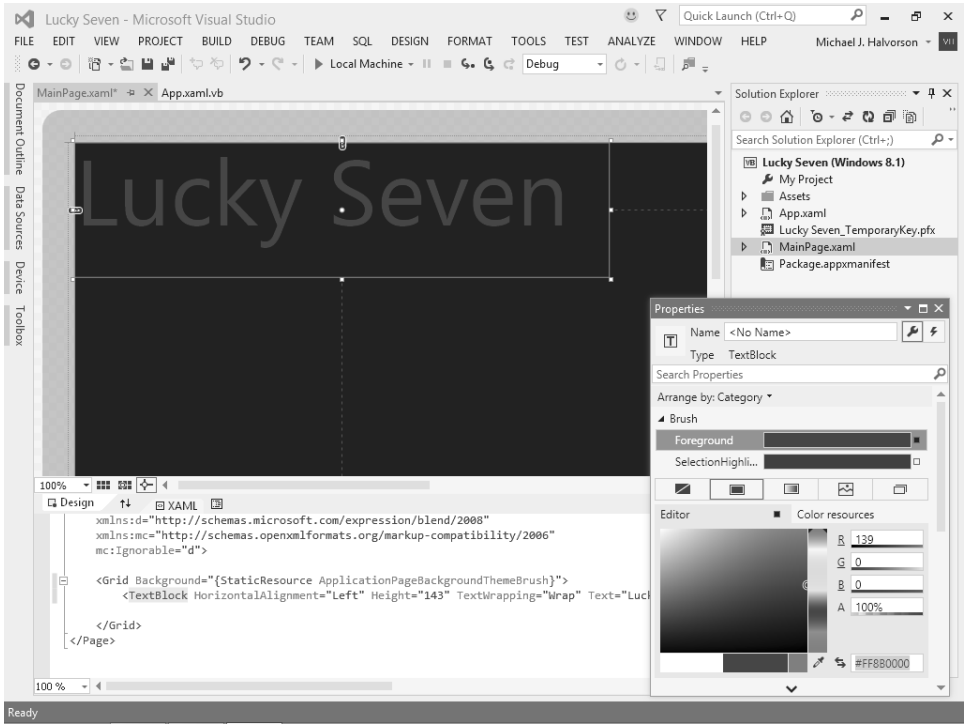
8. If you'd like more room to see the content of the Properties window, enlarge the window or configure the tool as a floating window so that you can see the Color Resources editor clearly.

9. Near the bottom of the editor, select the number containing the pound (#) sign.

   This eight-digit number is known as a hexadecimal color value—that is, a number expressed in base-16 arithmetic that specifies color by using RGBA values. When you specify a new color for text, you can specify individual values for red, green, and blue (R, G, and B), or you can use a standardized name, such as Red, DarkRed, White, Black, Purple, Lime, or Aquamarine.

10. Type **DarkRed** and press Enter.

    Note that after you press Enter in the Color Resources editor, Visual Studio converts "DarkRed" to the hexadecimal value #FF8B0000, as shown in the following screen:



11. Return the Properties window to its docked position if you moved or enlarged it.

Now you'll add three *TextBlock* controls below the Lucky Seven banner to display the randomly chosen numbers in the game. Each time that the user clicks Lucky Seven's Spin button, three new numbers will appear in these text blocks. If one of the numbers is a 7, the user wins and a sound is played.

Add text blocks for the random numbers

1. Double-click the *TextBlock* control in the Toolbox.

   Visual Studio creates a text block object on the page. In this case, the text block object is quite small, but you can resize it.

2. In the Properties window, click the *Text* category, click the *FontSize* box, type **72**, and then press Enter.

   Visual Studio expands the text block object to accommodate text in 72-point font.

3. In the Properties window, click the *Common* category, click the Text box, type **0**, and press Enter.

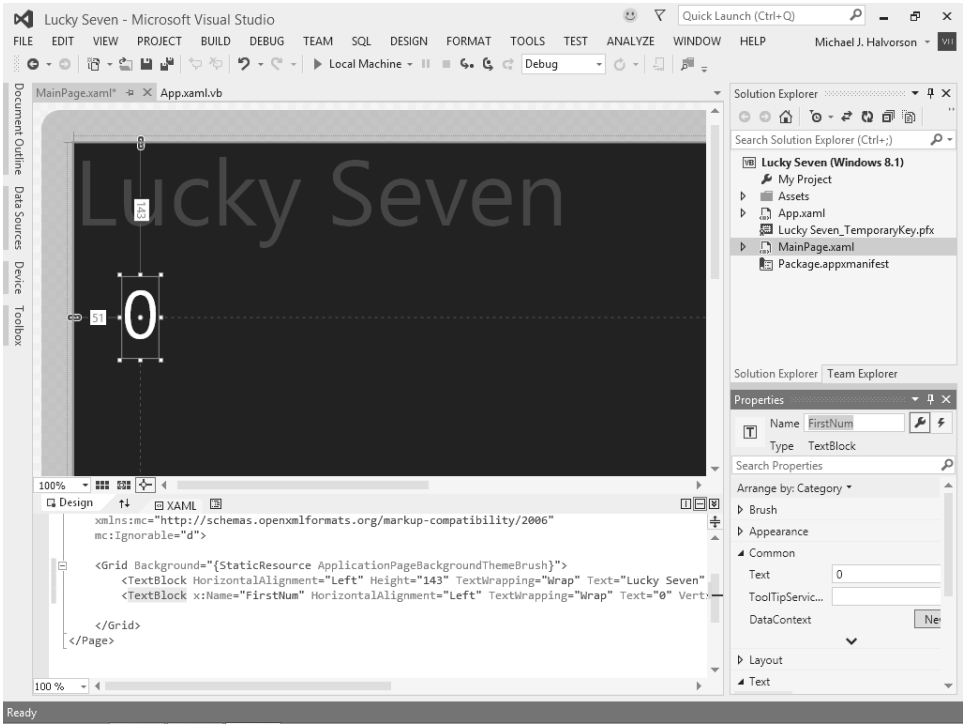
   0 will be an initial value for the first lucky number in the program.

4. At the top of the Properties window, change the *Name* property of the text block object to **FirstNum**.

   It is not required that all objects be named in your user interface, but it is important to name objects that will be referenced in program code. Because you'll be controlling the value of this lucky number in a Visual Basic event handler, you'll give it the name *FirstNum* here.

5. Drag the *FirstNum* text block object below the "u" in Lucky Seven.

   Your page should look something like this:



6. Double-click the *TextBlock* control in the Toolbox to create another text block object.

   This object will hold the second lucky number on the page.

7. Using the Properties window, set the *Name* property of the object to **SecondNum**, set the *FontSize* property to **72**, and set the *Text* property to **0**.

8. Move the new *SecondNum* object to the right of the *FirstNum* object, directly below the "y" in Lucky Seven.

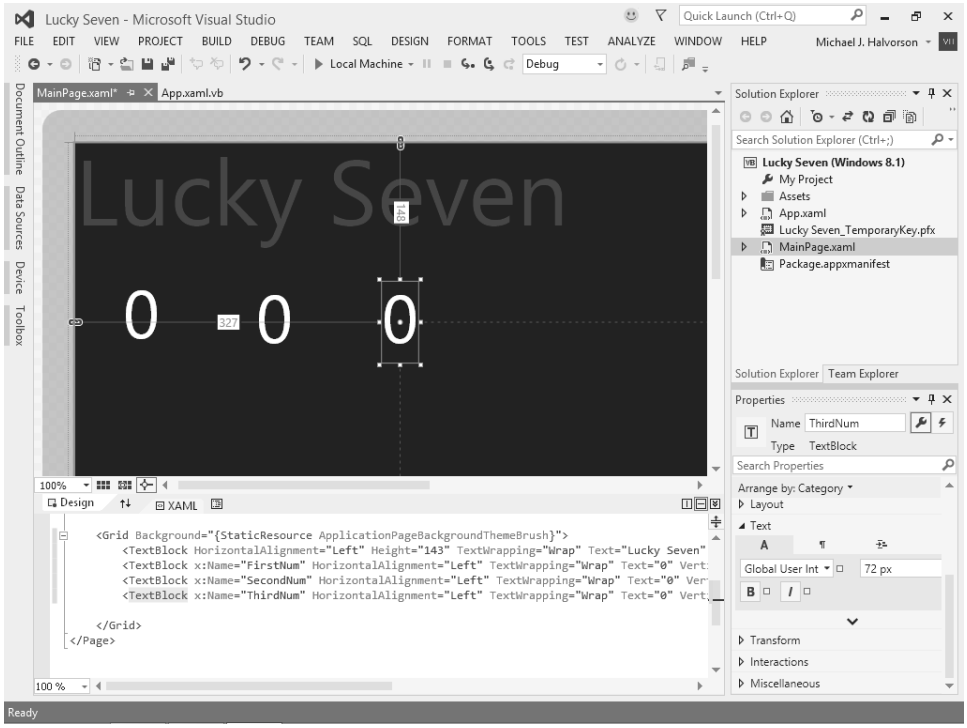
   Now you'll create the third lucky number for the page.

9. Double-click the *TextBlock* control in the Toolbox to create the last text block object.

10. Using the Properties window, set the *Name* property of the object to **ThirdNum**, set *FontSize* to **72**, and set *Text* to **0**.

11. Move the *ThirdNum* object to the right of the *SecondNum* object, directly below the first "e" in Lucky Seven.

    When you've finished, your four text block objects should look like those in this screen shot. (You can move your objects if they don't look quite right.)



Now you'll add a button control to the page.

Add a button control

1. Click the *Button* control in the Toolbox, and then move the mouse pointer over the application page.

2. Drag the pointer down and to the right. Release the mouse button to complete the button.

3. In the Properties window, in the *Common* category, change the *Content* property to **Spin** and press Enter.

   Note that a button object's contents are set via the *Content* property, rather than *Text* (like a text block object), because buttons can contain artwork and other data.

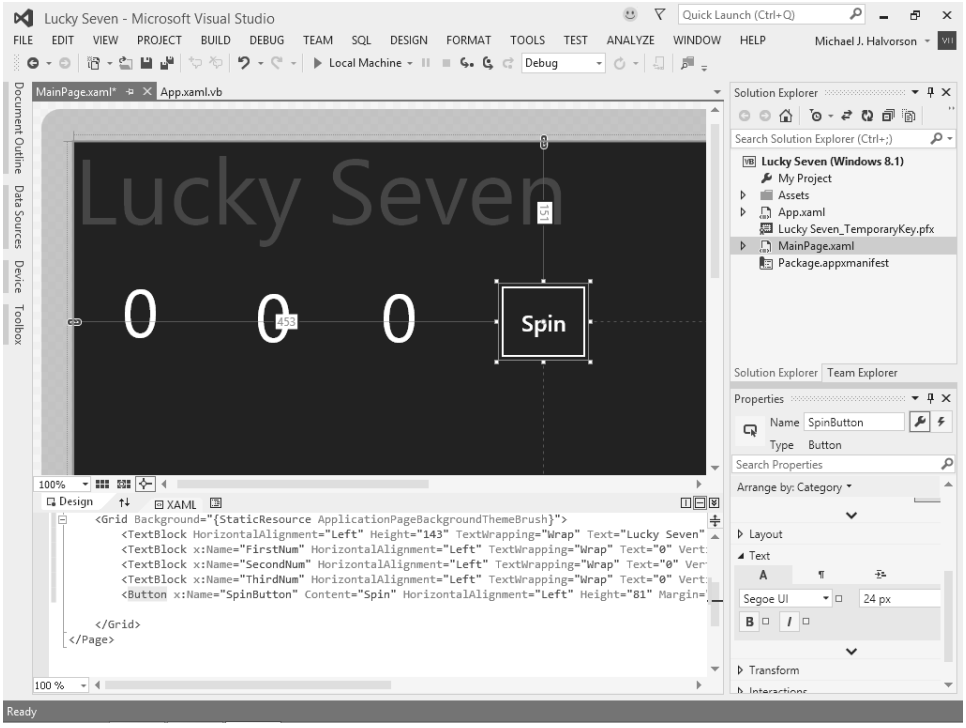
4. In the Properties window, change the button object's *Name* property to **SpinButton**.

5. In the Properties window, in the *Text* category, change the *FontSize* property to **24**.

6. Resize the *SpinButton* object so that it is 81 pixels high and 95 pixels wide.

7. Move the button object so that it is to the right of the third lucky number on the page. Snap lines will appear again as you move the object, and the top edge of the button will snap to the top edge of the three numbers when aligned.

   Your screen should look like this:



Now you'll add an image to the page to graphically display the payout you'll receive when you draw a 7 and hit the jackpot. An *Image* control is designed to display bitmaps, icons, digital photos, and other artwork—a major design feature of most Windows Store apps. One of the most common uses for an *Image* control is to display a PNG or JPEG file.

Add an image

1. Click the *Image* control in the Toolbox.
2. Using the control's drawing pointer, create a large rectangular box below the lucky numbers and the Spin button on the page.
3. If necessary, adjust the Zoom setting in the Designer window so that you can see more of the page in the Designer. For example, a Zoom setting of 50% might be useful.

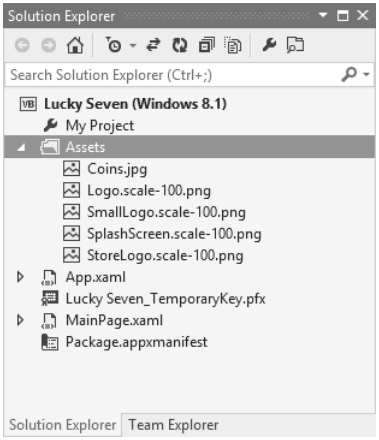
   It would be good if the image object covered most of the remaining area of the page below the numbers and the Spin button. Sometimes it is useful to reduce the size of a page in the Designer with the Zoom control to make these types of operations easier.

   Now you'll add a suitable photo to the project by using Solution Explorer and the Assets folder, a special container for resource files in your project.
4. If Solution Explorer is not visible now, open it by clicking Solution Explorer on the View menu.

   As you've already learned, Solution Explorer provides access to most of the files in your project, and prominently listed in Solution Explorer is the Assets folder, a container for your project's logo, splash screen, and other files. You'll add a digital photo to the Assets folder in the following step, which will make it available to your program.
5. Right-click the Assets folder in Solution Explorer to display a shortcut menu of useful Visual Studio commands.
6. Point to the Add command, and then click Existing Item.
7. In the Add Existing Item dialog box, browse to the My Documents\Visual Basic 2013 SBS\Chapter 03 folder and click Coins.jpg, a JPEG file containing coins from around the world—a visual representation of winnings in the Lucky Seven app.

8. Click Add to add the photo to your project in the Assets folder.

   Visual Studio inserts the file, and it appears now in Solution Explorer under Assets, as shown in the following illustration:



   When a file has been added to the Assets folder, it becomes part of the project you are working on, and it can be referenced via the Properties window. Most importantly, it becomes part of the project when the project is compiled for distribution—there is no need to remember where the file was originally located on your hard disk, because a copy will now travel with the project.

9. Select the image object (if it is not already selected) so that its properties are visible in the Properties windows.

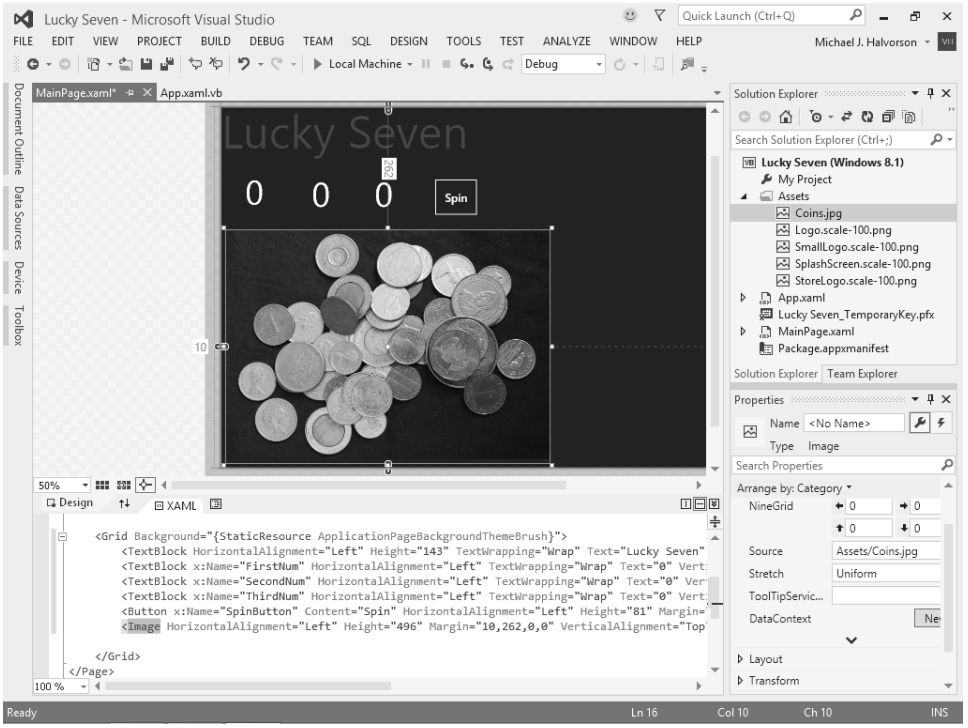
10. In the Properties window, in the Common category, click the *Source* text box, and then click Coins.jpg.

    You might need to expand the Properties window a little to see the drop-down list box arrow in the Source text box.

    After the file has been selected, a photo of coins from around the world fills the image object in the Designer.

11. Adjust the spacing of the image so that it takes up much of the left side of the page in the Designer.

    When you've finished, your page should look like this:



12. In the Properties window, change the *Name* property of the image object to **CoinImage**.

    Naming the image object is an important step, because you'll be referring to this object in Visual Basic code. Often you'll see me include the name of the control at the end of an object name so that its object type is clear.

Now you'll add a sound effect to the program so that the game plays a sound when the user spins a 7. You'll add this sound effect with the *MediaElement* control, which plays audio and video files in a Windows Store app. The sound you'll play is stored in a short WAV file named ArcadeRiff, created by Henry Halvorson.

Play audio media with the *MediaElement* control

1. In the Toolbox, expand the All XAML Controls category and double-click the *MediaElement* control.

   Visual Studio places a new media player object in the upper-left corner of the page. Like other new objects in the Designer, you can now move the object to a new location and customize it with property settings. However, the *MediaElement* control is essentially a behind-the-scenes tool; it is not visible to the user unless the control is displaying a video clip. For now, you can leave the media element object where it is.

   The *Source* property of the *MediaElement* control specifies the name of the media file that will be loaded into the control for playback. Before you can assign this property, you need to add a valid media file to the Assets folder, just as you did for the image control.

2. Right-click the Assets folder in Solution Explorer to display the shortcut menu.

3. Point to the Add command, and then click Existing Item.

4. In the Add Existing Item dialog box, browse to the My Documents\Visual Basic 2013 SBS\Chapter 03 folder and click ArcadeRiff.wav.

5. Click Add to add the music file to your project in the Assets folder.

   Visual Studio inserts the file, and it appears now in Solution Explorer under Assets.

   Now you're ready to name the media element object and assign it a music asset by using the *Source* property.

6. Click the media element object in the Designer window. (Zoom in on the Designer if necessary—remember that the object is invisible but it can be selected. You can always find it by clicking the *MediaElement* entry in the XAML tab of the Code Editor.)

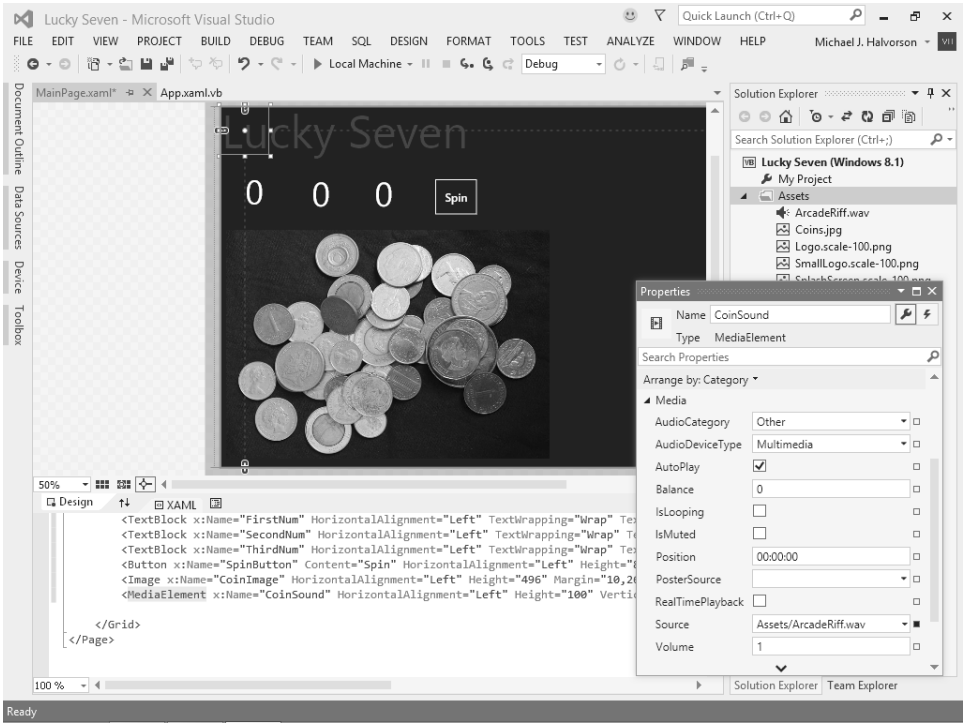
7. In the Properties window, change the *Name* property to **CoinSound**.

8. Expand the *Media* category, scroll down to the *Source* property, and click the Source list box.

   Your new media file (ArcadeRiff.wav) appears in the list.

   Click the ArcadeRiff.wav file to link it to the *CoinSound* object.

   Your screen will look like this (notice the entries in Solution Explorer and the Properties window):



The Properties window exposes a few other important media element properties that you can examine and adjust if desired.

For example, the AutoPlay check box is enabled by default, which directs the media control to automatically play the specified media file when the page loads. Because you don't want the sound to play until it is needed, disable that now.

9. Remove the check mark from the AutoPlay check box.

There are some other options you might notice now (but not adjust). The *Position* property specifies the location within the media file where playback will begin; this option is very useful if there is a specific place in the song or video where you want to start.

The *IsLooping* property is a Boolean value that allows you to run the media file over and over again if you like. Finally, *Volume* allows you to set an initial volume level for the media playback, which you can adjust with property settings in an event handler while the program is running.

Final property settings and adjustments

Your Lucky Seven page is almost complete. You just need to make a few final property settings, write the Visual Basic code, and design a splash screen that runs when your project starts.

Before you begin these tasks, let's think a little more specifically about how the program will operate when it runs. The game starts when the user opens the program and clicks the Spin button. When the Spin button is clicked, the app generates three random numbers and displays them in text block objects on the page. If and when the player hits the jackpot (that is, when at least one 7 appears in the text block objects), the object containing the photo of coins appears, and then the media element control plays a "celebration" sound.

Although the flow of events is pretty straightforward, the program needs to continue operating after the first "win." So, when the user clicks the Spin button, the coins image needs to disappear and remain hidden until another 7 appears, at which point the image is displayed again and the sound effect also run.

To get this behavior to work correctly, you need to find a mechanism to make the image object visible and invisible when you want. That can be accomplished by setting the image object's *Visibility* property, which is assigned *Visible* or *Collapsed* (invisible) values as needed. In fact, most objects in a Windows Store app can be made visible or invisible if you set this property—it is a built-in tool to control what appears on the screen. Give it a try here.

Set the *Visibility* property

1. Click the image object on the page.
2. In the Properties window, click the *Appearance* category, and then click the *Visibility* property.
3. In the drop-down list box that appears, click the *Collapsed* property.

   The image object on the page disappears. Don't worry—this is the desired effect. The object is not gone, it is just currently invisible. You'll make it reappear by using program code in an event handler.

Now you'll adjust the background color for the page. The default color value for Windows Store apps is Black, but a more colorful value can make the game more appealing. You can adjust this color by selecting the *Grid* object on the page and adjusting values in the *Brush* category by using the Properties window.

Set the page's background color

1. Select the *Grid* object by clicking the background page in the Designer (not one of the objects that you've just added).

   You can tell when you've selected the *Grid* object because its properties will fill the Properties window.

   As you'll learn in Chapter 7, "XAML markup step by step," each of the objects in a Windows Store app is defined by XAML markup codes and data that can be entered or adjusted in the Code Editor. The *Grid* object is the base layout element for a page, and all of the elements on a page are nested within this *Grid* object. In addition to serving as a useful container for objects, the *Grid* object also has settings that you can adjust, such as the background color that appears for your app. You'll set this now.

2. Click the *Brush* category, click the *Background* property, and then click the Solid Color Brush button.

3. Near the bottom of the Color Resources editor, select the number containing the pound (#) sign, replace the contents with **Green**, and press Enter.

   The alphanumeric value for green (#FF0080000) appears in the text box, and the background color of the *Grid* object changes to green. Feel free to experiment with other color values if you like.

   OK—that's it for the user interface design walkthrough. Save your work now, before you write the program code.

Save changes

1. Click the Save All command on the File menu to save all your additions to the Lucky Seven project.

   The Save All command saves everything in your project—the project file, the pages, the code-behind files, the assets, the package manifest, and other related components in your application. Because this is the first time that you have saved your project, the Save Project dialog box opens, prompting you for the name and location of the project. (If your copy of Visual Studio is configured to prompt you for a location when you first create your project, you won't see the Save Project dialog box now—Visual Studio just saves your changes.)

2. Browse and select a location for your files. I recommend that you use the My Documents\Visual Basic 2013 SBS\Chapter 03 folder (the location of the book's sample files), but the location is up to you. Because you used the "My" prefix when you originally opened your project, this version won't overwrite the practice file that I built for you on disk.

3. Clear the Create Directory For Solution check box.

   When this check box is selected, it creates a second folder for your program's solution files, which is not necessary for solutions that contain only one project (the situation for most programs in this book).

4. Click Save to save your files.

Tip If you want to save just the item you are currently working on (the page, the code module, or something else), you can use the Save command on the File menu. If you want to save the current item with a different name, you can use the Save As command.

Writing the code

Now you're ready to write the code for the Lucky Seven program. Because most of the objects you've created already "know" how to work when the program runs, they're ready to receive input from the user and process it. The inherent functionality of objects is one of the great strengths of Visual Studio and Visual Basic—after objects are placed on a page and their properties are set, they're ready to run without any additional programming.

However, the "meat" of the Lucky Seven game—the code that actually calculates random numbers, displays them in boxes, and detects a jackpot—is still missing from the program. This computing logic can be built into this Windows Store app only by using program statements—code that clearly spells out what the program should do at each step of the way. Because the Spin button drives the program, you'll associate the code for the game with an event handler designed for that button.

In the following steps, you'll enter the Visual Basic code for Lucky Seven in the Code Editor.

Use the Code Editor

1. In the Visual Studio Designer, click the *SpinButton* object.
2. Open the Properties window, and close the *Brush* category.
3. Near the top of the Properties window and to the right of the *Name* property and the Properties button, click the Event Handler button (a square button displaying a lightning bolt icon).

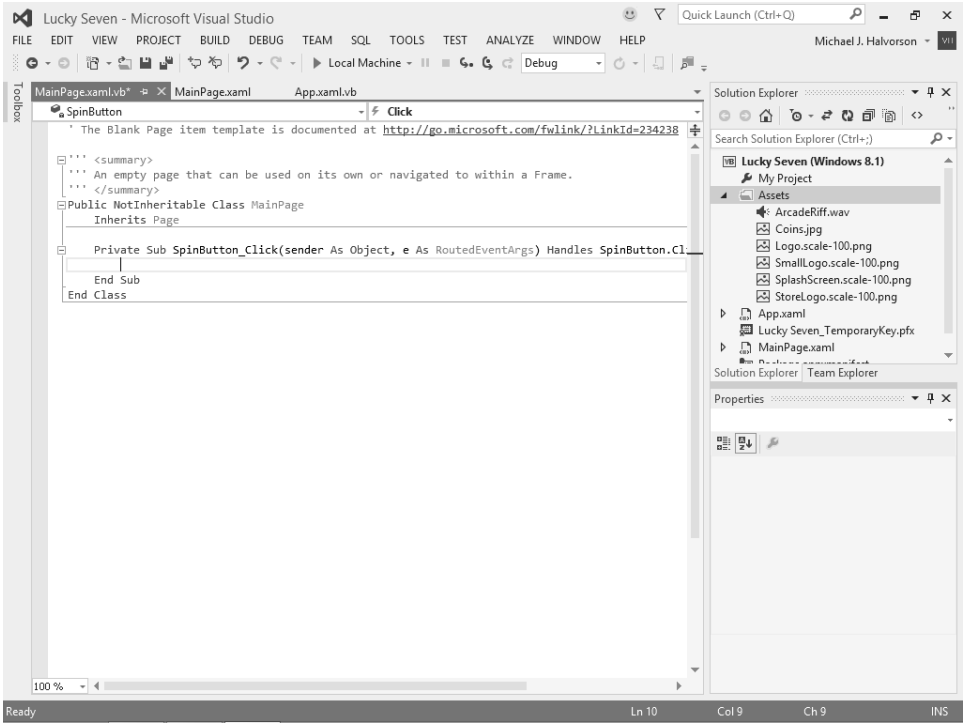
   A collection of actions or events that a button object can respond to fills the Properties window. Typical events that a button might recognize include *Click* (a mouse click), *DragOver* (an object being dragged over a button), Tapped (a button being touched by a finger), and *Drop* (an object being dragged over and dropped on a button).

Because Visual Basic is, at its core, an event-driven programming language, much of what you do as a software developer is create user interfaces that respond to various types of input from the user, and then you write event handlers that manage the input. Most of the time, you will need to write event handlers only for a few events associated with the objects in your programs. (However, the list of events is quite comprehensive to give you many options.)

To create an event handler for a particular event, you double-click the text box next to the event in the Properties window. Because you want to generate three random numbers each time that the user clicks the Spin button in your program, you'll write an event handler for the button's *Click* event.

4. Double-click the text box next to the *Click* event in the Properties window.

Visual Studio inserts an event handler named *SpinButton_Click* in the Click text box, and opens the MainPage.xaml.vb code-behind file in the Code Editor. Your screen should look like this:



Inside the Code Editor are program statements associated with the *MainPage* template that you opened when you started this project. This is Visual Basic program code, and you might notice right away that some of the code is organized into concise units, known as *procedures*. Near the bottom of the file is a new event handler procedure that you just created, called *SpinButton_Click*.

The *Sub* and *End Sub* keywords designate a procedure, and the keywords *Protected* and *Private* indicate how the procedure will be used. You'll learn more about these keywords later.

When you double-clicked the Click text box in the Properties window, Visual Studio automatically added the first and last lines of the *SpinButton_Click* event procedure, as the following code shows. (Your event procedure will not wrap as this one does. In print, I need to respect the book's margins.)

```
Private Sub SpinButton_Click(sender As Object, e As RoutedEventArgs) Handles SpinButton_
Click

End Sub
```

The body of a procedure fits between these lines and is executed whenever a user activates the interface element associated with the procedure. In this case, the event is a mouse click, but as you'll see later in the book, it could also be a different type of event. Programmers refer to this sequence as "triggering" or "firing" an event.

Tip You might also notice lines of text with green type in the Code Editor. In the default settings, green type indicates that the text is a *comment*, or an explanatory note written by the creator of the program, so that it might be better understood or used by others. The Visual Basic compiler does not execute, or *evaluate*, program comments.

5. Type the following program code, and press the Enter key after the last line:

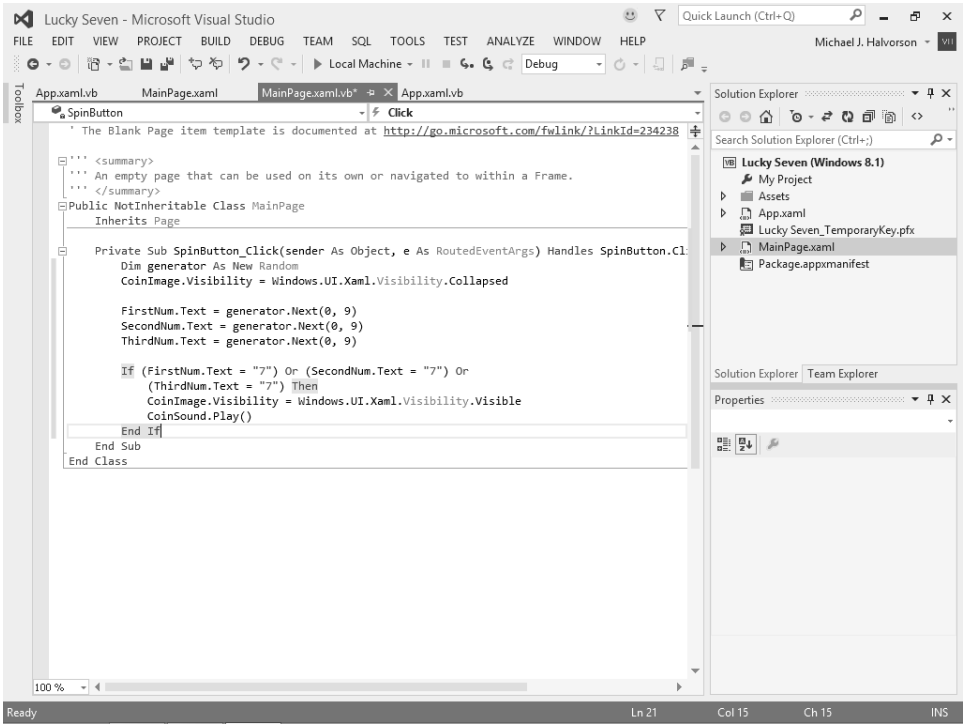
```
Dim generator As New Random
CoinImage.Visibility = Windows.UI.Xaml.Visibility.Collapsed

FirstNum.Text = generator.Next(0, 9)
SecondNum.Text = generator.Next(0, 9)
ThirdNum.Text = generator.Next(0, 9)

If (FirstNum.Text = "7") Or (SecondNum.Text = "7") Or
    (ThirdNum.Text = "7") Then
    CoinImage.Visibility = Windows.UI.Xaml.Visibility.Visible
    CoinSound.Play()
End If
```

As you enter the program code, Visual Studio formats the text and displays different parts of the code in color to help you identify the various elements. When you begin to type the name of an object property, Visual Basic also displays the available properties for the object that you're using in a list box, so you can click the property or keep typing to enter it yourself.

Your screen should now look like this:



Note If Visual Basic displays an additional error message, you might have misspelled a program statement. Check the offending line against the text in this book, make the necessary correction, and continue typing. (You can also delete a line and type it again from scratch.)

In Visual Studio, program statements can be composed of keywords, properties, object names, variables, numbers, special symbols, and other values. As you enter these items in the Code Editor, Visual Studio uses a feature known as IntelliSense to help you write the code. With IntelliSense, as Visual Studio recognizes language elements, it will automatically complete many expressions.

6. Click the Save All button to save your changes.

A look at the *SpinButton_Click* event handler

The *SpinButton_Click* event handler is executed when the user clicks the Spin button on the page. Essentially, the event handler performs four main tasks:

1. It declares a random number generator named *generator* in the program.
2. It hides the digital photo.
3. It creates three random numbers and displays them in text block objects.
4. It displays the Coins.jpg photo and plays a sound when the number *7* appears.

Let's look at each of these steps individually.

The random number generator is declared by this line of code:

```
Dim generator As New Random
```

You've probably declared and used variables before in programs. But notice the variable type here—the generator is declared using the type *Random*, which has been specifically designed to support the creation of so-called "pseudo-random" numbers—that is, numbers that don't follow a particular pattern and appear in a specific range. You'll use random numbers often in this book, and you'll learn much more about data types and conversion in Chapter 11, "Mastering data types, operators, and string processing."

Hiding the photo is accomplished by the following line:

```
CoinImage.Visibility = Windows.UI.Xaml.Visibility.Collapsed
```

As you learned earlier, the *Visibility* property determines whether or not an object on a page is visible. This specific syntax uses the objects in the .NET Framework to collapse (or hide) the photo of the coins. (This line is designed to restore the program to a neutral state if a previous spin had displayed the coins.)

The next three lines handle the random number computations. Does this concept sound strange? You can actually make Visual Basic generate unpredictable numbers within specific guidelines—that is, you can create random numbers for lottery contests, dice games, or other statistical patterns. The *generator* instance's *Next* method in each line creates a random number between 0 and 9—just what you need for this particular slot machine application.

```
FirstNum.Text = generator.Next(0, 10)
SecondNum.Text = generator.Next(0, 10)
ThirdNum.Text = generator.Next(0, 10)
```

The last group of statements in the program checks whether any of the random numbers is 7. If one or more of them is, the program displays the graphical depiction of a payout and plays the sound effect to announce the winnings.

```
If (FirstNum.Text = "7") Or (SecondNum.Text = "7") Or
    (ThirdNum.Text = "7") Then
    CoinImage.Visibility = Windows.UI.Xaml.Visibility.Visible
    CoinSound.Play()
End If
```

Each time the user clicks the Spin button, the *SpinButton_Click* event handler is executed, or called, and the program statements in the handler are run again. However, if you click the Spin button many times in rapid succession, you might miss one or more of the sound effects, because the media element object can play only one sound effect at a time.

Running Windows Store apps

Congratulations! You're ready to run your first Windows Store app. To run a Visual Basic program from the IDE, you can do any of the following:

- Click Start Debugging on the Debug menu.
- Click the Start Debugging button on the Standard toolbar. (You'll typically see "Local Machine" next to this button, because you debug on the local computer by default.)
- Press F5.

Try running your Lucky Seven program now. If Visual Basic displays an error message, you might have a typing mistake or two in your program code. Try to fix it by comparing the printed version in this book with the one you typed, or load Lucky Seven from your hard disk and run it.

Note I assume that you have named your project My Lucky Seven, but the instructions and screen shots below will show Lucky Seven because you might be running the sample project that I created.

Run the Lucky Seven program

1. Click the Start Debugging button on the Standard toolbar.

   The Lucky Seven program compiles and runs. After a few seconds, the user interface appears, just as you designed it.

2. Click the Spin button.

   The program picks three random numbers and displays them in the labels on the page. When a 7 appears, your screen will look like this:

The presence of a 7 also triggers the sound effect, which lasts a few seconds and sounds a bit like an electronic slot machine. You win!

3. Click the Spin button 15 or 16 more times, watching the results of the spins in the number text blocks.

   About half the time you spin, you hit the jackpot—pretty easy odds. (The actual odds are about 2.8 times out of 10; you're just lucky at first.) Later on, you might want to make the game tougher by displaying the photo only when two or three 7s appear, or by creating a running total of winnings.

4. When you've finished experimenting with your new creation, close the Windows Store app.

   The program stops, and the IDE reappears on your screen. Click the Stop Debugging button on the toolbar to end the program. Now you'll add a splash screen to the project.

Creating a splash screen for your app

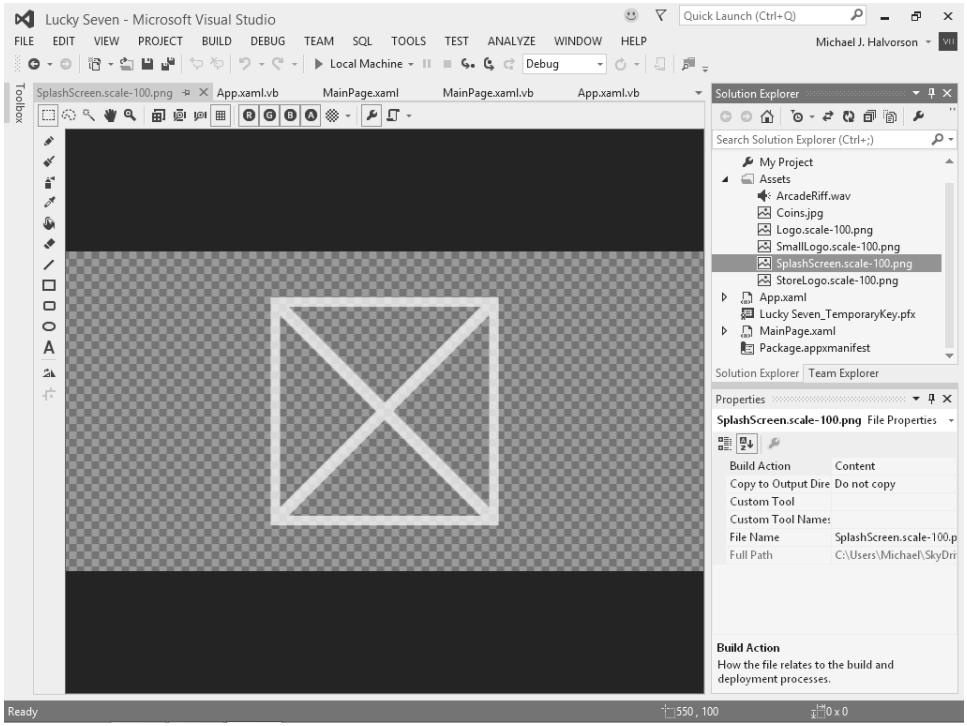
A *splash screen* is a transitional image that appears when your app first launches. Every Windows Store app must have a splash screen, which consists of an image (or text) and a surrounding background color. The splash screen is stored in the Assets folder within Solution Explorer, and every new Windows Store app has a basic splash screen that is created by default. You'll also see tile images in the Assets folder, which you'll learn to customize in Chapter 9, "Exploring Windows 8.1 design features: Command bar, flyout, tiles, and touch."

Although you can create a splash screen with Microsoft Paint or another third-party graphics program, you can also create a simple splash screen within Visual Studio. Just remember that a splash screen appears very briefly when you first launch your app. Accordingly, this is not the place to put elaborate program instructions or copyright information. You'll want to avoid placing advertisements or version information on a splash screen.

Instead, use the splash screen to offer a preview of the functionality of your app in some unique way. Consider an image or photo that will be easily adapted to other countries and cultures (that is, easily *localizable*) and that can be displayed effectively in different screen resolutions. Notice that Portable Network Graphics (.png) format is used because this file type is capable of displaying alpha transparency and 24-bit color images. When part of an image is formatted as transparent, the background color will be displayed behind it. (You'll see this in most splash screens and tiles in Windows Store and Windows Phone apps.)

Create a Lucky Seven splash screen

1. In Solution Explorer, open the Assets folder, and then double-click the file SplashScreen.scale-100.png.
2. This action opens the Image Editor Designer in Visual Studio, and loads the SplashScreen.scale-100.png file into the editor. Your screen looks like this:



The Solution Explorer and Properties windows are still visible. However, the Image Editor is active, and the design canvas is surrounded by graphics editing tools. The "X" shape in the center of the canvas is simply the default image for the SplashScreen.scale-100.png file. This is the image that you want to replace now.

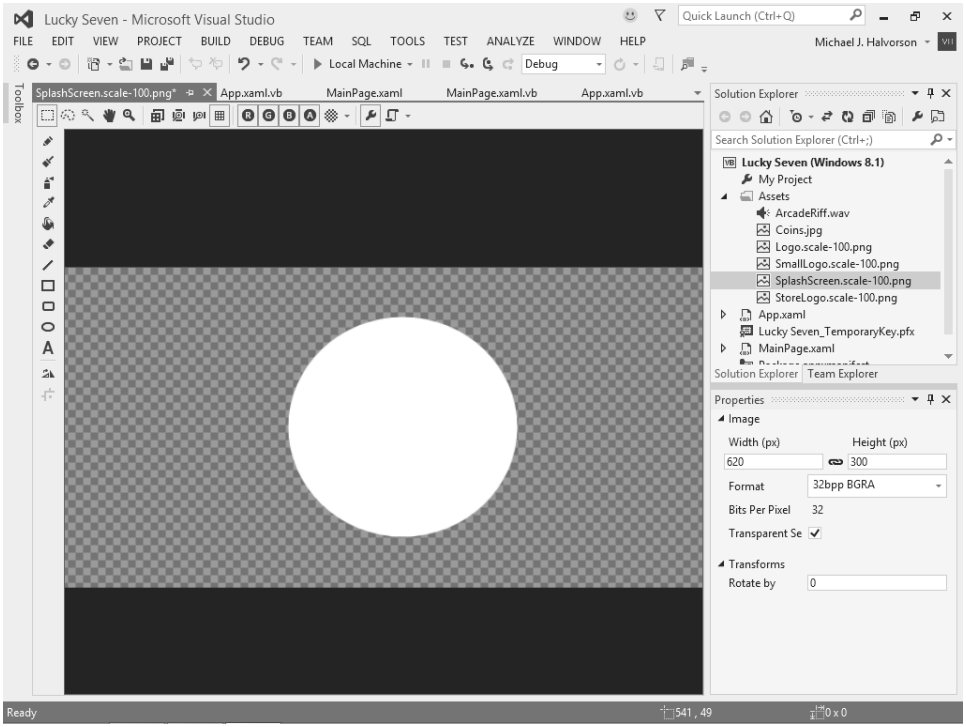
3. Click the Selection tool in the upper-left corner of the Image Editor, select the entire "X" shape, and press Delete.

   You now have a blank canvas on which to create your splash screen image. The alpha checkerboard pattern that you see is a color scheme that allows you to more easily see the transparent portions of your image—that is, what you see displayed as the checkerboard now will be replaced by the background when your splash screen is actually displayed on the screen.

4. Click the Ellipse tool on the left side of the Designer, and then create a circle shape in the middle of the splash screen.

   You can use the X- and Y-axis indicators in the lower-right corner of the screen to create your circle if you like. You can also use the Selection tool to move your shape to the center of the screen if you like.

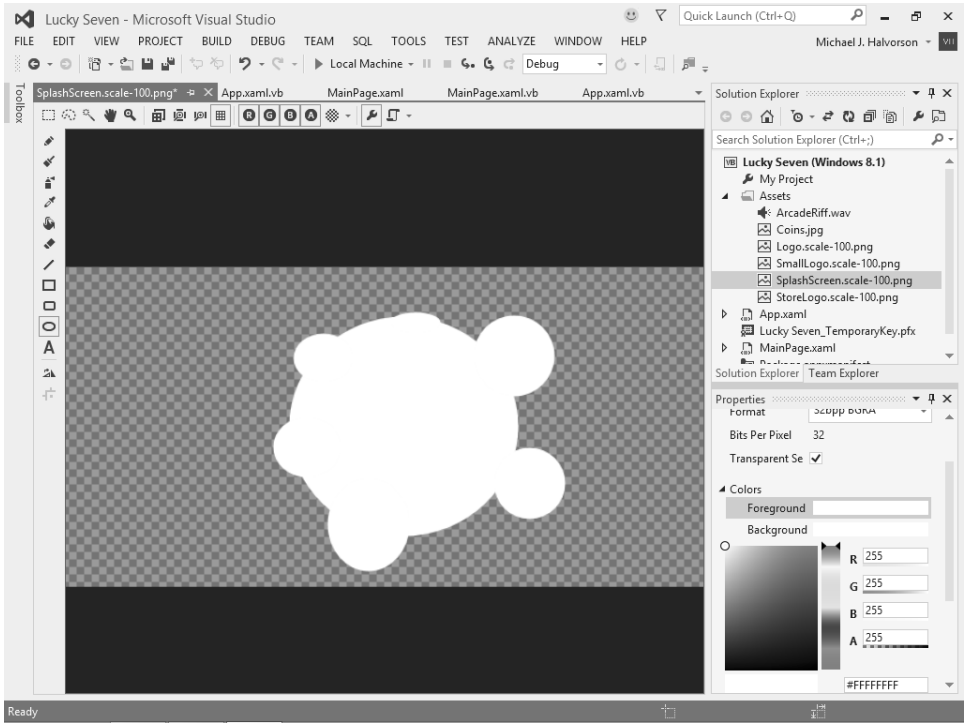
Your Image Editor will look like this:



5. Use the Ellipse tool to add four or five smaller circles around the edge of the circle that you have created.

   Typical splash screens show simple geometric shapes like this. Consider using a simplified version of your company logo.

   Your simple splash screen now looks like this:



You could add additional effects to this splash screen, embellishing it with colors, images, text, or animation. However, for this first walkthrough, you have something that will work just fine.

6. Click the Save All command on the File menu to save your changes.

7. Press F5 to run the project, and examine your splash screen.

   Notice that the splash screen comes and goes in just a few moments. Did you notice the ellipse shapes and the black background color?

8. Close the program, and then close the Image Editor Designer.

Now your project is complete—it is time to test and deploy the app by adding it to the Windows Start page on your local computer. However, note that if this were a commercial Windows Store app being prepared for distribution to other users via the Windows Store, you would now add additional items to your app as described in Table 1-1. For more information, see Chapter 1, "Visual Basic 2013 development opportunities and the Windows Store."

Sample projects on disk

If you didn't build the My Lucky Seven project from scratch (or if you did build the project and want to compare what you created to what I built for you as I wrote the chapter), take a moment to open and run the completed Lucky Seven project, which is located in the Visual Basic 2013 SBS\Chapter 03 folder on your hard disk (the default location for the practice files for this chapter). If you need a refresher course on opening projects, see the detailed instructions in Chapter 2.

This book is a step-by-step tutorial, so you will benefit most from building the projects on your own and experimenting with them. But after you have completed the projects, it is often a good idea to compare what you have with the practice file "solution" that I provide, especially if you have unexpected results. To make this easy, I will give you the name of the solution files on disk before you run the completed program in most of the step-by-step exercises.

After you have compared the My Lucky Seven project to the Lucky Seven solution files on disk, reopen My Lucky Seven and prepare to compile it as an executable file. If you didn't create My Lucky Seven, use my solution file to complete the exercise.

Building an executable file

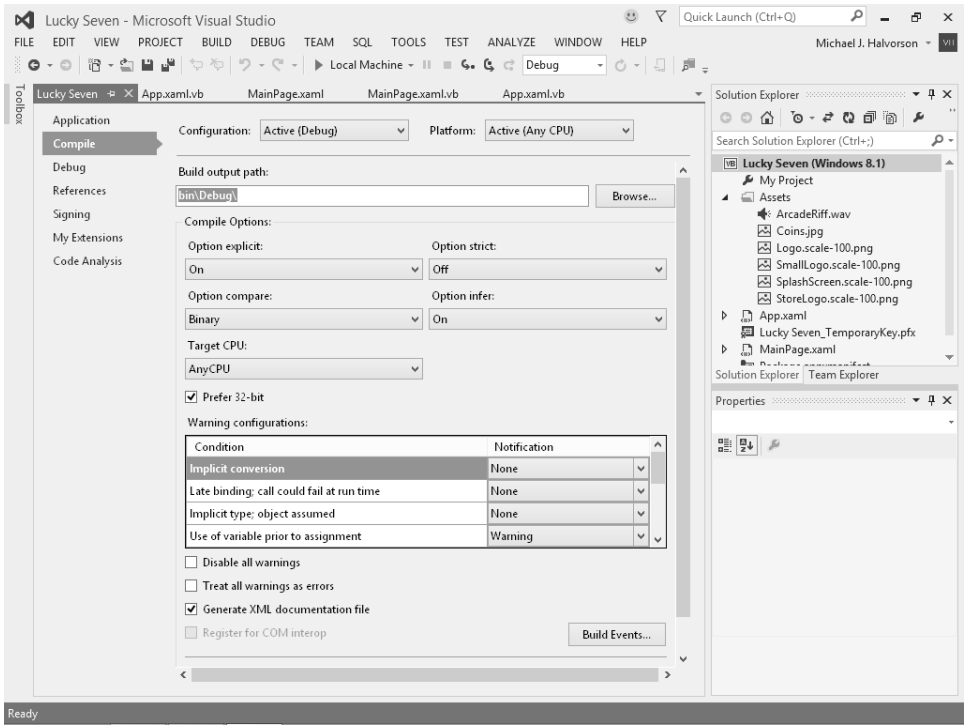
Your last task in this chapter is to complete the development process and create an application for Windows, or an *executable file*. Windows applications created with Visual Studio have the file name extension .exe and can be run on any system that contains Windows and the necessary support files. If you end up distributing your application via the Windows Store, the complete deployment package will be posted securely in the Store and made available to customers who would like to download it. However, you can also deploy your application to individual computers running Windows directly from within Visual Studio.

Because you just created a Windows Store app that targets the Windows 8.1 operating system, you need to be running Windows 8.1 to run this particular program. You won't post the sample app to the Windows Store yet, because it has not been registered or thoroughly tested. But you can deploy the app on your own computer, which does not have as many registration requirements as the Windows Store interface.

To assist in the testing and compilation process, Visual Studio allows you to create two types of executable files for your Windows application project: a *debug build* and a *release build*.

Debug builds are created automatically by Visual Studio when you create and test your program. They are stored in a folder called bin\Debug within your project folder. The debug executable file contains debugging information that makes the program run slightly slower.

Release builds are optimized executable files stored in the bin\Release folder within your project. To customize the settings for your release build, you click the *ProjectName* Properties command on the Project menu, and then click the Compile tab, where you'll see a list of compilation options that looks like the following screen. The Solution Configurations drop-down list box on the Standard Visual Studio toolbar indicates whether the executable is a debug build or a release build.



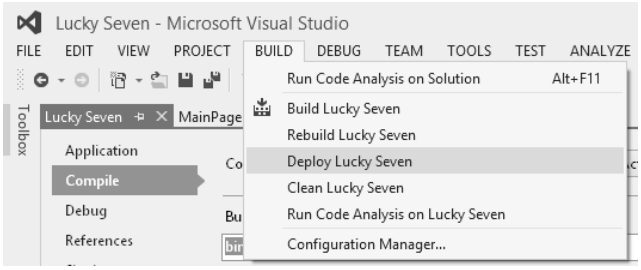
The process of preparing an executable file for a specific computer is called *deploying the application*. As noted, when you deploy an application with Visual Studio, the IDE handles the process of copying all the executable and support files that you will need to register the program with the operating system and run it. Visual Studio allows you to deploy applications *locally* (on the computer you are using) or *remotely* (on a computer attached to the network or Internet).

In the following steps, you'll deploy a release build for the My Lucky Seven application locally and create an application icon for the program on the Windows Start page.

Deploy a release build for the Lucky Seven app

1. Click the Solution Configurations drop-down list box on the Standard toolbar, and then click the Release option. Visual Studio will prepare your project for a release build, with the debugging information removed. The build output path is set to bin\Release\.

2. On the Build menu, click the Deploy Lucky Seven command.



   The Build command creates a bin\Release folder in which to store your project (if the folder doesn't already exist) and compiles the source code in your project. The Output window appears to show you milestones in the assembly and deployment process. The result is an executable file named Lucky Seven.exe, which Visual Studio registers with the operating system on your computer.

   Visual Studio deploys the application locally because Local Machine is currently selected on the toolbar next to the Start button. This is the desired behavior here, but you can also deploy applications on a remote machine (that is, a computer attached to yours via a network or the Internet) by selecting the Remote Machine option. If you select this option, you'll be presented with a dialog box asking for more information about the remote connection. Remember that remote deploying is mostly designed for testing purposes. The best way to install completed applications via the Internet is through the Windows Store.

   When you deploy an application built for the Windows 8.1 user interface, Windows automatically creates a new program icon for the application on the Start page. You can use this icon to launch the program whenever you want to run it. Try running My Lucky Seven now from the Start page on your computer.

3. Open the Windows Start page, and browse to the list of applications that are currently installed.

4. There are two possible locations for your new app: the main Start page, or the secondary Start page containing a longer list of app tiles. (This is where my Windows 8.1 system put the new Lucky Seven program.)

Because you didn't create a colorful Start page tile for your app, the default (gray) tile is shown. Your screen will look similar to this (note the Lucky Seven app in the second column):

5. Click the Lucky Seven application icon, and the Lucky Seven program will load and run in Windows.

6. Test the application again, clicking Spin several times and building up a few wins. When you are finished, close the app.

7. Return to Visual Studio, and close the Output window and the Lucky Seven properties page. Note that you can view and change compilation options whenever you want—the properties page is always available.

8. On the File menu, click Exit to close Visual Studio and the My Lucky Seven project.

9. Click Save if you are prompted to, and the Visual Studio IDE will close.

Congratulations on completing your first Windows Store app!

Summary

This chapter described how to create a Windows Store app named Lucky Seven by using Visual Studio 2013. The development process has much in common with earlier versions of Visual Basic and Visual Studio. You add Toolbox controls to a page, set properties, write program code, test the application, and prepare it for deployment. However, the XAML Toolbox for Windows Store apps is significantly different than the Toolbox used to create Windows Forms apps for the Windows desktop. In this chapter, we reviewed how to use XAML controls step by step. In the next chapter, you'll review how to use the Windows Forms Toolbox to create a desktop application for Windows 8.1, Windows 8, or Windows 7.

While creating the Lucky Seven slot machine game, you practiced using the *TextBlock* control, the *Button* control, the *Image* control, the *MediaElement* control, and setting the *Grid* control's background color. You also learned how to create a splash screen with the Visual Studio Image Editor. Finally, you tested and deployed your application to the Windows Start page. With a little more work, you'll also be able to deploy applications like Lucky Seven to the Windows Store.

Index

Symbols

& (ampersand)
    advanced arithmetic operator, 308
    arithmetic operator, 305
    shortcut operator, 313
    using with String class, 320
< (angle bracket), in XAML markup, 195
* (asterisk)
    arithmetic operator, 305
    shortcut operator, 313
* / (asterisk and backslash), formulas evaluated using, 314
\ (back slash)
    arithmetic operator, 305
    shortcut operator, 313
^ (caret)
    advanced arithmetic operator, 308
    arithmetic operator, 305
    formulas evaluated using, 314
' (comment character), in Visual Basic program code, 272
{ } (curly braces), indicator for markup extension, 517
= (equal) sign
    assignment operator, 293
    comparison operator, 343
    relational operator, 324
/ (forward slash)
    advanced arithmetic operator, 308–312
    arithmetic operator, 305
    shortcut operator, 313
/> (forward slash and a closing bracket), in XAML markup, 195
>= (greater than or equal to) sign
    comparison operator, 343
    relational operator, 324
> (greater than) sign
    comparison operator, 343
    relational operator, 324
#If DEBUG statements, 47, 117
<= (less than or equal to) sign
    comparison operator, 343
    relational operator, 324
< (less than) sign
    comparison operator, 343
    relational operator, 324
.mdb format (Microsoft Access), XML documents vs., 454
- (minus) sign
    arithmetic operator, 305
    formulas evaluated using, 314
<> (not equal to) sign
    comparison operator, 343
    relational operators, 324
@ Page directive, 559
( ) (parentheses), formulas evaluated using, 314
| (pipe symbol), adding items to Filter List using, 186
+ - (plus and minus) signs, formulas evaluated using, 314
+ (plus) sign
    arithmetic operator, 305
    shortcut operator, 313

A

Abs (absolute value) function, 564
Access
    establishing data base connection using Data Source Configuration Wizard, 492–501
    working with databases using ADO.NET, 492–505
    XML documents vs. .mdb format in, 454
access keys, adding to menu commands, 172–174

Acquire Developer License command, 12
AddButton_Click event handler, 406, 413, 427–428, 451
Add Class command, 467
Add Connection dialog box, 494–495
Add Existing Item dialog box, 59
addition
  arithmetic operator, 305
  shortcut operator, 313
Add method, 423, 428, 537
Add New Data Source command, 493
Add New Item dialog box
  Inherited Form template in, 461–462
  naming classes, 469
Add ToolStripButton arrow, 182
ADO.NET, database programming with
  about, 490
  building database app in Data Sources window, 501–505
  database terminology, 490–492
  establishing connection using Data Source Configuration Wizard, 492–501
  working with Access database, 492–505
All Windows Forms category, Button control in, 86
All XAML Controls category, in Toolbox tool, 59
Always Show Solution option, 21, 40
American National Standards Institute (ANSI), character set, 323
American Standard Code for Information Interchange (ASCII)
  character set, 323
  protecting text with encryption using, 331–339
  working with, 323–325
ampersand (&)
  advanced arithmetic operator, 308
  arithmetic operator, 305
  shortcut operator, 313
  using with String class, 320
AndAlso conditional statements, 352–353
And, logical operator
  about, 349–350
  adding password protection using, 350–352
angle bracket (<), in XAML markup, 195
ANSI (American National Standards Institute), character set, 323
AppBarButton control, 236–238
AppBar control, 236
AppBarToggleButton control, 236, 238, 242
Appearance category, in Properties window, 61
Application_Activated event handler, 632–633, 636
Application_Deactivated event handler, 632, 636
app listing page, of Windows Store, 10
apps. *See* desktop apps; *See* Windows Phone apps; *See* Windows Store apps
App.xaml
  creating style for, 222–227
  examining, 197–200
ArcadeRiff.wav file, 58–59
ArgumentException object, 387
ArgumentOutOfRangeException object, 387
ArithmeticException object, 387
arithmetic operators, 305–313
Array class, processing large arrays using methods in, 416–422
ArrayList class, 423
array literal, 402
array name, syntax element in array declaration, 398
arrays
  about, 398
  assigning initial values, 402–403
  creating, 398–399
  declaring multidimensional, 403
  declaring with set elements, 399–400
  extracting information using LINQ from
    about, 437–438
    extracting numeric information, 438–441
  processing large, 416–422
  setting aside memory for, 400–401
  setting size at runtime of, 409–414
  using LINQ locating overlapping elements of, 448–449
  using one-dimensional, 404–409
  using ReDim Preserve to preserve contents of, 414–416
  working with elements of, 401–402
Artboard, Blend, 193
As Boolean clause, 346
Asc function, 325
ASCII (American Standard Code for Information Interchange)
  character set, 323
  protecting text with encryption using, 331–339
  working with, 323–325
AscW method, 325
ASP.NET
  about, 544
  binding datasets to web applications via, 492
  building web forms website with, 550–556

customizing website template, 570–572
displaying database records on webpage, 573–580
editing document and site master properties, 581–583
hosting web applications, 569
server controls
about, 552–553
adding to website, 561–563
software requirements for developing in, 550–551
tags, 559
using Web Designer, 557–560
WebMatrix and, 547
writing event handlers for webpage controls, 563–569
ASP.NET Development Server, 551, 561–563, 565
ASP.NET MVC
creating web applications using, 546
Web Pages with Razor and, 547–548
ASP.NET Web Forms
about, 545
ASP.NET Web Forms, 552
building website with, 550–556
ASP. NET Web Pages (with Razor)
about, 547
make up of, 559
assembly, 31
Assets folder
adding images to, 128–129
creating splash screen from, 70–73
for designing custom tiles for apps, 249
in Solution Explorer, 56–57
assignment operator (=), 293
asterisk (*)
arithmetic operator, 305
shortcut operator, 313
asterisk and backslash (* /), formulas evaluated using, 314
attributes, as properties, 196
audio media, using MediaElement control for playing, 58–60
Audio Playback template, Windows Phone, 607
Auto Hide command, 38
Auto Hide pushpin button, 37–39
AutoPlay check box, 60
AutoSize property, in Properties window, 95

B

background color, setting page, 62
background image, app with generic list and, 425–432
back slash (\)
advanced arithmetic operator, 308
arithmetic operator, 305
formulas evaluated using, 314
shortcut operator, 313
badge notification, appearing in Lock screen, 250
base class
creating, 466–476
inheriting, 476–479
referencing, 481
BaseDiscount class, 481, 483–484
BasedOn property, using to inherit style, 228–231
binary operators, 350
binding
about, 516–517
elements for data, 516–517
to XAML controls, 516–526
Binding class, binding objects using, 517
Birthday program, creating using DateTimePicker control, 152–154
BitArray class, 423
blank lines, in LINQ code blocks, 439
Blend, 202
add controls in, 202
XAML in, 193
blog for developers, preparing for Windows Store, 12
Boolean
data type
about, 298
ListBox control and, 300–301
expressions, 346
IsChecked property, 204
IsCompact property, 237
IsMuted property, 137
setting properties, 98
ToggleButton control as, 203
bound controls, 502
browsers, web, creating, 142–146
Brush category, in Properties window, 51, 62
Build command, 76
Build My Form Inheritance command, 461
Button1_Click event handler, 350–351

button controls
adding to desktop apps, 86–88
adding to phone app, 614, 616
adding to program, 54–55
adding to toolbar, 181–182
adding to website, 562–563
AppBarButton, 236–238
AppBarToggleButton, 236, 238, 242
controlling music playback with, 135–138
for opening flyout on page, 245–248
in list app, 425
in phone app, 614, 616
looping, 243
navigate to web address in browser, 142
setting properties of, 93–94
using with date time picker, 150
buttons
Auto Hide pushpin, 37–39
Categorized, 93
Check Drive, 379–381, 385, 389
Collapse Pane/Expand Pane, 30
Debugging, 116–117
Expand Pane/Collapse Pane, 30
Fill Array, 419
Horizontal Split, 30
looping, 243
New Connection, 494
Publish Now, 106
Solid Color Brush, 51
Sort Array, 420
Start Debugging, 103
Stop Debugging, 382
Test Connection, 495
Vertical Split, 30
Zoom drop-down, 49
Byte data type
about, 298
ListBox control and, 300–301

C

CalculateButton_Click event handler, 564–565
Calculate_Click event handler, 307
Calendar app (Windows Phone 8), 590
calendar information
running Birthday program, 152–154
using DateTimePicker control for gathering, 148–152
camel-casing style, when declaring variables, 272, 293
Canvas control, 209
caret (^)
advanced arithmetic operator, 308
arithmetic operator, 305
formulas evaluated using, 314
shortcut operator, 313
Categorized button, in Properties window, 93
CDbl function, 564
certification, planning for Windows Store developer, 12
Change Data Source dialog box, 494
Char data type
about, 298
ListBox control and, 300–301
charms vs. command bars, 239
CheckBox control
creating, 155–158
running Checkbox program, 158–159
CheckButton_Click event handler, 379, 384, 386, 391
Check Drive button, 379–381, 385, 389
CheckedChanged event handler, 163
checklists, Windows Store requirements, 12–15
child element, in XAML, 196
Choose Data Source dialog box, 494
“chrome” (persistent user interface) features, presenting users in Windows Store app with, 236
ChrW method, 325
classes
binding control to, 517–521
binding objects using Binding class, 517
creating base, 466–476
for maintaining lists in System namespace, 423
identifying in Designer, 460
inheriting base, 476–479
Inherits statement in, 467
in polymorphism, 482
referencing base, 481
Class View tool, 24
Clear method, Array class, 416
Click event, in Properties window, 64–65
ClickOnce Security and Deployment, publishing desktop app using, 105–107
cloud, Windows Azure applications for, 6
CLR (Common Language Runtime), Windows, 544
Code Editor
buttons, 30
displaying XAML markup in Designer, 25–30

docking as tabbed document, 37
green type in, 65
IntelliSense feature, 204, 500
keyword indicator in, 100
using for desktop apps, 99–101
using for Windows Phone apps, 617–620
using for Windows Store apps, 63–66
using LINQ with, 437
XAML tab of
about, 23
adding elements using, 202–212
adjusting Background property in, 201–202
displaying markup in Designer, 23
examining XAML project files, 198–200
setting property for text box object, 196
Collapsed (invisible) property, 61, 67
Collapse Pane/Expand Pane button, 30
collections
about, 422–423
app with generic list and background image, 425–432
generic
about, 423–424
binding to ListBox control, 522–526
declaring, 424–425
LINQ retrieving data from, 451–453
using LINQ with, 450–451
ColorDialog control, 184, 187
Color Resources editor, 51
color value, hexadecimal, 51
columns (fields), 491
command bar
creating to manage tasks, 236–243
vs. charms, 239
CommandBar control, 236–238, 240–243
commas (,), displaying with data types using Format() function, 299
comment character ('), in Visual Basic program code, 272
Comment-out, 530
comments, in Code Editor, 65
Common category, in Properties window, 53, 55
Common Controls tab (Toolbox tool), 502, 506
Common Language Runtime (CLR), Windows, 544
comparison operators
about, 343
using with Select Case structure, 355–361
component tray pane, non-visible objects displayed in, 171
concatenation (combination), string
advanced arithmetic operator, 308
arithmetic operator, 305
shortcut operator, 313
using with String class, 320
conditional expressions
order in decision structure of, 345
using, 343–344
using logical operators in, 349–352
conditional statement, in Do loop, 369
console applications
about, 267
Console object in, 272
creating in Visual Studio
about, 268–275
modules and procedures, 270–271
opening Console Application template, 269–270
Roll-The-Dice application, 271–275
Sub Main() procedure, 271–275
temperature conversion application, 271–275
math games, interactive, 275–284
Visual Basic opportunities for, 6
Console class
in Find-The-Number console application, 276–277
in simulate rolling dice console application, 280–281
constants
about declaring, 292
defining, 295–296
guidelines for naming, 295–296
Const keyword, 295
constructor, 481, 520
Contains method, 122–124, 321, 443, 445
controls
about, 111
adding controls using Toolbox tool, 49–52
AppBar, 236
AppBarButton, 236–238
AppBarToggleButton, 236, 238, 242
bound, 502
button
adding to desktop apps, 86–88
adding to program, 54–55
adding to toolbar, 181–182
adding to website, 562–563
controlling music playback with, 135–138
for opening flyout on page, 245–248
in list app, 425

in phone app, 614, 616
looping, 243
navigate to web address in browser, 142
setting properties of, 93–94
using with date time picker, 150
Canvas, 209–212
CheckBox, 155–159
ColorDialog, 184
ColorDialog, properties, 187
CommandBar, 236–238, 240–243
DateTimePicker control, 148–154
dialog box, 183–185
displaying database information using toolbox, 506–509
FlipView, 127–132
FolderBrowserDialog, 184
FontDialog, 184
Grid, 201
GridView, 574–578
GroupBox, 159–164
HTML, 553–554
HyperLink, 578–579
Image
about, 55
adding images to program, 56–58
adding to phone app page, 612–614
ceating, 205
style set for, 219
in Windows Phone Store, 601, 603
Label
adding to form, 175
adding to website, 562–563
in Windows Forms Toolbox, 88–89
setting properties, 95–97
ListBox, 164–168, 297–304, 355, 522–526
MaskedTextBox, 346–348, 350–351, 506–509
MediaElement
about, 133
controlling playback, 135–138
in Toolbox tool, 58–60
playing music using, 133–135
playing videos using, 139–141
Source property of, 59, 134
using with CommandBar control, 240–243
using with Flyout control, 245
MenuStrip
about, 169–170
adding access keys to menu commands, 172–174
changing order of menu items, 175
creating menu, 170–172
processing menu choices, 175–180
OpenFileDialog, 184
PageSetupDialog, 184
PictureBox
creating rectangle above check box, 155
creating rectangle beneath group box with, 161
drawing square object on form, 184
in Windows Forms Toolbox, 89–90
PrintDialog, 184
PrintDocument, 183
Print-PreviewControl, 183
PrintPreviewDialog, 184
ProgressBar, 416–417
RadioButton, 159–164
RangeValidator, 570
RequiredFieldValidator, 570
SaveFileDialog, 184
server, 552–553
TextBlock
adding text blocks for random numbers, 52–54
in list app, 425
in phone app, 614–617
using, 50–52
TextBox
adding to website, 561–562
assigning to variable, 118–120
binding control to class using, 517–521
check spelling in, 124–127
holding web address in browser, 142
in Data Sources window, 502–503
in list app, 425
multiline, 120–124
opening and displaying contents of XML documents, 517–521
receiving input using, 113–118
using loop to fill text box with string data, 362
ToggleButton, 203–206
ToolStrip, 180–183
understanding, 112–113
webpage validator, 570
WebView, displaying live web content using, 141–146
conversion functions, 319–320
Convert class, converting data types using, 318–319
Copy method, Array class, 416

core definition (root) element, in XAML documents, 198
counter variables, data types for loops using, 362
Count method, 428
crashes, program, 376, 383
CreateArrayButton_Click event handler, 412
CreateButton_Click event handler, 365
Create Directory For Solution check box, 63
"Cryptographic Tasks" (MSDN), 339
CSS (Cascading Style Sheet) information in websites, 552
curly braces ({ }), indicator for markup extension, 517

D

data access, TwoWay, 517
database app, building in Data Sources window, 501–505
database objects, 498
databases
  about, 490
  backward compatibility of, 490
  combining, 498
  commercial application that uses, 497
  displaying records on webpage, 573–580
  relationship to datasets, 505
  terminology used about, 490–492
  using part of, 498
  using toolbox controls to display information, 506–509
data binding
  elements for, 516–517
  to XAML controls, 516–526
DataBindings property settings, about, 492
data entry, user interface for, 540–541
DataException exception object, 387
data, managing
  about, 397
  preserving array contents using ReDim Preserve, 414–416
  processing large arrays, 416–422
  with LINQ
    about, 435–436
    debugging strategies, 450
    extracting information from arrays, 437–449
    using with collections, 450–453
    using with XML documents, 454–458
  working with arrays of variables, 398–409
  working with Collections, 422–432
data navigator, 491–492
Dataset Designer, 500–501
dataset objects, binding masked text box control to, 506–509
datasets
  about, 492
  database objects and, 498
  disconnected data sources in, 498
  displayed in Data Sources window, 501
  relationship to databases, 505
  typed, 500
Data Source Configuration Wizard
  about, 491–492
  connecting datasets to, 492
  establishing data base connection using, 492–501
  filtering data with, 509
  in adding GridView control using, 574–575
  writing Windows Forms app in, 492
data sources
  binding to Texbox control, 518–520
  for Windows Store apps, 516
Data Sources window
  about, 23, 501–502
  binding datasets to controls, 492
  creating database objects on form using, 502–505
  Data Source Configuration Wizard, 493
DataTypeListBox_SelectionChanged event handler, 303–304
data types
  converting, 315–320
  ListBox control and, 297–304
data type, syntax element in array declaration, 398
DateButton_Click event handler, 151–152
Date data type
  about, 298
  ListBox control and, 300–301
DateString property, 177
DateTimePicker control
  about, 148
  creating, 148–152
  running Birthday program, 152–154
DateToolStripMenuItem_Click event handler, 177
debug build executable file, 75, 104
Debugging button, 116–117
debugging counters, 47
Debugging Not Enabled dialog box, 565–566
debugging strategies, LINQ, 450
Debug menu, Start Debugging command on, 30

Debug toolbar, Step Into button on, 450
Debug windows, 24
Decimal data type
about, 298
ListBox control and, 300–301
using in loops with counter variables, 362
decision structures, 344
declaring arrays
about, 398–399
assigning initial values, 402–403
setting array size at runtime, 409–414
setting aside memory for, 400–401
using one-dimensional arrays, 404–409
with set elements, 399–400
working with array elements, 401–402
defensive programming logic, 393–394
deploying applications
desktop apps, 105–107
on web server, 569
Windows Store, 75
derived class, 480
derived classes, 467
Designer
about, 23–24
add controls in, 202
buttons, 30
displaying XAML markup in, 25–30
docking as tabbed document, 37
Ellipse tool in, 71–72
Fit All option in, 129
identifying classes in, 460
media element object in, 59
navigating, 48–49
"Designing UX for apps" (MSDN article), 236
design mode, Visual Basic, 87
Design tab (Web Designer)
about, 556
adding text, 557–558, 573
editing text, 581–582
desktop apps
about, 79
building executable file, 104–105
ColorDialog control properties, 187
controls
CheckBox, 155
DateTimePicker, 148–154
GroupBox, 159–164
ListBox, 164–168, 297–304
MenuStrip, 169–183
creating
adding number labels, 88–89
adding .wav file to Resources folder, 92–93
naming objects for clarity, 98
new project for, 83–85
picture box properties, 97–98
setting button properties, 93–94
setting descriptive label properties, 96–97
setting number labels properties, 95–96
setting title bar text of form, 97
SpinButton_Click event handler, 101–103
user interface, 85–87
using PictureBox control, 89–90
writing code, 99–101
database controls for
about, 489
displaying database information using toolbox controls, 506–509
programming with ADO.NET, 490–505
SQL statements and filtering data, 509–514
dialog box controls, 184–185
event handlers managing common dialog boxes, 185–190
MaxLength property of TextBox controls, 363
program crashes in, 383
publishing, 105–107
running, 103–105
starting, 80–81
Visual Studio 2013 and, 81–82
Details page, Windows Store, 15
developers
annual registration fee for Windows Phone, 596
getting license for Windows, 18
Windows Phone Store
planning for certification, 595–596
selling apps in, 595
developers, Windows Store
planning for certification, 12
registering as, 11
Windows Store requirements checklist, 12–15
Device charms, 239
device drivers, Visual Basic opportunities for, 6
Device window, 23–24
dialog box controls, 183–185
dialog boxes, managing with event handlers common, 185–190
Dictionary class, 423
Dim statements
Boolean variables created by, 346
creating generic collection, 427

declaring variables with, 292–294
for data types, 301
in LINQ queries, 436, 439
DirectoryNotFoundException object, 387
disconnected data sources, 498
DisplayArray_Click event handler, 413
DisplayButton_Click event handler, 407
DisplayImageCheckBox_CheckedChanged event handler, 157–158
DisplayToggleBtn_Click event handler, 206, 208
DivideByZeroException object, 387
division sign
advanced arithmetic operator, 308–312
arithmetic operator, 305
formulas evaluated using, 314
shortcut operator, 313
.dll file format, inheriting form using, 463
docking
Code Editor or Designer as tabbed documents, 37–38
programming tools manually, 37–38
Document Outline window, 23–24
Do loops
avoiding endless loop, 369
converting temperatures using, 370–372
using Until keyword in, 372–373
Double data type
about, 298
in math program, 307
ListBox control and, 300–301
Do...Until Loop, in Find-The-Number console application, 276–277
Do...While structure, 549
drive and path name errors, 378–382

E

Ease Of Access Center, displaying underline or small box for access keys using, 173
Element method, 532–533, 536–537
Ellipse tool, in Designer, 71–72
Else conditional statements, 344
ElseIf conditional statements, 344
encryption, protecting text with, 331–339
End as keyword, in Visual Studio 2013, 100
End Class statements, 477
End If conditional statements, 344
endless loop, avoiding, 369
EndOfStreamException object, 387
End Sub and Sub keywords, 65
End Sub statement, 483–484
EndsWith method, 325
End Try statement, 384
EnterButton_Click event handler, 478
Entity Framework, 490
equal (=) sign
assignment operator, 293
comparison operator, 343
relational operator, 324
error handlers
about, 375
comparing with defensive programming techniques, 393–394
Exception objects, 387
Exit Try statement, 394
processing errors using Try...Catch statement, 376–384
specifying retry period, 390–392
using Finally clause to perform cleanup tasks, 385
using nested Try...Catch blocks, 391–392
writing flash drive error handler, 384–385
error messages
correcting, 66
Unrecognized Database Format, 496
errors (exceptions)
raising your own, 390
unhandled, 377, 382
Windows Store apps built-in handling of, 383
event-driven programming, 342
event handlers
AddButton_Click, 406, 427–428, 451
Application_Activated, 632–633
Application_Deactivated, 632
Button1_Click, 350–351
CalculateButton_Click event handler, 564–565
Calculate_Click, 307
CheckButton_Click, 379, 384, 386, 391
CheckedChanged, 163
CreateArrayButton_Click, 412
CreateButton_Click, 365
creating, 64, 136, 205–208
DataTypeListBox_SelectionChanged, 303–304
DateButton_Click, 151–152
DateToolStripMenuItem_Click, 177
DisplayArray_Click, 413
DisplayImageCheckBox_CheckedChanged, 157–158
DisplayToggleBtn_Click, 206, 208

EnterButton_Click, 478
FillButton_Click, 419
Form1_Load, 370–371, 417–418, 422
in Windows Store app, 67–68
managing common dialog boxes with, 185–190
NavigateButton_Click, 143
OpenItem_Click, 333–334, 336
OpenToolStripButton_Click, 185
OpenTool-StripMenuItem_Click, 330
PauseButton_Click, 137
RecordScoreButton_Click, 619, 620
ReverseButton_Click, 421
RunQuery_ Click, 439, 442–444, 446, 448, 451, 455–457
SaveAsItem_Click, 333, 336–337
ShowButton_Click, 428
SortButton_Click, 420–421
SortTextToolStripMenu-Item_Click, 328
SpinButton_Click, 101–103
StopButton_Click, 137
Sub procedure and, 270–271
TestButton_Click, 116, 481–482, 484
TimeToolStripMenuItem_Click, 176–177
ToolStripButton1_Click, 186–187
writing for webpage controls, 563–569
XmlTestButton_Click, 530
events, supported by Visual Basic objects, 342
exception handling, 383
Exception objects, 376, 387–390
exceptions (errors)
about, 311, 313
raising your own, 390
unhandled, 377, 382
Windows Store apps built-in handling of, 383
executable file, building
console applications using release build, 271–275
for destop app, 104–105
for Windows Store app, 74–77
Exit For statement, 367–368
Exit Try statement, 377–378, 394
Expand Pane/Collapse Pane button, 30
explicitly declaring variables, 292–293
exponentiation (raising to a power) operator
advanced arithmetic operator, 308–312
arithmetic operator, 305
shortcut operator, 313
Express for Web, 4
Express for Windows, 4
Express for Windows Desktop, 4
Express for Windows Phone, 4
Extensible Application Markup Language (XAML)
about, 191–192
All XAML Controls category, 59
as root of Windows Store app controls, 112–113
data binding expressed as markup extension, 517
defining list box using, 357–361
elements
about, 194–196
adding using tab of Code Editor, 202–212
examining project files, 196–202
Grid element, 201
introduction to, 192–202
markup to define FlipView control, 129–131
namespaces in, 196, 199
resource dictionary, 218
root element in documents, 198
styles
about, 215–216
building new styles from existing styles, 228–231
creating, 217–221
IDE shortcuts for applying, 231–232
practicing, 221–227
referencing, 220
StandardStyles.xaml, 216–217
using explicit and implicit, 220–221
tab of Code Editor
about, 23
adding elements using, 202–212
adjusting Background property in, 201–202
displaying markup in Designer window, 25–30
examining XAML project files, 198–200
setting property for text box object, 196
< tag and /> tag in markup, 195
Toolbox controls
about, 49
AppBar, 236
AppBarButton, 236–238
AppBarToggleButton, 236, 238, 242
binding to data using, 516–526
Canvas, 209–212
CommandBar, 236–238, 240–243
Flyout, 243–248
gesture support using, 259–260
Grid, 201
Image, 205, 612–613
in Windows Phone Store, 600, 603
ListBox, 522–526
ProgressRing, 417

TextBlock, 425, 614–617
TextBox, 425
ToggleButton, 203–206
WPF and, 26, 82, 112
Extensible Markup Language (XML)
about, 454, 490
documents
about, 515, 526
accessing data in, 526–540
adding node with data to, 538–540
locating child elements in XML hierarchy, 532–533
modifying element in, 537–538
opening and displaying contents of, 527–530
reading selection of tagged elements, 530–532
searching for items in file, 533–536
elements
locating in XML hierarchy child, 530–532
modifying, 537–538
reading selection of tagged, 530–532
files
about, 526
reading, 526–533
searching for items in, 533–536
writing to, 536–540
using LINQ with, 454–458
vs. Microsoft Access .mdb format, 454

F

FahrenheitTemp variable, 272
fields (columns), 491
FileNotFoundException object, 387
files, selecting contiguous or noncontiguous, 129
Fill Array button, 419
FillButton_Click event handler, 419
Filter List, using pipe symbol (|) for adding items to, 186
Finally statement, 377–378, 385–386
FindDiscount method, 481–484
Find method, Array class, 416
Find-The-Number console application, 275–280
firing (triggering) events, 65
Fit All option, in Designer, 129
flash drives, writing error handler for, 384–385
FlipView control, 127–132
Flyout control, 243–248
Focus method, 413, 428
FolderBrowserDialog control, 184
FontDialog control, 184
Font property, in Properties window, 96
Font Size text box, 51
For Each...Next loop, 428, 440–441
ForeColor property, in Properties window, 96–97
ForegroundColor property, 276–277
Foreground property, 51
Form1_Load event handler, 370–371, 417–418, 422
Format function, displaying commas with data types using, 299
formulas, 304, 314–315
For...Next loop
counter variables, 326, 330
in arrays, 407–410, 420
in simulate rolling dice console application, 280–281
mastering
about, 361
complex loops, 363–368
Exit For statement, 367–368
using to convert distances, 364–367
using to fill Textbox with string data, 362–363
placing Stop statement for, 450
forward slash (/)
advanced arithmetic operator, 308–312
arithmetic operator, 305
shortcut operator, 313
forward slash and a closing bracket (/>), in XAML markup, 195
free
offering apps as, 11
versions of Visual Studio 2013 development suite, 4
From clause, in LINQ queries, 436, 443, 448, 456, 535
FromFile method, 379
FromFile statement, 384
Function procedures, in Visual Basic application, 270–271
functions, conversion, 319–320

G

gathering input, controls for
CheckBox, 155
GroupBox and RadioButton, 159–164
ListBox, 164–168

generic collections
about, 423–424
binding to ListBox control, 522–526
declaring, 424–425
LINQ retrieving data from, 451–453
gestures
common, 260–262
on phone app development, 619
support for, 259–260
Get block, 470
GetLowerBound method, 404–409
GetUpperBound method, 404–410, 418
Global.asax files (global web application information), 551
greater than or equal to (>=) sign
comparison operator, 343
relational operator, 324
greater than (>) sign
comparison operator, 343
relational operator, 324
green type, in Code Editor, 65
Grid element, in XAML, 201
Grid object, 62
GridView control, 574–578
GroupBox control
about, 158–159
gathering input with RadioButton control and, 160–163
running Radio Button program, 160–163
Group method, 470–472, 475, 478

H

Hashtable class, 423
hexadecimal color value, 51
Horizontal Split button, 30
.htm (HTML page files), 551
HTML
tags, 559
viewing markup for webpage, 559–560
HTML5 and JavaScript programming, 548–550
HTML controls, 553–554
HyperLink control, 578–579

I

IDE (Integrated Development Environment)
about, 5, 17
about development environment, 19–21
component tray displaying non-visible objects, 171
configuring for step-by-step exercises, 39–42
editing Windows form app with, 460–464
exploring Windows Phone, 609–610
in deploying application, 75
menu commands pertaining to Windows store, 12
running program from, 67–68
shortcuts for applying styles, 231–232
tools for
important, 22–24
organizing tools, 24
XAML in, 192–193
If...Else structure, 549
If...Then...ElseIf structure, 431
If...Then...Else structure, 407
If...Then structure, 122, 186, 207, 308, 344–353, 393, 413
IIS (Internet Information Services), Microsoft, 544, 569
Image control
about, 55
adding images to program, 56–58
adding to phone app page, 612–614
creating, 205
style set for, 219
Image Editor tool, using to design custom tiles, 251–253
Image Gallery program, using FlipView control in, 130–132
image objects
naming in program, 58
setting Visibility property, 61
images
adding to Assets folder, 128–129
adding to Resources folder, 90–92
hiding, 67
splash screen, 70
implicitly declaring variables, 292
IndexOutOfRangeException object, 387
inheritance
about, 460
polymorphism as type of, 480
inheritance picker, 460–466
Inheritance Picker dialog box, 462–463
Inherited Form template, 462
Inherits statement
about, 467
using for inheriting base class, 476–479

input gestures, on touch-enabled screen, 261
Input Mask dialog box, 347
InputScope property, 616
Integer data type
about, 297
displaying commas with, 299
ListBox control and, 299–301
using in loops with counter variables, 362
integer (whole number) division
arithmetic operator, 305
shortcut operator, 313
Integrated Development Environment (IDE)
about, 5, 17
about development environment, 19–21
component tray displaying non-visible objects, 171
configuring for step-by-step exercises, 39–42
editing Windows form app with, 460–464
exploring Windows Phone, 609–610
in deploying application, 75
menu commands pertaining to Windows store, 12
running program from, 67–68
shortcuts for applying styles, 231–232
tools for
important, 22–24
organizing tools, 24
XAML in, 192–193
IntelliSense feature, Code Editor, 500
Internet Explorer, intranet settings are turned off warning in, 567
Internet Information Services (IIS), Microsoft, 544, 569
intranet settings are turned off warning, 567
IOException object, 388
IsChecked property, 204
IsCompact property, 237
IsEnabled property, 452
IsLooping property, 60, 242
IsMuted property, 137
IsolatedStorageSettings class, 617, 636
IsSpellCheckEnabled property, 124–127
Items Collection Editor (Properties window), 504

J

JavaScript
about, 549
and HTML5 programming, 548–550
Windows Store apps designed with, 549
JPEG file, displaying, 55

K

keyboard shortcuts
displaying Properties window, 34
moving from one zoom setting to another, 49
Open Project dialog box, 20
selecting contiguous or noncontiguous files, 129

L

Label controls
adding to form, 175
in Windows Forms Toolbox, 88–89
setting properties, 95–97
Layout category, AutoSize property in, 95
Length property, using with String data, 116–117
less than or equal to (<=) sign
comparison operator, 343
relational operator, 324
less than (<) sign
comparison operator, 343
relational operator, 324
LINQ (Language Integrated Query). *See also* queries, LINQ
about, 435–436
blank lines in code blocks, 439
debugging strategies, 450
extracting information from arrays, 437–438
extracting string data, 443–447
query syntax for, 436
using Code Editor with, 437
using complex Where clause, 441–443
using with collections, 450–458
using with XML documents, 454–458, 526, 530–536
ListBox control
about, 164–165
binding generic collection to, 522–526
creating, 165–168
data types and, 297–304
running ListBox program, 168–169
using with Select Case structure, 355
List class, 423
lists
classes for maintaining, 423
creating collections and generic, 423–424, 451–453
using LINQ with XML document, 454–458

live tiles
about, 257
in Microsoft Weather application, 251
pinning on Start Page, 590
programming, 257–259
receiving notification, 258
local deployment of apps, 75–77
local machine, running program on, 31
local notification, 258
Lock screen, badge notification appearing in, 250
logical operators, using in conditional expressions, 349
Long data type
about, 298
displaying commas with, 299
ListBox control and, 300–301
using in loops with counter variables, 362
looping button, 243
Lucky Seven app, Windows desktop
about, 79–80, 82–83
adding number labels, 88–89
adding .wav file to Resources folder, 92–93
building executable file, 104–105
creating user interface, 85–87
naming objects for clarity, 98
new project for creating, 83–85
picture box properties, 97–98
publishing, 105–107
running, 103–105
setting button properties, 93–94
setting descriptive label properties, 96–97
setting number labels properties, 95–96
setting title bar text of form, 97
SpinButton_Click event handler, 101–103
using PictureBox control, 89–90
writing code, 99–101
Lucky Seven app, Windows Store
about, 44–45
building executable file, 74–77
creating splash screen, 70–73
designing user interface
adding button control, 54–55
adding image, 56–58
adding text blocks for random numbers, 49–50
creating new project, 45–47
navigating Designer, 48–49
opening Toolbox, 49–50
playing audio media, 58–60
using TextBlock control, 50–52
running program, 67–68
setting background color of page, 62
setting Visibility property, 61
SpinButton_Click event handler, 67–68
using Code Editor, 63–66
using Save All command, 62–63

M

MainPage.xaml, edit XAML markup in, 197–200
managed provider (provider), in database connection string, 496
managing data
about, 397
preserving array contents using ReDim Preserve, 414–416
processing large arrays, 416–422
with LINQ
about, 435–436
debugging strategies, 450
extracting information from arrays, 437–449
using with collections, 450–453
using with XML documents, 454–458
working with arrays of variables, 398–409
working with Collections, 422–432
Manifest Designer
adjusting tile options in, 254–257
setting options in Windows Phone manifest file, 637–638
markup extension
XAML data binding expressed as, 517
MaskedTextBox control, 346–348, 350–351, 506–509
Math class, 272
math games, interactive, 275–284
MaxLength property, of TextBox control, 363
MediaElement control
about, 133
controlling playback, 135–138
in Toolbox tool, 58–60
playing music using, 133–135
playing videos using, 139–141
Source property of, 59, 134
using with CommandBar control, 240–243
using with Flyout control, 245
media element object, in Designer, 59
menu commands
Acquire Developer License, 12
adding access keys to, 172–174

Auto Hide, 38
Open Developer Account, 12
Reserve App Name, 12
Start Debugging, 30
menu conventions, 173
menu items, changing order of, 175
MenuStrip control
about, 169–170
adding access keys to menu commands, 172–174
changing order of menu items, 175
creating menu, 170–172
processing menu choices, 175–180
Merry-go-round video file, 139
Message property, Exception object, 387
methods
syntax for overriding, 480–481
system clock, 180
vs. properties, 151
Microsoft Access
establishing data base connection using Data Source Configuration Wizard, 492–501
working with databases using ADO.NET, 492–505
XML documents vs. .mdb format in, 454
Microsoft Calendar app, 590
Microsoft Developer Network (MSDN), Windows Phone Development Center, 595
Microsoft IntelliSense feature, Code Editor, 204, 500
Microsoft Internet Information Services (IIS), 544, 569
Microsoft .NET Framework
specify version of, 84
Windows Forms and, 81
Microsoft OLE DB, as database provider, 496
Microsoft Silverlight, Windows Phone 8 and, 112
Microsoft user experience (UX), 236
Microsoft Visual Studio. *See* Visual Studio
Microsoft Visual Studio 2013. *See* Visual Studio 2013
Microsoft Visual Studio Express for Web, 4
Microsoft Visual Studio Express for Windows, 4
Microsoft Visual Studio Express for Windows Desktop, 4
Microsoft Visual Studio Express for Windows Phone, 4
Microsoft Visual Studio website, 4
Microsoft Weather application, tiles in, 250–251
minus (-) sign
formulas evaluated using, 314
shortcut operator, 313
mobile phone programming
about, 587
app life cycle considerations, 626–636
closing apps, 626–627
creating apps
about, 607
adding Image control, 612–614
adjusting settings in PhoneApplicationPage, 611
creating new project, 607–608
designing user interface, 614–617
exploring IDE, 609–610
mouse input, 619
writing code, 617–620
deactivating apps, 627
features of Phone 8, 589–592
hardware requirements for, 590
IsolatedStorageSettings class, 636
opportunities in platform, 588
PhoneApplicationService class, 628–635
registering apps, 620–621
setting options in Windows Phone manifest file, 637–638
testing apps, 620–626
Mod
advanced arithmetic operator, 308
arithmetic operator, 305
formulas evaluated using, 314
Model-View-Controller (MVC) architecture, 546
mouse input, on phone apps, 619
Movie Maker, Windows, 141
MSDN (Microsoft Developer Network), Windows Phone Development Center, 595
MsgBox statement, 384
multidimensional arrays, declaring, 403
multiline TextBox controls, 120–124
multiplication sign
arithmetic operator, 305
shortcut operator, 313
multiproject solution, opening, 22
multitasking, in phone environment, 627
music
playing using MediaElement control, 133–135, 240–243
using with Flyout control, 245
MustOverride keyword, 481
MVC (Model-View-Controller) architecture, 546
MyBase syntax, 481

N

Name property, in Properties window, 87–88, 98, 115–116
namespaces
  in Visual Studio programming terminology, 199
  in XAML, 196
NavigateButton_Click event handler, 143
Navigate method, 143
navigation toolbar, 504
.NET Framework
  manipulating strings using, 320
  Math class of, 272
  specify version of, 84
  Windows Forms and, 81
New Connection button, 494
New Project dialog box, 45, 83–85, 128–129
New Web Site command, 554
New Web Site dialog box, 547–548, 555
Next method
  in Find-The-Number console application, 276–277
  in Lucky Seven app, 67
  in simulate rolling dice console application, 280–281
not equal to (<>) sign
  comparison operator, 343
  relational operators, 324
notification
  live tiles receiving, 258
  Start page tile as, 258
Not, logical operator, 349
NotOverridable keyword, 481
number of dimensions, syntax element in array declaration, 399
number of elements, syntax element in array declaration, 399

O

Object Browser tool, 23–24
Object data type, 298
Object list, switching between objects using, 94
Object-Oriented Programming (OOP)
  about, 459
  constructor, 481, 520
  create properties statement, 470
  creating base classes, 466–476
  creating method statement, 471
  creating objects statement, 473
  inheriting base classes, 476–479
  inheriting form using inheritance picker, 460–466
  Inherits statement, 467, 476–479
  polymorphism, 480–485
objects, naming for clarity, 98
object terminology, 36
Office applications, Visual Basic opportunities for, 6
OLE DB, Microsoft, as database provider, 496
Open Developer Account command, 12
OpenFileDialog control, 184
OpenItem_Click event handler, 333–334, 338
Open Project dialog box, keyboard shortcut for, 20
OpenToolStripButton_Click event handler, 185
OpenTool-StripMenuItem_Click event handler, 330
operators
  arithmetic, 305–313
  binary, 350
  comparison, 343, 355–361
  relational, 324
  shortcut, 313
Option Explicit, 41
Option Infer, 41
Option Infer statements, 294, 403
Options dialog box, 39–41
Option Strict, 41
OrElse conditional statements, 352–353
organizing tools
  in IDE, 24
  moving and docking tools, 37–38
Or, logical operator, 349–350
OutOfMemoryException object, 388
OverflowException object, 388
Overridable keyword, 481
Overrides keyword, 481–482

P

Page element, root element and, 200
page-level resource definition, 217
PageSetupDialog control, 184
panning movement, on touch-enabled screen, 260
parentheses (( )), formulas evaluated using, 314
Parse method, converting data types using, 316–318
PasswordChar property, 352
password protection, adding using And operator, 350–352
path name and drive errors, 378–382
PauseButton_Click event handler, 137

People hub (Windows Phone 8), 590
periodic notification, 258
persistent user interface ("chrome") features, presenting users in Windows Store app with, 236
Phone 8 development, Windows
  about, 587
  app life cycle considerations, 626–636
  closing apps, 626–627
  creating apps
    about, 607
    adding Image control, 612–614
    adjusting settings in PhoneApplicationPage, 611
    creating new project, 607–608
    designing user interface, 614–617
    exploring IDE, 609–610
    mouse input, 619
    writing code, 617–620
  deactivating apps, 627
  hardware requirements for, 590
  IsolatedStorageSettings class, 636
  opportunities in platform, 588
  PhoneApplicationService class, 628–635
  registering apps, 620–621
  setting options in Windows Phone manifest file, 637–638
  testing apps, 620–626
Phone 8, Windows
  features of, 589–592
  installing apps, 595
  Microsoft Silverlight and, 112
  Visual Basic opportunities on, 6
  website for, 591
PhoneApplicationPage class, 610–611
PhoneApplicationService class, 628–635
Phone apps, Windows, binding datasets to, 492
Phone Emulator, Windows, using, 621–626
Phone Software Development Kit (SDK)
  about, 596
  on virtual machine environment, 598
  working with version 8.0, 596–599
Phone Store, Windows
  about, 590
  accessing, 591–596
  installing Windows Phone app, 595
  planning for certification, 595–596
  selling apps in, 595
  Windows Phone Store vs., 600–604
photos (pictures)
  adding to Assets folder, 128–129
  adding to Resources folder, 90–92
  hiding, 67
  splash screen, 70
PictureBox control
  creating rectangle above check box, 161
  creating rectangle beneath group box with, 161
  drawing square object on form, 184
  in Windows Forms Toolbox, 89–90
picture box properties, in desktop app, 97–98
pinning live tiles, on Start Page, 590
Pin To Start command, 286
pipe symbol (|), adding items to Filter List using, 186
plus and minus (+ -) signs, formulas evaluated using, 314
plus (+) sign
  arithmetic operator, 305
  shortcut operator, 313
Pmt function, 564
PNG (Portable Network Graphics) format
  displaying file in, 55
  for splash screen, 70
polymorphism, 480–485
portable class libraries, 604
Position property, in Properties window, 60
precedence, changing order of, 315
Preserve keyword, using with ReDim statement, 415
price tier, in Windows Store, 10–11
PrintDialog control, 184
PrintDocument control, 183
Print-PreviewControl, 183
PrintPreviewDialog control, 184
Private and Protected keywords, 65
procedures, 64
program crashes, 376, 383
programmers, Windows Store
  planning for certification, 12
  registering as, 11
  Windows Store requirements checklist, 12–15
programming tools
  manually docking, 37–38
  organizing, 36–39
programming Window Store app
  about, 44–45
  building executable file, 74–77
  creating splash screen, 70–73
  designing user interface
    adding button control, 54–55
    adding image, 56–58

adding text blocks for random numbers, 49–50
creating new project, 45–47
navigating Designer, 48–49
opening Toolbox, 49–52
playing audio media, 58–60
using TextBlock control, 50–52
running program, 67–68
setting background color of page, 62
setting Visibility property, 61
SpinButton_Click event handler, 67–68
using Code Editor, 63–66
using Save All command, 62–63
program statements, in Visual Studio, 66
ProgressBar control, 416–417
ProgressRing controls, 417
Project menu
Add Class command, 467
Add New Item command, 461–462
Project Properties Designer, opening, 92
projects
about, 22
file extension for, 22
Projects folder, default, 20
properties
attributes as, 196
syntax for overriding, 480–481
system clock, 180
vs. methods, 151
Properties window
about, 23–24
Appearance category in, 61
AutoSize property in, 95
Behavior category of, 98
Brush category, 51, 62
Categorized button in, 93
changing Name property of, 87, 88
changing property settings, 35
Click event in, 64–65
Common category in, 53, 55
displaying, 34
Font property in, 96
ForeColor property in, 96–97
identifying classes in, 460
IsLooping property in, 60
Items Collection Editor in, 504
manually docking, 37–38
Name property in, 87, 88, 98, 115–116
Position property in, 60
setting properties in Windows Phone Store app, 611
TextAlign property in, 95
Text category in, 51–52
Text property in, 94, 116
Visible property in, 98
working with, 33–36
property terminology, 36
Protected and Private keywords, 65
provider (managed provider), in database connection string, 496
Public Class statements, 477, 483–484
Public keyword, 295, 470
Public Sub New procedure, 520
publishing console apps, 285
publishing desktop app, using ClickOnce Security and Deployment, 105–107
Publish Now button, 106
push notification, 258

Q

queries, LINQ
extracting numeric information from array, 438–441
extracting string data, 443–447
From statements in, 436, 443, 448, 456
reading selection of tagged XML elements, 530–532
retrieving data from XML document, 455–458
searching for items in XML file, 533–536
Select clause in, 437, 448, 450
syntax for, 436
using complex Where clause, 441–443
Where clause in, 437, 440–443, 448, 456, 534–535
Query Builder, Visual Studio, creating SQL statements with, 509–514
Queue class, 423

R

RadioButton control
about, 159–160
gathering input with GroupBox control and, 160–163
running Radio Button program, 160–163
RAD (Rapid Application Development), ASP.NET Web Forms and, 545

random number generator, declaring, 67
RangeValidator control, 570
Rapid Application Development (RAD), ASP.NET Web Forms and, 545
Razor, ASP. NET Web Pages with
  about, 547
  make up of, 559
ReadAllText method, 330
ReadKey method, 272
  in Find-The-Number console application, 276–277
  in simulate rolling dice console application, 280–281
ReadLine method, 272
RecordScoreButton_Click event handler, 619, 620
records (rows), 491
ReDim statement
  preserving size of array using, 414–415
  specifying size of array at runtime using, 410–414
  using for three-dimensional arrays, 415–416
registering, as Windows Store developer, 11
relational database, 491
relational operators, 324
release build executable file
  about, 75
  building executable file, 104–105
  creating console applications using, 271–275
  deploying, 76–77
remainder division operator
  advanced arithmetic operator, 308–312
  arithmetic operator, 305
remote deployment of apps, 75
Replace method, 325
RequiredFieldValidator control, 570
requirements checklist, Windows Store, 12–15
Reserve App Name command, 12
resource dictionary file, adding XAML, 218
Resources folder
  adding photo to, 90–92
  adding sound file to, 92–93
Retries variable, 390–392
ReverseButton_Click event handler, 421
Reverse method, Array class, 416, 420–421
Rnd function, VBMath class, 419
Roll-The-Dice console application, 284–287
root (core definition) element
  in XAML documents, 198
  Page element and, 200
Round method, 272
rows (records), 491
runningTotal variable, 295
RunQuery_ Click event handler, 439, 442–444, 446, 448, 451, 455–457
run-time errors, 311, 313, 378–382, 391–392

S

sales information, in Windows Store, 10–11
Save All command, 62–63
Save As command, 63
SaveAsItem_Click event handler, 333, 336–337
SaveFileDialog control, 184
Save method, 537
Save New Projects option, 40
SByte data type
  about, 298
  ListBox control and, 301
scheduled notification, 258
ScheduleTileNotification object, 258
SDK (Software Development Kit), Windows Phone
  about, 596
  on virtual machine environment, 598
  working with version 8.0, 596–599
Search charm, 239
Search Criteria Builder dialog box, 510, 512–513
security and permissions settings, 263–266
SecurityException object, 388
security issues
  associating web browser with unknown websites, 146
  in deploying desktop apps, 107
Select Case decision structure
  evaluating records with, 476
  functions using, 471–472
  handling group assignments, 474
  in Find-The-Number console application, 276–277
  inputing conditions into, 475
  mastering, 353–361
Select clause, in LINQ queries, 437, 448
SelectedIndex property, 165, 302
Select Resource dialog box, 91
SELECT statement, SQL, 509
server controls
  about, 552–553
  adding to website, 561–563
Server Explorer tool, 23–24

Set block, 470
Settings charms, 239
SetValue method, 536–537
Share charm, 239
ShellSort procedure, 326, 329–330
ShellSort Sub procedure, 325–326
shortcut operators, 313
Short data type
  about, 297
  displaying commas with, 299
  ListBox control and, 300–301
ShowButton_Click event handler, 428
ShowDialog method, 185
Silverlight, Windows Phone 8 and, 112
simulate rolling dice console application, 280–284
Single data type
  about, 298
  ListBox control and, 300–301
site master page title, editing, 581–583
slide gesture, on touch-enabled screen, 260
.sln (solution file extension), 22
Software Development Kit (SDK), Windows Phone
  about, 596
  on virtual machine environment, 598
  working with version 8.0, 596–599
Solid Color Brush button, 51
Solution Explorer
  about, 23–24
  Assets folder in
    about, 56–57
    creating splash screen from, 70–73
  creating Resources folder, 90–92
  displaying, 26
  double-clicking files in, 27
  opening, 48
solutions
  about, 22
  file extension for, 22
Sort Array button, 420
SortButton_Click event handler, 420, 421
SortedList class, 423
Sort method, 416, 420, 428, 432
SortTextToolStripMenu-Item_Click event handler, 328
sound effect
  adding to desktop apps, 92–93
  adding to program, 58–60
Source property, of MediaElement control, 59, 134
Source tab (Web Designer)
  about, 556–557
  viewing HTML and ASP.NET markup for webpage, 559–560
spelling, checking in TextBox controls, 124–127
SpinButton_Click event handler
  in desktop app, 101–103
  in Windows Store app, 67–68
splash screen, creating, 70–73
Split method, 326, 329
Spotlight area, of Windows Store, 10
SqlException object, 388
SQL Server, as database provider, 496
SQL (Structured Query Language)
  filtering data and, 509–514
  LINQ and, 436
  SELECT statement, 509
standard charms vs. command bars, 239
StandardStyles.xaml, 216–217
Start Debugging button, 103
Start Debugging command, 30
Start Here! Learn Microsoft Visual Basic 2012 (Microsoft Press), 5, 111
Start Page, pinning live tiles on, 590
Start page tile, as notification, 258
StaticResource dictionary, 517
step-by-step exercises, configuring IDE for, 39–42
Step Into button, on Debug toolbar, 450
StopButton_Click event handler, 137
Stop Debugging button, 382
Stop statements, 450
StreamReader class, 325
StreamWriter class, 325
String class, processing strings with
  about, 320
  common tasks, 320–322
  sorting strings in textbox, 325–330
  sorting text, 322
  Visual Basic equivalents of elements in, 321–322
string concatenation (combination)
  advanced arithmetic operator, 308–312
  arithmetic operator, 305
  shortcut operator, 313
  using with String class, 320
String data type
  about, 298
  Length property using with, 116
  ListBox control and, 300–301
string data, using LINQ to extract, 443–447

string keyword, in Dim statement, 293
string variable, using to hold TextBox input, 118–120
structured error handlers, 376
Structured Query Language (SQL)
  filtering data and, 509–514
  LINQ and, 436
  SELECT statement, 509
styles, XAML
  about, 215–216
  building new styles from existing styles, 228–231
  creating, 217–221
  IDE shortcuts for applying, 231–232
  practicing, 221–227
  referencing, 220
  StandardStyles.xaml, 216–217
  using explicit and implicit, 220–221
Sub and End Sub keywords, 65
Sub Main method, creating console applications using, 271–275
Sub procedures
  adding, 483–484
  creating method adding to class, 471
  in Visual Basic application, 270–271
Substring method, 325
subtraction sign
  advanced arithmetic operator, 308–312
  arithmetic operator, 305
  formulas evaluated using, 314
system clock, properties and methods, 180
System namespace
  classes for maintaining lists in, 423
  Collections namespace within, 423

T

tabbed documents
  about, 37
  docking Code Editor or Designer window as, 37
table adapter, 491–492
tables, 491
tap gesture, on touch-enabled screen, 260
TargetType (control name), assigning styles matching, 220
temperature conversion console application, creating, 271–275
TestButton_Click event handler, 116, 481–482, 484
Test Connection button, 495
text
  protecting with encryption, 331–339
  sorting using String class, 322–323
TextAlign property, in Properties window, 95
TextBlock control
  adding text blocks for random numbers, 52–54
  in list app, 425
  in phone app, 614–617
  in Toolbox tool, 50–52
TextBox controls
  assigning to variable, 118–120
  binding control to class using, 517–521
  check spelling in, 124–127
  holding web address in browser, 142
  in Data Sources window, 502–503
  in list app, 425
  multiline, 120–124
  opening and displaying contents of XML documents, 517–521
  receiving input using, 113–118
  using loop to fill text box with string data, 362–363
text boxes, sorting strings in, 325–330
text box object, XAML setting property for, 196
Text category, in Properties window, 51–52
Text property, in Properties window, 94, 116
ThemeResource, 201
tiles
  about, 257
  designing for app custom
    about, 249
    Assets folder for, 249
    required tiles for, 249–257
  live
    about, 257
    in Microsoft Weather application, 251
    pinning on Start Page, 590
    programming, 257–259
    receiving notification, 258
  programming, 257–259
  size of tiles for, 250
"Tiles, badges, and notifications (Windows Store apps)" (MSDN), 259
TileUpdateManager class, 258
TimeToolStripMenuItem_Click event handler, 176–177
title bar text, setting, 97
ToggleButton control, 203–206

toggle button object, creating event handler for, 205–208
tombstoning, 602, 630
Toolbox tool
about, 23–24
adding button control, 54–55
adding controls using, 49–52
All XAML Controls category in, 59
Common Controls tab on, 502, 506
controls as XAML controls, 49
HyperLink control, 578–579
MediaElement control in, 58–60
opening, 49–52
using controls to display database information, 506–509
using TextBlock control, 50–52
Toolbox, Web Forms
about controls in, 552
Button controls, 562–563
GridView control, 574–578
Label controls, 562–563
RangeValidator control, 570
server controls in, 552–553
TextBox control, 561–562
webpage validator controls, 570
Toolbox, Windows Forms
about, 81
adding button controls, 86–88
adding number labels, 88–89
CheckBox control, 155
Common Controls category of, 149
DateTimePicker control in, 150
dialog box controls, 184–185
displaying, 86
GroupBox control in, 159
Label controls
adding to form, 175
in Windows Forms Toolbox, 88–89
setting properties, 95–97
ListBox control in, 159–164
MaskedTextBox control in, 346–348, 350–351, 506–509
PictureBox control
creating rectangle above check box, 161
creating rectangle beneath group box with, 161
in Windows Forms Toolbox, 89–90
ProgressBar control in, 416–417
RadioButton control, 160
ToolStrip control, 180–183
Toolbox, Windows Phone
about controls in, 612–614
Button controls, 614, 616
Image control, 612–614
TextBlock control, 614–617
Toolbox, XAML, controls. *See also* Windows Store apps, controls
AppBar, 236
AppBarButton, 236–238
AppBarToggleButton, 236, 238, 242
binding to data, 516–521
Canvas, 209–212
CommandBar, 236–238, 240–243
Flyout, 243–248
gesture support using, 259
Grid, 201
Image, 205
in Windows Phone Store, 601, 603
ListBox control, 522–526
ProgressBar control in, 417
ProgressRing, 417
TextBlock, 425
TextBox, 425, 517–521
ToggleButton, 203–206
Toolbox controls
Image, 612–614
TextBlock, 614–617
ToolStripButton1_Click event handler, 186–187
ToolStrip control, 180–183
tool windows, hiding, 38–39
ToString method, 316, 520, 521
touch input
gestures, 260–262
planning for, 259–263
ToUpper method, 321, 445
trapping errors
about, 375
comparing error handlers with defensive programming techniques, 393–394
Exception objects, 387–390
Exit Try statement, 394
processing errors using Try...Catch statement, 376–384
specifying retry period, 390–392
using Finally clause to perform cleanup tasks, 385
using nested Try...Catch blocks, 391–392
writing flash drive error handler, 384–385
triggering (firing) events, 65
TrimEnd method, 428, 445

Trim method, 321, 445
Try...Catch statement
  comparing error handlers with defensive programming techniques, 393–394
  Exception objects, 387–390
  Exit Try statement, 394
  nested, 377–378, 392–393
  processing errors using, 376–384
  specifying retry period, 390–392
  writing flash drive error handler, 384–385
TwoWay data access, 517
typed datasets, 500

U

UInteger data type
  about, 298
  ListBox control and, 301
ULong data type
  about, 298
  ListBox control and, 301
UnauthorizedAccessException object, 388
uncomment, 530
unhandled errors or exceptions, 377, 382
Unrecognized Database Format message, 496
Until keyword, in Do loop, 372–373
USB flash drives, writing error handler for, 384–385
user experience (UX), Microsoft, 236
user interface, creating for desktop apps, 85–87
user interface, designing
  adding button control, 54–55
  adding image, 56–58
  adding text blocks for random numbers, 49–50
  creating console applications
    about, 267
    in Visual Studio, 268–275
  creating new project, 45–47
  for data entry, 540–541
  for phone apps, 614–617
  navigating Designer, 48–49
  opening Toolbox, 49–52
  playing audio media, 58–60
  using TextBlock control, 50–52
users, validate using If...Then decision structure, 346–349
UShort data type
  about, 297
  ListBox control and, 301
UX (user experience), Microsoft, 236

V

validate users, using If...Then decision structure, 346–349
validator controls, 570
variables
  about, 118
  about declaring, 292
  assigning TextBox control to, 118–120
  Boolean, 346–348
  declaring as constant, 295–296
  explicitly declaring, 292–293
  guidelines for naming, 295–296
  implicitly declaring, 292
  measurement of, 297
  using assignment operator (=), 293
VBMath class, Rnd function in, 419
.vbproj (project file extension), 22
.vb (website code module files), 551
Vertical Split button, 30
videos, playing using MediaElement control, 139–141
virtual machine environment, Windows Phone SDK 8.0 on, 598
Visibility property
  setting, 61
  setting in desktop app, 98
  syntax for, 67
Visual Basic
  about, 4–7
  about upgrade, xvii
  advanced arithmetic operator, 308
  as event-driven programming language, 64
  data types in, 297
  design mode, 87
  events supported by objects in, 342
  formulas, 304, 314–315
  multiplatform approach to learning, 7
  operators
    arithmetic, 305–313
    binary, 350
    comparison, 343, 355–361
    relational, 324
    shortcut, 313
  running program from IDE, 67–68
Visual Basic Blank App (XAML) template, 196–197
Visual Studio
  about, 5
  about development environment, 19–21

creating console applications in
about, 268
interactive math games, 275–284
modules and procedures, 270–271
opening Console Application template, 269–270
Roll-The-Dice application, 271–275
Sub Main() procedure, 271–275
temperature conversion application, 271–275
exiting, 42
gesture support in, 259
menu commands pertaining to Windows store, 12
namespaces in programming terminology, 199
program statements, 66
Visual Studio 2013
about, 4, 79–80
Blend for
add controls in, 202
XAML in, 193
building executable file, 104–105
creating desktop app
adding number labels, 88–89
adding .wav file to Resources folder, 92–93
naming objects for clarity, 98
new project for, 83–85
picture box properties, 97–98
setting button properties, 93–94
setting descriptive label properties, 96–97
setting number labels properties, 95–96
setting title bar text of form, 97
SpinButton_Click event handler, 101–103
user interface, 85–87
using PictureBox control, 89–90
writing code, 99–101
databases and, 490
desktop apps and, 81–82
End as keyword in, 100
free versions of, 4
publishing desktop app, 105–107
running desktop app, 103–105
starting, 18–19
Visual Studio Code Editor. *See* Code Editor
Visual Studio Express for Web, 4
Visual Studio Express for Windows, 4
Visual Studio Express for Windows Desktop, 4
Visual Studio Express for Windows Phone, 4
Visual Studio Query Builder, creating SQL statements with, 509–514
Visual Studio website, 4
volume level, setting for media playback initial, 60

W

.wav file, adding to Resources folder, 92–93
Weather application, tiles in, 250–251
web applications. *See also* website, application
binding datasets to, 492
creating Windows Store apps with JavaScript in, 549–550
hosting ASP.NET web applications, 569
using ASP.NET MVC, 546
using ASP.NET Web Forms application, 545
Windows Store apps and, 548–549
web browsers, creating, 142–146
Web.config files, 551
web content, displaying live, 142–146
Web Designer
about, 552, 556–557
Design tab
about, 556
adding text, 557–558, 573
editing text, 581–582
including information and resources using, 570–572
inserting controls with, 561–563, 573
Source tab (Web Designer)
about, 556–557
viewing HTML and ASP.NET markup for webpage, 559–560
using, 557–560
web development, Visual Basic opportunities for, 6
Web Forms
ASP.NET
about, 545
building website with, 550–556
Web Pages vs., 552
Web Forms Toolbox
about controls in, 552
Button controls, 562–563
GridView control, 574–578
Label controls, 562–563
RangeValidator control, 570
RequiredFieldValidator control, 570
server controls in, 552–553
TextBox control, 561–562
webpage validator controls, 570
WebMatrix, 547
webpage controls
Button controls, 562–563
GridView control, 574–578
inserting, 561–563, 573

Label controls, 562–563
RangeValidator control, 570
TextBox control, 561–562
validator controls, 570
writing event handlers for, 563–569
Web Pages (with Razor), ASP. NET
about, 547
make up of, 559
Windows Forms vs., 552
web servers, Windows Azure applications for, 6
website, application. *See also* web applications
adding text in Design view (Web Designer), 557–558
building web forms, 550–556
creating, 554–557
customizing website template, 570–572
deploying application on web server, 569
displaying database records on webpage, 573–580
editing document and site master properties, 581–583
validating input fields on webpage, 570
writing event handlers for webpage controls, 563–569
WebView control, displaying live web content using, 141–146
Where clause, in LINQ queries, 437, 440–443, 448, 456, 534–535
While keyword, in Do loop, 372–373
Windows 8.1 design
creating command bar to manage tasks, 236–243
designing custom tiles for apps
about, 249
Assets folder for, 249
required tiles for, 249–257
live tiles
about, 257
in Microsoft Weather application, 251
programming, 257–259
receiving notification, 258
Microsoft user experience guidelines for, 236
planning for touch input, 259–263
security and permissions settings, 263–266
using charms, 239
Windows 8.1, Visual Basic opportunities on, 6
Windows 8, Windows 7, and Windows Server, Visual Basic opportunities on, 6
Windows Azure applications for web servers and cloud
Visual Basic opportunities for, 6
Windows Common Language Runtime (CLR), 544
Windows desktop apps
about, 79
building executable file, 104–105
ColorDialog control properties, 187
controls
CheckBox, 155–159
DateTimePicker, 148–154
GroupBox, 159–164
ListBox, 164–168, 355
ListBox control, 297–304
MenuStrip, 169–183
creating
adding number labels, 88–89
adding .wav file to Resources folder, 92–93
naming objects for clarity, 98
new project for, 83–85
picture box properties, 97–98
setting button properties, 93–94
setting descriptive label properties, 96–97
setting number labels properties, 95–96
setting title bar text of form, 97
SpinButton_Click event handler, 101–103
user interface, 85–87
using PictureBox control, 89–90
writing code, 99–101
database controls for
about, 489
displaying database information using toolbox controls, 506–509
programming with ADO.NET, 490–505
SQL statements and fi ltering data, 509–514
dialog box controls, 184–185
event handlers managing common dialog boxes, 185–190
MaxLength property of TextBox controls, 363
program crashes in, 383
publishing, 105–107
running, 103–105
starting, 80–81
TextBox control, using loop to fill text box with string data, 362
Visual Studio 2013 and, 81–82
Windows Explorer, 286
Windows Forms
about, 79–80
about Visual Studio and, 81
building executable file, 104–105
ColorDialog control properties, 187

controls
about, 147–154
CheckBox, 155–159
GroupBox, 159–164
ListBox, 164–168, 297–304, 355
MenuStrip, 169–180
ToolStrip, 180–183
creating desktop app
adding number labels, 88–89
adding .wav file to Resources folder, 92–93
naming objects for clarity, 98
new project for, 83–85
picture box properties, 97–98
setting button properties, 93–94
setting descriptive label properties, 96–97
setting number labels properties, 95–96
setting title bar text of form, 97
SpinButton_Click event handler, 101–103
user interface, 85–87
using PictureBox control, 89–90
writing code, 99–101
desktop apps and, 81
dialog box controls, 184–185
editing app with IDE, 460–464
event handlers managing common dialog boxes, 185–190
limitations of, 107
.NET Framework and, 81
publishing desktop app, 105–107
running desktop app, 103–104, 105
writing apps with Data Source Configuration Wizard, 492–505
Windows Forms Designer
about, 81
binding masked text box control to dataset object, 506–509
creating user interface, 85–87
displaying information in dataset with, 501
Windows Forms Toolbox
about, 81
adding button controls, 86–88
adding number labels, 88–89
CheckBox control, 155
Common Controls category of, 149
DateTimePicker control in, 150
dialog box controls, 184–185
displaying, 86
Label controls
adding to form, 175
in Windows Forms Toolbox, 88–89
Label properties, setting, 95–97
ListBox control in, 159–164
MaskedTextBox control in, 346–348, 350–351, 506–509
MenuStrip control, 170
PictureBox control
about, 89–90
creating rectangle above check box, 161
creating rectangle beneath group box with, 161
drawing square object on form, 184
ProgressBar control in, 416–417
RadioButton control, 160
ToolStrip control, 180–183
Windows, getting license for developers, 18
Windows Library for JavaScript (WinJS), 549
Windows Lock screen, badge notification appearing in, 250
Windows Movie Maker, 141
Windows Phone 8
features of, 589–592
installing apps, 595
Microsoft Silverlight and, 112
Visual Basic opportunities on, 6
website for, 591
Windows Phone 8 development
about, 587
app life cycle considerations, 626–636
closing apps, 626–627
creating apps
about, 607
adding Image control, 612–614
adjusting settings in PhoneApplicationPage, 611
creating new project, 607–608
designing user interface, 614–617
exploring IDE, 609–610
mouse input, 619
writing code, 617–620
deactivating apps, 627
hardware requirements for, 590
IsolatedStorageSettings class, 636
opportunities in platform, 588
PhoneApplicationService class, 628–635
registering apps, 620–621
setting options in Windows Phone manifest file, 637–638
testing apps, 620–626
Windows Phone apps. *See also* Windows Phone Store
binding datasets to, 492

Windows Phone apps, controls
about, 601
button controls, adding to phone app, 614, 616
Image controls, adding to phone app page, 612–614
TextBlock control, 614–617
Windows Phone Audio Playback template, 607
Windows Phone Development Center, 595
Windows Phone Emulator, using, 621–626
Windows Phone manifest file, setting options in, 637–638
Windows Phone Software Development Kit (SDK)
about, 596
on virtual machine environment, 598
working with version 8.0, 596–599
Windows Phone Store
about, 590
accessing, 591–596
installing Windows Phone app, 595
planning for certification, 595–596
selling apps in, 595
setting properties for apps in, 611
Windows Phone Store vs., 600–604
Windows Store vs., 600–604
Windows Presentation Foundation (WPF)
ASP.NET Web Forms and, 545
binding datasets to, 492
controls as root of Windows Store app controls, 112–113
using for desktop apps in Visual Studio 2013 development suite, 82
XAML markup language and, 26, 82, 112
Windows Push Notifi cation Service (WNS), 258
Windows Server and Windows 7, Visual Basic opportunities on, 6
Windows Store
about, 8
accessing, 9–10
app listing page, 10
connecting datasets to, 492
Details page, 15
installing apps from, 10
offering free apps, 11
planning for certification, 12
price tier in, 10–11
requirements checklist, 12–15
resources for developers preparing for, 12
sales information, 10–11
Spotlight area of, 10
Windows Phone Store vs., 600–604
Windows Store apps
about, 7
about programming, 44–45
adjusting Background property in, 201–202
binding datasets to, 492
check spelling in, 124–127
creating for web with JavaScript, 549–550
data sources for, 516
designing
custom tiles for, 249–257
live tiles, 257–259
planning for touch input, 259
presenting users with “chrome” features in, 236
replacing traditional menu bars and toolbars, 236–243
security and permissions settings, 263–266
size of tiles for, 250
designing custom tiles for
about, 249
Assets folder for, 249
required tiles for, 249–257
edit XAML markup in MainPage.xaml, 200–202
examining App.xaml, 197–200
exception handling, 383
live tiles
about, 257
in Microsoft Weather application, 251
programming, 257–259
receiving notification, 258
planning for touch input, 259
presenting users with “chrome” features in, 236
replacing traditional menu bars and toolbars, 236–243
running and testing, 30–33
security and permissions settings, 263–266
size of tiles for, 250
TextBox controls, using loop to fill text box with string data, 362–363
web applications and, 548–549
XAML styles
about, 215–216
building new styles from existing styles, 228–231
creating, 217–221
IDE shortcuts for applying, 231–232
practicing, 221–227
referencing, 220
StandardStyles.xaml, 216–217
using explicit and implicit, 220–221

XML documents in
about, 526
locating child elements in XML hierarchy, 532–533
opening and displaying contents of, 527–530
reading selection of tagged elements, 530–532
searching for items in file, 533–536
Windows Store apps, controls. *See also* Toolbox, XAML, controls
about, 111, 191
about understanding, 112–113
AppBar, 236
AppBarButton, 236–238
AppBarToggleButton, 236, 238, 242
binding to data using XAML, 516–526
Canvas controls, 209
CommandBar, 236–238, 240–243
displaying live web content using WebView control, 141–146
FlipView control, 127–132
Flyout, 243–248
gesture support using, 259
Image controls
about, 55
adding images to program, 56–58
creating, 205
style set for, 219
Label controls
in Windows Forms Toolbox, 88–89
Label properties, setting, 95–97
ListBox control, 522–526
MediaElement control
about, 133
controlling playback, 135–138
playing music using, 133–135, 240–243
playing videos using, 139–141
using with Flyout control, 245
PictureBox control, drawing square object on form, 184
TextBlock controls, adding text blocks for random numbers, 52–54
TextBox controls
assigning to variable, 118–120
binding control to class using, 517–521
check spelling in, 124–127
multiline, 120–124
opening and displaying contents of XML documents, 517–521
receiving input using, 113–118
ToggleButton control, 203–206
Windows Store apps, creating
building executable file, 74–77
Canvas controls, 209
creating splash screen, 70–73
designing user interface
adding button control, 54–55
adding image, 56–58
adding text blocks for random numbers, 49–50
creating new project, 45–47
navigating Designer, 48–49
opening Toolbox, 49–50
playing audio media, 58–60
using TextBlock control, 50–52
Image control, 205
new project for, 197–200
running program, 67–68
setting background color of page, 62
setting Visibility property, 61
SpinButton_Click event handler, 67–68
ToggleButton control, 203–206
using Code Editor, 63–66
using Save All command, 62–63
Windows Store developers
planning for certification, 12
registering as, 11
Windows Store requirements checklist, 12–15
WinJS (Windows Library for JavaScript), 549
WNS (Windows Push Notifi cation Service), 258
WPF (Windows Presentation Foundation)
ASP.NET Web Forms and, 545
binding datasets to, 492
controls as root of Windows Store app controls, 112–113
using for desktop apps in Visual Studio 2013 development suite, 82
XAML markup language and, 26, 82, 112
WriteLine method, 272
in Find-The-Number console application, 276–277
in simulate rolling dice console application, 280–281
Write method
in Find-The-Number console application, 276–277
in simulate rolling dice console application, 280–281
Xor operator and, 337
writing code. *See* Code Editor

X

XAML (Extensible Application Markup Language)
- about, 191–192
- All XAML Controls category, 59
- as root of Windows Store app controls, 112–113
- data binding expressed as markup extension, 517
- defining list box using, 357–361
- elements
  - about, 194–196
  - adding using tab of Code Editor, 202–212
- examining project files, 196–202
- Grid element, 201
- introduction to, 192–202
- markup to define FlipView control, 129
- namespaces in, 196, 199
- resource dictionary, 218
- root element in documents, 198
- styles
  - about, 215–216
  - building new styles from existing styles, 228–231
  - creating, 217–221
  - IDE shortcuts for applying, 231–232
  - practicing, 221–227
  - referencing, 220
  - StandardStyles.xaml, 216–217
  - using explicit and implicit, 220–221
- tab of Code Editor
  - about, 23
  - adding elements using, 202–212
  - adjusting Background property in, 201–202
  - displaying markup in Designer window, 25–30
  - examining XAML project files, 198–200
  - setting property for text box object, 196
- < tag and /> tag in markup, 195
- Toolbox controls
  - about, 49
  - AppBar, 236
  - AppBarButton, 236–238
  - AppBarToggleButton, 236, 238, 242
  - binding to data using, 516–526
  - Canvas, 209–212
  - CommandBar, 236–238, 240–243
  - Flyout, 243–248
  - gesture support using, 259–260
  - Grid, 201
  - Image, 205, 612–614
  - in Windows Phone Store, 601, 603
  - ListBox, 522–526
  - ProgressRing, 417
  - TextBlock, 425, 614–617
  - TextBox, 425, 517–526
  - ToggleButton, 203–206
- WPF and, 26, 82, 112

Xbox 360, Visual Basic opportunities for, 6
x: characters, namespaces prefaced by, 196
XDocument class, 526–527, 529, 532–533, 537
XDocument object, 538
XElement class, 527, 532–533
XElement object, 537, 538
XML (Extensible Markup Language)
- about, 454, 490
- documents
  - about, 515, 526
  - accessing data in, 526–540
  - adding node with data to, 538–540
  - locating child elements in XML hierarchy, 532–533
  - modifying element in, 537–538
  - opening and displaying contents of, 527–530
  - reading selection of tagged elements, 530–532
  - searching for items in file, 533–536
- elements
  - locating in XML hierarchy child, 530–532
  - modifying, 537–538
  - reading selection of tagged, 530–532
- files
  - about, 526
  - reading, 526–533
  - searching for items in, 533–536
  - writing to, 536–540
- using LINQ with, 454–458
- vs. Microsoft Access .mdb format, 454

XML schema file, 491–492
XmlTestButton_Click event handler, 530
Xor operator, 334–339, 349–350

Z

Zoom control, 115
Zoom drop-down button, 49
zoom in and out, on touch-enabled screen, 261–262
Zoom tool, in Designer, 49

About the author

MICHAEL HALVORSON is the author or co-author of more than 35 books, including *Start Here! Learn Microsoft Visual Basic 2012*, *Microsoft Visual Basic 2010 Step by Step*, *Microsoft Office XP Inside Out*, and *Microsoft Visual Basic 6.0 Professional Step By Step*. He has been the recipient of numerous non-fiction writing awards, including the Computer Press Best How-to Book Award (Software category) and the Society for Technical Communication Excellence Award (Writing category). Halvorson earned a bachelor's degree in Computer Science from Pacific Lutheran University in Tacoma, Washington, and master's and doctoral degrees in History from the University of Washington in Seattle. He was employed at Microsoft Corporation from 1985 to 1993, and he has been an advocate for Visual Basic programming since the product's original debut at Windows World in 1991. Halvorson is currently an associate professor at Pacific Lutheran University. You can learn more about his books and ideas at *http://michaelhalvorsonbooks.com*.