

Alessandro Del Sole

Visual Basic 11

UNLEASHED



SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Alessandro Del Sole

Visual Basic® 2012

UNLEASHED

SAMS

800 East 96th Street, Indianapolis, Indiana 46240 USA

Visual Basic® 2012 Unleashed

Copyright © 2013 by Pearson Education, Inc.

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

ISBN-10: 0-672-33631-6

ISBN-13: 978-0-672-33631-7

Library of Congress Cataloging-in-Publication Data is on file.

Printed in the United States of America

First Printing: January 2013

Trademarks

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

Warning and Disclaimer

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

Bulk Sales

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

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

For sales outside of the U.S., please contact:

International Sales
+1-317-581-3793
international@pearsontechgroup.com

Editor-in-Chief

Greg Wiegand

Executive Editor

Neil Rowe

Acquisitions Editor

Brook Farling

Development Editor

Mark Renfrow

Managing Editor

Kristy Hart

Project Editor

Anne Goebel

Copy Editor

Megan Wade

Indexer

WordWise Publishing
Services

Proofreader

Debbie Williams

Technical Editor

Matt Kleinwaks

Publishing Coordinator

Cindy Teeters

Cover Designer

Anne Jones

Compositor

Nonie Ratcliff

Contents at a Glance

Part I	Learning the Basics of VB	
1	Introducing the .NET Framework 4.5	1
2	Getting Started with the Visual Studio 2012 IDE	11
3	The Anatomy of a Visual Basic Project	63
4	Data Types and Expressions	89
5	Debugging Visual Basic 2012 Applications	179
6	Handling Errors and Exceptions	207
Part II	Object-Oriented Programming with Visual Basic 2012	
7	Class Fundamentals	225
8	Managing an Object's Lifetime	269
9	Organizing Types Within Namespaces	283
10	Modules	301
11	Structures and Enumerations	305
12	Inheritance	323
13	Interfaces	347
14	Generics and Nullable Types	367
15	Delegates and Events	379
16	Working with Collections and Iterators	393
17	Creating Objects: Visual Tools and Portable Libraries	423
Part III	Advanced Language Features	
18	Manipulating Files and Streams	453
19	The My Namespace	477
20	Advanced Language Features	511
Part IV	Data Access with ADO.NET and LINQ	
21	Introducing ADO.NET and DataSets	539
22	Introducing LINQ	549
23	LINQ to Objects	557
24	LINQ to SQL	587

25	LINQ to DataSets	621
26	Introducing ADO.NET Entity Framework	629
27	Manipulating XML Documents with LINQ and XML Literals	671
 Part V Building Windows Desktop Applications		
28	Creating WPF Applications	693
29	WPF Common Controls	725
30	Brushes, Styles, Templates, and Animations in WPF	757
31	Manipulating Media and Documents	793
32	Introducing Data-Binding	811
33	Localizing Applications	841
 Part VI Building Web Applications		
34	Building ASP.NET Web Applications	851
35	Publishing ASP.NET Web Applications	883
36	Building Rich Internet Applications with Silverlight 5	893
37	Building and Deploying Applications for Windows Azure	929
38	Building Apps for Windows Phone 7.5	955
 Part VII Networking and Exposing Data Through Networks		
39	Creating and Consuming WCF Services	991
40	Implementing and Consuming WCF Data Services	1013
 Part VIII Advanced .NET Framework with VB 2012		
41	Serialization	1035
42	Processes and Multithreading	1057
43	Parallel Programming and Parallel LINQ	1069
44	Asynchronous Programming	1103
45	Working with Assemblies	1143
46	Reflection	1157
47	Coding Attributes	1181
48	Platform Invokes and Interoperability with the COM Architecture	1191
49	Documenting the Source Code with XML Comments	1207

Part IX Applications Deployment

50	Understanding the Global Assembly Cache	1221
51	Setup and Deployment Projects with InstallShield for Visual Studio	1229
52	Deploying Applications with ClickOnce	1245

Part X Mastering the Visual Studio 2012 IDE

53	Advanced IDE Features	1261
54	Introducing the Visual Studio Extensibility	1287
55	Advanced Analysis Tools	1309
56	Testing Code with Unit Tests, Test-Driven Development, and Code Contracts	1337

Appendix

A	Useful Resources and Tools for Visual Basic	1357
	Index	1361

Table of Contents

Part I Learning the Basics of VB

1	Introducing the .NET Framework 4.5	1
	What Is the .NET Framework?	1
	Where Is the .NET Framework?	2
	The .NET Framework Architecture	2
	Differences Between .NET 4.0 and .NET 4.5	3
	The Common Language Runtime	4
	Writing Managed Code	4
	.NET Assemblies	4
	The Base Class Library	5
	Programming Languages Included in .NET 4.5	6
	Additional Tools Shipping with the .NET Framework	7
	Windows Software Development Kit	7
	What's New in .NET Framework 4.5	7
	How .NET Meets Windows 8 and the Windows Runtime	8
	Summary	9
2	Getting Started with the Visual Studio 2012 IDE	11
	What's New in Visual Studio 2012	11
	Status Bar and Start Page	12
	Get Started Tab	14
	The Latest News Tab	14
	Working with Projects and Solutions	16
	Creating Visual Basic Projects	16
	Multi-targeting	18
	Accessing Recent and Online Templates	19
	Searching for Installed Templates	20
	Finding Code Samples on the Internet	21
	Creating Reusable Projects and Items Templates	22
	Creating Your First Visual Basic 2012 Project	22
	Finding Visual Basic Projects	24
	Working with the Code Editor	24
	Working with Tool Windows	27
	The Solution Explorer Window	28
	The Error List Window	30

The Properties Window	31
The Output Window	31
My Project	33
Application Tab	34
Compiling Projects	39
Debug and Release Configurations	40
Other Compile Options	43
Advanced Compile Options	46
Debugging Overview	48
Debugging an Application	48
Breakpoints and Data Tips	50
Runtime Errors	53
Edit and Continue	54
Browsing the Visual Basic and .NET Documentation	55
Online Help and the MSDN Library	56
Object Browser Window	56
Quick Launch Tool	59
Showing the Hierarchy of Method Calls	60
Summary	61
3 The Anatomy of a Visual Basic Project	63
Brief Overview of Types and Members	63
Classes	64
Properties	64
Methods	64
Modules	65
Structures	65
Inheritance	65
Namespaces	66
Accessing Members	67
Imports Directives	68
Region Directives	69
Attributes	69
Implicit Line Continuation	70
Visual Basic 2012 Reserved Keywords	72
Understanding Project Files	74
Dissecting My Project	74
Application.MyApp	75
AssemblyInfo.vb	76
Resources and the Resources.resx File	77
Application Settings	81

Understanding References	83
Adding References to COM Libraries	85
Deploy Without PIAs	86
Final Considerations	88
Summary	88

4 Data Types and Expressions 89

Introducing the Common Type System	89
Everything Is an Object	90
Introducing Value Types and Reference Types	90
System.Object and System.ValueType	91
Understanding Value Types	92
.NET Framework Primitive Value Types	93
Using Value Types	95
Working with BigInteger	102
Building Custom Value Types	103
Understanding Reference Types	103
.NET Framework Primitive Reference Types	105
Differences Between Value Types and Reference Types	106
Memory Allocation	106
Object-Oriented Differences	107
Performance Differences	110
What Custom Type Should I Choose?	111
Converting Between Value Types and Reference Types	111
Understanding Implicit Conversions	111
Boxing and Unboxing	113
Deep Copy and Shallow Copy	115
The GetType Keyword	119
Understanding Conversion Operators	120
Widening and Narrowing Conversions	120
Working with .NET Fundamental Types	125
Working with Strings	125
Working with Dates	137
Working with Time	143
Working with TimeZone and TimeZoneInfo	144
Working with GUIDs	147
Working with Arrays	148
Common Operators	155
Arithmetic Operators	155
Assignment Operators	157
Logical, Bitwise, and Shift Operators	158

Concatenation Operators	163
Comparison Operators	163
Iterations, Loops, and Conditional Code Blocks	166
Iterations	166
Loops	170
Conditional Code Blocks	172
Constants	175
With...End With Statement	176
Summary	177
5 Debugging Visual Basic 2012 Applications	179
Preparing an Example	179
Debugging Instrumentation	180
Debugging in Steps	180
Mixed Mode Debugging	182
“Just My Code” Debugging	182
Working with Breakpoints and Trace Points	184
Locals Window	187
Command Window	187
Call Stack Window	188
Watch Windows	189
Threads Window	191
Autos Window	192
Inspecting Object Details with Debugger Visualizers	192
Debugging in Code	193
The Debug Class	194
The Trace Class	195
Understanding Trace Listeners	196
Using Debug Attributes in Your Code	202
Summary	206
6 Handling Errors and Exceptions	207
Introducing Exceptions	207
Handling Exceptions	208
Tips for Visual Basic 6 Migration	209
System.Exception, Naming Conventions, and Specialization	209
Handling Exceptions with Try...Catch...Finally Blocks	209
The Throw Keyword	218
Catching Exceptions Without a Variable	223
Summary	224

Part II Object-Oriented Programming with Visual Basic 2012

7 Class Fundamentals	225
Declaring Classes	225
Nested Classes	226
Fields	227
Avoiding Ambiguities with Local Variables	228
Properties	229
Read-Only Properties	231
Write-Only Properties	231
Exposing Custom Types	232
Accessing Properties	232
Default Properties	233
Types and Members Visibility: Scope	234
Executing Actions with Methods	235
Invoking Methods	236
Methods Arguments: <code>ByVal</code> and <code>ByRef</code>	237
Overloading Methods	242
Exit from Methods	246
Partial Classes	248
Partial Methods	251
Constructors	252
Overloading Constructors	255
Object Initializers	258
Shared Members	259
Shared Classes	259
Shared Fields	259
Shared Properties	260
Shared Methods	260
Shared Constructors	262
Common Language Specification	263
Where Do I Need to Apply?	263
Marking Assemblies and Types as CLS-Compliant	264
Naming Conventions	264
Rules About Classes	266
Rules About Properties	267
Rules About Methods	267
Rules About Arrays	267
Summary	267

8	Managing an Object's Lifetime	269
	Understanding Memory Allocation	269
	Understanding Garbage Collection	270
	Understanding the <code>Finalize</code> Method	271
	Understanding <code>Dispose</code> and the <code>IDisposable</code> Interface	273
	<code>Using...End Using</code> Statement	275
	Putting <code>Dispose</code> and <code>Finalize</code> Together	275
	Restoring Objects with Object Resurrection	278
	Advanced Garbage Collection	279
	Interacting with the Garbage Collector	279
	Understanding Generations and Operation Modes	280
	Summary	282
9	Organizing Types Within Namespaces	283
	Understanding Namespaces	283
	Organizing Types Within Namespaces	284
	Why Are Namespaces So Useful?	287
	Nested Namespaces	288
	Scope	291
	Root Namespace	291
	<code>Imports</code> Directives	292
	Namespaces and Common Language Specification	295
	Global Namespaces and the <code>Global</code> Keyword	295
	Summary	299
10	Modules	301
	Modules Overview	301
	Scope	302
	Differences Between Modules and Classes	303
	No Constructor	303
	No Inheritance Support	303
	No Interface Implementation	303
	Summary	303
11	Structures and Enumerations	305
	Understanding Structures	305
	Assigning Structures to Variables	308
	Passing Structures to Methods	308
	Members' Visibility	308
	Inheritance Limitations and Interface Implementation	309

Memory Allocation	309
Organizing Structures	310
Overloading Operators	310
Overloading CType	313
Structures and Common Language Specification	314
Enumerations	315
Using Enumerations	315
Useful Methods from System.Enum	316
Using Enums As Return Values from Methods	319
Enum Values As Bit Flags	320
Enumerations and Common Language Specification	320
Summary	321
12 Inheritance	323
Applying Inheritance	324
Illustrating System.Object in Detail	328
Introducing Polymorphism	329
Overriding Members	331
NotOverridable Keyword	334
Overloading Derived Members	334
Conditioning Inheritance	334
NotInheritable Keyword	335
MustInherit and MustOverride Keywords	336
Accessing Base Classes Members	337
MyBase Keyword	337
MyClass Keyword	339
Constructors' Inheritance	341
Shadowing	342
Overriding Shared Members	343
Practical Inheritance: Building Custom Exceptions	344
Summary	346
13 Interfaces	347
Defining Interfaces	347
Implementing and Accessing Interfaces	348
Passing Interfaces as Method Arguments	351
Interfaces and Polymorphism	352
Interfaces Inheritance	353
Defining CLS-Compliant Interfaces	354

Most Common .NET Interfaces	355
The IEnumerable Interface	356
The IComparable Interface	358
The IConvertible Interface	360
The IFormattable Interface	363
Summary	365
14 Generics and Nullable Types	367
Introducing Generics	367
Creating and Consuming Generics	368
Consuming Generic Types	370
Implementing Generic Methods	371
Understanding Constraints	372
Overloading Type Parameters	375
Introducing Nullable Types	376
Summary	377
15 Delegates and Events	379
Understanding Delegates	379
Declaring Delegates	380
Combining Delegates: Multicast Delegates	382
Handling Events	383
Registering for Events: AddHandler and RemoveHandler	383
Declaring Objects with the WithEvents Keyword	384
Offering Events to the External World	385
Raising Events	385
Creating Custom Events	389
Summary	391
16 Working with Collections and Iterators	393
Understanding Collections Architecture	394
Working with Nongeneric Collections	394
The ArrayList Collection	394
The Queue Collection	397
The Stack Collection	398
The Hashtable Collection	398
The ListDictionary Collection	399
The OrderedDictionary Collection	399
The SortedList Collection	400
The HybridDictionary Collection	400

The <code>StringCollection</code> Collection	400
The <code>StringDictionary</code> Collection	400
The <code>NameValueCollection</code> Collection	401
The <code>BitArray</code> Collection	401
The <code>BitVector32</code> Collection	402
Working with Generic Collections	403
The <code>List(Of T)</code> Collection	403
Working with Collection Initializers	405
The <code>ReadOnlyCollection(Of T)</code> Collection	406
The <code>Dictionary(Of TKey, TValue)</code> Collection	407
The <code>SortedDictionary(Of TKey, TValue)</code> Collection	408
The <code>ObservableCollection(Of T)</code> Collection	408
The <code>ReadOnlyObservableCollection(Of T)</code> Collection	410
The <code>LinkedList(Of T)</code> Collection	410
The <code>Queue(Of T)</code> and <code>Stack(Of T)</code> Collections	412
Building Custom Collections	413
Concurrent Collections	413
Iterators in Visual Basic	414
Understanding the Benefits of Iterators in Code	415
Simple Iterators	417
Exiting from Iterators	418
Iterators with <code>Try..Catch..Finally</code>	418
Anonymous Iterators	419
Implementing an Iterator Class	419
Summary	422
17 Creating Objects: Visual Tools and Portable Libraries	423
Visual Studio Class Designer	424
Enabling the Class Designer	424
Adding and Designing Objects	425
Implementing Derived Classes	429
Creating Multiple Diagrams	431
Exporting the Diagram	432
Class View Window	432
Generate from Usage	433
Generating Shared Members	436
On-the-Fly Code and Object Initializers	437
Generating Complex Objects	437
Creating Portable Classes	440
Creating a Sample Portable Library	442
Summary	451

Part III Advanced Language Features

18	Manipulating Files and Streams	453
	Manipulating Directories and Pathnames	453
	The <code>System.IO.Path</code> Class	454
	The <code>System.IO.Directory</code> Class	455
	The <code>System.IO.DirectoryInfo</code> Class	458
	The <code>System.IO.DriveInfo</code> Class	459
	Handling Exceptions for Directories and Pathnames	459
	Manipulating Files	460
	The <code>System.IO.File</code> Class	460
	The <code>System.IO.FileInfo</code> Class	462
	Handling File Exceptions	463
	Understanding Permissions	463
	Introducing Streams	464
	Reading and Writing Text Files	464
	Reading and Writing Binary Files	465
	Using Memory Streams	466
	Using Streams with Strings	467
	Compressing Data with Streams	467
	Networking with Streams	474
	Summary	475
19	The <code>My</code> Namespace	477
	Introducing the <code>My</code> Namespace	477
	<code>My.Application</code>	478
	Retrieving Assembly Information	478
	Working with Cultures	479
	Deployment and Environment Information	480
	<code>My.Computer</code>	482
	Working with the File System	483
	Working with the Clipboard	484
	Playing Audio Files	485
	Managing the Keyboard	485
	Working with the Registry	486
	Accessing the Network	487
	Getting Computer Information	488
	<code>My.Settings</code>	490
	<code>My.Settings</code> Events	495
	<code>My.Resources</code>	497
	Getting Resources by Name in Code	500

My.User	500
My.WebServices	502
Extending My	502
Extending My.Application and My.Computer	504
Extending My.Resources and My.Settings	506
My in Different Applications	506
Understanding Application Events	509
Summary	510
20 Advanced Language Features	511
Local Type Inference	511
Option Infer Directive	513
Local Type Inference Scope	514
Array Literals	515
Bug Fix: Return Type in Array Literals	515
Multidimensional and Jagged Arrays	516
Extension Methods	517
Coding Custom Extension Methods	521
Exporting Extension Methods	523
Anonymous Types	524
Relaxed Delegates	526
Lambda Expressions	526
Type Inference and Lambda Expressions	529
Multiline Lambdas	530
Sub Lambdas	531
Lexical Closures	532
Ternary If Operator	533
Generic Variance	535
Covariance	535
Contra Variance	536
Summary	537
Part IV Data Access with ADO.NET and LINQ	
21 Introducing ADO.NET and DataSets	539
System Requirements	539
Introducing ADO.NET	540
Data Providers	540
Connection Modes	541
Understanding Connections and Data Readers	541
Introducing DataSets	543
Creating DataSets	543
Summary	547

22	Introducing LINQ	549
	What Is LINQ?	549
	LINQ Examples	551
	Language Support	552
	Understanding Providers	553
	Overview of LINQ Architecture	554
	Summary	555
23	LINQ to Objects	557
	Introducing LINQ to Objects	557
	Querying in Memory Objects	558
	Understanding Deferred Execution	565
	Introducing Standard Query Operators	568
	Projection Operators	568
	Restriction Operators	569
	Aggregation Operators	570
	Understanding the <code>Let</code> Keyword	572
	Conversion Operators	572
	Generation Operators	574
	Ordering Operators	575
	Set Operators	576
	Grouping Operators	577
	Union Operators	579
	Equality Operators	582
	Quantifiers	582
	Concatenation Operators	583
	Elements Operators	583
	Partitioning Operators	584
	Summary	586
24	LINQ to SQL	587
	Introducing LINQ to SQL	588
	Prerequisites and Requirements for This Book	588
	Understanding LINQ to SQL Classes	589
	Behind the Scenes of LINQ to SQL Classes	599
	Querying Data with LINQ to SQL	600
	Insert/Update/Delete Operations with LINQ	604
	Inserting Entities	605
	Updating Entities	608
	Deleting Entities	609
	Mapping Stored Procedures	610
	Using the Log	613

Advanced LINQ to SQL	613
Custom Validations	614
Handling Optimistic Concurrency	616
Using SQL Syntax Against Entities	617
LINQ to SQL with SQL Server Compact Edition 3.5	617
Writing the Connection String	619
Summary	619
25 LINQ to DataSets	621
Querying Datasets with LINQ	621
Building Complex Queries with Anonymous Types	623
LINQ to DataSets' Extension Methods	624
Understanding CopyToDataTable	624
Understanding Field(Of T) and SetField(Of T)	626
Summary	627
26 Introducing ADO.NET Entity Framework	629
Introducing Entity Framework	629
Understanding Entity Data Models	630
Understanding the DbContext Class: The Visual	
Basic Mapping	638
Entity Designer Tool Windows	640
Insert/Update/Delete Operations for Entities	645
Instantiating the DbContext	645
Adding Entities	646
Deleting Entities	647
Updating Entities	648
Handling Optimistic Concurrency	648
Validating Data	650
Querying EDMs with LINQ to Entities	652
Querying EDMs with Entity SQL	653
Mapping Stored Procedures	654
Introducing the Code First Approach	657
Downloading Additions to the Entity Framework 5	658
Coding Your Model	659
Generating the Database and Executing Data Operations	660
Introducing Data Annotations	663
Introducing the Fluent APIs	665
Compatibility with the Past and with Other Technologies	668
Summary	669

27	Manipulating XML Documents with LINQ and XML Literals	671
	Introducing LINQ to XML	672
	The <code>System.Xml.Linq</code> Namespace	672
	Writing XML Markup in VB with XML Literals	677
	LINQ Queries with XML Literals	680
	Understanding Embedded Expressions	682
	XML Schema Inference	685
	Summary	691
Part V	Building Windows Desktop Applications	
28	Creating WPF Applications	693
	What Is WPF?	694
	Improvements in WPF 4.5	695
	WPF and Windows 8: The Future of Desktop Development	695
	Introducing the WPF Architecture	696
	Building WPF Applications with Visual Studio 2012	697
	Understanding the eXtensible Application Markup Language	699
	Declaring and Using Controls with the Designer and XAML	701
	Understanding Visual Tree and Logical Tree	704
	Handling Events in WPF	706
	A More Thorough Discussion: Introducing the Routed Events	707
	Arranging Controls with Panels	709
	The Grid Panel	709
	The StackPanel Panel	711
	The WrapPanel Panel	713
	The Canvas Panel	714
	The DockPanel Panel	714
	The ViewBox Panel	716
	Managing Windows	716
	Instantiating Windows at Runtime	718
	Introducing the Application Object	719
	Brief Overview of WPF Browser Applications	721
	Summary	724
29	WPF Common Controls	725
	Introducing WPF Controls Features	725
	Understanding the ContentControl	726
	Understanding Common Controls	727
	The Border Control	727
	The Button Control	728

Showing Dates with the Calendar Control	728
Items Selection with the CheckBox Control	729
Selecting Values from a List with the ComboBox Control	730
Presenting Tabular Data with the DataGrid Control	731
Selecting Dates with the DatePicker Control	732
Viewing XPS Documents with the DocumentViewer Control	733
Drawing Shapes: The Ellipse	733
Organizing Controls with the Expander	734
Frame	734
Organizing Controls with the GroupBox Control	735
Displaying Images with the Image Control	736
Displaying Text Messages with the Label Control	736
Presenting Data with the ListBox Control	736
Presenting Data with the ListView Control	738
Playing Audio and Video with the MediaElement Control	739
Building Effective User Interfaces with the Menu Control	740
Entering Passwords with the PasswordBox Control	741
Showing the Progress of an Operation with the ProgressBar Control	743
Accepting User Choices with the RadioButton Control	744
Drawing Shapes: The Rectangle	745
Editing Text with the RichTextBox Control	745
Extended View with the ScrollBar Control	745
Scrolling the Visual Tree with the ScrollViewer Control	746
Separating Visual Elements with the Separator Control	746
Value Selection with the Slider Control	747
Displaying Information with the StatusBar Control	747
Organizing User Interfaces with the TabControl Control	748
Presenting Text with the TextBlock Control	749
Entering Text with the TextBox Control	750
Offering Commands with the ToolBar Control	750
Presenting Hierarchical Data with the TreeView Control	751
Accessing the Web with the WebBrowser Control	752
Windows Forms Interoperability with the WindowsFormsHost Control	753
Using Common Dialogs	754
Summary	755
30 Brushes, Styles, Templates, and Animations in WPF	757
Introducing Brushes	757
Applying a SolidColorBrush	759
Applying a LinearGradientBrush	760
Applying a RadialGradientBrush	762

Applying an ImageBrush	762
Applying SelectionBrush and CaretBrush	765
Applying a VisualBrush	767
Applying a DrawingBrush	768
Applying a BitmapCacheBrush	769
Introducing Styles	770
Styles Inheritance	773
Understanding Triggers	773
Introducing Control Templates	775
Introducing Transformations	778
Rotating Visual Elements with RotateTransform	780
Dynamically Resizing Visual Elements with ScaleTransform	780
Changing Visual Elements' Angles with SkewTransform	780
Dynamically Moving Visual Elements with TranslateTransform	781
Applying Multiple Transforms	782
Introducing Animations	782
Applying DoubleAnimation	783
Applying ColorAnimation	786
Working with Animation Events	787
Creating Animations with Visual Basic	789
Summary	791
31 Manipulating Media and Documents	793
Viewing Images	793
Playing Media	795
Manipulating Documents	799
Understanding the RichTextBox Control	806
Viewing XPS Documents	808
Summary	809
32 Introducing Data-Binding	811
Introducing the Data-Binding in WPF	811
Binding UI Elements with the Binding Markup Extension	812
Understanding the DataGrid and the ObservableCollection	814
Discussing the Drag'n'Drop Data-Binding	818
Creating Tabular Data Forms	819
Creating Master-Details Forms	826
Understanding Views and Binding Lists	830
Implementing String Formatters and Value Converters	835
Summary	840

33	Localizing Applications	841
	Introducing .NET Localization	841
	Windows Forms Localization	842
	WPF Localization	844
	Preparing the LocBaml Tool	845
	Localizing a WPF Application	845
	Summary	850

Part VI Building Web Applications

34	Building ASP.NET Web Applications	851
	Introducing the ASP.NET Model	851
	Understanding Page Requests	852
	Scalability and Performance	852
	Available Project Templates	854
	Web Forms and Master Pages	855
	Web Forms	855
	ASP.NET Controls	858
	Server Controls	858
	HTML Controls	859
	Handling Events	860
	Understanding State Management	861
	The Application State	861
	The Cache State	862
	The Context State	863
	Using Cookies for Saving Information	863
	The Session State	864
	The ViewState State	864
	Creating a Web Application with VB 2012 with Navigation and Data-Binding	864
	Master Pages	865
	Adding the Data Model	868
	Adding a New Web Form	868
	Adding Data Controls	870
	Adding Filtering Capabilities	873
	Adding Navigation Controls	874
	Running the Application	875
	New in ASP.NET 4.5: Strongly Typed Data Controls and Model Binding	875
	Configuring a Web Application for Security	879
	Summary	882

35	Publishing ASP.NET Web Applications	883
	Deployment Overview	883
	The 1-Click Deployment	884
	Classic Publishing	884
	MSDeploy Publish	886
	Understanding Packages	886
	Web Deploy with MSDeploy	887
	Packaging Web Applications for Manual Installation	889
	Summary	891
36	Building Rich Internet Applications with Silverlight 5	893
	Introducing Silverlight	894
	Creating Silverlight Projects with Visual Basic 2012	895
	Adding Controls and Handling Events	897
	How Silverlight Applications Are Packaged	899
	Playing Media	900
	Animating UI Elements	905
	Introducing Navigation Applications	908
	Introducing WCF RIA Services	911
	Adding the Data Source	912
	Adding the Domain Service Class	913
	Data-Binding to Controls	916
	Running the Application	920
	Filtering Data with the <code>PivotViewer</code> Control	920
	“Out-of-Browser” Applications	923
	Elevated Permissions and Security Considerations	926
	XAML Debugging	926
	Additional New Features in Silverlight 5	928
	Summary	928
37	Building and Deploying Applications for Windows Azure	929
	Overview of the Windows Azure Platform	929
	Registering for the Windows Azure Developer Portal	931
	Downloading and Installing Tools for Visual Studio	931
	Additional Tools	932
	Creating a Demo Project	933
	Understanding Web Roles and Web Configuration	934
	Building the Silverlight 5 Project	936
	Testing the Application Locally	942

Deploying Applications to Windows Azure	944
Introducing the Windows Azure Management Portal	949
Summary	952
38 Building Apps for Windows Phone 7.5	955
Introducing Windows Phone	955
Developer Registration to the Windows Phone Marketplace	956
Downloading the Developer Tools	957
Training and Learning Resources	958
The Windows Phone 7.5 Programming Model	958
Creating Apps with Visual Basic	959
Starting and Debugging Windows Phone Apps	963
Understanding Pages, Orientations, and the	
Navigation Framework	963
Using System Functionalities with Launchers	967
Showing Multiple Contents with the Panorama Control	974
Local Data Storage	980
Understanding the Application Bar	982
Accessing the Pictures Hub	982
Understanding the Execution Model	984
Setting Properties, Icons, and Splash Screen	985
Submitting Apps to the Marketplace	987
Summary	989
 Part VII Networking and Exposing Data Through Networks	
39 Creating and Consuming WCF Services	991
Introducing Windows Communication Foundation	992
Understanding Endpoints	992
Address, Binding, Contract: The ABC of WCF	992
Implementing WCF Services	993
Implementing Custom Logic for the WCF Service	997
Consuming WCF Services	1001
Creating the Client and Adding a Service Reference	1002
Understanding the Proxy Class	1002
Invoking Members from the Service	1003
Handling Exceptions in WCF	1007
Hosting WCF Services in Internet Information Services	1008
Configuring Services with the Configuration Editor	1009
Summary	1011

40	Implementing and Consuming WCF Data Services	1013
	What Are Data Services?	1013
	Querying Data via HTTP Requests	1014
	Open Data Protocol for WCF Data Services	1015
	Implementing WCF Data Services	1016
	Deploying WCF Data Services to Internet Information Services	1021
	Consuming WCF Data Services	1022
	Creating a Client Application	1022
	Querying Data	1026
	Implementing Service Operations	1027
	Implementing Query Interceptors	1030
	Understanding Query Interceptors	1030
	Understanding Change Interceptors	1032
	Understanding Server-Driven Paging	1033
	Summary	1034
Part VIII	Advanced .NET Framework with VB 2012	
41	Serialization	1035
	Objects Serialization	1036
	Binary Serialization	1036
	SOAP Serialization	1039
	Providing Serialization for Custom Objects	1040
	NonSerialized Events	1042
	XML Serialization	1043
	Customizing XML Serialization	1044
	Custom Serialization	1045
	Serialization Events	1047
	Serialization with XAML	1048
	Serialization in Windows Communication Foundation	1050
	JSON Serialization	1053
	Serialization in the ADO.NET Entity Framework	1053
	Summary	1054
42	Processes and Multithreading	1057
	Managing Processes	1058
	Querying Existing Processes	1059
	Introducing Multithreading	1060
	Creating Threads	1060
	Passing Parameters	1061

Understanding the .NET Thread Pool	1061
Getting and Setting Information in the Thread Pool	1063
Threads Synchronization	1063
The <code>SyncLock..End SyncLock</code> Statement	1064
Synchronization with the <code>Monitor</code> Class	1065
Read/Write Locks	1065
Summary	1067
43 Parallel Programming and Parallel LINQ	1069
Introducing Parallel Computing	1070
What's New in .NET 4.5: Custom Task Scheduling	1070
Introducing Parallel Classes	1071
Understanding and Using Tasks	1072
What Is a Task?	1072
Running Tasks with <code>Parallel.Invoke</code>	1072
Creating, Running, and Managing Tasks: The <code>Task</code> Class	1073
Creating Tasks That Return Values	1074
Exception Handling	1075
Canceling Tasks	1077
The <code>Barrier</code> Class	1078
Parallel Loops	1080
<code>Parallel.For</code> Loop	1081
<code>Parallel.ForEach</code> Loop	1083
Debugging Tools for Parallel Tasks	1086
Concurrent Collections	1087
<code>ConcurrentBag(Of T)</code>	1088
<code>ConcurrentQueue(Of T)</code>	1089
<code>ConcurrentStack(Of T)</code>	1089
<code>ConcurrentDictionary(Of TKey, TValue)</code>	1090
<code>BlockingCollection(Of T)</code>	1091
Introducing Parallel LINQ	1092
Simulating an Intensive Work	1093
Measuring Performances of a Classic LINQ Query	1093
Measuring Performances of a PLINQ Query	1095
Ordering Sequences	1096
<code>AsParallel</code> and Binary Operators	1097
Using <code>ParallelEnumerable</code>	1097
Controlling PLINQ Queries	1097
Handling Exceptions	1100
Summary	1101

44	Asynchronous Programming	1103
	Overview of Asynchrony	1104
	Before .NET 4.5: Event-based Asynchrony	1104
	Before .NET 4.5: The Asynchronous Programming Model	1106
	.NET 4.5: Introducing the Async Pattern	1107
	Where Do I Use Async?	1111
	When and Why to Use Async/Await and Comparisons with the TPL	1111
	Getting Started with Async/Await	1112
	The Synchronous Approach	1112
	Event-based Asynchrony and Callbacks	1116
	Asynchrony with Async/Await	1120
	How Async and Await Work Behind the Scenes	1122
	Documentation and Examples of the Async Pattern	1126
	Exception Handling in Async	1127
	Implementing Task-based Asynchrony	1127
	Switching Threads	1128
	Using Combinators	1129
	Cancellation and Progress	1131
	Implementing Cancellation	1131
	Reporting Progress	1134
	Asynchronous Lambda Expressions	1136
	Asynchronous I/O File Operations in .NET 4.5	1137
	Summary	1141
45	Working with Assemblies	1143
	Assembly Overview	1143
	Information Stored Within Assemblies	1144
	Assembly Location	1144
	Signing Assemblies	1145
	Assembly Information and Attributes	1145
	Understanding Application Domains	1145
	Creating Application Domains and Executing Assemblies	1145
	Security Model in .NET 4.5	1148
	Implementing and Requiring Permissions	1149
	The Transparency Level 2	1150
	Sandboxing	1152
	Conditional APTCA	1154
	Migrating from Old CAS-Based Code	1155
	Summary	1155

46	Reflection	1157
	Introducing Reflection	1157
	Understanding Assemblies' Metadata	1158
	Preparing a Sample Assembly	1159
	Getting Assembly Information	1160
	Reflecting Types	1162
	Reflecting a Single Type	1167
	Invoking Code Dynamically	1169
	Generating Code at Runtime with <code>Reflection.Emit</code>	1171
	Late Binding Concepts	1176
	Caller Information	1177
	Summary	1180
47	Coding Attributes	1181
	Applying Attributes	1181
	Coding Custom Attributes	1184
	Applying Custom Attributes	1186
	Applying Attributes Multiple Times	1187
	Defining Inheritance	1187
	Reflecting Attributes	1189
	Summary	1190
48	Platform Invokes and Interoperability with the COM Architecture	1191
	Importing and Using COM Objects	1192
	Importing COM Components into Visual Studio	1192
	Using COM Objects in Code	1195
	Catching Exceptions	1195
	Releasing COM Objects	1195
	Calling COM Objects from WPF	1196
	Exposing .NET Objects to the COM World	1197
	P/Invokes and Unmanaged Code	1199
	Understanding P/Invokes	1200
	Encapsulating P/Invokes	1201
	Converting Types to Unmanaged	1202
	The <code>StructLayout</code> Attribute	1203
	The <code>VBFixedString</code> Attribute	1205
	Handling Exceptions	1205
	References to the Win32 API Calls	1206
	Summary	1206

49	Documenting the Source Code with XML Comments	1207
	Understanding XML Comments	1208
	Enabling XML Comments	1209
	Implementing XML Comments	1210
	Defining Complex Code Documentation	1212
	Generating Compiled Help Files	1220
	Summary	1220

Part IX Applications Deployment

50	Understanding the Global Assembly Cache	1221
	The DLL Hell Problem	1221
	XCopy Deployment	1222
	The Global Assembly Cache	1222
	Installing and Uninstalling Assemblies	1223
	Signing Assemblies with Strong Names	1224
	Top Reasons for Installing (or Not) Assemblies to the GAC	1227
	Summary	1227
51	Setup and Deployment Projects with InstallShield for Visual Studio	1229
	Windows Installer Overview	1230
	Introducing InstallShield	1231
	Obtaining Your Copy of InstallShield LE	1231
	Creating a Setup Project	1232
	Application Information	1233
	Installation Requirements	1234
	Application Files	1236
	Application Shortcuts	1238
	Application Registry	1239
	Installation Interview	1241
	Specifying Environment Variables	1242
	Configuring the Setup Project	1242
	Building and Deploying the Windows Installer Package	1244
	Summary	1244
52	Deploying Applications with ClickOnce	1245
	Introducing ClickOnce	1245
	How ClickOnce Handles Applications	1246
	When to Use ClickOnce	1246

Deploying Applications with ClickOnce	1247
Understanding the Structure of a ClickOnce Deployment	1250
Configuring ClickOnce	1251
Application Files	1251
Prerequisites	1252
Publishing Application Updates	1252
Options	1253
Security Considerations	1255
Providing Certificates	1255
Programmatically Accessing ClickOnce	1257
Registration-Free COM	1258
Summary	1260

Part X Mastering the Visual Studio 2012 IDE

53 Advanced IDE Features	1261
Exporting Templates	1261
Exporting Project Templates	1261
Exporting Item Templates	1263
Customizing Visual Studio 2012	1267
Customizing the Tools Menu	1267
Customizing Commands and Toolbars	1268
Managing User Settings	1271
Exporting Settings	1271
Importing Settings	1273
Customizing the Toolbox	1275
Using, Creating, and Managing Reusable Code Snippets	1275
Consuming Code Snippets	1276
The Code Snippet Manager	1278
Creating and Consuming Custom Code Snippets	1279
Managing Libraries with NuGet	1283
Summary	1286
54 Introducing Visual Studio Extensibility	1287
Introducing Visual Studio Extensibility	1287
About Extensibility with Visual Studio 2012	1288
The Visual Studio 2012 SDK	1288
Building a Visual Studio Package	1289
Deploying Visual Studio Extensions	1299
Managing Extensions with Extensions and Updates	1302

Managing Add-Ins with the Add-In Manager	1304
Extending the Code Editor	1304
Summary	1308
55 Advanced Analysis Tools	1309
Introducing Analysis Tools	1309
Performing Code Analysis	1310
Calculating Code Metrics	1315
Code Clone Detection	1317
Profiling Applications	1319
Profiling External Executables	1327
Historical Debugging with IntelliTrace	1328
IntelliTrace Options	1329
Creating a Sample Application	1330
Tracking Application Events and Exceptions with IntelliTrace	1331
Analyzing IntelliTrace Logs	1333
Using IntelliTrace for Unit Tests	1334
Generating Dependency Graphs	1334
Summary	1336
56 Testing Code with Unit Tests, Test-Driven Development, and Code Contracts	1337
Testing Code with Unit Tests	1337
Creating Unit Tests	1338
Running Unit Tests	1342
Enabling Code Coverage	1343
Introducing Test Driven Development	1344
Creating a Test Project	1344
Creating Unit Tests	1345
Refactoring Code	1349
Understanding Code Contracts	1350
Setting Up the Environment	1350
Setting Contracts Properties	1350
Tools for Code Contracts	1351
Understanding Preconditions	1352
Post-Conditions	1353
Invariants	1354
Assertions and Assumptions	1355
Contract Events	1355
Summary	1355

Appendix

A	Useful Resources and Tools for Visual Basic 2012	1357
	Visual Basic Resources in MSDN	1357
	Useful Developer Tools for Visual Basic	1358
	Coding Tools	1358
	Networking	1359
	Data Access	1359
	Diagnostics and Performance	1359
	Miscellaneous	1359
	Where Do I Find Additional Tools?	1360
	Index	1361

Foreword

From his blog posts (in Italian and English) to his forum answers, from his online articles to his code snippets, from his technical speeches and user-groups to his books, the breadth of Alessandro's reach has been astounding.

Now is a particularly good time for Alessandro to write this book, because the current release of Visual Basic has so many new features. I'd like to tell the story behind the most important of them, "Asynchronous Programming," explained in Chapter 44. We introduced this feature because everyone wants their interactive applications to be fast and responsive, but the code to achieve this has traditionally been too convoluted. Now, thanks to asynchronous programming, the code has become straightforward. Although the code itself is straightforward, our small design group at Microsoft knew we were producing something unfamiliar as we sat down to design the feature in early 2010—something not taught in college courses, something whose success would depend on the ability of authors like Alessandro to explain it (which he does well).

As Language Designer of VB, I implemented the initial prototype of the feature within the VB and C# compilers. Our team first revealed it behind closed doors to MVPs ("Most Valued Professionals") such as Alessandro at a small gathering in New Orleans, hoping that the attendees would see its worth but daunted by the challenge of explaining something so unusual. The MVPs understood the feature's potential. They also asked us to integrate it better with other parts of Visual Studio such as debugging (Chapter 5) and unit-tests (Chapter 56), which we then did.

Indeed, there are manual useful parts to Visual Studio and .NET, and Alessandro knows them all. In this book, he makes sure that you will know all the important ones, too, and he sets them in context by explaining when and why to use them. I myself was happy to learn how to do localization (Chapter 33) and deployment (Chapters 51 and 52) from this book—both topics that I was aware of but hadn't explored before.

Alessandro has found just the right blend of text, code, and screenshots to explain Visual Studio effectively. Beginners and experts alike will find much of value in this book.

—Lucian Wischik
Seattle, Washington
December 2012

About the Author

Alessandro Del Sole, a Microsoft Most Valuable Professional (MVP) for Visual Basic since 2008, is well known throughout the global VB community. He is a community leader on the Italian Visual Basic Tips and Tricks website (<http://www.visual-basic.it>) that serves more than 42,000 VB developers, as well as a frequent contributor to the MSDN Visual Basic Developer Center. He enjoys writing articles on .NET development, writing blog posts on both his Italian and English blogs, and producing instructional videos as well as apps for Windows 8 and Windows Phone. You can find him online in forums or news-groups. Alessandro has been awarded MVP of the Year for Visual Basic in 2009, 2010, and 2011.

Dedication

To my parents, I'm learning from you how one should never give up even when life becomes the hardest possible. As usual, you taught me the most important lessons in life. The more time goes on, the more I understand what you have done for me. Thank you!

To my best friend Nadia, who always finds a way to make me smile when things become so difficult and is always able to find positive perspectives when I cannot. You have really been one of the most important people in my whole life for more than 10 years, and I will always be there for you.

Acknowledgments

First of all, I would like to thank Neil Rowe, Mark Renfrow, Anne Goebel, and all at Sams Publishing for trusting me enough to write the second edition of this book about Visual Basic. Writing books like this is hard work, not only for the author, but also for all the people involved in the reviews and in the production process. Working with these guys made the process much more pleasant. Thank you!

Very special thanks to Matthew Kleinwaks, who has been the technical editor for this book and who is a Microsoft Visual Basic MVP. Matthew did an incredible job, walking through every single word and line of code, engaging interesting technical discussions. His suggestions have been so important and made the content as accurate and interesting as possible. You can visit Matthew's blog at <http://zerosandtheone.com/>. Thank you so much, Matt!

Special thanks to all the Microsoft Visual Basic Team for doing a great job; particularly I would like to give my special thanks to Lucian Wischik, Lisa Feigenbaum, Beth Massi, and Anthony D. Green for providing opportunities of continuous interactions and technical discussions. They contribute and keep the passion for Visual Basic alive around the globe.

Great thanks also to the guys from the Italian subsidiary of Microsoft: Alessandro Teglia (my MVP lead), who has a great passion for his work and for the MVP community; he did a really great job when I had some important questions related to information required for this book, quickly pointing me in the right direction. Also, special thanks to the local Development and Platform Evangelism Division (Marco Agnoli, Francesca Longoni, Francesca Marinoni, Lorenzo Barbieri, Vito Lorusso, Pietro Brambati, Mario Fontana, Gabriele Castellani, and Irina Turcu) for their continuous support and encouragement for my activities.

Thanks to Alfonso Ghiraldini and Francesco Bosticco for their cordiality.

I would like to thank my everyday friends who are always ready to encourage me even if they are not developers and will never read my books. Most importantly, these people always support me when I need their help. So my deep thanks to Roberto Bianchi, Alessandro Ardovini, Michela Santini, Leonardo Amici, Francesca Bongiorno, Paolo Leoni, Nadir Mariotti, Blerina Spahiu, Karin Meier, and Sara Gerevini. You guys really rock!

I'm a community leader and vice president in the Italian Visual Basic Tips and Tricks community (www.visual-basic.it); therefore, I would like to thank all those guys who are the right stimulus for making things better every day; all those people who visited my blogs at least once or who read even one article of mine; and all those people who visit our website and follow us on forums, videos, articles, and blogs. Special thanks to my MVP colleagues Diego Cattaruzza, Antonio Catucci, Renato Marzaro, Massimo Bonanni, Matteo Pagani, and Raffaele Rialdi for their great support and valuable suggestions. Thanks to Marco Notari for his continuous support and encouragement.

We Want to Hear from You!

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

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

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

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

Email: feedback@sampublishing.com

Mail: Neil Rowe
Associate Publisher
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

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

This page intentionally left blank

CHAPTER 2

Getting Started with the Visual Studio 2012 IDE

You develop Visual Basic applications using the Visual Studio 2012 Integrated Development Environment (IDE), which is the place where you will spend most of your developer life. Before diving deep into the Visual Basic language, you need to know what instruments you need to develop applications. Although the Visual Studio IDE is a complex environment, this chapter provides you with an overview of the most common tasks you will perform from within Visual Studio 2012 and the most important tools you will utilize so that you can feel at home within the IDE. You get an introduction to some of the new features introduced by the new version of the development environment, which can provide the basis for the rest of the book. You also learn about other advanced IDE features in the last part of the book.

What's New in Visual Studio 2012

The Visual Studio 2012 IDE retakes the infrastructure that was first introduced by its predecessor, which is written in managed code and in which several parts are based on the *Windows Presentation Foundation* framework, such as the code editor, menus, and floating windows. On the other hand, Visual Studio 2012 has a completely different look, which is now based on the Microsoft Design Style, with a simplified approach to commands and tools via a flattened user interface in which there are only a few colors.

IN THIS CHAPTER

- ▶ What's New in Visual Studio 2012
- ▶ Status Bar and Start Page
- ▶ Working with Projects and Solutions
- ▶ Working with Tool Windows
- ▶ My Project
- ▶ Compiling Projects
- ▶ Browsing the Visual Basic and .NET Documentation
- ▶ "Quick Launch" Tool

WHAT IS THE MICROSOFT DESIGN STYLE?

The Microsoft Design Style is a design language for user interfaces that Microsoft started to introduce with the first version of Windows Phone; it is also one of the most important characteristics of Windows 8. This design language relies on the principles of abstraction, simplicity, and geometric shapes. Its goal is avoiding complex and overcolored user interfaces in favor of simple drawings. For example, it is common in the Microsoft Design Style to find monochromatic shapes rather than drawings with millions of colors that can be confusing. After its adoption in Windows Phone and Windows 8, Microsoft is bringing the new design style into a number of other programs. Visual Studio 2012 is an example of how the layout of a powerful and complex desktop application can be simplified by following some of the Microsoft Design Style principles. You can find a more detailed explanation of such principles at: <http://goo.gl/uqA5V>.

The goal is helping the developer focus on writing code or on designing the application, not on the development environment. So, the environment now has fewer colors than in the previous versions, but commands are still recognizable via familiar icons; parts of the IDE that you interact the most with (typically the code editor and designers) are still rich with helpful colorizations. Although this innovation is important, you will still feel at home with the new version. This is because the instrumentation is located and behaves the same as in the past. This chapter gives you an overview of the most common tools you need for developing your Visual Basic applications. (Deeper details on advanced IDE features are provided starting from Chapter 53, “Advanced IDE Features.”)

Status Bar and Start Page

When you first run Visual Studio 2012, you notice a new layout and the Start Page, as shown in Figure 2.1.

You can immediately notice the new look of Visual Studio 2012 based on Microsoft Design Style. The colored status bar is new in Visual Studio 2012, and its color changes according to the particular task Visual Studio is running. Following is a list of possible colors for the status bar (see Figure 2.2 for a graphical representation):

- ▶ **Violet**—This is the color for the status bar when Visual Studio is ready (for example, at startup).
- ▶ **Light blue**—This is the color for the status bar at development time, which means coding, designing the user interface, or any other task you run before compiling the code and running the application.
- ▶ **Blue**—This is the color for the status bar when Visual Studio 2012 is building the solution and compiling the code.
- ▶ **Orange**—This is the color for the status bar when you are running the application in debugging mode (that is, by pressing F5).

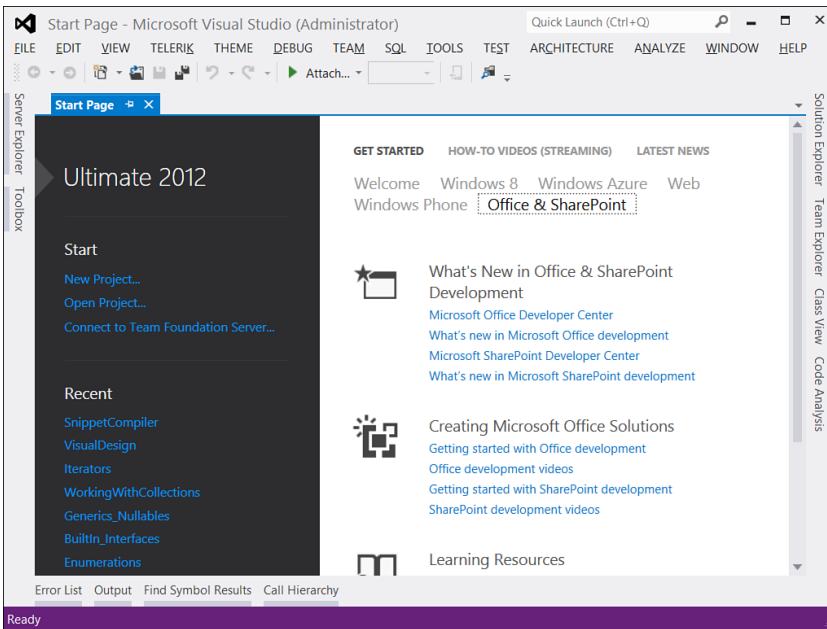


FIGURE 2.1 The Start Page in Visual Studio 2012 and the new look.

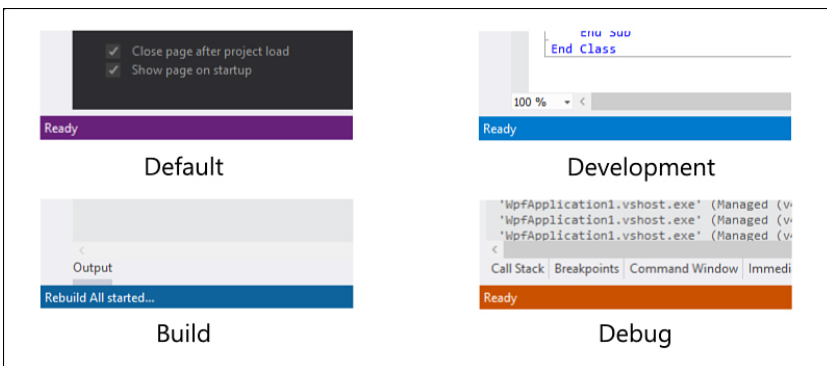


FIGURE 2.2 The status bar colors and related moments.

THEMES FOR VISUAL STUDIO 2012

Visual Studio 2012 ships with two themes, Light and Dark. Light is the default setting and is also the one that is used in this book. The Dark theme reproduces the look of Microsoft Expression Blend and is based on the black color and dark tones. You can change the theme by going to Tools, Options and then selecting a different theme in the Color Theme drop-down box in the Environment tab. At the time this chapter is being

written, there are no additional themes, but you can download a nice extension called Visual Studio 2012 Color Theme Editor, available at: <http://visualstudiogallery.msdn.microsoft.com/366ad100-0003-4c9a-81a8-337d4e7ace05>. This extension provides a number of ready-to-use themes and allows creating custom ones.

The Start Page is a central point in the IDE. First, it offers a better organization of the most common tasks, based on *tabs*. Tabs are located on the right side of the screen and enable access to specific contents. On the left side of the screen you can instead find links for creating new projects and opening existing projects, as well as the list of recently opened projects. You can easily remove recent projects by right-clicking the project name and then selecting the deletion command. It is worth mentioning that the Start Page relies on the Windows Presentation Foundation *technology and is completely written in XAML code*. This means that it can be customized according to your needs. Customizing the Start Page is beyond the scope of this chapter, whereas a deeper discussion on the default settings is absolutely necessary. The two default tabs are Get Started and Latest News. The following paragraphs discuss them in detail.

Get Started Tab

The Get Started tab (refer to Figure 2.1) offers links to important resources, such as MSDN Walkthroughs (which are step-by-step tutorials on specific topics related to Visual Studio 2012); community and learning resources, and extensions for the IDE; such as custom add-ins or third-party components. (This topic is discussed later in the book.) This tab is divided into subcategories, each related to a specific development area such as Windows 8, Web, and Windows Azure. When you click each subcategory, you access a number of links to resources for learning about the selected area. The Get Started tab's purpose is to offer links to useful resources about the development environment and to new and existing .NET technologies.

The Latest News Tab

As in its predecessors, Visual Studio 2012 can also show a list of news based on RSS feeds so that you can stay up-to-date with your favorite news channels. Now the list appears in the Latest News tab, as shown in Figure 2.3.

By default, the news channel is an RSS feed pointing to the MSDN developer portal, but you can replace it with one of your favorites. To accomplish this, you have the following alternatives:

- ▶ Open the Latest News tab and replace the default link in the RSS feed field with a valid XML feed link (this is the easiest way).
- ▶ Open the Options window (for example, by clicking the Settings link in the Visual Studio tab), and then select the Startup item. In the text box named Start Page news channel (see Figure 2.4), you can insert your favorite link. Just be sure you are writing a valid XML feed link.

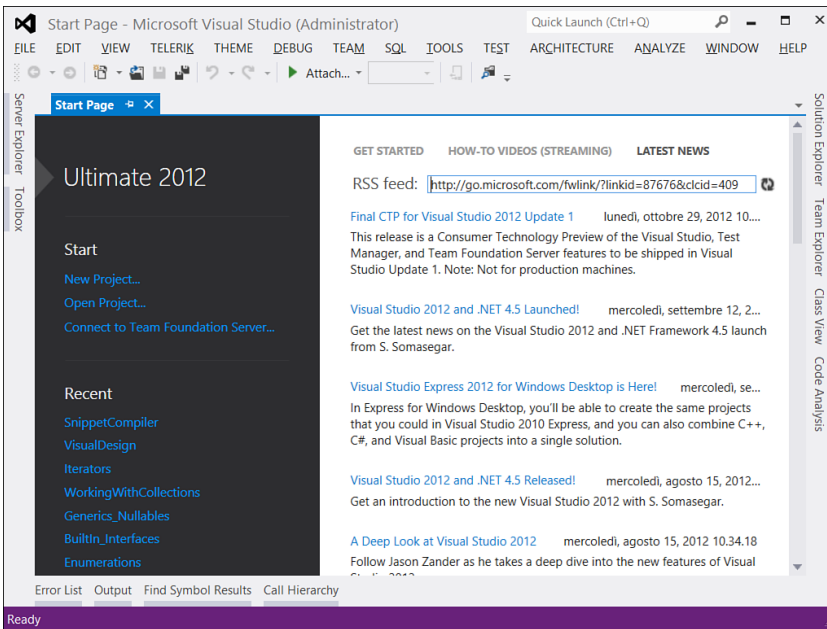


FIGURE 2.3 The Latest News tab shows updated news from the specified RSS channel.

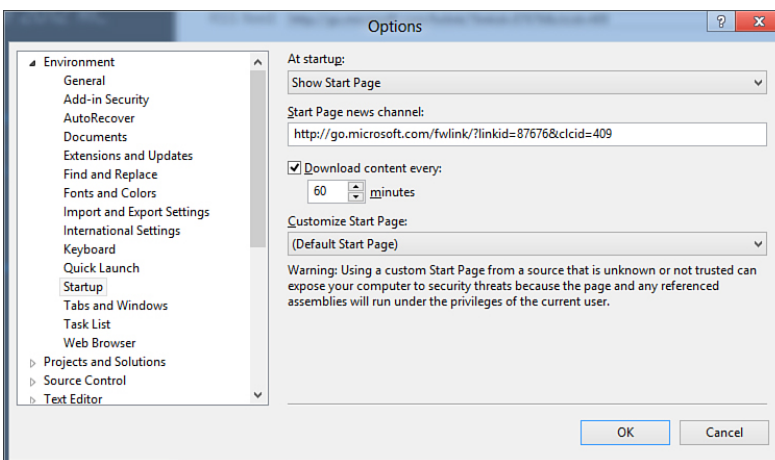


FIGURE 2.4 The Options window enables customizing the RSS Feeds news channel.

If you want to stay up-to-date with Visual Basic news, consider replacing the default news channel with the Visual Basic page of the Visual Studio Developer Center, which is at the following: http://sxp.microsoft.com/feeds/3.0/msdntn/VB_featured_resources. After this brief overview of the Start Page, we can begin discussing the tooling that you use for creating Visual Basic projects within Visual Studio 2012.

Working with Projects and Solutions

Each time you want to develop an application, you create a *project*. A project is a collection of files, such as code files, resources, data, and every kind of file you need to build your final assembly. A Visual Basic project is represented by a .Vbproj file, which is an Extensible Markup Language (XML) file containing all the information required by Visual Studio to manage files that constitute your project. Projects are organized into solutions. A *solution* is basically a container for projects. In fact, solutions can contain infinite projects of different kinds, such as Visual Basic projects, projects produced with programming languages other than Visual Basic, class libraries, projects for Windows client applications, Windows Communication Foundation services, and so on. In other words, a solution can include each kind of project you can create with Visual Studio 2012. Solutions also can contain external files, such as documents or help files, and are represented by a .Sln file that has an XML structure and stores information required to manage all the projects contained in the solution. Visual Studio 2012 can also open solutions created with previous versions of the IDE.

PROJECT UPGRADES AND “ROUND-TRIPPING”

You can upgrade previous versions of your solutions by simply opening them in Visual Studio 2012. The new version of the IDE introduces the so-called project round-tripping, which means that solutions created with Visual Studio 2010 with Service Pack 1 can be opened in Visual Studio 2012 with no upgrades; then you can open the same solution back in Visual Studio 2010 with SP 1. For solutions created with other versions, the Upgrade Wizard can guide you through the upgrade process in a few steps. There are some exceptions to project round-tripping, which you can read in the specific MSDN page: [http://msdn.microsoft.com/en-us/library/hh266747\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/hh266747(v=vs.110).aspx).

Typically you manage your projects and solutions using the Solution Explorer window discussed later in this chapter. We now focus on the creation of Visual Basic 2012 projects.

Creating Visual Basic Projects

Creating a new Visual Basic project is a simple task. You can click either the **File, New Project** command or the **New Project** link from the Start Page. In both cases, Visual Studio shows the New Project window, which you can see in Figure 2.5.

As you can see, the look of the New Project window is quite different from previous versions of Visual Studio and still retakes the Windows 8 style. To understand what kind of Visual Basic applications you can create, you simply need to select the **Visual Basic** node on the left side of the window. After you click the node, you see a lot of different kinds of applications you can create with Visual Basic 2012, such as Windows applications, web applications, Office System customizations, Silverlight applications, Windows Communication Foundation Services, the new Windows 8 Store Apps for Windows 8, and so on.

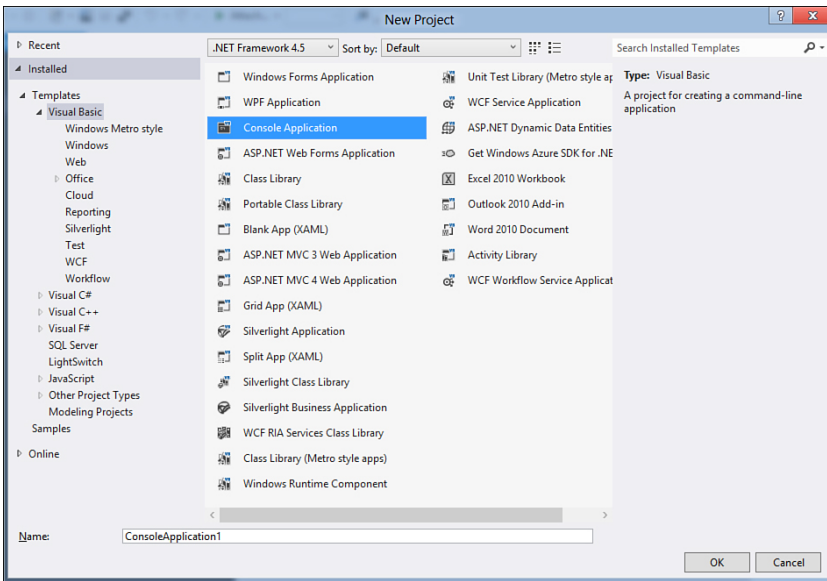


FIGURE 2.5 The New Project window for Visual Basic.

NOTE ABOUT AVAILABLE PROJECT TEMPLATES

The list of installed project templates can vary depending on either the Visual Studio 2012 edition or additional items installed later (for example, from the Visual Studio Gallery).

Each kind of application is represented by a specific project template, which provides a skeleton of a project for that particular kind of application, including all the references or the basic code files required. For example, the WPF Application project template provides a skeleton of a project for creating a Windows application using the Windows Presentation Foundation technology, therefore including references to the `WindowsBase.dll` assembly, specific `Imports` directives, and WPF objects represented by the appropriate code files. Moreover, Visual Studio will automatically enable the WPF designer. Notice the detailed description for each template on the right side of the window every time you select a particular template.

NOTE

In the second part of this book you find several kinds of applications you can build with Visual Basic 2012. For this reason, a full description of each project template will be provided in the proper chapters. At the moment, the description provided by Visual Studio for project templates is sufficient. Also consider that in this first part of the book the Console Application project template is used because it is the most appropriate for learning purposes.

When you create a new project, Visual Studio usually creates a solution containing that project. If you plan to add other projects to the solution, it can be a good idea to create a directory for the solution. This allows for a better organization of your projects, because one directory can contain the solution file and this directory can then contain subdirectories, each of them related to a specific project. To accomplish this, ensure that the Create Directory for Solution check box is selected. The New Project window also offers some interesting features: the .NET Framework multi-targeting, the ability of searching through templates, the ability of managing templates, and finding samples on the Internet. We now discuss each of these features.

Multi-targeting

Like in Visual Studio 2010, in Visual Studio 2012 you can choose which version of the .NET Framework your application targets. This can be useful if you plan to develop applications with high compatibility levels and without new language features but still want to take advantage of the new features of the IDE. You can choose one of the following:

- ▶ .NET Framework 4.5 (proposed by default)
- ▶ .NET Framework 4.0
- ▶ .NET Framework 3.5
- ▶ .NET Framework 3.0
- ▶ .NET Framework 2.0

To accomplish this, just select the appropriate version from the combo box at the top of the window, as shown in Figure 2.6.

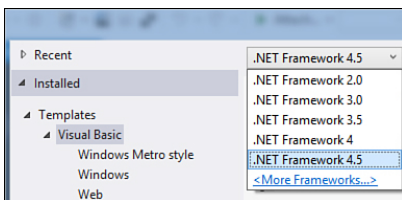


FIGURE 2.6 Choosing the .NET Framework version for your application.

NOTE

Remember that, depending on the version of the .NET Framework you choose, you might not be able to use some libraries or technologies. For example, if you choose .NET Framework 3.0 as the target version, you cannot use LINQ, which is instead exposed by .NET Framework 3.5 and higher. So keep in mind these limitations when developing your applications on previous Framework versions.

Accessing Recent and Online Templates

Visual Studio 2012 provides the capability to access the most recently used templates and install additional templates from the Internet. You can easily access the most recently used project templates by clicking the **Recent Templates** item on the left side of the New Project window. In this way, you get a list of the recently used project templates in case you still need them (see Figure 2.7). You can also find additional online templates and install them to the local system. To accomplish this, simply click the **Online Templates** item in the New Project window. Visual Studio checks for the availability of online templates and shows a list of all the available templates (see Figure 2.8).

As you can see, Visual Studio is listing all the online templates for both Visual Basic and Visual C#, showing a description of the template, information about the author, and a small picture with ratings when available. To download and install a template, simply double-click its name. After a few seconds you will be prompted to agree to the installation. You will be warned in the case that the new extension for Visual Studio does not contain a digital signature. If you trust the publisher and want to continue, click **Install**. After a few seconds, you will see the newly installed project templates available among the other ones.

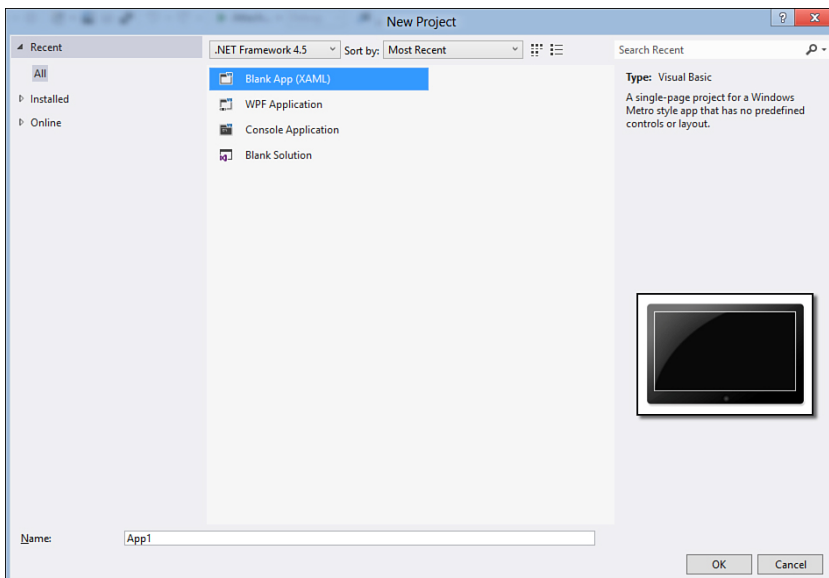


FIGURE 2.7 Accessing the most recently used projects templates.

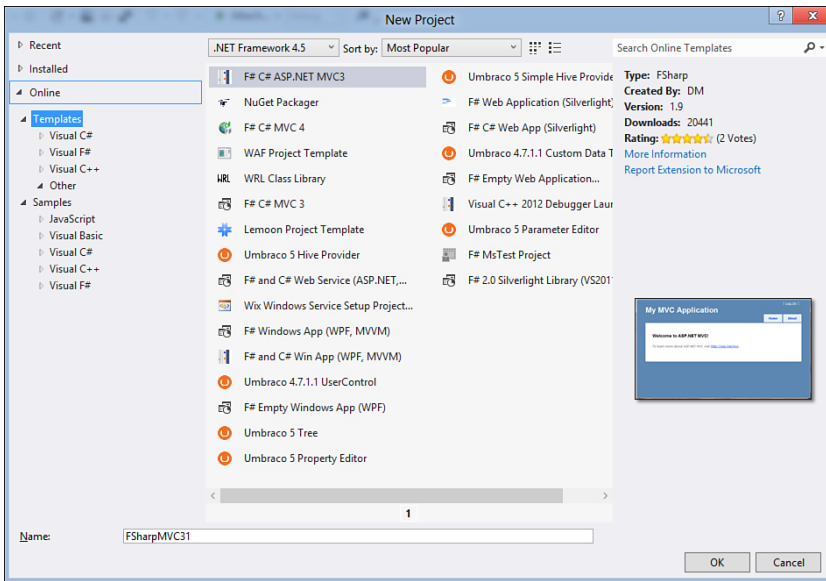


FIGURE 2.8 Additional online templates you can add to Visual Studio 2012.

As in the previous versions of Visual Studio, you can still export projects as reusable project templates. We discuss this feature later in Chapter 53, “Advanced IDE Features.”

Searching for Installed Templates

Visual Studio 2012 provides lots of default project templates, and as you saw before, you have the ability of adding your own custom ones. Therefore, finding the necessary template at a certain moment can be difficult. Because of this, the New Project window provides a search box that is located in the upper-right corner of the window (see Figure 2.9). Just begin typing the name of the project template you are looking for, and the New Project window shows all the project templates that match your choice. Each time you type a character, the window updates showing the results matching your search string. As you can see in Figure 2.9, Visual Studio is showing all the project templates whose names match the *wor* search string.

NOTE

Remember that the search functionality can retrieve all project templates related to your search string. This means that the search result can contain not only Visual Basic projects but also every template name matching your criteria (for example, Visual C#, Visual F#, and so on).

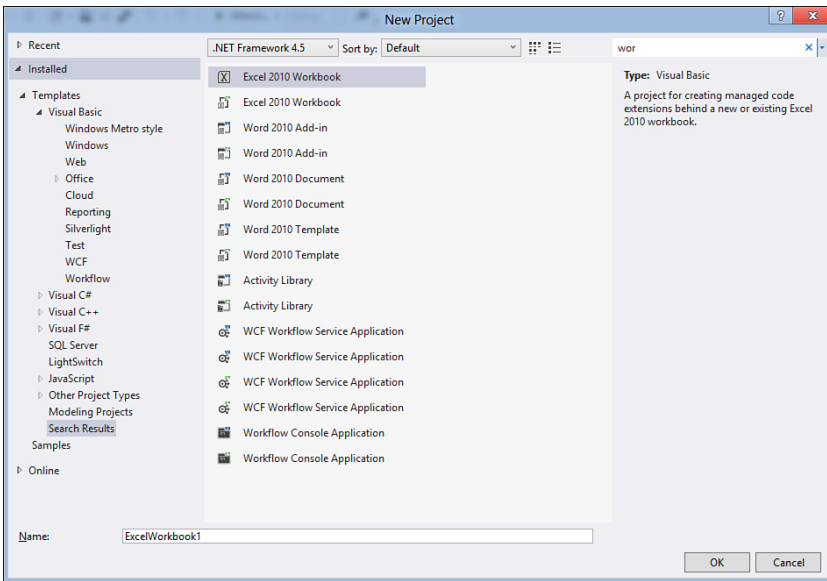


FIGURE 2.9 Searching for installed project templates using the new search feature.

Finding Code Samples on the Internet

One of the great new features of Visual Studio 2012 is the ability to search for sample code on the Internet directly from the New Project dialog box. Visual Studio performs this search inside the MSDN Code Gallery (<http://code.msdn.microsoft.com>), a website where developers can publish sample code they want to share with others. To find code samples, click the Samples item at the bottom of the list of templates in the New Project dialog box (see Figure 2.5). After you select this item, you will be able to browse the Code Gallery from within the dialog box. For example, if you search for a specified language, you will then be able to discover samples of each language based on technologies (such as Windows, Web, Cloud, Windows Phone, and so on). For each of these categories, you will find additional subcategories where samples are divided by framework; for instance, Windows development samples are divided into WPF, Windows Forms, Windows Runtime, and so on. Figure 2.10 shows how you can browse samples online from Visual Studio 2012.

You select a sample from the list and then click OK or simply double-click the sample. Visual Studio 2012 will download the sample to your hard disk and will ask you to accept the license agreement that the developer who published the code added to the sample. After it's downloaded, the sample is opened in Visual Studio 2012 under the form of a normal project. This is possible because, to upload a sample to the gallery, you must supply a full project (or an entire solution) to enable other developers to reuse your code with ease.

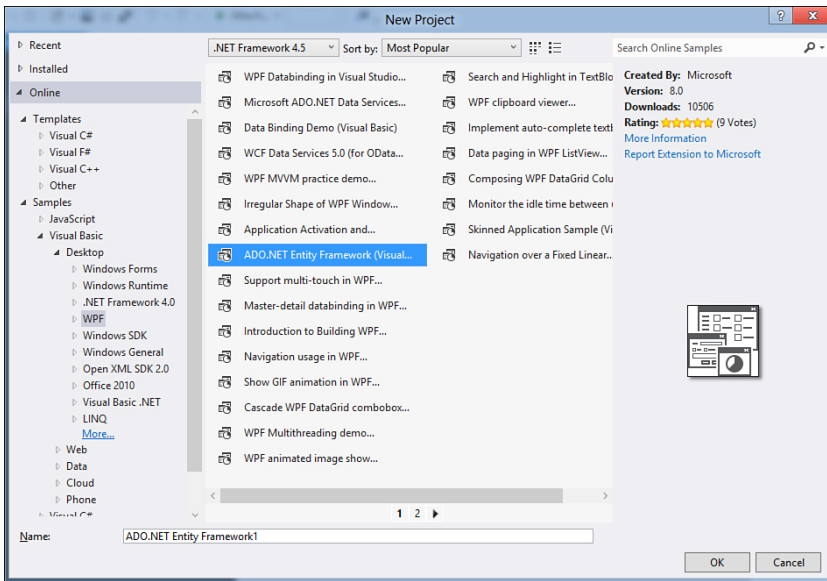


FIGURE 2.10 Browsing online code samples from Visual Studio 2012.

Creating Reusable Projects and Items Templates

As in the previous versions, in Visual Studio 2012 you can create your custom projects and items templates, exporting them to disk and making them available within the IDE. (This topic is discussed in Chapter 53.) Now that you have seen the main new features for project creation, you are ready to create your first Visual Basic project.

Creating Your First Visual Basic 2012 Project

This section shows how easily you can create a Visual Basic 2012 application. If you have experience with Visual Studio, you'll notice the differences in the development environments between the new version and the previous ones. You can create a new project for the Console by first opening the New Project window and then selecting the **Console Application** project template.

NOTE

Until Part IV, “Data Access with ADO.NET and LINQ,” all code examples, listings, and code snippets are based on Console applications, so remember to create a Visual Basic project for the console when testing the code.

Name the new project **MyFirst2012Program** and then click **OK** (see Figure 2.11).

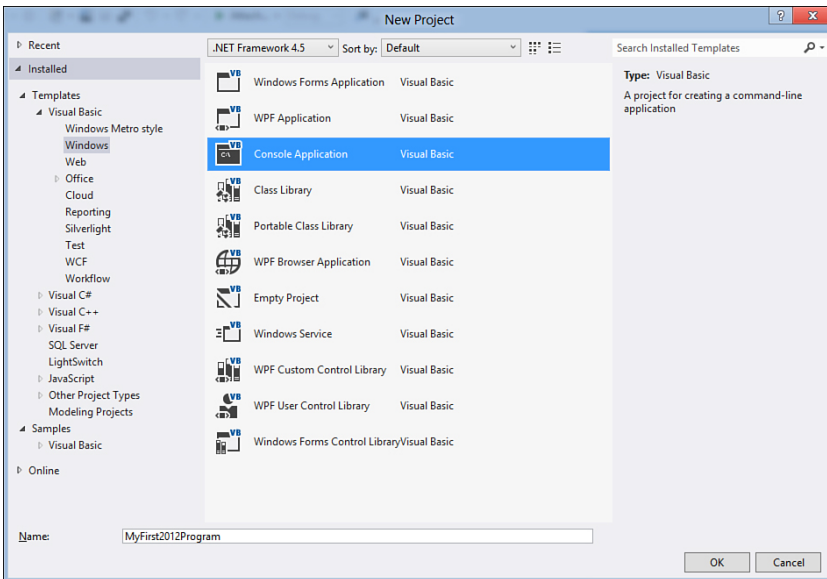


FIGURE 2.11 Creating your first VB 2012 application.

After a few seconds the new project is ready to be edited. Figure 2.12 shows the result of the project creation.

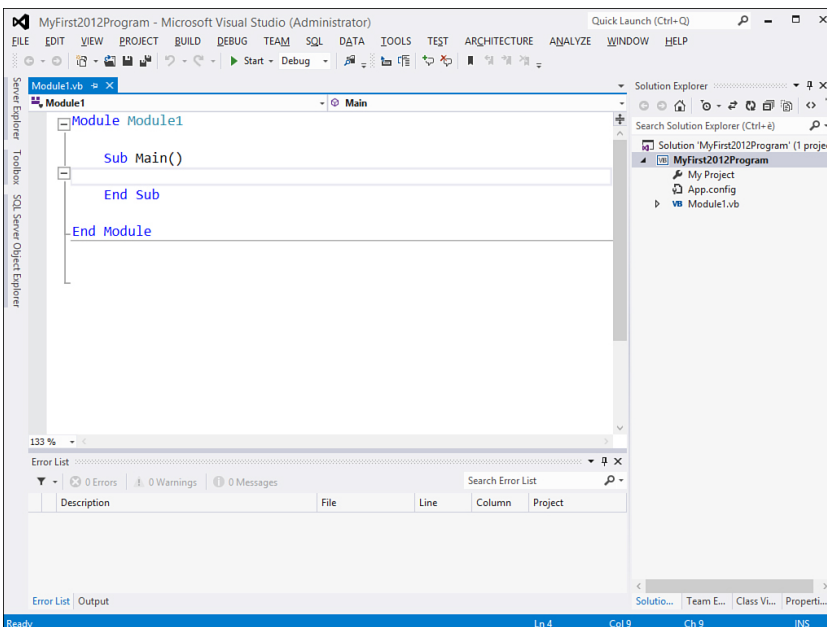


FIGURE 2.12 The creation of the first VB 2012 project.

As mentioned at the beginning of this chapter, the code editor and floating windows are based on Windows Presentation Foundation. You see the new Windows 8 look for the tooling, but this change does not modify your development experience. You should feel at home with the new version of the environment because you can still find tools as in the previous versions—maybe just with fewer colors. For example, you can access Solution Explorer, the Properties window, or the Error List exactly as you did before. An interesting feature in the code editor is that by pressing **Ctrl** and moving up and down the mouse wheel, you can zoom in and out with the code editor without the need to change the font settings each time the Visual Studio options changes. For our testing purposes, we could add a couple of lines of code to the `Main` method, which is the entry point for a Console application. Listing 2.1 shows the complete code for creating a VB 2012 application.

LISTING 2.1 Creating the First VB 2012 Application

```
Module Module1

    Sub Main()
        Console.WriteLine("Hello Visual Basic 2012!")
        Console.ReadLine()
    End Sub
End Module
```

WHAT IS A CONSOLE?

In a Console application, the `System.Console` class is the main object for working with the Windows console. Such a class provides methods for reading and writing from and to the Console and for performing operations against the Console itself.

The code simply shows a message in the Console window and waits for the user to press a key. This is obviously a basic application, but you need it as the base for understanding other topics in this chapter.

Finding Visual Basic Projects

As in the previous versions, Visual Studio 2012 stores by default its information in a user-level folder called **Visual Studio 2012** that resides inside the My Documents folder. Here you can find settings, add-ins, code snippets, and projects. For example, if you run Windows Vista, Windows 7, or Windows 8, your projects should be located in `C:\Users\UserName\Documents\Visual Studio 2012\Projects`, in which `UserName` stands for the user logged in to Windows. Of course, you can change the default projects directory by opening the Options window (Tools, Options command), selecting the Projects and Solutions item on the left side, and replacing the value for the Projects location text box.

Working with the Code Editor

The code editor, together with designers, is the place where you spend most of your developer life in Visual Studio, so knowing how to get the best out of it is very important. This

is what I explain throughout the whole book. In this chapter you see just a few of the code editor features because the other ones are related to specific topics discussed later. We now focus on the zoom functionality in the code editor and IntelliSense.

Zooming the Code

You can zoom the code editor in and out by simply pressing the **Ctrl** key and moving the mouse wheel up and down. This is a useful feature, particularly if you are a presenter of technical speeches because you can enlarge the code without modifying Visual Studio settings in the Options window. Figure 2.13 shows an example of this feature; notice how the font for the code is greater than the default settings and how the zoom percentage is visible at the bottom-left corner.

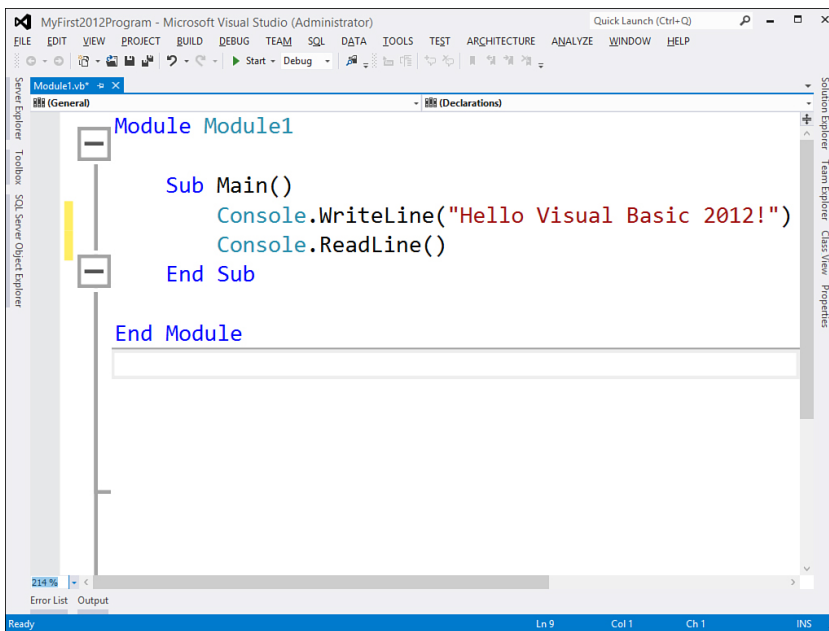


FIGURE 2.13 The code editor zoom feature enables real-time enlarging and reducing of the font size of the code without changing settings in Visual Studio.

If you already used this feature in Visual Studio 2010, you might remember how the scrollbar on the right also changed its size according to the zoom. This has been fixed in Visual Studio 2012; the bar size is always the same and independent from the zoom.

IntelliSense Technology

IntelliSense is one of the most important technologies in the coding experience with Visual Studio. IntelliSense is represented by a pop-up window that appears in the code editor each time you begin typing a keyword or an identifier and shows options for auto-completing words. Figure 2.14 shows IntelliSense in action when adding a new instruction in code.

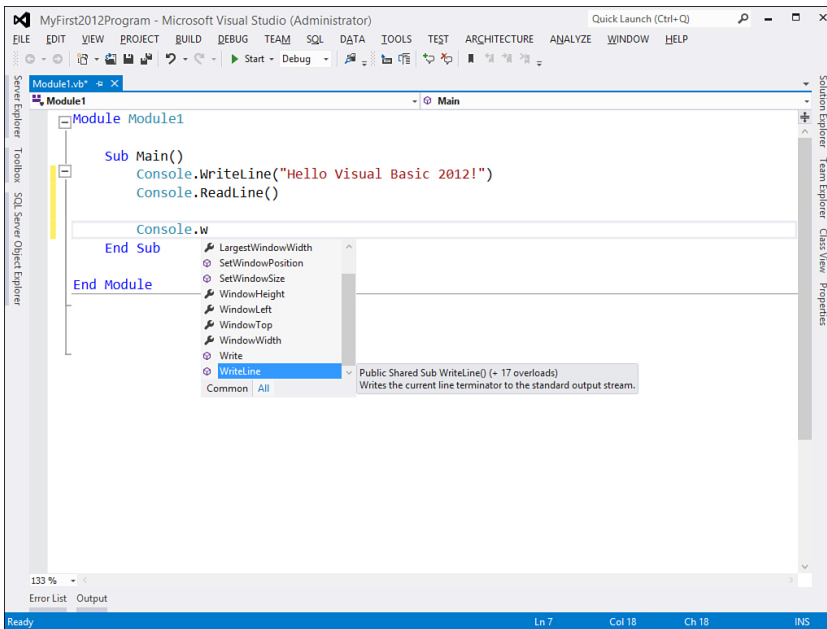


FIGURE 2.14 IntelliSense in action.

To auto-complete your code typing, you have the following alternatives:

- ▶ **Tab**—Pressing this auto-completes your words and enables you to write other code.
- ▶ **Space**—Pressing this auto-completes your words, adding a blank space at the end of the added identifier and enabling you to write other code.
- ▶ **Enter**—Pressing this auto-completes your words, adding a couple of parentheses at the end of the completed identifier and positioning the cursor on a new line. Use this technique when you need to invoke a method that does not require arguments.
- ▶ **Left parenthesis**—Pressing this auto-completes your words, adding a left parenthesis at the end of the completed identifier and waiting for you to supply arguments.
- ▶ **Ctrl + Space**—Pressing this brings up the full IntelliSense listing.

IntelliSense has been improved since Visual Studio 2008. In fact, it is activated just when typing one character and is also active versus Visual Basic reserved words. Moreover, it remembers the last member you supplied to an object if you invoke that particular object more than once. For example, if you use IntelliSense to provide the `WriteLine` method to the `Console` object as follows:

```
Console.WriteLine()
```

and then try to invoke IntelliSense on the Console object again, it proposes as the first alternative of the WriteLine method you supplied the first time. IntelliSense is important because it lets you write code more quickly and provides suggestions on which member to add to your code.

Working with Tool Windows

As in the previous versions, lots of the Visual Studio tools are provided via *tool windows*. Tool windows are floating windows that can be docked to the IDE interface and are responsible for a particular task. As a general rule, in the View menu you can find the tool windows provided by Visual Studio 2012. Exceptions to this rule are the Test tool windows and the analysis tool windows that can be invoked from the Test and Analyze menus, respectively, and tool windows specific for debugging, available in the Debug menu (see Chapter 5, “Debugging Visual Basic 2012 Applications”). In this book we utilize several tool windows, and in this chapter you get an overview of the most frequently used ones. In particular, we now focus on Solution Explorer, Error List, Properties, and Output windows. This is because these are the tool windows you will use in each of your projects. Other tool windows will be analyzed when applied to specific topics. To dock a tool window to the desired position in the IDE, just move the window onto the most appropriate arrow in the graphical cross that you see on the IDE and then release. Figure 2.15 represents this situation.

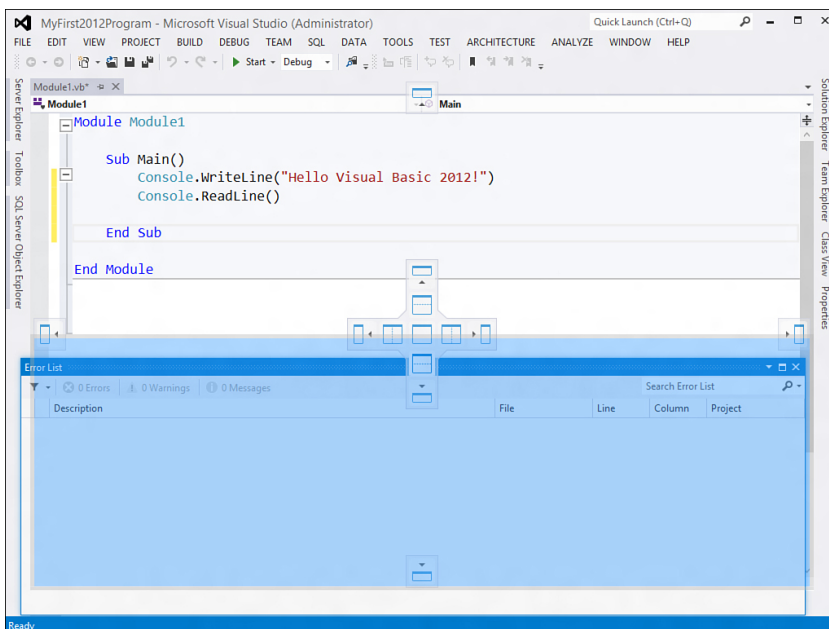


FIGURE 2.15 Docking a floating tool window to the IDE's interface.

Visual Studio 2012 automatically positions some tool windows in specific places of the IDE, but you can rearrange tool windows as much as you like. We now discuss the previously mentioned tool windows.

The Solution Explorer Window

Solution Explorer is a special tool window that enables managing solutions, projects, and files in the projects or solution. It provides a complete view of which files compose your projects and enables adding or removing files and organizing files into subfolders. Figure 2.16 shows how a WPF project is represented inside Solution Explorer.

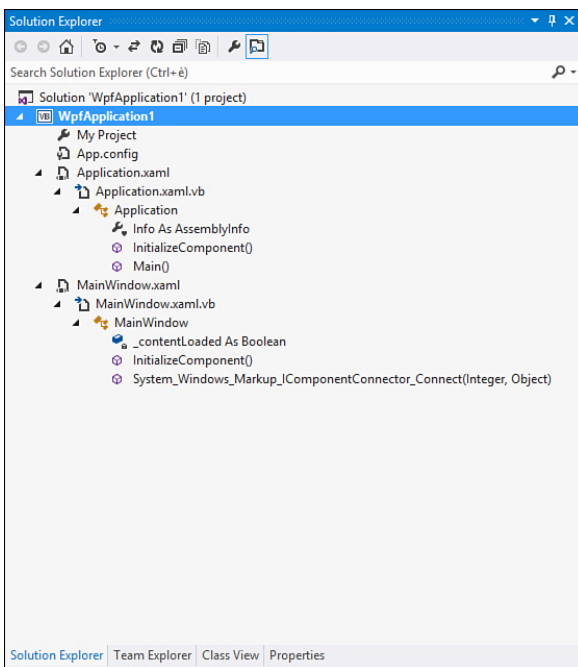


FIGURE 2.16 An example of the Solution Explorer tool window.

As you can see, at the root level there is the project. Nested are code files, subfolders containing pictures, data, and documents. You can also get a list of all the references in the project. You use Solution Explorer for adding and managing items in your projects as well as getting a representation of which files constitute the project itself. By default, Solution Explorer shows only the items that compose the project. If you need a complete view of references and auto-generated code files, you need to click the **Show All Files** button located on the upper-left portion of the window. Solution Explorer has been deeply revisited in Visual Studio 2012 and, if you had a chance of using the Power Tools extension for Visual Studio 2010, you might notice how it retakes most of the features of a tool window called Solution Navigator. Solution Explorer now shows the list of

types and members that a code file defines, by expanding the name of a code file. For instance, you can see in Figure 2.16 how the `MainWindow.xaml.vb` code file defines a class called `MainWindow`, which exposes a field called `_contentLoaded` of type `Boolean`, and two methods, `InitializeComponent` and `System_Windows_Markup_IComponentConnector_Connect`. By double-clicking a member you will be automatically redirected to its definition in the code file. The window also shows arguments and their type. This is useful when you have hundreds of files and just need to know which types are defined within a file without searching inside all the others. Also, Solution Explorer has a useful search box in which you can type a search key and the IDE will search for all the items in the solution that contain the specified word(s). Another interesting feature of the new Solution Explorer is that you can discover the following behaviors for members defined in code:

- ▶ Calls that the member makes
- ▶ Members that are calling the selected member
- ▶ Members that are using the selected member

You can simply right-click a member defined inside a code file and then select the desired command. At that point, Solution Explorer will show the list of members that are invoked by, are calling, or are using the selected member. To go back to the full list you simply use the Back button in the toolbar (see Figure 2.16). The new Solution Explorer also provides interaction with Team Foundation Server and the possibility of generating visual representation of the object graph for the selected node. To manage your project's items, you just need to right-click the project name and select the most appropriate command from the pop-up menu that appears. Figure 2.17 shows this pop-up menu.

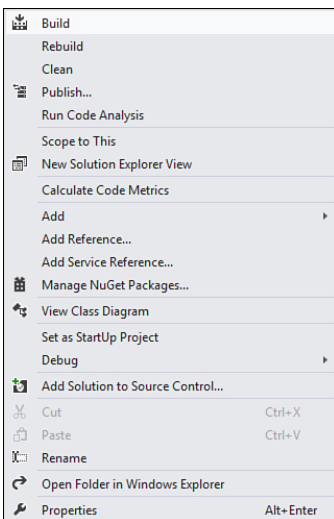


FIGURE 2.17 Solution items can be managed in Solution Explorer using the pop-up menu.

As you can see from Figure 2.17, the pop-up menu shows several tasks you can perform on your projects or solutions. You can easily add new items by selecting the **Add** command; you also can perform tasks against specific files if you right-click items in the solution instead of the project's name.

NOTE ABOUT ADDING ITEMS TO PROJECTS

In this book you will be asked lots of times to add new items to a project or to a solution, so you should keep in mind that this can be accomplished by right-clicking the project name in Solution Explorer and then clicking the Add, New Item command from the pop-up menu.

You can easily find Solution Explorer by pressing **Ctrl+Alt+L** if it is not available yet in the IDE. The enhancements that Visual Studio 2012 brings to Solution Explorer make this tool window a very important productivity tool, rather than a simple project browser.

The Error List Window

The Error List tool window can show a list of all messages, including warnings and information, which are generated by Visual Studio during the development of your applications. Figure 2.18 shows the Error List tool window.

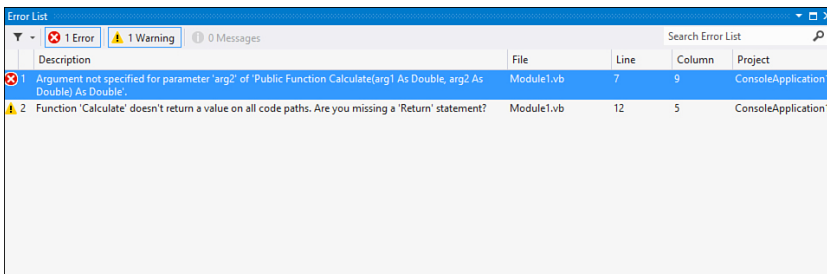


FIGURE 2.18 The Error List tool window.

Typically the Error List window can show three kinds of messages:

- ▶ Error messages
- ▶ Warning messages
- ▶ Information messages

Error messages are related to errors that prevent your application from running—for example, if your code cannot compile successfully. This can include any kind of problems that Visual Studio or the background compiler encounter during the development process, such as attempting to use an object that has not been declared, corrupted auto-generated Windows Presentation Foundation files that must be edited in the Visual Studio designer,

or corrupted Visual Studio files in situations that throw error messages you can see in the Error List. Another kind of message is a warning. Warnings are related to situations that will not necessarily prevent your code from being successfully compiled or your application from running; in such cases warnings can be simply ignored. It's good practice to try to solve the problems that caused these messages to be shown. For example, running the Code Analysis tool will throw warnings on code that is not compliant with Microsoft guidelines for writing code. This means that the application will probably work, but something in your code should be improved. In both error and warning messages, you can be redirected to the code that caused the message by double-clicking the message itself. You can also get some help about the message by right-clicking it and then selecting the **Show Error Help** command from the pop-up menu, or simply by pressing F1. Information messages are just to inform you about something. Usually information messages can be ignored with regard to the code, although they could be useful for understanding what the IDE wants to tell us. By default, the **Error List** shows the three kinds of messages together, but you could also choose to view only some of them—for example, error messages excluded and so on. To filter the Error List results, click the tab related to the kind of message you do not want to be shown. For instance, click the **Messages** tab if you want information messages to be excluded by the errors list. To include back information messages, click the same tab again. The Errors List can also be easily invoked by pressing **Ctrl+**, **Ctrl+E**.

The Properties Window

In the .NET development, everything has *properties*, which are the characteristics of a particular item. Classes can have properties; files can have properties; and so on. For example, the filename is a property of a file. You often need to set properties for your code, for .NET objects you use in your code, and for your files. To make things easier, Visual Studio provides a tool window named Properties window, which is a graphical tool for setting items' properties. Figure 2.19 represents the Properties window showing properties for a button in Windows Presentation Foundation.

We could define the Properties window's structure as a two-column table, in which the left column specifies the property and the right column gets or sets the value for the property. Although you often need to set properties in code, the Properties window provides a graphical way to perform this assignment and can be particularly useful when designing the user interface or when you need to specify how a file must be packaged into the executable assembly. The Properties window can be easily invoked by pressing **F4**.

The Output Window

Visual Studio often recurs to external tools for performing some actions. For example, when compiling your projects, Visual Studio invokes the Visual Basic command-line compiler, so the IDE captures the output of the tools it utilizes and redirects the output to the Output window. The Output window's purpose is to show results of actions that Visual Studio has to perform or that you require to be performed by Visual Studio. So, when you compile your projects, the Output window shows the results of the build process. Figure 2.20 shows the Output window containing the results of the build process of a Visual Basic project.

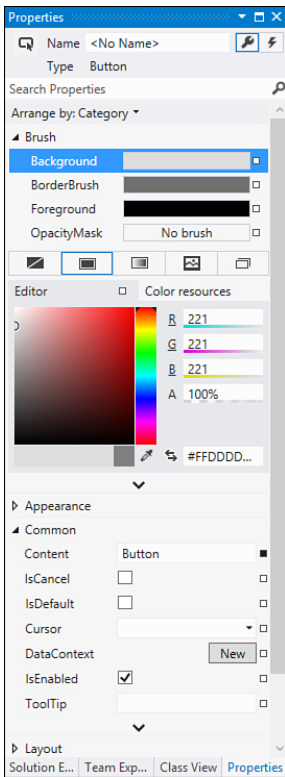


FIGURE 2.19 The Properties window.

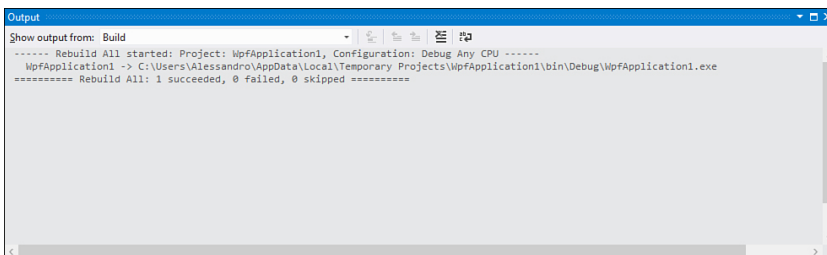


FIGURE 2.20 The Output window showing results of a build process.

The Output window is also interactive. To continue with the example of compiling a program, if the compiler throws any errors, these are shown in the Output window. You can click the **Go to Next Message** or **Go to Previous Message** button to navigate error messages. After you do this, the current error message is highlighted. Each time you move to another error message, you will be redirected to the code that caused the error. The Output window can capture the output not only of the compiler, but also of other tools

Visual Studio needs to use, such as the debugger. By the way, you can get a list of the available outputs by clicking the **Show Output From** combo box. After this first look to the main tool windows, we can begin examining another important feature of the Visual Studio IDE for Visual Basic: the My Project window.

My Project

My Project is a special tool that enables developers to set project properties. My Project is a window that can be invoked by double-clicking the **My Project** item in Solution Explorer or by clicking the **Project, Properties** command (the menu item text also includes the current project name). Figure 2.21 shows how My Project looks regarding the sample application we previously created.

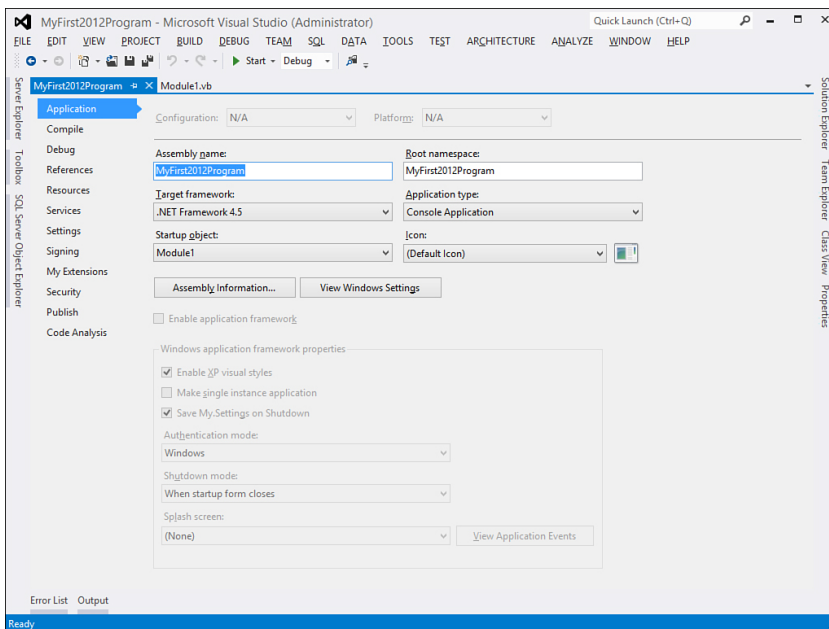


FIGURE 2.21 The My Project window.

My Project is organized in tabs; each tab represents a specific area of the project, such as application-level properties, external references, deployment options, compile options, debugging options, and many more. At the moment you don't need to learn each tab of My Project because during the rest of the book we use it a lot, learning the meaning and purpose of each tab when appropriate. What you instead need now is to

- Understand what My Project is.
- Remember how you can open it.
- Learn the usage of the Application tab, which we will discuss first.

UNDERSTANDING MY PROJECT

Understanding My Project is also important for another reason: It provides most of the infrastructure for the `My` namespace that will be discussed in Chapter 19, “The My Namespace.” Most of settings you can specify in My Project are then accessible by invoking `My`. In Chapter 3, “The Anatomy of a Visual Basic Project,” we describe the structure of My Project, so carefully read how it is composed.

Application Tab

Each application has some settings. Application settings can be the executable’s name, icon, or metadata that will be grabbed by the operating system, such as the program version, copyright information, and so on. The purpose of the Application tab in My Project is to provide the ability to edit these kinds of settings. The Application tab is shown by default when you first open My Project. Figure 2.21 shows the Application tab. Some settings are common to every kind of Visual Basic project, whereas other ones are related to specific project types. In this chapter you get an overview of the common settings, whereas specific ones will be discussed when required in the next chapters.

Assembly Name

The Assembly name field sets the name of the compiled assembly—that is, your executable. By default, Visual Studio assigns this setting based on the project name, but you can replace it as needed.

Root Namespace

This particular field sets the root-level namespace identifier. Namespaces will be discussed later in this book. You can think of the root namespace as the object that stores all that is implemented by your project. According to Microsoft specifications, the root namespace should be formed as follows: `CompanyName.ProductName.Version`. This convention is optimal when developing class libraries or components but might not be necessary when developing standalone executables. By default, Visual Studio sets the root namespace based on the project name.

Application Type

This represents the application type (for example, Console application, Class Library, Windows Forms application) and is automatically set by Visual Studio on the appropriate choice. To ensure you will avoid any problems, you should not change the default setting.

Icon

This field allows setting an icon for the executable file. You can browse the disk and select an existing `.ico` file as the executable icon.

NOTE ABOUT ICONS

Assigning an icon to the executable file will not automatically assign icons to Windows Forms windows or WPF windows when developing client applications. In such scenarios you need to explicitly assign icons for each window because the Icon item in the Application tab will just set the icon for the executable.

2

Startup Object

By setting the Startup Object field, you can specify which object will be executed first when running your application. For example, imagine you have a Windows Presentation Foundation application with more than one window. You might want to decide which window must be the application's main window. With the Startup Object field, you can make this decision. Notice that the startup object changes based on the project type. For example, in a WPF application the startup object is a `Window` object, whereas in a Console Application the default startup object is a `Module` in which the `Sub Main` method is located. The name of the field also changes based on the project type. In a WPF application it is called Startup URI, whereas in a Console Application it is called Startup Object.

CHANGING THE STARTUP OBJECT

Please be careful when changing the Startup Object. A wrong choice could cause errors on your application and prevent it from running.

Assembly Information

By clicking the Assembly Information button, you get access to a new window called Assembly Information, as shown in Figure 2.22.

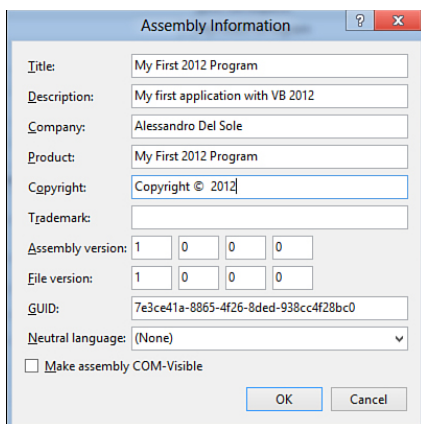


FIGURE 2.22 The Assembly Information window.

From this window you can specify several properties for your executable that will be visible both to the .NET Framework and to the Windows operating system. Table 2.1 explains each property.

TABLE 2.1 Assembly Information Explained

Property	Description
Title	The title for your application, for example, “My First 2012 Program.”
Description	The description for your application, for example, “My first application with VB 2012.”
Company	Your company name.
Product	The product name, for example, “Suite of My New 2012 Applications.”
Copyright	Copyright information on the author.
Trademark	Trademarks information.
Assembly version	Specifies the version number for the assembly in the style Major.Minor.Build.Revision. This information identifies the assembly for the .NET Framework.
File version	Specifies the version number for the executable in the style Major.Minor.Build.Revision. This information is visible to the Windows operating system.
GUID	A globally unique identifier assigned to the assembly. You can replace it with a new one or leave it as the GUID provided by the IDE.
Neutral language	Specifies what local culture is used as the neutral language.
Make assembly COM-Visible	.NET assemblies can be exposed to COM. By marking this flag, you can accomplish this task later.

The Assembly Information tool is important because it enables you to specify settings that you want to be visible to your customers and other settings needed by the .NET Framework. Behind the scenes, all this information is translated into Visual Basic code, which is discussed more in Chapter 3.

View Windows Settings

Starting from Windows Vista, Microsoft introduced to the Windows operating system an important component, known as User Account Control (UAC). When enabled, this mechanism requires the user to explicitly grant elevated permissions to applications being run. Because of this and starting from Visual Studio 2008, you have the ability of specifying the permissions level your application will require for the UAC. For example, if your application needs to write to the Program Files folder (this is just an example and rarely a good idea), you need to ask for elevated permissions to the UAC. You can specify UAC settings for your application by clicking the **View Windows Settings** button. At this point Visual Studio generates a new XML manifest that will be packaged into your executable and that you can edit within the IDE. This file contains information for UAC settings and

for specifying the operating systems that the application is designed to work for (see the `supportedOS` node). Listing 2.2 shows an excerpt of the default content of the manifest, related to the UAC settings.

LISTING 2.2 The UAC Manifest Content

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-
com:asm.v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <!-- UAC Manifest Options
          If you want to change the Windows User Account Control level replace the
          requestedExecutionLevel node with one of the following.

          <requestedExecutionLevel level="asInvoker" uiAccess="false" />
          <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
          <requestedExecutionLevel level="highestAvailable" uiAccess="false" />

          Specifying requestedExecutionLevel node will
          disable file and registry virtualization.
          If you want to utilize File and Registry Virtualization for backward
          compatibility then delete the requestedExecutionLevel node.
        -->
        <requestedExecutionLevel level="asInvoker" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>

  <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
    <application>
      <!-- A list of all Windows versions that this application is designed
        to work with.
        Windows will automatically select the most compatible environment.-->

      <!-- If your application is designed to work with Windows Vista, uncomment the
        following supportedOS node-->
      <!--<supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"></supportedOS>-->

      <!-- If your application is designed to work with Windows 7, uncomment the
        following supportedOS node-->
      <!--<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}" />-->
```



```

    <!-- If your application is designed to work with Windows 8, uncomment the
following supportedOS node-->
    <!--<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"></supportedOS-->

    </application>
</compatibility>

<!-- Enable themes for Windows common controls and dialogs (Windows XP and later)
-->
<!-- <dependency>
    <dependentAssembly>
        <assemblyIdentity
            type="win32"
            name="Microsoft.Windows.Common-Controls"
            version="6.0.0.0"
            processorArchitecture="*"
            publicKeyToken="6595b64144ccf1df"
            language="*"
        />
    </dependentAssembly>
</dependency>-->

</asmv1:assembly>

```

Notice how a huge number of comments help you understand the file content and how you should manage the behavior of your application according to the Windows version that your application is going to target. The `requestedExecutionLevel` element enables you to specify which permission level must be requested to the UAC. You have three possibilities, explained in Table 2.2.

TABLE 2.2 UAC Settings

Setting	Description
<code>asInvoker</code>	Runs the application with the privileges related to the current user. If the current user is a standard user, the application will be launched with standard privileges. If the current user is an administrator, the application will be launched with administrative privileges. This is the default selection in the manifest.
<code>requireAdministrator</code>	The application needs administrative privileges to be executed.
<code>highestAvailable</code>	Requires the highest privilege level possible for the current user.

To specify a privilege level, just uncomment the line of XML code corresponding to the desired level. You can also delete the `requestedExecutionLevel` node in case you want to use file and registry virtualization for backward compatibility with older versions of the Windows operating system.

PAY ATTENTION TO UAC REQUIREMENTS

Be careful of the combination of activities that you need to execute on the target machine and the user privileges because bad UAC settings could cause big problems. A good practice is selecting the `asInvoker` level and architecting your application in a way that it will work on user-level folders and resources. In some situations, you will need deeper control of the target machine and administrator privileges, but these should be considered exceptions to the rule.

2

The Application Framework

In the lower part of the screen is the Enable Application Framework group, a feature that allows executing special tasks at the beginning and at the end of the application lifetime. For Console applications it is not available, but it is relevant to other kinds of applications; for instance, in the Windows Forms application it enables the setting of a splash screen or establishing which form is the main application form. The application framework is discussed in Chapter 19.

Target Framework

From the Target Framework combo box, you can select the version of the .NET Framework that your application will target. The main difference with the same selection that you can do when creating a new project is that here you can target the .NET Framework Client Profile for versions 4.0, 3.5 Service Pack 1, and 3.5 Server Core. The .NET Framework Client Profile is a subset of the .NET Framework that provides the infrastructure for client applications and that can be included in your deployments instead of the full version. Microsoft removed the .NET Framework Client Profile in version 4.5.

NOTE FOR VISUAL STUDIO 2010 USERS

Until Visual Studio 2010, the Target Framework option was available in the Advanced Compile Options dialog box. In Visual Studio 2012 it has been moved to the Application tab of My Project.

Compiling Projects

Compiling a project (or *building* according to the Visual Studio terminology) is the process that produces a .NET assembly starting from your project and source code (according to the .NET architecture described in Chapter 1, “Introducing the .NET Framework 4.5”). An assembly can be a standalone application (.exe assembly) or a .NET class library

(.dll assembly). To compile your project into an assembly, you need to click the **Build** command in the Build menu. Notice that the Build command is followed by the name of your project. When invoking this command, Visual Studio launches, behind the scenes, the Visual Basic command-line compiler (Vbc.exe) and provides this tool all the necessary command-line options. For solutions containing different kinds of projects, Visual Studio launches MSBuild.exe, a command-line utility that can compile entire solutions containing several projects written in different languages and of different types. At the end of the build process, Visual Studio shows a log inside the Output window. Figure 2.23 shows the output log of the build process for the MyFirst2012Program sample application.

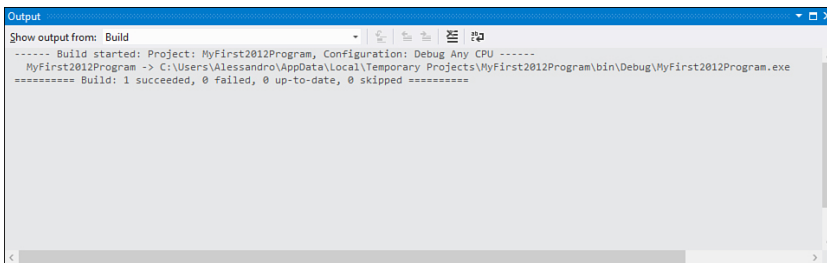


FIGURE 2.23 The Output window shows the compilation process results.

The compilation log shows useful messages that help you understand what happened. In this case there were no errors, but in situations in which the compilation process fails because of some errors in the code, you will be notified of the errors found by the compiler. The Error List window shows a complete list of error messages and warnings and enables you to easily understand where the errors happened by double-clicking the error message. This operation redirects you to the code that generated the error. The executable (or the executables, in the case of more than one project in the solution) will be put in a subfolder within the project's directory, called Bin\Debug or Bin\Release, depending on the output configuration you chose. Configurations are discussed next.

Debug and Release Configurations

Visual Studio provides two default possibilities for compiling your projects. The first one is related to the debugging phase in the development process and includes debug symbols that are necessary for debugging applications. The second one is related to the end of the development process and is the one you will use when releasing the application to your customers. Both ways are represented by configurations. By default, Visual Studio offers two built-in configurations: Debug and Release. When the Debug configuration is active, the Visual Basic compiler generates debug symbols that the Visual Studio debugger can process. Without these symbols, you cannot debug your applications with the Visual Studio debugger. The Release configuration basically excludes debug symbols from the build process. It is the configuration you will use when building the final version of your

application—that is, the executable you will release to your customers. To set the current configuration, you have two possibilities:

- ▶ Use the combo box located on the Visual Studio toolbar.
- ▶ Access the Compile options inside the My Project window.

Figure 2.24 shows the Compile tab in My Project.

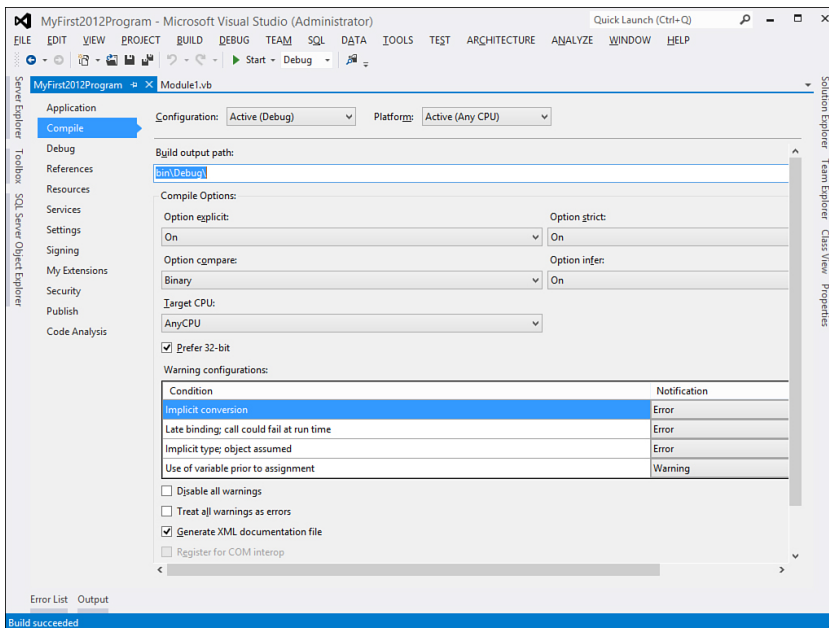


FIGURE 2.24 Compile options in the My Project window.

At the top of the window is a combo box called Configuration. There you can select the most appropriate configuration for you. By default, Visual Studio 2012 is set up on the Debug configuration. You could also consider building a custom configuration (although both Debug and Release can be customized instead of making new ones), which will be discussed next. For our purposes, it's suitable to leave unchanged the selection of the Debug configuration as the default because we will study the Visual Studio debugging features in depth.

Creating Custom Configurations with Configuration Manager

You might have situations in which both the Debug and Release configurations are not enough for your needs. As mentioned in the previous paragraph, with Visual Studio 2012 you can also create your custom configuration. To accomplish this, you need to access the Configuration Manager tool (see Figure 2.25), which is reachable using the Configuration

Manager command in the Build menu. There you can edit an existing configuration or create a new one.

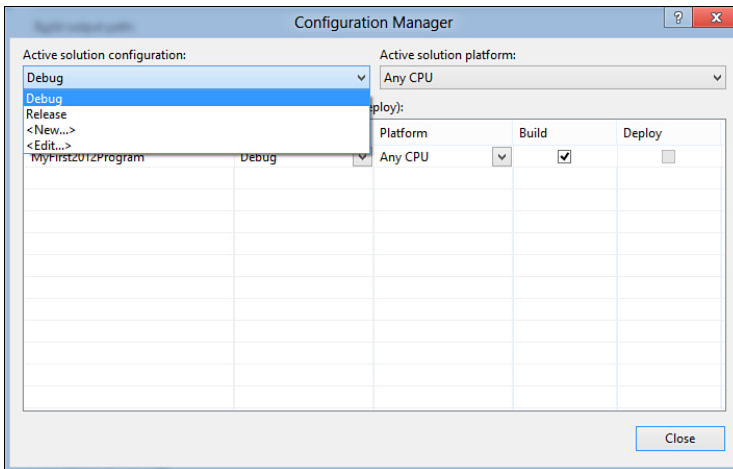


FIGURE 2.25 The Configuration Manager.

To create a new custom configuration, perform the following steps:

1. Click the **Active Solution Configuration** combo box and select the **New** option.
2. In the New Solution Configuration window, specify a name for the new configuration and select an existing configuration (such as Debug) from which settings will be copied. Figure 2.26 shows an example.

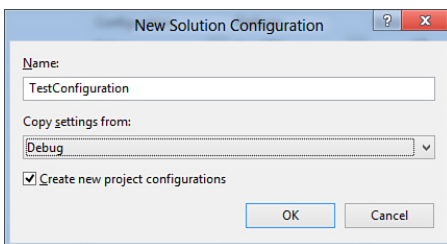


FIGURE 2.26 Creating a custom configuration that imports settings from an existing one.

3. When done, click **Close**.
4. Click the **Advanced Compiler Options** button in the Compile tab and specify which compile options must affect the new configuration. For example, you could decide to affect just compilations against 64-bit processors, so you would need to change the value in the Target CPU combo box. This is the point at which you can modify your real configuration settings and how your project should be built.

Such modification influences just the new configuration. Moreover, if you decide to use the new configuration, you will have a new subfolder under the Bin subfolder in your project's main folder, which takes the name of your custom configuration and contains the output of the build process made with that configuration. For our MyFirst2012Program sample, the project folder is named MyFirst2012Program and contains the Bin subfolder, which also contains the default Debug and Release subfolders. With your custom configuration, a new TestConfiguration subfolder will be available under Bin.

Background Compiler

Visual Basic 2012 offers a great feature: the background compiler. While you write your code, the IDE invokes the Visual Basic compiler that will immediately compile the code and notify you about errors that occur, writing messages in the Error List window. This is possible because the Visual Basic compiler can compile your code on-the-fly while you type. As you can imagine, this feature is important because you will not necessarily need to build your project each time to understand whether your code can be successfully compiled. Typical examples of the background compiler in action are error messages shown in the Error List window when typing code. Refer to Figure 2.19 to get an idea of this feature. You can double-click the error message to be redirected to the line of code that caused the error. Also, the IDE underlines code containing errors with squiggly lines so that it is easier to understand where the problem is.

Other Compile Options

Visual Studio 2012 enables developers to get deep control over the build process. With particular regard to Visual Basic, you can control other compile options that are specific to the language. Table 2.3 lists them in detail.

TABLE 2.3 Visual Basic Compile Options

Option	Meaning
Option Explicit	When set to On, the developer must declare an object before using it in code.
Option Strict	When set to On, the developer must specify the type when declaring objects. In other words, Object is not automatically assigned as the default type. Moreover, Option Strict On disallows late binding and conversions from one type to another where there is a loss of precision or data. You should always set Option Strict On unless required.
Option Compare	Determines which method must be used when comparing strings (Binary or Text). The Binary option enables the compiler to compare strings based on a binary representation of the characters, while the Text option enables string comparisons based on textual sorting, according to the local system international settings.
Option Infer	When set to On, enables local type inference (this feature is discussed in Chapter 20, “Advanced Language Features”).

OPTION STRICT ON

By default, `Option Strict` is `Off`. You can set it to `On` each time you create a new project, but you can also change the default setting by clicking Tools, Options. Then in the Options dialog box expand the Projects and Solutions node; finally select the VB Defaults element and change the default setting.

Options shown in Table 2.3 are also considered by the background compiler, so you will be immediately notified when your code does not match these requirements. You can also specify how the Visual Basic compiler has to treat some kind of errors. This is what you see next.

Target CPU

You can specify the CPU architecture your applications will target. You can choose among 32-bit architectures (x86), 64-bit architectures (x64), Itanium processors, ARM (in the case of Windows Store Apps) or any architecture (AnyCPU). Until Visual Studio 2010 this setting was available in the Advanced Compile Options dialog box.

Warning Configurations

Warning configurations state how the Visual Basic compiler should notify the developer of some particular errors, if just sending warning messages (which will not prevent from compiling the project) or error messages (which will instead prevent from completing the build process).

DO NOT IGNORE WARNING MESSAGES

Even if warning messages will not prevent the completion of the build process, they should never be blindly ignored. They could be suggestions of potential exceptions at runtime. You should always accurately check why a warning message is thrown and, possibly, solve the issue that caused the warning. A typical example of when warnings could be ignored is when running code analysis on code that does not need to be compliant with Microsoft specifications (for example, the user interface side of a WPF application). In all other situations, you should be careful about warnings.

Depending on how you set the Visual Basic compile options discussed in the previous paragraph, Visual Studio will propose some default scenarios for sending notifications (and, consequently, influencing the build process). Table 2.4 lists the available warning conditions.

TABLE 2.4 Warning Condition Details

Condition	Description
Implicit conversion	<p>Checked when trying to assign an object of a type to an object of another type. For example, the following code will cause the condition to be checked (implicit conversion from <code>Object</code> to <code>String</code>):</p> <pre>Dim anObject As Object = "Hi!" Dim aString As String = anObject</pre>

Condition	Description
Late binding	Checked when trying to assign at runtime a typed object to another one of type <code>Object</code> .
Implicit type	<p>Checked when not specifying the type for an object declaration. If <code>Option Infer</code> is on, this condition is checked only for declarations at class level. For example, the following class-level declaration would cause the condition to be checked:</p> <pre>Private Something</pre> <p>This condition is determined by <code>Option Strict On</code>.</p>
Use of variable prior of assignment	<p>Checked when attempting to use a variable that doesn't have a value yet. This is typical with instance variables. The following code causes this condition to be checked:</p> <pre>Dim p As Process Console.WriteLine(p.ProcessName.ToString)</pre> <p>In this case <code>p</code> must get an instance of the <code>Process</code> object before attempting to use it.</p>
Function/operator without return value	Checked when a <code>Function</code> method or an operator definition performs actions without returning a value.
Unused local variable	Checked when a variable is declared but never used. It's a good practice to remove unused variables both for cleaner code and for memory allocation.
Instance variable accesses shared members	Checked when trying to invoke a member from an instance object that is instead a shared member.
Recursive operator or property access	Checked when trying to use a member (properties or operators) inside the code block that defines the member itself.
Duplicate or overlapping catch blocks	<p>Checked when a <code>Catch</code> clause inside a <code>Try...Catch...End Try</code> code block is never reached because of inheritance. The following code causes the condition to be checked because the <code>FileNotFoundException</code> inherits from <code>Exception</code> and therefore should be caught before the base class; otherwise, <code>Exception</code> would be always caught before derived ones:</p> <pre>Try Catch ex As Exception Catch ex As FileNotFoundException End Try</pre>

You also have the ability to change single notifications; just select the most appropriate notification mode for your needs. Based on the explanations provided in Table 2.4, be careful about the consequences that this operation could cause. If you are not sure about consequences, the best thing is leaving default options unchanged. Three other compile options are listed at the bottom of the `Compile` tab and described in Table 2.5.

TABLE 2.5 Additional Compile Options

Option	Description
Disable all warnings.	The Visual Basic compiler will not produce warning messages.
Treat all warnings as errors.	The Visual Basic compiler will treat all warning messages as if they were errors.
Generate XML documentation file.	When flagged, enables Visual Studio to generate an XML file for documenting the source code. If XML comments are included in the code, this file also contains descriptions and detailed documentation for the code. This is useful when you need to automate the documentation process for your class libraries.

Advanced Compile Options

You can specify advanced settings for the build process. To accomplish this, you need to click the **Advanced Compile Options** button.

COMPILER SETTINGS AND CONFIGURATIONS

Advanced compiler settings are at the configuration level. This means that the Debug configuration has its own advanced settings, the Release configuration has its own settings, and your custom configurations will have their own settings. Please remember this when providing advanced settings.

Figure 2.27 shows the Advanced Compiler Settings window.

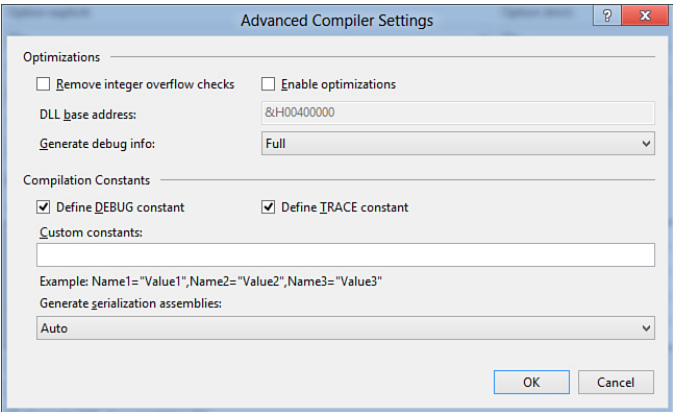


FIGURE 2.27 The Advanced Compiler Settings window.

Here you can set compiler options to drive the build process. Next we discuss options in detail.

Optimizations

The Optimization tab offers options that would potentially lead to building a smaller and faster executable. This tab is composed of four options that we discuss.

Remove Integer Overflow Checks When you make calculations in your code against `Integer` or `Integer`-style data types, the Visual Basic compiler checks that the result of the calculation falls within the range of that particular data type. By default, this option is turned off so that the compiler can do this kind of check. If you flag this check box, the compiler will not check for such overflows, and the application execution might result faster. Be careful about this choice, especially if your code implements calculations.

Enable Optimizations When this check box is flagged, the compiler basically removes some opcodes that are required for interacting with the debugger. Moreover, the Just-In-Time compilation is optimized because the runtime knows that a debugger will not be attached. On the other hand, this can result in major difficulties when debugging applications. For example, you might not use breakpoints at specific lines of code and, consequently, perform debugging tasks although the optimization process could produce a smaller and faster executable.

DLL Base Address This option is available when developing class libraries and user controls and provides the ability to specify the Base Address for the assembly. As you might know, the base address is the location in memory where a .dll file is loaded. By default, Visual Studio assigns a base address and represents it in hexadecimal format. If you need to provide a custom base address, this is the place where you can do it.

Generate Debug Information Generating debug information when building your project allows you to use the debugger against your application. By default, this option is set to Full, which means that full debug information is generated so that the debugger can be fully used to debug an application. (This is the case of the Debug configuration.) If you set this option to None, no debug information will be generated; if you set this option to pdb-only, the compiler will produce just a .pdb file containing debug symbols and project state information.

Compilation Constants

You can use compilation constants to conditionally compile blocks of code. Conditional compilation relies on the evaluation to True of constants that will be included in the final assembly. The Visual Basic compiler defines some default constants you can evaluate within your code; you also have the ability to declare custom constants. In the Advanced Compiler Settings window, you can specify whether the compiler needs to include the `DEBUG` and `TRACE` constants. The first one enables you to understand if the application is running in debug mode; in other words, if the application has been compiled using the Debug configuration. The second one is also related to debugging tasks; particularly, the .NET Framework exposes a class called `Trace` that is used in debugging and that can send the tracing output to the Output window when the `TRACE` constant is defined. If not, no output is generated because invocations versus the `Trace` class are ignored. A full list of built-in constants can be found at MSDN Library at [http://msdn.microsoft.com/en-us/library/dy7yth1w\(VS.110\).aspx](http://msdn.microsoft.com/en-us/library/dy7yth1w(VS.110).aspx). Evaluating constants in code is simple. You can use the `#If`, `#Else`, `#ElseIf`, and `#EndIf` directives. For example, if you want to evaluate

whenever an application has been compiled with the Debug configuration, you could write the following code:

```
#If DEBUG Then
    Console.WriteLine("You are in Debug configuration")
#Else
    Console.WriteLine("You are not in Debug configuration")
#End If
```

which essentially verifies if the constant is defined and takes some action at that point. In our example, if the `DEBUG` constant is defined in the assembly, this means that it has been built via the Debug configuration.

Custom Constants You can also define custom constants. This can be basically accomplished in two ways. The first is adding custom constants in the appropriate field of the Advanced compiler settings window. Each constant must have the form of `Name="Value"`, and constants are separated by commas. The second way for providing custom constants is adding a `#Const` directive in your code. For example, the following line of code

```
#Const TestConstant = True
```

defines a constant named `TestConstant` whose value is set to `True`. The big difference in using a `#Const` directive is that it defines just private constants that have visibility within the code file that defines them.

Generate Serialization Assemblies

As we discuss in Chapter 41, “Serialization,” serialization in .NET development is a technique that allows persisting the state of an object. Among several alternatives, this can be accomplished using a class called `XmlSerializer`. In such situations, the Visual Basic compiler can optimize applications that use the `XmlSerializer` class, generating additional assemblies for better performances. By default, this option is set to **Auto** so that Visual Studio generates serialization assemblies only if you are effectively using XML serialization in your code. Other options are **On** and **Off**.

Debugging Overview

In this section you get an overview of the debugging features in Visual Studio 2012 for Visual Basic applications. Although the debugger and debugging techniques are detailed in Chapter 5, “Debugging Visual Basic 2012 Applications,” here we provide information on the most common debugging tasks, which is something that you need to know in this first part of your journey through the Visual Basic programming language.

Debugging an Application

To debug a Visual Basic application, you basically need to perform two steps:

- ▶ Enable the Debug configuration in the compile options.
- ▶ Press F5 to start debugging.

By pressing F5, Visual Studio runs your application and attaches an instance of the debugger to the application. Because the Visual Studio debugger needs the debug symbols to proceed, if you do not choose the Debug configuration, you cannot debug your applications. The instance of the debugger detaches when you shut down your application.

TIP

As an alternative you can click the Start button on the Visual Studio standard toolbar. If the Debug configuration is selected, this action will be the equivalent of pressing F5. If the Release configuration is selected, this is the equivalent of launching the application with CTRL + F5.

The debugger monitors your application's execution and notifies for runtime errors; it allows you to take control over the execution flow as well. Figure 2.28 shows our sample application running with the Visual Studio debugger attached.

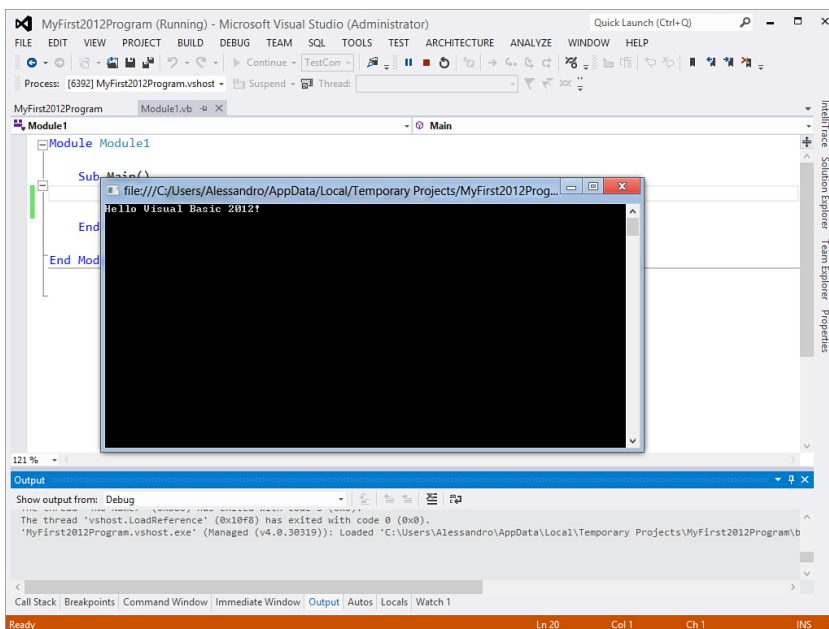


FIGURE 2.28 Our sample application running with an attached instance of the Visual Studio debugger.

In the bottom area of the IDE, you can notice the availability of some tabs, such as Locals, Watch 1, Watch 2, Call Stack, Breakpoints, Command Window, Immediate Window, and Output. Each tab represents a tool window that has specific debugging purposes. Also, notice how the status bar becomes orange and an orange border is placed around the IDE, to remind the developer that the IDE is running in debugging mode. The Visual Studio debugger is a powerful tool; next you learn the most important tasks in debugging

applications. Before explaining the tooling, it is a good idea to modify the source code of our test application so that we can cause some errors and see the debugger in action. We could rewrite the `Sub Main` method's code, as shown in Listing 2.3.

LISTING 2.3 Modifying the Sub Main for Debugging Purposes

```
Sub Main()  
  
    'A text message  
    Dim message As String = "Hello Visual Basic 2012!"  
  
    Console.WriteLine(message)  
  
    'Attempt to read a file that does not exist  
    Dim getSomeText As String =  
        My.Computer.FileSystem.ReadAllText("FakeFile.txt")  
  
    Console.WriteLine(getSomeText)  
    Console.ReadLine()  
  
End Sub
```

NEW TO VISUAL BASIC .NET?

If you are not an existing Visual Basic .NET developer, you may not know some of the objects and keywords shown in the code listings of this chapter. The code is the simplest possible, so it should be easy to understand; comments are also provided. The next chapters guide you to the details of the programming language, so everything used here will be explained. At the moment, it is important for you to focus on the instrumentation more than on the code.

The code simply declares a message object of type `String`, containing a text message. This message is then shown in the Console window. This is useful for understanding breakpoints and other features in the code editor. The second part of the code will try to open a text file, which effectively does not exist and store its content into a variable called `getSomeText` of type `String`. We need this to understand how the debugger catches errors at runtime, together with the edit and continue feature.

Breakpoints and Data Tips

Breakpoints enable you to control the execution flow of your application. A breakpoint breaks the execution of the application at the point where the breakpoint itself is placed so that you can take required actions (a situation known as *break mode*). You can then resume the application execution. To place a breakpoint on a specific line of code, just place the cursor on the line of code you want to debug and then press **F9**.

TIP

To add a breakpoint, you can also right-click the line of code you want to debug and select the Breakpoint, Insert breakpoint command from the pop-up menu or just click the left-most column in the code window.

A breakpoint is easily recognizable because it highlights in red the selected line of code (see Figure 2.29).

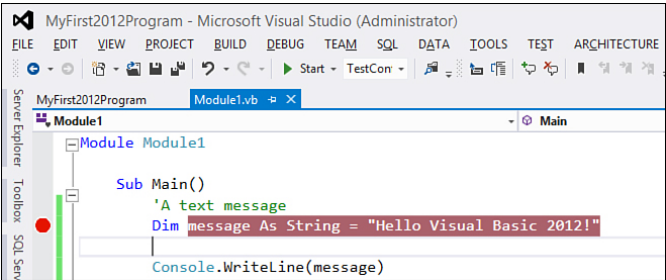


FIGURE 2.29 Placing a breakpoint in the code editor.

To see how breakpoints work, we can run the sample application by pressing F5. When the debugger encounters a breakpoint, it breaks the execution and highlights in yellow the line of code that is being debugged, as shown in Figure 2.30, before the code is executed.

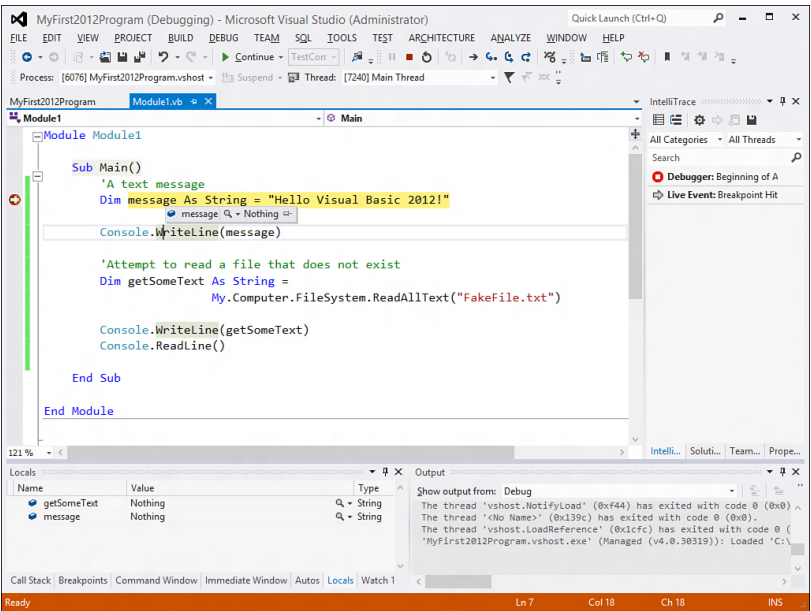


FIGURE 2.30 When encountering a breakpoint, Visual Studio highlights the line of code that is currently debugged.

If you take a look at Figure 2.30, you notice that, if you pass with the mouse pointer over the message variable, IntelliSense shows the content of the variable itself, which at the moment contains no value (in fact, it is set to `Nothing`). This feature is known as Data Tips and is useful if you need to know the content of a variable or of another object in a particular moment of the application execution.

THE VISUAL STUDIO HISTORICAL DEBUGGER

If you run the Microsoft Visual Studio 2012 Ultimate edition, you also notice another window called IntelliTrace. This window is also known as the Visual Studio Historical Debugger and is specific to the Visual Studio Ultimate instrumentation. It is discussed in Chapter 55, “Advanced Analysis Tools.”

You can then execute just one line of code at a time, by pressing **F11**. For example, supposing we want to check if the message variable is correctly initialized at runtime. We could press F11 (which is a shortcut for the Step Into command in the Debug menu). The line of code where the breakpoint is placed will now be executed, and Visual Studio will highlight the next line of code. At that point, you can still pass the mouse pointer over the variable to see the assignment result, as shown in Figure 2.31.

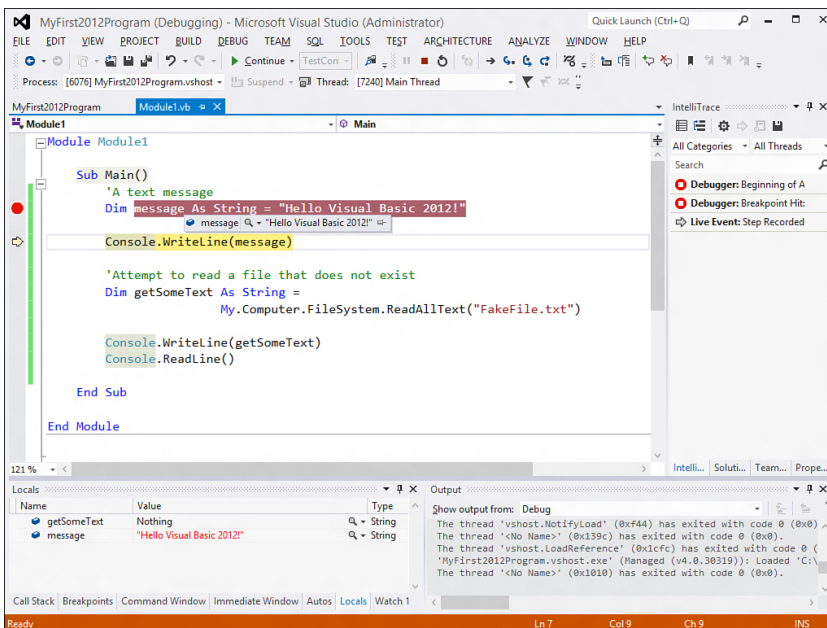


FIGURE 2.31 Using the Step Into command lets us check whether the variable has been assigned correctly.

When you finish checking the assignments, you can resume the execution by pressing **F5**. The execution of the application continues until another breakpoint or a runtime error is encountered. We discuss this second scenario next.

Runtime Errors

Runtime errors are particular situations in which an error occurs during the application execution. These are not predictable and occur due to programming errors that are not visible at compile time. Typical examples of runtime errors are when you create an application and you give users the ability to specify a filename, but the file is not found on disk, or when you need to access a database and pass an incorrect SQL query string. Obviously, in real-life applications you should predict such possibilities and implement the appropriate error handling routines (discussed in Chapter 6, “Handling Errors and Exceptions”), but for our learning purposes about the debugger, we need some code that voluntarily causes an error. Continuing the debugging we began in the previous paragraph, the application’s execution resumption causes a runtime error because our code is searching for a file that does not exist. When the error is raised, Visual Studio breaks the execution as shown in Figure 2.32.

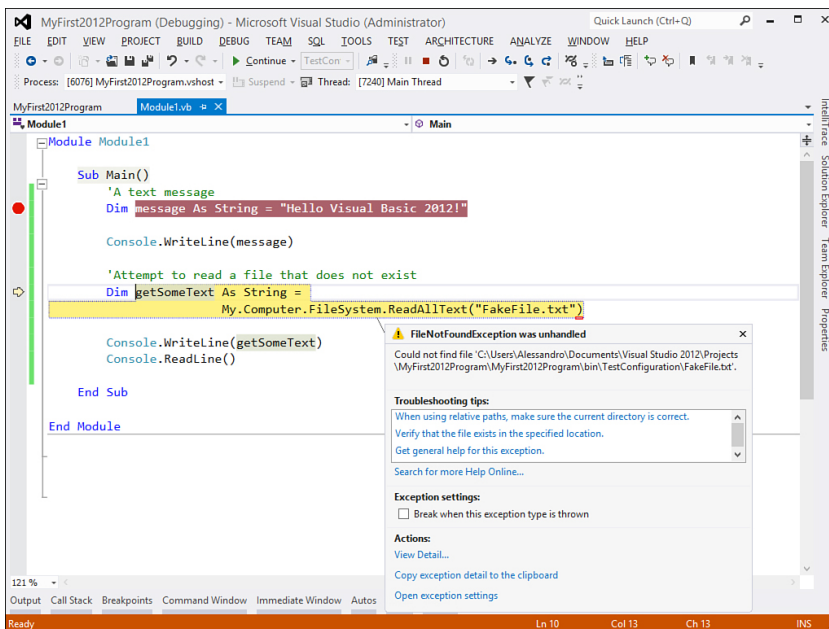


FIGURE 2.32 The Visual Studio debugger encounters a runtime error.

As you can see, the line of code that caused the error appears highlighted. You also can see a pop-up window that shows some information about the error. In our example, the code searched for a file that does not exist, so a `FileNotFoundException` was thrown and was not handled by error handling routines; therefore, the execution of the application

was broken. Visual Studio also shows a description of the error message. (In our example it communicates that the code could not find the FakeFile.txt file.) Visual Studio also shows some suggestions. For example, the Troubleshooting tips suggest some tasks you could perform at this point, such as verifying that the file exists in the specified location, checking the pathname, or getting general help about the error. By clicking a tip, you are redirected to the MSDN documentation about the error. This can be useful when you don't exactly know what an error message means. There are other options within the Actions group. The most important is View Detail. This enables you to open the View Detail window, which is represented in Figure 2.33. Notice how the StackTrace item shows the hierarchy of calls to classes and methods that effectively produced the error. Another interesting item is the InnerException. In our example it is set to Nothing, but it's not unusual for this item to show a kind of exceptions tree that enables you to better understand what actually caused an error. For example, think of working with data. You might want to connect to SQL Server and fetch data from a database. You could not have sufficient rights to access the database, and the runtime might return a data access exception that does not allow you to immediately understand what the problem is. Browsing the InnerException can help you understand that the problem was caused by insufficient rights. Going back to the code, this is the point where you can fix it and where the Edit and Continue features comes in.

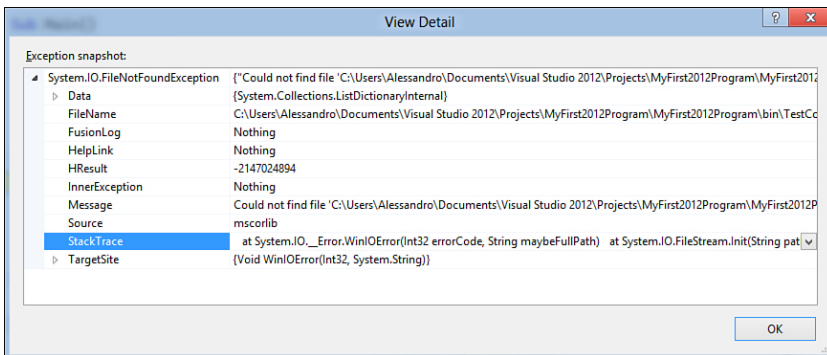


FIGURE 2.33 The View Detail window enables developers to examine what caused an exception.

Edit and Continue

The Edit and Continue features enable you to fix bad code and resume the application execution from the point where it was broken, without having to restart the application. You just need to run the application by pressing F5; then you can break its execution by pressing **Ctrl+Alt+Break** or either selecting the **Break All** command in the Debug menu or pressing Pause on the Debug toolbar.

AVAILABILITY OF EDIT AND CONTINUE

Generally, you can use the Edit and Continue features, but there are situations in which you will not. For example, if fixing your code might influence the general application behavior, you need to restart the application. Also, Edit and Continue is not available when running configurations that target 64-bit CPUs nor in Silverlight applications.

In our example we need to fix the code that searches for a not existing file. We can replace the line of code with this one:

```
Dim getSomeText As String = "Fixed code"
```

This replaces the search of a file with a text message. At this point we can press **F5** (or **F11** if we want to just execute the line of code and debug the next one) to resume the execution. Figure 2.34 shows how the application now runs correctly. The Edit and Continue feature completes the overview of the debugging features in Visual Studio. As we mentioned before, this topic is covered in detail in Chapter 6.

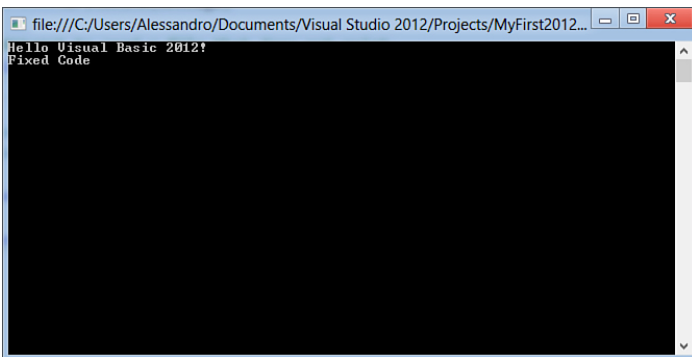


FIGURE 2.34 The sample application running correctly after fixing errors.

After this brief overview of the debugging features in Visual Studio 2012, it's time to talk about another important topic for letting you feel at home within the Visual Studio 2012 IDE when developing applications: getting help and documentation.

Browsing the Visual Basic and .NET Documentation

The *.NET Framework Base Class Library* is very large, and remembering all the objects that you can use in your applications (or the ones that .NET Framework relies on) is not possible. What is instead important is to know where to search for information. You have different tools available to browse the .NET Framework and its documentation, for Visual Basic, too. Because the goal of this chapter is to provide information on the primary tools you need for developing Visual Basic applications, getting help with the language and with the tooling is absolutely one of the primary necessities, as discussed next.

Online Help and the MSDN Library

Visual Studio 2012 ships with the MSDN Library, which is the place where you can find documentation for Visual Basic 2012 and the .NET Framework 4.5. There are basically two ways to access the MSDN Library: offline and online. To access the MSDN Library offline, you have the following alternatives:

- ▶ Click the **View Help** command from the Help menu in Visual Studio.
- ▶ Press **F1** from wherever you are.
- ▶ Open the **Microsoft Help Viewer** shortcut in Windows's Start, All Programs, Microsoft Visual Studio 2012 menu; this launches the local help viewer.

If you are writing code or performing a particular task on a tool within the IDE, pressing **F1** is the best choice because you will be redirected to the help page related to that instruction, code statement, or tool. If you are instead searching for information about a particular technology or framework, such as WPF or the Visual Studio Tools for Office, you could consider one of the other choices. To access the MSDN Library online, you just need an Internet connection. Then you can specify to always use the online help by selecting **Help, Set Help Preference** and then clicking **Launch in Browser** or **Launch Help Viewer**, or you can manually open one of the following websites, which are the main points of interest for a Visual Basic developer:

- ▶ The MSDN Library portal at <http://msdn.microsoft.com/en-us/library/default.aspx>
- ▶ The .NET Framework reference at [http://msdn.microsoft.com/en-us/library/w0x726c2\(VS.110\).aspx](http://msdn.microsoft.com/en-us/library/w0x726c2(VS.110).aspx)
- ▶ The Visual Basic page on the Visual Studio Developer Center at <http://msdn.com/vbasic>

You can also quickly find information on particular objects using built-in tools, such as the Object Browser.

MANAGING HELP CONTENTS

You can download additional documentation or remove documentation that is already on your system by selecting **Help, Add or Remove Help Content**.

Object Browser Window

The Object Browser is a special tool window that enables you to browse the .NET Framework class library as well as any referenced libraries and types defined in your projects. You can get a hierarchical view of the Base Class Library and of all the types defined in your solution, including types defined in referenced external assemblies. The Object Browser can be activated by pressing **CTRL+ALT+J**; it is useful because you can understand how a type is defined, which members it exposes, which interfaces it implements,

and which other classes it derives from. If the types are documented, you can get a description for each object or member.

Figure 2.35 represents, as an example, the Object Browser showing members of the `System.Windows.ContentElement` class.

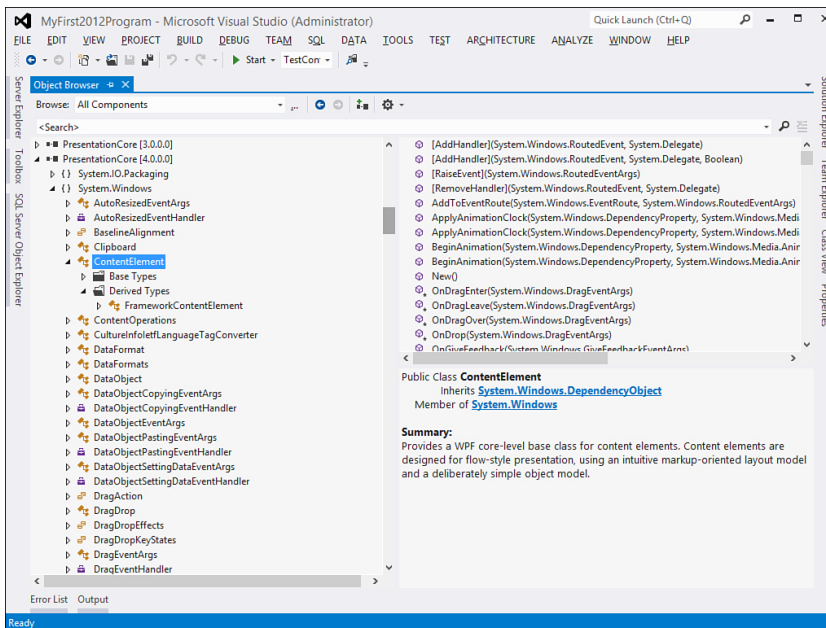


FIGURE 2.35 The Object Browser enables exploring .NET objects showing information.

The right side of the window lists methods and properties exposed by the selected object. When you click on a method or on a member of the object in the left side of the window, a short description of the object should appear in the bottom-right side of the Object Browser. If the description is not useful enough to understand the meaning of an object or of one of its members, you can press **F1**, and Visual Studio shows the online help (if available) for the object or member. The Object Browser also provides links to objects used by the one you are exploring. Considering the example shown in Figure 2.35, you not only can see the description of a method, but also can also click the parameters' identifiers to be redirected to the definition of the parameter. The Object Browser can also be invoked when writing code, as discussed next.

Invoking the Object Browser from the Code Editor

Often you need to know how particular .NET objects or members are structured or how they work. Visual Studio 2012 provides the ability of invoking the Object Browser directly from the code editor by right-clicking the object you want to browse and selecting the **Go to Definition** command from the pop-up menu. For example, imagine you

want to know how the `Console` class is defined. To accomplish this, you can revisit the `MyFirst2012Program` example. When in the code editor, right-click the `Console` object and select `Go To Definition`. By doing this, Visual Studio opens the Object Browser that automatically selects the `Console` class, showing its methods on the right side of the screen (see Figure 2.36).

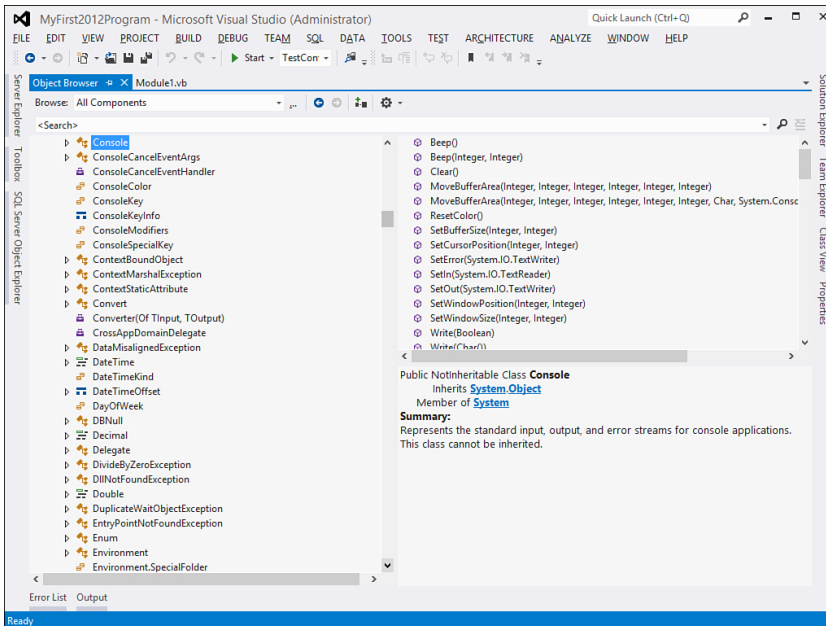


FIGURE 2.36 The Object Browser can be invoked from the code editor by clicking the `Go to Definition` command.

This technique works with shared classes or, more generally, with declarations of noninstance classes; however, in some situations you might want to learn about data types or instance members' definitions. For example, consider the following code:

```
Dim text As String
text = "Hi!"
```

If you try to run the `Go to Definition` command for the `text` identifier, you will be redirected to the first line of code, which effectively defines the `text` object. But what if you want to browse the `String` object? Fortunately there is another command you can choose in such situations—`Go to Type Definition`. It is still available in the pop-up menu. Invoking `Go to Type Definition` redirects to the definition of the type that characterizes the object you declared (in our example, `String`). The result will be the same as in Figure 2.36, of course referring to the selected type.

NOTE

Instance and shared members are discussed in detail in Chapter 7, “Class Fundamentals.”

Although the Object Browser’s purpose is not typically to provide help, it is a good place for learning about .NET objects, both if you need information on their structure and if you need descriptions on their usage.

2

Quick Launch Tool

Visual Studio 2012 introduces a new tool called Quick Launch. The purpose of this tool is making it easy to find commands, options, and recently opened files. The tool is available through a search box located at the upper right corner of the IDE. For example, imagine you need to launch the SQL Server Object Explorer window but do not remember where the command for launching such a window is and do not want to waste time browsing every menu. If you type “SQL” in the Quick Launch search box, Visual Studio will show all menu commands, options, and recently opened files (if any) containing the “SQL” word. As you can see from Figure 2.37 the SQL Server Object Explorer is the first search result, so you can simply click it to open the desired tool.

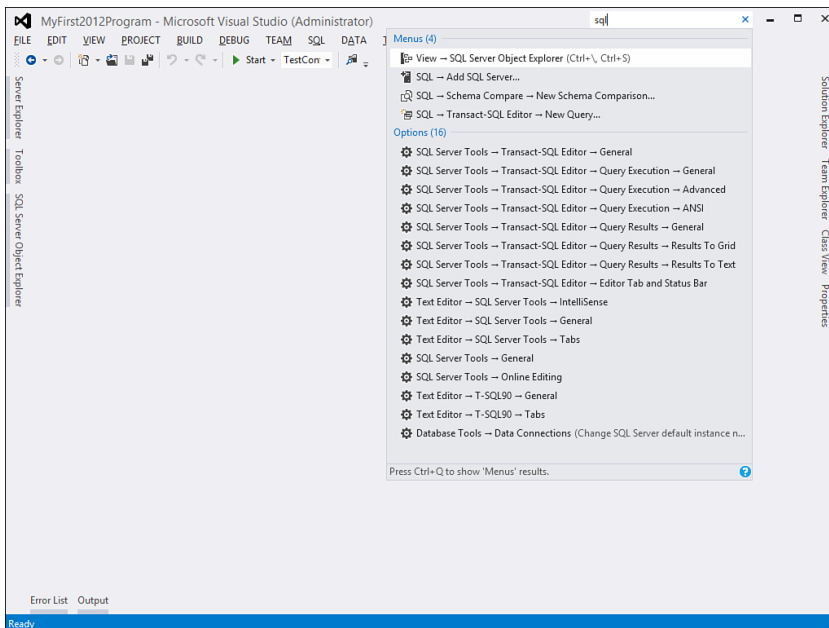


FIGURE 2.37 Finding commands and options with Quick Launch.

To understand how it works, repeat the search and select the SQL Server Tools -> General option from the list. You will see how Visual Studio will open the Options dialog box pointing to the requested setting. This is a useful tool that can save a lot of time in finding the necessary tools.

Showing the Hierarchy of Method Calls

Visual Studio 2012 brings to Visual Basic a tool window named Call Hierarchy, which was already introduced for Visual C# in Visual Studio 2010. As its name implies, Call Hierarchy enables the showing of the hierarchy of calls to and from one or more methods. To understand how it works, let's consider the following code:

```
Module Module1

    Sub Main()
        DoSomething()
    End Sub

    Sub DoSomething()
        DoSomethingElse()
    End Sub

    Sub DoSomethingElse()
        DoNothing()
    End Sub

    Sub DoNothing()
        '
    End Sub

End Module
```

The code defines some method, without performing any particular tasks, but it demonstrates a nested hierarchy of method calls. If you right-click one of the method's names and then select View Call Hierarchy from the pop-up menu, you will be able to see the method call hierarchy as demonstrated in Figure 2.38.

As you can see from Figure 2.38, the tool shows the first level hierarchy, but you can also expand method names to show the full hierarchy, including nested method calls. On the right side of the window is the line number in the code file where the selected method is defined. Also, if you double-click a method name in Call Hierarchy, the code editor will automatically focus on the method definition.

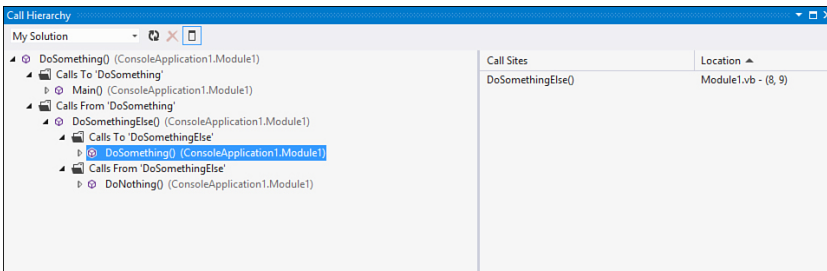


FIGURE 2.38 Analyzing method calls with Call Hierarchy.

Summary

In this chapter we discussed basic things you need to know as a Visual Basic developer to feel at home within the Visual Studio 2012 Integrated Development Environment. Tasks such as creating projects, compiling projects, debugging applications, and searching for documentation and tools are the most common in a developer's life, and this chapter offers a fast way to understand all the primary tools you need for building applications with Visual Basic 2012. You also saw some new tools introduced by Visual Studio 2012, such as Quick Launch and Call Hierarchy. Now that you know how you can move inside the IDE, it's time to begin working with the Visual Basic programming language.

This page intentionally left blank

Index

NUMBERS

1-Click deployment, 884
3-D graphics, Silverlight applications, 908
64-bit browsers, support, 928

SYMBOLS

* (multiplication) operator, 155
+ (addition) operator, 155, 313
- (pointer to address in memory) keyword, 95
- (subtraction) operator, 155, 313
/ (division) operator, 155, 313
<> (inequality) operator, 104, 164, 313
< (less than) operator, 164, 313
<< (left-shift) operator, 162, 313
<= (less than or equal to) operator, 164, 313
= (equality) operator, 126, 164, 313, 570
> (greater than) operator, 164, 313
>= (greater than to equal to) operator, 164, 313
>> (right-shift) operator, 162, 313
\
(integer division) operator, 155, 313
^ (exponentiation) operator, 155, 313
_(underscore) character, 70, 265, 355, 552

A

ABC (Address, Binding, and Contract) properties, 992
abstract classes, 336
 CLS, 337
 code, 427
 contra variance, 536
 inheritance, 337
 interfaces, implementing, 428
accepting licenses, NuGet, 1285

Access (Microsoft), 541
accessing
 base class members, 337-341
 CAS, 1148
 ClickOnce, 1257-1258
 databases, 540
 Developer Center (Windows Phone), 987
 directories, 455
 Generate from Usage feature, 433-439
 interfaces, 348-352
 LINQ, 550. *See also* LINQ
 LINQ to SQL, 589, 618
 local file systems, 926
 members, Visual Basic 2012 projects, 67-68
 MSDN Library, 56
 My.Resources property, 500
 properties, 45, 229-234, 232
 Registry, 486-487
 resources, 81
 role configuration options, 935
 System.IO.DriveInfo class, 459
 templates, 19-20
 Visual Basic 2012 tools, 1359
 WCF RIA Services, 911-923
Access property, 428
accounts
 Storage Account service, 930
 UAC, configuring, 36-39
actions
 methods, executing with, 235-247
 XBAP, 722
Activated event, 509
Activity log (Windows Azure), 947
AddAfter method, 410
AddAfterSelf method, 674
AddBeforeSelf method, 674
Add command, 424
AddFirst method, 410, 674
AddHandler keyword, 383-384

adding. *See also* insert operations

- breakpoints, 51
 - class diagrams, 431
 - classes, LINQ to SQL, 589
 - code snippets, 1282
 - columns, 710
 - controls, 870-873
 - Silverlight applications, 897-900
 - XAML, 701
 - DataSets, LINQ to DataSets, 622
 - Domain Service Class, 913-916
 - EDMs, 631
 - environment variables, 1242
 - expressions to Watch windows, 190
 - filtering, 873-874
 - forms, 868-869
 - Imports directives, 286
 - ink notes, 806
 - items
 - to projects, 30
 - to templates, 1265
 - members to interfaces, 426
 - models, 868
 - multiple roles, 936
 - navigation controls, 874
 - objects, Visual Studio Class Designer, 425-428
 - Option Infer directives, 513
 - output to projects (InstallShield), 1236
 - pages, Windows Phone, 966
 - references
 - to COM libraries, 85-86
 - to Data Services, 1023
 - resources, 498
 - services, 1002, 1016
 - setup projects (InstallShield), 1232
 - sources, Silverlight applications, 912-913
 - stored procedures to EDMs, 654
 - strong names to projects, 1225
 - Windows Forms, 1193
 - XML schemas, 1112
- Add-in Manager tool, 1304
- add-ins
- deploying, 1300
 - managing, 1304
 - Windows Phone, 957

- addition (+) operator, 155, 313
- AddLast method, 410
- AddMemoryPressure method, 280
- Add method, 370, 394, 660
 - concurrent collections, 1087
- Add New Item dialog box, 589, 1267
- AddNew method, 833
- AddParticipant method, 1080
- AddParticipants method, 1080
- AddProduct method, 606
- AddRange method, 395
- Add Reference command, 84
- Address, Binding, and Contract properties.
 - See* ABC properties, 992
- addresses, ABC properties, 992
- AddressOf clause, 381
- Add Service Reference dialog box, 1002
- AdHost class, 1194
- Adjust Shapes Width command, 424
- ADO.NET, 539
 - connection modes, 541
 - databases, connecting, 541-543
 - data providers, 540-541
 - disconnected modes, 541
 - Entity Framework, 629
 - Code First approach, 657
 - compatibility, 668-669
 - delete operations, 647-648
 - downloading additions to, 658
 - EDMs, 630-643
 - Fluent APIs, 665-668
 - handling optimistic concurrency, 648-650
 - insert operations, 646-647
 - instantiating, 645
 - LINQ to Entities, querying EDMs with, 652-653
 - mapping stored procedures, 654-657
 - modifying, 645-652
 - overview of, 629-630
 - SQL, querying EDMs with, 653-654
 - update operations, 648
 - validating data, 650-652
- overview of, 540-543
- partial methods, 251
- serialization, 1053-1054

- AdRotator control, 858
- Advanced Compile Options dialog box, 44
- Advanced Compiler Settings window, 46-47
- advanced features, IDEs, 1261
- advanced garbage collection, 279-281
- advanced LINQ to SQL, 613-617
- agents, Windows Phone, 960-963
- Aggregate clause, 570
- AggregateException, 1076, 1101
- Aggregate method, 519
- aggregation operators, 570-572
- Ajax
 - applications, 1015
 - controls, 854
- aligning text, 802
- All method, 519
- allocating memory
 - objects, managing, 269-270
 - reference types/value types, comparing, 106-108
 - structures, 309
 - value types, 94
- AllowMultiple property, 1187
- AllowPartiallyTrustedCallers attribute, 1154
- ambiguities in namespaces, avoiding, 284, 294-295
- analysis tools (Visual Studio 2012), 1309
 - Code Analysis, 1309-1315
 - Code Clone Detection, 1310, 1317-1319
 - Code Metrics, 1309, 1315
 - Dependency Graphs, generating, 1334-1335
 - IntelliTrace, 1310, 1328-1334
 - overview of, 1309-1310
 - Profiler, 1310, 1319-1328
- analyzing
 - code, 264
 - error messages, 208
 - method calls, 61
 - Microsoft code analysis rules, 1311
 - value types, 100
- AndAlso operator, 158-159, 570
- And operator, 158-162, 313, 570
- angles, modifying elements, 780
- animations
 - ColorAnimation, applying, 786-788
 - DoubleAnimation, applying, 783-785
 - events, 787-789
 - Silverlight applications. *See* Silverlight applications
 - UI elements, 905-908
 - Visual Basic 2012, 789-790
 - Windows Phone, 959
 - WPF, 782-790
- annotations
 - applying, 805
 - Data Annotations, Code First approach, 663-665
 - flow documents, 803
 - services, implementing, 804
- anonymous iterators, 419
- anonymous types
 - languages, 524-525
 - LINQ, 552
 - LINQ to DataSets, 623
 - LINQ to SQL, 603
- AnyCPU, 44
- Any method, 519
- APIs (application programming interfaces), 1191
 - Fluent APIs, Entity Framework (ADO.NET), 665-668
 - structures, passing, 310
 - Win32 API calls, references to, 1206
- APM (Asynchronous Programming Model), 1106-1107
- App.Current property, 924
- AppDomain class, 1146
- AppDomain.CreateDomain method, 1147
- AppDomainUnloadedException, 1147
- ApplicationBar class, Windows Phone applications (apps), 982
- Application Building Blocks, 930
- Application class, members, 720
- ApplicationDeployment class, 1257
- Application Deployment tool, 957
- Application Files (ClickOnce), 1251
- Application Files group (InstallShield),
- ApplicationIcon.png, 986

Application Information group
(InstallShield), 1233

Application.myapp file, 75-76

Application object, 719-721

Application Registry (InstallShield), 1239

applications (apps)

AJAX, 1015

animations, 782-790

application-level only settings, 492-493

ASP.NET, 851. *See also* ASP.NET

configuring security, 879-882

controls, 858-860

creating with Visual Basic 2012,
864-862

deploying, 883-884

handling events, 860-861

MSDeploy, 886-891

overview of, 851-855

publishing, 883, 884-885

state management, 861-864

web forms, 855-857

breakpoints, 50-53

ClickOnce, 1245. *See also* ClickOnce

accessing, 1257-1258

applying, 1246-1247

configuring, 1251-1255

deploying, 1247-1251

overview of, 1246

Registration-Free COM, 1258-1259

security, 1255

updating, 1252-1253

clients

ClickOnce, 1247

creating, 1022-1027

instances, 606

code, debugging, 193-206

Console, 24, 622

data services, hosting, 1016

desktop, Transparency Level 2, 1151

domains, assemblies, 1145-1147

Edit and Continue feature, 54-55

events, 509-510

exceptions, 207-208, 224

frameworks

My.Application property, 478

WPF, 508

HTTP, 8

InstallShield, 1229. *See also* InstallShield

IntelliTrace, 1330-1331

localizing, 841

.NET Framework, 841-842

Windows Forms, 842-843

WPF, 844-850

logs

managing, 199-200

writing entries to, 481-482

Metro-style. *See* Metro-style apps, 7-8, 407

multi-targeting, 18

My namespace, applying, 506-510

.NET Framework, 2. *See also* .NET
Framework

PIAs, deploying without, 86-87

project templates, 17-18

references, GAC, 1227

root namespaces, 291-292

running, 875

runtime errors, 53-54

serialization, 1035

ADO.NET, 1053-1054

customizing, 1045-1048

objects, 1036-1042

WCF, 1050-1053

XAML, 1048-1050

settings, Visual Basic 2012 projects, 81-83

Silverlight, 55, 893. *See also* Silverlight
applications

debugging XAML, 926-927

deploying, 899

drag'n'drop data-binding, 916-919

new features, 898, 928

out-of-browser applications, 923-926

overview of, 894

packages, 899

permissions, 926

security, 926

Silverlight Navigation Applications,
908-910

Visual Basic 2012, 895-897

WCF RIA Services, 911-923

Store (Windows 8), 1111

types, My Project, 34

video, playing, 798

Visual Basic 2012

- applying breakpoints/trace points, 184-187

- Autos window, 192

- Call Stack window, 188-189

- Command window, 187-188

- debugging, 179-180

- inspecting objects, 192-193

- Just My Code debugging, 182-184

- Locals window, 187

- Mixed Mode debugging, 182

- Threads window, 191-192

- tools for debugging, 180-192

- Watch windows, 189-191

Visual Studio 2012, 11. *See also* Visual Studio 2010

Visual Studio 2012 Express for Web, 932

web, deploying, 887-888

Windows Azure, 929

- creating demo projects, 933-944

- deploying, 944-952

- Management Portal, 949-952

- overview of, 929-931

- registering the Developer Portal, 931

- testing, 942-944

Windows Phone, 955

- AppBar class, 982

- customizing, 985-987

- debugging, 963-964

- executing, 984-985

- launchers, 967-720

- local data storage, 980-981

- overview of, 955-956

- pages, 963-966

- panorama controls, 974-980

- Pictures Hub, 982-984

- programming models, 958-959

- starting, 963-964

- submitting to Marketplace, 987-989

- tools, 957-958

- Visual Basic 2012, 959-985

WPF

- Application object, 719-721

- architecture, 696-697

- arranging controls with panels, 709-716

- Browser Applications, 721-724

- contra variance, 536

- controls, 725. *See also* controls

- creating, 693

- handling events, 706-709

- Logical Tree/Visual Tree, 704-705

- managing windows, 716-719

- overview of, 694-695

- Visual Studio 2012, 697-699

- XAML, 699-704

Application Shortcuts group (InstallShield), 1238-1239

Application state, 861

Application tab (My Project), 34-39

applying

- annotations, 805

- arrays, 148-155

- Async pattern, 1111

- attributes, 1181-1187

- BigInteger, 102-103

- BitmapCacheBrush, 769-771

- breakpoints, 184-187

- brush properties, 758

- CaretBrush, 765-767

- ClickOnce, 1246-1247

- Code Contracts, 1350-1355

- code editor, 24-27

- Code First approach, 657-668

- collections

- generics, 403-412

- nongeneric, 394-403

- ColorAnimation, 786-788

- COM

- objects, 1192-1196

- P/Invoke, 1200

- combinators, 1129-1131

- constants, 175-176

- cookies, 863

- CopyToDataTable method, 624-626

- CreateInstance, 154

- CSS, 856

- dates, 137-143

- DoubleAnimation, 783-785

- DrawingBrush, 768

- EDMs, 630-643

- embedded expressions, 682-685
 - enumerations, 315
 - Expression Blend, 775
 - fields, 227
 - Finalize method, 271-272
 - fundamental types, 125-155
 - GUIDs, 147-148
 - highlights, 805
 - ImageBrush, 762-765
 - inheritance, 324-327
 - initializers, objects, 558-559
 - iterations, 166-170
 - iterators, LINQ, 562-565
 - LinearGradientBrush, 760-761
 - logs, SQL, 613
 - loops, 95-172
 - multiple transforms, 782
 - My.Application property, 478-482
 - My namespace, 506-510
 - My Project, 33-39
 - Object Browser tool, 394
 - projects, Visual Studio 2010, 16-27
 - query interceptors, 1030-1033
 - RadialGradientBrush, 762
 - reference types, 103-106
 - reserved words as keywords, 72
 - SelectionBrush, 765-767
 - SolidColorBrush, 759-760
 - streams
 - memory, 466
 - strings, 467
 - strings, 125-137
 - structures, 305-308
 - tasks, parallel computing, 1072-1080
 - time, 143-144
 - trace listeners, 196-202
 - value types, 92-103
 - VisualBrush, 767-768
 - Visual Studio 2012 Express for Web, 932
 - With..End With statements, 176-177
- architecture
- collections, 394
 - CPUs, 44
 - interoperability between COM/.NET, 86
 - LINQ, 554
 - .NET Framework, 2-3
 - WPF, 696-697
- archives (Zip)
- support, 8
 - Visual Basic 2012, 472-474
- AreEqual method, 1341
- AreNotEqual method, 1341
- AreSame method, 1341
- ArgumentException, 359, 459, 614
- ArgumentNullException, 218-219, 223, 459, 615
- arguments
- command-line, retrieving, 482
 - empty strings, passing, 221
 - lambda expressions, 518
 - methods, 237-242, 351
 - nullable, 241
 - ParamArray, 239-240
 - stringToPrint, 238
- arithmetic operators, 155-157
- ArrayList collection, 394-397
- Array literals, 151, 515-516
- arrays
- applying, 148-155
 - collections, comparing, 148-149
 - copying, 152-155
 - formatting, 152-155
 - generics, 369-370
 - initializing, 150
 - inspecting, 152-155
 - iterators, 169
 - jagged, 151-152, 516
 - multidimensional, 151, 516
 - passing, 239
 - rules, 267
 - sorting, 152-155
 - support, 8
 - zero-based, 149
- As clause, 96, 373, 525
- AsDataView method, 623
- AsEnumerable method, 517-519
- AsNoTracking method, 654
- AsParallel, invoking, 1097

- ASP.NET, 6
 - Administration tool, 879
 - applications, 851
 - configuring security, 879-882
 - controls, 858-860
 - creating with Visual Basic 2012, 864-862
 - deploying, 883-884
 - handling events, 860-861
 - MSDeploy, 886-891
 - publishing, 883-885
 - state management, 861-864
 - web forms, 855-857
 - handlers in, 8
 - overview of, 851-855
 - page requests, 852
 - scalability, 852-853
 - templates, 854-855
- assemblies, 3, 1143
 - application domains, 1145-1147
 - attributes, 1145, 1184
 - Base Addresses for, 47
 - BCL (GAC), 1223
 - Caller Information, 1177-1180
 - CLS, 264
 - COM, registering for interoperability, 1197
 - GAC, 84
 - installing/uninstalling, 1223-1224
 - signing with strong names, 1224-1226
 - inspecting, 1161
 - locations, 1144
 - metadata, 1158-1160
 - MsCorlib.dll, 6
 - My.Application property, retrieving information, 478-479
 - .NET Framework, 4-5
 - overview of, 1143-1145
 - PIAs, 86-87
 - Portable Class Library, 441
 - references, 84
 - reflection, 1158. *See also* reflection
 - invoking dynamic code, 1169-1171
 - retrieving information from, 1160-1162
 - types, 1162-1169
 - sandboxed models, 1152-1154
 - security models, 1148-1155
 - serialization, generating, 48
 - sharing, 853
 - signing, 1145
 - types, 1144
 - versions, 1144
- Assembly Information dialog box, 77
- Assembly Information window (My Project), 35-36
- AssemblyInfo.vb file, 76-77
- Assembly name field (My Project), 34
- Assertion dialog box, 196
- assertions, contracts, 1355
- Assert method, 194
- Asset class, methods, 1341
- assigning
 - field inlines, 228
 - identifiers, 72
 - images to PictureBoxes, 499
 - scope, 234
 - structures to variables, 308
 - styles to buttons, 771
 - value types, 98-99
 - variables, 45
- assignment operators, 157-158
- associations, 593
- assumptions, contracts, 1355
- asynchronous calls, 1109
- asynchronous programming, 466, 1103
 - APM, 1106-1107
 - Async pattern, 1107-1112, 1120-1122.
 - See also* Async pattern
 - canceling, 1131-1134
 - exception handling, 1127
 - callbacks, 1116-1120
 - EAP, 1104-1106
 - event-based asynchrony, 1116-1120
 - input/output (I/O), 8
 - I/O file operations (.NET 4.5), 1137-1141
 - lambda expressions, 1136-1137
 - overview of, 1104
 - requests, 853
 - synchronous approach to, 1112-1116
 - task-based asynchrony, 1127-1131

Asynchronous Programming Model. See APM, 1106-1107

Async pattern, 1107-1112

- asynchronous programming, 1120-1122
- canceling, 1131-1134
- documentation, 1126
- exception handling, 1127
- progress, reporting, 1134-1136

attributes

- AllowPartiallyTrustedCallers, 1154
- applying, 1181-1184
- assemblies, 1145
- CallerMemberName, 1179
- ChangeInterceptor, 1032
- CLSCompliant, 70, 336, 1182
- code, 1181, 1184-1188
- Column, 664
- ComVisible, 1198
- connectionString, 660
- customizing, 428
- Data Annotations, 664
- DebuggerBrowsable, 202-203
- DebuggerDisplay, 202-203
- DebuggerStepperBoundary, 202-203
- DebuggerTypeProxy, 202, 205
- DebuggerVisualizer, 202
- debugging, 202-206
- DocumentProperties, 1187
- Flags, 320
- ForeignKey, 664
- getting/setting, 456
- inheritance, defining, 1187
- Just My Code, 183
- Key, 664
- MaxLength, 664
- MinLength, 664
- parameters, types, 1185
- QueryInterceptor, 1030
- reflection, 1189-1190
- Required, 650, 664
- security, 1151
- SecurityCritical, 1151
- SecurityRules, 1151
- SecuritySafeCritical, 1151
- SecurityTransparent, 1151,
- Serializable, 70, 1182

StringLength, 664

StructLayout, 310, 1203-1204

Table, 664

VBFixedString, 1205

Visual Basic 2012 projects, 69-70

WebGet, 1028

WebInvoke, 1028

XmlRoot, 1045

audio, 77

- playing, 485

Audio property, 482, 485

Authenticode certificate, 1255

auto-completing code, 26

auto-generating

- complex types/function imports, 657
- contracts, 995-996
- partial classes, 248
- XAML code for details, 827

auto-implementing properties, 229-231

Autos window, 192

availability

- Async patterns, 1111

- For..Each, 168

- of templates, 19-20

Average method, 519

avoiding

- ambiguities with local variables, 228-229
- boxing/unboxing, 115

awaiters, customizing, 1141

Await keyword, 1109

- asynchronous programming, 1120-1122

Azure. See Windows Azure

B

Background property, 717

backgrounds

- compilers, 43
- images (Windows Phone), 986

BAML (Binary Application Markup Language), 844

Barrier class, 1078-1080

Base Addresses for assemblies, 47

BaseClassDemo class, 340

- base classes
 - contra variance, 536
 - members, accessing, 337-341
- Base Class Library. *See* BCL
- BasicHttpBinding, 1005
- BasicHttpContextBinding, 1005
- BCL (Base Class Library), 2, 5-6
 - assemblies (GAC), 1223
 - documentation, 55
 - generics, customizing types, 368
 - namespaces, 284
 - reference types, 105
 - Silverlight, 897
 - viewing, 57
- BeginRead method, 464
- BeginWrite method, 464
- behavior
 - extension methods, 521
 - members, 29
- BigInteger, applying, 102-103
- Binary Application Markup Language.
 - See* BAML, 844
- binary files
 - reading, 465
 - writing, 465
- BinaryFormatter class, 1036
- binary numbers, 160-162
- binary operators, invoking AsParallel, 1097
- Binary Rewriter, 1351
- BinarySearch method, 153
- binary serialization, 1036-1038
- binding. *See also* late binding, 45, 115, 1176
 - ABC properties, 992
 - assemblies, 1144
 - data-binding. *See* data-binding
 - WCF, 1005
- Binding markup extension, 812-813
- Bing, searching, 972
- BingMapsDirectionsTask, 967
- BingMapsTask, 967-969
- BitArray collection, 401
- bit flags, enumeration values as, 320
- BitmapCacheBrush, 758, 769-771
- Bitvector32 collection, 402-403
- bitwise operators, 160-162
- Blob Storage, 930-931
- blocks
 - catch, 45
 - Class..End Class, 348
 - conditional code, 172-175
 - If..Then..Else code, 172-173
 - Interface..End Interface, 348
 - Namespace..End Namespace, 284, 288
 - nested Try..Catch..Finally, 217-218
 - Structure..End Structure, 305
 - Try..Catch..Finally, 1127
 - handling exceptions, 209
 - iterators, 418-419
- Boolean keyword, 95
- Boolean user-level settings, 491
- Boolean values, operators, 164
- Border control, 727
- BoundedCapacity property, 1091
- boxing types, converting reference/value, 113
- breakpoints
 - applications, 50-53
 - applying, 184-187
 - XAML, placing in, 926
- Breakpoints window, 184
- Browse command, 492
- Browser Applications (WPF), 721-724
- Browser for a Destination dialog box, 1238
- brushes
 - properties, applying to, 758
 - WPF, 757-771
- bubbling strategy, 708
- bug fixes, return types, 515
- build engines, 7
- building
 - applications
 - .NET Framework, 2. *See also* .NET Framework
 - Visual Studio 2012 Express for Web, 932
 - CLS-compliant structures, 314
 - code snippets, 1281
 - InstallShield packages, 1244
 - MSBuild.exe, 40
 - packages, Visual Studio 2012 extensibility, 1289-1299
 - projects, Silverlight applications, 936-942

- build processes, viewing results, 31
- built-in extension methods, 519
- BulletedList control, 859
- Button controls, 728, 859
- buttons, assigning styles, 771
- ByRef keyword, 237-242
- Byte keyword, 95
- ByVal keyword, 237-242

C

- caches
 - events, 856-857
 - GAC, 84, 1144, 1221
 - InstallShield, 1237
 - overview of, 1222-1227
 - troubleshooting DLLs, 1221-1222
- Cache state, 862-863
- CalculateDiscount method, 354
- CalculatePerimeter method, 1349
- calculating
 - arithmetic operators, 156. *See also* arithmetic operators, 155-157
 - Code Metrics, 1309, 1315
- Calendar control, 728-729, 859
- callbacks, asynchronous programming, 1116-1120
- Caller Information assemblies, 1177-1180
- CallerMemberName attribute, 1179
- calls
 - asynchronous, 1109
 - from COM objects WPF, 1196
 - methods, hierarchies, 60
 - synchronous, 1108
 - Win32 API calls, references to, 1206
- Call Stack window, 188-189
- cameras, capturing Silverlight applications, 904
- canceling
 - Async pattern, 1131-1134
 - PLINQ queries, 870
 - tasks, 1077-1078
- CancellationTokenSource class, 1077, 1134
- Cancel property, 496

- CanFilter property, 832
- CanGoBack property, 965
- CanGoForward property, 965
- CanGroup property, 832
- CannotUnloadAppDomainException, 1147
- CanRead method, 464
- CanSeek method, 464
- CanSort property, 832
- CanUserAddRows property, 818
- CanUserRemoveRows property, 818
- CanUserReorderColumns property, 818
- CanUserResizeColumns property, 818
- CanUserResizeRows property, 818
- CanUserSortColumns property, 818
- Canvas panel, 714
- CanWrite method, 464
- Capacity property, 394
- capturing cameras (Silverlight applications), 904
- CaretBrush, 758, 765-767
- CAS (Code Access Security), 1148
 - migrating old CAS-based code, 1155
- Cascade Style Sheets. *See* CSS, 856
- case sensitivity
 - Select Case statements, 174-175
 - Visual Basic 2012, 72
 - XML comments, 1213
- Cassini Web Server, 852
- Cast method, 519
- catch blocks, 45
- catching
 - events, 383-385
 - exceptions, 217
 - COM, 1195
 - without variables, 223-224
- Catch statement, 210
- Category class, 592, 595
- CBool function, 121
- CByte function, 121
- CChar function, 121
- CDate function, 121
- CDbl function, 121
- certificates
 - ClickOnce, 1255
 - X.509, 926

- ChangeConflictException, 616
- ChangeExtension method, 454
- ChangeInterceptor attribute, 1032
- change interceptors, 1032
- Change Members Format command, 424
- channels, RSS, 15
- characters
 - LINQ queries, 552
 - literal type, 98
 - underscore (_), 70, 265, 355
- Char keyword, 95
- CheckBox control, 729-730, 859
- CheckBoxList control, 859
- CheckMailAddress method, 381
- CIL (Common Intermediate Language), 5
- Clnt function, 121
- Class.End Class statements, 225, 348
- classes, 225
 - abstract, 336
 - CLS, 337
 - code, 427
 - implementing interfaces, 428
 - inheritance, 337
 - AdHost, 1194
 - AppDomain, 1146
 - Application, members, 720
 - AppBar, Windows Phone applications (apps), 982
 - ApplicationDeployment, 1257
 - Asset, methods, 1341
 - Barrier, 1078-1080
 - BaseClassDemo, 340
 - BCL, 2, 5-6
 - BinaryFormatter, 1036
 - CancellationTokenSource, 1077, 1134
 - Category, 592, 595
 - CLS, 263-267
 - CompositeType, 997
 - ConcurrentExclusiveSchedulerPair, 852
 - constructors, 252-259
 - Contacts, 358
 - contra variance, 536
 - control properties, 428
 - CultureInfo, 842
 - CustomConverter, 839
 - DataContext, 600, 619
 - DataContractSerializer, 1050
 - DataServiceContext, 1026
 - DbContext, 638-640, 645
 - DbSet, 653
 - Debug, 194-195
 - declaring, 225-227
 - diagrams
 - adding, 431
 - files, 425
 - Document, 348, 352
 - Domain Service Class, adding, 913-916
 - DownloadStringCompletedEventArgs, 1106
 - DropCreateDatabaseIfModelChanges, 665
 - fields, 227-229
 - File, 461
 - FileStream, 68
 - generating, 437
 - generics, defining, 370
 - ILGenerator, 1175
 - inheritance, preventing, 335
 - interfaces, defining, 348
 - Iterator, implementing, 419-422
 - LastNameChangedEventArgs, 388
 - libraries, 226
 - LINQ to SQL, 589-599
 - mapping, 1335
 - methods
 - executing actions with, 235-247
 - partial methods, 251-252
 - modules, comparing, 303
 - Monitor, synchronization, 1065
 - namespaces, 284
 - NavigationService, 965
 - nested, 226-227, 348
 - NestedClass, 227
 - NetworkStream, 474
 - ObjectContext, 916
 - ObservableCollectionHelper, 503
 - Page, 856
 - Parallel, 1071
 - ParallelEnumerable, 1097
 - parallelism, 1071
 - ParallelLoopState, 1085
 - ParallelOptions, 1071

- partial, 248-251
- portable, creating, 440-451
- Portable Class Libraries, 7, 440-451
- ProcessStartInfo, 1058
- properties, 229-234
- proxy, WCF, 1002
- reference types, 90, 103-106
- requirements for COM exposure, 1197
- ResourceManager, 80, 500
- ResurrectDemo, 279
- rules, 266
- scope, 234-235
- shared members, 259-263
- SoapFormatter, 1039
- SqlCommand, 541
- Stream, methods, 1138
- StreamWriter, 464
- Style, 771
- support, 301
- System.Array, 105, 152-155
- System.Attribute, 69
- System.Boolean, 95
- System.Byte, 95
- System.Char, 95
- System.Collections, 394
- System.Convert, methods, 123
- System.DateTime, 95, 139
- System.Decimal, 95
- System.Delegate, 380
- System.Diagnostics.Process, 1058
- System.Double, 95
- System.Enum, applying methods from, 316-319
- System.Exception, 105, 209, 214-216, 344
- System.GC, 270
- System.Guid, 95
- System.Int16, 95
- System.Int32, 95
- System.Int64, 95
- System.IntPtr, 95
- System.IO.Directory, 455-458
- System.IO.DirectoryInfo, 458-459
- System.IO.DriveInfo, 459
- System.IO.File, 293, 460-461
- System.IO.FileInfo, 462-463
- System.IO.Path, 454-455
- System.IO.Stream, 105, 464
- System.Math, 157
- System.Numerics.BigInteger, 95
- System.Object, 90, 105
 - inheritance, 325-329
 - methods, 92
 - naming, 91
 - WPF, 696
- System.SByte, 95
- System.SerializableAttribute, 70
- System.Single, 95
- System.String, 105, 126
- System.Threading.ThreadPool, 1062
- System.TimeSpan, 95, 142
- System.TimeZone, 95
- System.UInt16, 95
- System.UInt32, 95
- System.UInt64, 95
- System.ValueType, 91-92
- System.Windows.ContentElement, 57
- TaskFactory, 1071
- TaskScheduler, 1071
- Test, 227
- TextRange, 807
- Trace, 195
- Visual Basic 2012 projects, 64
- Visual Studio Class Designer, 424-432
- Window, 928
- XamlServices, 1048
- XDocument, members, 674
- XmlSerialization, 1043
- XmlSerializer, 48
- XmlWriterTraceListener, 196-197
- XpsDocument, 808
- Class Library, 226
- Class View window, 432-433
- clauses
 - As, 96, 373, 525
 - Into, 570
 - AddressOf, 381
 - Aggregate, 570
 - Handles, 385
 - OrderBy, 916

- Clear method, 396, 404, 410, 484
- ClickOnce, 1229
 - accessing, 1257-1258
 - applications
 - applying, 1246-1247
 - deploying, 1247-1251
 - overview of, 1246
 - certificates, 1255
 - configuring, 1251-1255
 - overview of, 1245-1247
 - Registration-Free COM, 1258-1259
 - security, 1255
 - updating, 1252-1253
- ClientBin, 899
- clients
 - applications
 - ClickOnce, 1247
 - creating, 1022-1027
 - exception handling, 220
 - formatting, 1002
 - instances, 606
 - validating, 1004
 - WCF, configuring, 1004
- Clipboard property, 482
- Clone method, 117, 352, 374
- clones, searching code, 1310, 1317-1319
- Close method, 194, 464
- CloudBerry Explorer for Windows Azure, 932
- cloud services
 - settings, 946
 - Windows Azure, 930. *See also* Windows Azure
- Cloud Storage Studio (Cerebrata), 932
- CLR (Common Language Runtime), 2-5, 1191
 - assemblies, 1144-1147
 - deferred execution, 565-568
 - exceptions, 207
 - Finalize method, 272
 - finalizers, 110
 - LINQ, 552
 - LINQ to SQL, 588
 - managed code, writing, 4
 - metadata, 1158
 - namespaces, 284
 - objects, invoking destructors, 270
 - POCO, 640
 - programming languages, 7
 - Stack, comparing reference types/value types, 106-108
 - structures, allocating memory, 309
 - value types, 92
 - WPF, 696
- CLS (Common Language Specification), 72, 263-267
 - abstract classes, 337
 - assemblies, 264
 - AssemblyInfo.vb file, 77
 - Code Analysis, 1310
 - enumerations, 320-321
 - inheritance, 327
 - interfaces, 354-355
 - namespaces, 295
 - naming conventions, 264-266
 - rules
 - arrays, 267
 - classes, 266
 - methods, 267
 - properties, 267
 - structures, 314
 - types, 264
- CLSCompliant attribute, 70, 336, 1182
- CObj function, 121
- code. *See also* languages; programming
 - abstract classes, 427
 - analysis, 264
 - attributes, 1181
 - applying, 1181-1184
 - customizing, 1184-1188
 - reflection, 1189-1190
 - auto-complete, 26
 - clones, searching, 1310, 1317-1319
 - COM
 - components, 87
 - objects, 1195
 - compiling, 5
 - components, applying in code, 87
 - conditional code blocks, 172-175
 - documentation, 1207. *See also* XML comments

- dynamic
 - creating at runtime, 1147
 - invoking, 1169-1171
- editors
 - creating projects, 24
 - extending, 1304-1307
 - placing breakpoints in, 51
- extension methods, customizing, 521-523
- files
 - Visual Basic 2012 projects, 72-83
 - Visual Studio 2012 packages, 1293
- Imports directives, positioning, 293
- iterators, benefits of, 415-417
- lambda expressions, 528
- LINQ, examples, 551-552
- LocBaml, 844-845
- managed, writing, 4
- MediaElement control, 797
- Microsoft code analysis rules, 1311
- models, Code First approach, 659-660
- MSBuild.exe, localization, 847
- MSDN Code Gallery, 1303
- old CAS-based code, migrating from, 1155
- on-the-fly, compiling, 1297
- opposed, 1175
- refactoring, 1349
- resources, accessing by name, 500
- reusing, 368
- samples, searching, 21
- snippets
 - deploying, 1300
 - reusing, 1275-1283
- SQL, database objects, 655
- System.Reflection.Emit namespace, 1171-1177
- testing, 1337
 - applying Code Contracts, 1350-1355
 - TDD, 1344-1349
 - unit tests, 1337-1344
- tools, Visual Basic 2012, 1358
- unmanaged
 - COM, 1199
 - writing, 4
- Visual Basic 2012
 - debugging, 179-180
 - debugging in, 193-206
 - XAML, 700
 - zooming, 25
- Code Access Security. *See* CAS, 1148, 1155
- Code Analysis, 1309, 1310-1315. *See also* analyzing
- codebases, assemblies, 1144
- Code Clone Detection, 1310, 1317-1319
- Code Contracts, applying, 1350-1355
- Code Coverage, enabling, 1343
- code editors, applying, 24-27
- Code First approach, 630, 657
 - Data Annotations, 663-665
 - databases, generating, 660-663
 - migrating, 665
 - WCF Data Services, 1016
- Code Gallery, 21
- Code Metrics, 1309, 1315
 - Result tool window, 1317
- CodePlex, 6, 932
- CodeRush Xpress, 1358
- Code Snippet Editor, 1358
- Code Snippet Manager, 1278
- code tag (XML comments), 1214
- coercion, 244
- CollectionChanged event, 816
- collections, 393
 - architecture, 394
 - ArrayList, 394-397
 - arrays, comparing, 148-149
 - BitArray, 401
 - Bitvector32, 402-403
 - concurrent, 413
 - customizing, 413
 - debugging, 405
 - Dictionary (Of TKey, TValue), 407
 - generics
 - applying, 403-412
 - optimizing, 395
 - serialization, 1049
 - HashTable, 398-399

- HybridDictionary, 400
- initializers, 405-406
- in-memory, querying, 558
- iterators, 169, 415. *See also* iterators
- LinkedList (Of T), 410-412
- List (Of T), 403-405
- ListDictionary, 399
- NameValueCollection, 401
- nongeneric, 394-403, 406
- ObservableCollection (Of T), 408-410, 814-818
- OrderedDictionary, 399
- parallel computing, 1087-1092
- Queue, 397-398
- Queue (Of T), 412
- ReadOnlyCollection (Of T), 406-407
- SortedDictionary (Of TKey, TValue), 408
- SortedList, 400
- Stack, 398
- Stack (Of T), 412
- StringCollection, 400
- StringDictionary, 400
- CollectionViewSource objects, 830
- Collect method, 279
- ColorAnimation, 786-788, 905
- colors
 - Microsoft Design Style, 11
 - Start Page, 12
- Color Theme drop-down box, 14
- Color Theme Editor (Visual Studio 2012), 14
- Column attribute, 664
- ColumnGap property, 803
- columns
 - adding, 710
 - design, 823
 - foreign keys, support, 633
 - tabular data forms, formatting, 819-825
- COM (Component Object Model), 1191
 - assemblies, registering for interoperability, 1197
- Automation, 926
 - components, applying in code, 87
 - exceptions, catching, 1195
 - libraries, adding references, 85-86
 - objects
 - applying, 1192-1196
 - exposing, 1197-1199
 - P/Invoke, 1200
 - converting types to unmanaged, 1202-1203
 - encapsulating, 1201-1202
 - handling exceptions, 1205-1206
 - Registration-Free (ClickOnce), 1258-1259
 - unmanaged code, 1199
 - Win32 API calls, references to, 1206
- combinators, 1129-1131
- combining delegates, 382-383
- ComboBox control, 730-732
- command-line
 - arguments, retrieving, 482
 - tools, 7
- commands. *See also* functions
 - Add, 424
 - Add Reference, 84
 - Adjust Shapes Width, 424
 - Browse, 492
 - Change Members Format, 424
 - customizing, 1268-1270
 - Export Diagram as Image, 424
 - Generate Database from Model, 657
 - Generate Other, 434
 - Go to Definition, 58
 - Group Members, 424
 - Hit Count, 185
 - Insert breakpoint, 51
 - Install ApplicationName onto This Computer, 924
 - Layout Diagram, 424
 - New Event Handler, 706
 - New Item, 30
 - Run to Cursor, 181
 - Settings, 433
 - Show Data Sources, 917
 - Step Into, 181
 - Step Out, 181
 - Step Over, 181
 - When Hit, 186
 - Zoom, 424
- Command window, 187-188

comments

- Data Annotations, Code First approach, 663-665
- resources, 80
- XML, 1207
 - generating Help files, 1220
 - implementing, 1210-1219
 - overview of, 1208-1209
- common dialogs, WPF, 754-755
- Common Intermediate Language. *See* CIL, 5
- Common Language Runtime. *See* CLR
- Common Language Specification. *See* CLS
- Common Type System, 89-93
 - objects, 90
 - reference types, 90-93
 - value types, 90-93
- Compare method, 127
- comparing
 - arrays/collections, 148-149
 - classes/modules, 303
 - reference types/value types, 106-111
 - strings, 126-128
- comparison operators, 163-165
- compatibility
 - Entity Framework (ADO.NET), 668-669
 - Windows Forms, 75
- compilers, backgrounds, 43
- compiling, 5
 - constants, 47-48
 - Help files, generating, 1220
 - multicore JIT compilation, 853
 - namespaces, 6
 - on-the-fly code, 1297
 - options, 41-44
 - Output window, 33
 - projects, 39-48
 - types, local type inference, 512
 - Visual Basic Compiler (vbc.exe), 2
- complex objects, generating, 437-439
- complex types, auto-generating, 657
- compliance, CLS, 264. *See also* CLS
- Component Object Model. *See* COM
- components
 - COM, importing, 1192-1194
 - Windows Media Player, 86-87

- CompositeType class, 997
- compressing data with streams, 467-474
- Compute Emulator, 943
- ComVisible attribute, 1198
- concatenating
 - concatenation operators, 163, 583
 - strings, 136-137
- Concat method, 519
- Conceptual Schema Definition Language. *See* CSDL, 634
- concurrency
 - concurrent collections, 413, 1087-1092
 - concurrent operations, managing, 1129
 - handling, 648-650
 - optimistic, 616
- ConcurrentBag (Of T), 1087
- ConcurrentDictionary (Of TKey, TValue), 1090-1092
- ConcurrentExclusiveSchedulerPair class, 852
- ConcurrentQueue (Of T), 1089
- ConcurrentStack (Of T), 1089
- conditional code blocks, 172-175
- conditional exception handling, 222. *See also* exceptions
- conditioning
 - attribute inheritance, 1188
 - inheritance, 334-337
- conditions
 - contracts
 - post-conditions, 1353-1354
 - preconditions, 1352-1353
 - debugging, 187
 - warnings, 45
- configuration files
 - listener settings, 200-202
 - naming, 493-495
- Configuration Manager, 42
- configuring
 - Advanced Compiler Settings window, 46
 - ClickOnce, 1251-1255
 - clients, WCF, 1004
 - Code Analysis, 1312
 - contracts, properties, 1350-1351
 - Data Source Configuration Wizard, 544
 - Debug configuration, 49

- Debug configuration (Visual Studio 2012), 40-43
- Domain Service Class, 913
- IIS servers, 994
- InstallShield, 1229. *See also* InstallShield
- item properties, 31
- MSDeploy, 889-888
- out-of-browser, enabling, 923
- project-level default Imports directives, 294
- Release configuration (Visual Studio 2012), 40-43
- roles, 881
- security, ASP.NET applications, 879-882
- settings, 490
- toolbars, 1271
- UAC, 36-39
- users, 881, 1271-1273
- View Windows Settings (My Project), 36-39
- Visual Studio 2012 projects, 1289-1299
- warnings, 44-46
- WCF Service Configuration Editor, 1009-1010
- Windows Azure subscriptions, 944
- conflicts, avoiding with different namespaces, 287
- connecting
 - ADO.NET, 541
 - databases, 541-543
 - EDMs, 632
 - LINQ to DataSets, 622
 - LINQ to SQL, 618
 - strings, writing, 619
- ConnectionSettingsTask, 967, 970
- connectionString attribute, 660
- Console
 - Application project template, 22
 - applications, 24
 - Entity Framework 5, 658
 - LINQ to DataSets, 622
 - value types, applying, 96
- ConsoleTraceListener, 196, 200
- Console.WriteLine statement, 181, 186
- constants, 175-176
 - compiling, 47-48
 - customizing, 48
- constraints
 - generics, 372-374
 - methods, 373
 - multiple, 374
 - New keyword, 373
 - types, 373
- constructors, 252-259
 - defaults, viewing, 253
 - inheritance, 341-342
 - modules, 303
 - overloading, 255-257
 - private, 257
 - types, comparing reference types/value, 110
- consuming
 - code snippets, 1276-1277
 - Data Services, 1022-1027
 - generics, 370-375
 - WCF services, 1001-1007
- Contacts class, 358
- Contains method, 396, 410, 519
- ContainsAudio method, 484
- ContainsData method, 484
- ContainsFileDropList method, 484
- ContainsImage method, 484
- ContainsText method, 484
- content, reproduction, 901
- ContentControl control, 726-727
- ContentFrame_Navigated event handlers, 910
- Content property, 726
- context, objects, 645
- Context state, 579
- continuation
 - implicit line, 552, 1182
 - LINQ, 559-561
- contracts
 - ABC properties, 992
 - auto-generating, 995-996
 - Code Contracts, applying, 1350-1355
 - customizing, 997-1001
 - events, 1355
 - invariants, 1354
 - post-conditions, 1353-1354
 - preconditions, 1352-1353
 - properties, configuring, 1350-1351
 - WCF, 993

contra variance, 536-537. *See also*
covariance, 535

controls

- adding, 870-873
- AdRotator, 859
- Ajax, 854
- ASP.NET, 858-860
- BulletedList, 859
- Button, 859
- Calendar, 859
- CheckBox, 859
- CheckBoxList, 859
- data-binding, WCF RIA Services, 916-919
- DataGrid, data-binding, 814-818
- DataList, 859
- DataPager, 919
- DetailsView, 859
- DocumentViewer, 808
- DomainDataSource, 919-921
- DropDownList, 859
- FileUpload, 859
- FlowDocumentReader, 799, 802
- GridView, 859
- HiddenField, 859
- HTML, 858-860
- HtmlAnchor, 858
- HtmlButton, 858
- HtmlForm, 858
- HtmlGenericControl, 858
- HtmlInputButton, 858
- HtmlInputCheckBox, 858
- HtmlInputFile, 858
- HtmlInputHidden, 858
- HtmlInputImage, 858
- HtmlInputRadioButton, 858
- HtmlInputText, 858
- HtmlTable, 858
- HtmlTableCell, 858
- HtmlTableRow, 858
- HtmlTextArea, 858
- HyperLink, 859
- Image, 793-795, 858-859
- ImageButton, 859
- ImageMap, 859
- Label, 859
- LinkButton, 859

- ListBox, 859
- managing, 702
- MediaElement, 795, 900
- MultiView, 859
- navigation, adding, 874
- Panel, 859
- panels, managing, 709-716
- PictureBox, 499
- PivotViewer, filtering data with, 920-923
- properties, binding, 813
- RadioButton, 859
- RadioButtonList, 859
- RangeValidator, 859
- RequiredFieldValidator, 859
- resizing, 711
- RichTextBox, 806-808
- servers, 858
- Silverlight applications, 726, 897-900
- strongly typed data, 875-862
- Substitution, 859
- Table, 859
- TextBox, 812, 857-859
- View, 859
- VirtualizingStackPanel, 714
- WindowsFormsHost, 1196
- Wizard, 859
- WPF, 725
 - Border, 727
 - Button, 728
 - Calendar, 728-729
 - CheckBox, 729-730
 - ComboBox, 730-732
 - ContentControl, 726-727
 - DataGrid, 731
 - DatePicker, 733
 - DocumentViewer, 733
 - Expander, 734
 - features, 725-726
 - Frame, 734-735
 - GroupBox, 735
 - Image, 736
 - Label, 736
 - ListBox, 736
 - ListView, 738
 - MediaElement, 739

- Menu, 740
- PasswordBox, 741
- ProgressBar, 743
- RadioButton, 744
- Rectangle, 745
- ScrollView, 746
- Separator, 746
- StatusBar, 747
- TabControl, 748
- templates, 775-778
- TextBlock, 749
- TextBox, 750
- ToolBar, 750
- TreeView, 751
- WebBrowser, 735, 753
- WindowsFormsHost, 753
- XAML, adding, 701
- Xml, 859
- conventions
 - Code First, 663
 - CType, 314
 - naming, 8
 - CLS, 264-266
 - Code Analysis, 1314
 - exceptions, 209, 345
 - identifiers, 72
 - interfaces, 355
 - value types, 94
 - Pascal, 355
- conversion operators, 120-125, 572-574
 - widening/narrowing, 120-125
- Converter property, 840
- converting
 - dates to strings, 138-139
 - between decimal and binary numbers, 160-162
 - implicit conversions, 45
 - IValueConverter interfaces, implementing
 - between reference types/value types, 111-119
 - types to unmanaged code, 1202-1203
 - values, 835-840
- ConvertTimeBySystemZoned method, 147
- cookies, applying, 863
- copying
 - arrays, 152-155
 - objects with serialization, 1038
 - strings, 131
- CopyToAsync method, 1138
- CopyToDataTable method, applying, 624-626
- CopyTo method, 396, 410
- Counter property, 343
- Count method, 307, 396, 519
- Count property, 410
- covariance, 535
- CPUs (central processing units), architecture, 44
- Create Directory for Solution check box, 18
- CreateInstance, applying, 154
- create/read/update/delete. See CRUD, 913, 1014
- cref tag (XML comments), 1214
- CRUD (create/read/update/delete), 913, 1014
- CByte function, 121
- CSDL (Conceptual Schema Definition Language), 634
- CShort function, 121
- CSng function, 121
- CSS (Cascade Style Sheets), 856
- CStr function, 121
- c tag (XML comments), 1214
- .ctor method, 216
- CType, 112, 117, 123-125
 - overloading, 312-314
- CUInt function, 121
- CULong function, 121
- CultureInfo class, 842
- Culture property, 479
- cultures, My.Application property, 479-480
- Current property, 720
- CurrentCell property, 818
- CurrentDeployment property, 1257
- CurrentItem property, 832
- CurrentPhaseNumber property, 1080
- CurrentPosition property, 832
- CurrentPrincipal method, 500
- CUShort function, 121
- CustomConverter class, 839
- Custom Event keywords combination, 389

customizing

- applications (Windows Phone), 985-987
- attributes, 428
- awaiters, implementing, 1141
- code attributes, 1184-1188
- collections, 413
- commands, 1268-1270
- compiling, 43-44
- configurations, 42-43
- constants, 48
- contracts, 997-1001
- dates, formatting, 138
- Debug configurations, 41
- debugger visualizers, 193
- events, 389-391
- exceptions, inheritance, 344-346
- extensions
 - libraries, testing, 524
 - methods, 521-523
- formats (symbols), 130
- generic types, 368
- InstallShield, 1244
- IntelliTrace, 1329
- item templates, 1264
- object serialization, 1040-1042
- operators, structures, 310-314
- out-of-browser applications, 924
- prerequisites, 1252
- properties, deploying packages, 1301
- Release configurations, 41
- RSS Feeds news channel, 14
- serialization, 1045-1048
- tasks, scheduling, 1070
- templates, 519
- text, 898
- themes, 14
- toolbars, 1268-1270
- toolboxes, 1275
- types
 - exposing, 232
 - local type inference, 513
 - selecting, 111
- validations, 614-616
- value types, 103
- Visual Studio 2012, 1267-1270, 1289-1299
- XML serialization, 1044-1045

D

- Dark theme (Visual Studio 2012), 14
- Data Annotations, Code First approach, 663-665
- Database First, 657
- databases. *See also* LINQ
 - ADO.NET, overview of, 540-543
 - Code First approach, generating, 660-663
 - connecting, 541-543
 - data sources, adding, 912-913
 - EDMs, formatting from existing, 631
 - LINQ, 550. *See also* LINQ
 - LINQ to DataSets, 622
 - LINQ to SQL, accessing, 589
 - objects, adding, 655
 - products, saving, 607
 - schemes, modifying, 664-665
 - SQL Server 2012 (Local DB), 882
- data-binding, 811
 - ASP.NET applications, creating with Visual Basic 2012, 864-862
 - controls, WCF RIA Services, 916-919
 - drag'n'drop, 818-840
 - modes, 813
 - overview of, 811-818
 - strings, formatting, 835-840
 - values, converting, 835-840
 - views, 830-835
 - Visual Basic 2012, 814
- data centers, 929
- DataContext class, 600, 619
- DataContext.Log property, 613
- DataContract, 993
- DataContractSerializer class, 1050
- data controls, adding, 870-873
- DataGrid control, 731, 814-818
- DataList control, 859
- data models, adding, 868
- data operations, executing, 660-663
- DataPager control, 919
- Data property, 214
- data providers, ADO.NET, 540-541
- DataServiceContext class, 1026

- DataServiceException, 1031
- Data Services (WCF)
 - consuming, 1022-1027
 - IIS, deploying, 1021
 - implementing, 1013, 1016-1021
 - overview of, 1013-1015
 - querying data, 1026
 - query interceptors, applying, 1030-1033
 - server-driving paging, 1033-1034
- DataSets, 539
 - data-binding to, 834
 - formatting, 543-547
 - LINQ to, 621
 - extension methods, 624-627
 - queries, 621-624
 - overview of, 543-547
- Data Source Configuration Wizard, 544
- data sources
 - adding (Silverlight applications), 912-913
 - Show Data Sources command, 917
- Data Source window, 820
- data types, 89
 - arrays, applying, 148-155
 - Common Type System, 89-93
 - conditional code blocks, 172-175
 - constants, 175-176
 - converting between reference types/value, 111-119
 - dates, applying, 137-143
 - extension methods, 517-524
 - fundamental types, 125-155
 - GUIDs, applying, 147-148
 - iterations, 166-170
 - loops, 95-172
 - objects, 90
 - operators, 155-165
 - arithmetic, 155-157
 - assignment, 157-158
 - bitwise, 160-162
 - comparison, 163-165
 - concatenation, 163
 - logical, 158-159
 - shift, 162-163
 - short-circuiting, 159-160
 - reference types, 90-93
 - applying, 103-106
 - primitive, 105-106
 - time, applying, 143-144
 - TimeZone, 144-147
 - TimeZoneInfo, 144-147
 - value types, 90-93
 - applying, 92-103
 - conversion operators, 120-125
 - customizing, 103
 - differences between reference types/value types, 106-111
 - methods, 100-101
 - .NET Framework, 93-94
 - optimizing, 101
 - With..End With statements, 176-177
- data validations, customizing, 614-616
- DatePicker control, 733
- dates
 - applying, 137-143
 - formatting, 139-142
 - strings, converting to, 138-139
 - subtracting, 142-143
- DbContext class, 638-640
 - instantiating, 645
- dbentityvalidationexception, 651
- DbSet class, 653
- DbUpdateConcurrencyException, 649
- Deactivated event, 509
- deallocating memory, 110. *See also* allocating
- Debug class, 194-195
- Debug configuration, 40-43, 49
- DebuggerBrowsable attribute, 202-203
- DebuggerDisplay attribute, 202-203
- DebuggerHidden attribute, 183
- DebuggerNonUserCode attribute, 183
- DebuggerStepperBoundary attribute, 202-203
- DebuggerStepThrough attribute, 183
- DebuggerTypeProxy attribute, 202, 205
- DebuggerVisualizer attribute, 202
- debugging
 - applications
 - breakpoints, 50-53
 - Visual Basic 2012, 50

- attributes, 202-206
- collections, 405
- debug information, generating, 47
- Edit and Continue feature, 54-55
- IntelliTrace, 1310, 1328-1334
- Output window, 33
- processes, 180-182
- runtime errors, 53-54
- tasks, 1086
- Visual Basic 2012, 179
 - applying breakpoints/trace points, 184-187
 - Autos window, 192
 - Call Stack window, 188-189
 - in code, 193-206
 - Command window, 187-188
 - inspecting objects, 192-193
 - Just My Code debugging, 182-184
 - Locals window, 187
 - Mixed Mode debugging, 182
 - preparing examples for, 179-180
 - Threads window, 191-192
 - tools, 180-192
 - Watch windows, 189-191
- Visual Studio 2010, 48-55
- Windows Phone applications, 963-964
- XAML (Silverlight applications), 926-927
- Debug menu (Visual Studio 2012), 181
- Decimal keyword, 95
- decimal numbers, converting, 160-162
- declarative mode, XAML, 703
- declaring
 - anonymous types, 524
 - classes, 225-227
 - constants, 176
 - controls, 726
 - delegates, 380-382
 - events, 386
 - instances, LINQ to SQL, 600
 - interfaces, media players, 796
 - objects, WithEvents keyword, 384-385
 - properties, 64
 - reference types, 110
 - structures, 65, 306
 - variables, storing value types, 96
- decompressing streams, 468
- deep copy
 - serialization, 1038
 - types, converting between reference/value, 115-118
- Default.aspx pages, replacing, 950
- default constructors, viewing, 253
- DefaultIfEmpty method, 519
- Default keyword, 233
- Default mode, 813
- default properties, 233-234
- DefaultTraceListener, 196
- default types, selecting, 493
- default values
 - for optional arguments, 241
 - thread pools, 1063
- deferred execution, LINQ, 565-568
- defining
 - complex code documentation, 1212-1213
 - default properties, 233
 - generics, 370
 - inheritance, 1187
 - interfaces, 347-348
 - media players, 796
 - multiple modules, 302
 - resources, 77
 - settings, 492
 - Settings property, 83
 - styles, 771
 - views
 - models, 446
 - Silverlight applications, 449
 - WPF, 448
- DeflateStream object, 467
- Delegate keyword, 380
- delegates, 379
 - combining, 382-383
 - declaring, 380-382
 - generating, 437
 - generics, defining, 370
 - namespaces, 284
 - overview of, 379-383
 - ParameterizedThreadStart, 1061
 - relaxed, 526

- Delete method, 457
- DeleteObject method, 1025
- delete operations
 - databases, 542
 - DataSets, 547
 - Entity Framework (ADO.NET), 647-648
 - LINQ to SQL, 609
- deleting
 - directories, 457
 - integer overflow checks, 47
 - toolbars, 1270
- DelimitedListTraceListener, 196-197
- demo projects, creating (Windows Azure), 933-944
- Dependency Graphs, generating, 1334-1335
- deploying
 - 1-Click deployment, 884
 - add-ins, 1300
 - applications
 - Windows Azure, 944-952
 - without PIAs, 86-87
 - ASP.NET applications, 883-884
 - ClickOnce, 1229
 - accessing, 1257-1258
 - applications, 1247-1251
 - configuring, 1251-1255
 - overview of, 1245-1247
 - Registration-Free COM, 1258-1259
 - security, 1255
 - updating, 1252-1253
 - code snippets, 1300
 - InstallShield. *See also* InstallShield
 - overview, 1231
 - packages, 1244
 - MSDeploy, 884-891
 - My.Application property, 480-482
 - portable libraries, 451
 - Silverlight applications, 899
 - Visual Studio 2012 extensibility, 1299-1302
 - WCF Data Services to IIS, 1021
 - web applications, 887-888
 - Windows Azure, viewing status, 947
 - Windows Installer, 1230
 - XBAP, 724
 - XCopy deployment, 1144, 1222
- Deployment Label (Windows Azure), 947
- Deploy Without PIAs feature, enabling, 88
- Dequeue method, 398
- deriving
 - classes, Visual Studio Class Designer, 429-431
 - members, overriding, 334
- Descendants method, 675
- deserialization
 - binary files, 1036
 - events, 1047
- design
 - ClickOnce, 1251-1255
 - Code Analysis, 1312
 - columns, 823
 - DataSets, 544
 - EDMs, 640
 - Installation Requirements (InstallShield), 1234
 - LINQ to SQL, 590
 - Microsoft Design Style, 11-12
 - Resources Designer, 497
 - settings, 490
 - templates, customizing, 519
 - tools
 - adding columns, 710
 - XAML, 699
 - Visual Studio Class Designer, 424-432
 - XML schemas, 689
- desktop applications (apps), 8
 - Transparency Level 2, 1151
- destructors
 - Finalize method, 271-272
 - implementing, 275
 - invoking, 270
- DetailsView control, 859
- Developer Center (Windows Phone),
 - accessing, 987
- Developer Portal (Windows Azure),
 - registering, 931
- development
 - TDD, 1344-1349
 - Windows 8 (WPF), 695
 - Windows Phone tools, 957-958
- Device Emulator, 957

diagnostics (Visual Basic 2012 tools), 1359.

See *also* troubleshooting

diagrams

classes

adding, 431

files, 425

exporting, 432

formatting, 431

dialog boxes

Add New Item, 589, 1267

Add Service Reference, 1002

Advanced Compile Options, 44

Assembly Information, 77

Assertion, 196

Browser for a Destination, 1238

common dialogs (WPF), 754-755

Export Diagram as Image, 424, 432

Extension and Updates, 1302

External Tools, 1268

Generate New Type, 438, 1346

Infer XML Schema Set from XML, 687

Manage NuGet Packages, 658

New Interface, 425

New Project, 21, 854, 1263

Options, 513

Prerequisites, 1252

Publish Web, 887

Reference Manager, 84-85, 936

Visual Studio Output Selector, 1236

Dictionary (Of TKey, TValue) collection, 407

Dim keyword, 308

DirectCast, 123-125

directives

Imports, 17

adding, 286

multithreading, 1060

namespaces, 292-295

System.Globalization, 843

Visual Basic 2012 projects, 68-69

Option Infer, 513

Region (Visual Basic 2012 projects), 69

directories

deleting, 457

exceptions, handling, 459

modifying, 453-459

System.IO.Directory class, 455-458

System.IO.DirectoryInfo class, 458-459

DirectoryNotFoundException, 459

direct routing strategy, 708

DirectX (WPF), 696

disabling

Just My Code, 183, 1078

warnings, 44

disconnected modes (ADO.NET), 541

Dispatcher property, 720

DispatcherUnhandledException, 509

Dispose interface, implementing, 273-278

Dispose method, 271, 275-276, 347, 404

Distinct method, 519

Divide method, 533

division (/) operator, 155, 313

DLLs (dynamic link libraries), 2

Base Addresses for assemblies, 47

Interop.AssemblyName.dll, 86

Interop.WMPLib.dll, 87

Microsoft Core Library (MsCorlib.dll), 6

System.ServiceModel.dll, 6

troubleshooting, 1221-1222

WMPLib.dll, 86

DoCancel method, 873

docking tool windows, 27

DockPanel panel, 714

documentation

Async pattern, 1126

code, 1207. See *also* XML comments

external files, 1217-1218

.NET Framework, 55-59

permission requirements, 1219

Visual Basic 2012, 55-59

Document class, 348, 352

Document Outline window, 704

DocumentProperties attribute, 1187

documents

flow

implementing, 799

viewing, 803

- modifying, 799-808
- RichTextBox control, 806-808
- spell check, implementing, 808
- WPF, 793
- XML, formatting, 672
- XPS, viewing, 808-809
- Document Type Definition. *See* DTD, 672
- DocumentViewer control, 733, 808
- Do..Loops, 170-171
- DomainDataSource control, 919-921
- domains
 - applications, assemblies, 1145-1147
 - Domain Service Class, adding, 913-916
- DoSomething method, 229
- DoubleAnimation, 783-785, 905
- Double keyword, 95
- DownloadFile method, 487
- downloading
 - InstallShield, 1231
 - IronPython, 6
 - subscriptions (Windows Azure), 945
 - templates, 19
 - Visual Studio 2012 tools (Windows Azure), 931-932
- DownloadStringCompletedEventArgs class, 1106
- DownloadString method, 1115
- drag'n'drop data-binding, 818-840
 - Silverlight applications, 916-919
- DrawingBrush, 758, 768
- drives
 - System.IO.DriveInfo class, 459
 - Windows Azure, 931
- DropCreateDatabaseIfModelChanges class, 665
- DropDownList control, 859
- DTD (Document Type Definition), 672
- duplicate catch blocks, 45
- dynamic code
 - reflection, running, 1169-1171
 - runtime, creating at, 1147
- dynamic link libraries. *See* DLLs
- DynamicResource, 772

E

- EAP (Event-based Asynchronous Pattern), 1104-1106
- early binding, 115
- Edit and Continue feature, 54-55
- Edit Breakpoints Labels window, 184
- editing
 - breakpoints, 184
 - code, placing breakpoints in, 51
 - code editors
 - applying, 24-27
 - extending, 1304-1307
 - Color Theme Editor (Visual Studio 2012), 14
 - localization, 848
 - root namespaces, 292
 - spell check, implementing, 808
 - strings, 106, 133-136
 - Visual Studio 2012 built-in image editors, 498
 - WCF Service Configuration Editor, 1009-1010
 - XAML, 700. *See also* XAML
- EDMs (Entity Data Models), 630
 - adding, 868
 - applying, 630-643
 - LINQ to SQL, exposing, 1001
 - mapping, 638-640, 1335
 - Navigation Properties, 640
 - Open Data Protocol (OData), 1015
 - queries with LINQ to Entities, 652-653
 - serialization, 1054
 - Silverlight applications, 912
 - viewing, 820
- efficiency, optimizing structures, 310
- ElementAt method, 519
- ElementAtOrDefault method, 519
- Element method, 674
- ElementName property, 812
- elements
 - angles, modifying, 780
 - Ellipse, 733
 - moving, 781
 - operators, 583-584

- resizing, 780
- rotating, 780
- UI
 - adding to Silverlight applications, 897
 - animating, 905-908
 - Logical Tree, 704
 - XML comments, 1215-1217
- elevated privileges (Visual Studio 2012), 933
- Ellipse element, 733
- EmailComposeTask, 967, 971
- embedding expressions, applying, 682-685
- Embed Interop Types property, 87
- empty strings
 - checking for, 128
 - passing, 221
- Empty Web Application template (ASP.NET), 855
- Enable Application Framework group (My Project), 39
- enabling
 - Code Coverage, 1343
 - Deploy Without PIAs feature, 88
 - InstallShield, 1231
 - Just My Code debugging, 183
 - media players, content reproduction, 901
 - optimization, 47
 - out-of-browser settings, 923
 - spell check, 808
 - tracing, 1010
 - View All Files, 638
 - Visual Studio Class Designer, 424-425
 - XML comments, 1209
- encapsulating P/Invoke, 1201-1202
- endpoints (WCF services), 992
- EndRead method, 464
- EndWrite method, 1107
- entities
 - LINQ to SQL
 - delete operations, 609
 - insert operations, 605-608
 - update operations, 608-607
 - serialization, 1054
 - SQL, applying against, 617
- EntityClient, 540
- Entity Data Models. See EDMs
- Entity Data Model Wizard, 630
- Entity Framework (ADO.NET), 629
 - Code First approach, 657
 - compatibility, 668-669
 - delete operations, 647-648
 - downloading additions to, 658
 - EDMs, 630-643
 - LINQ to Entities queries, 652-653
 - SQL queries, 653-654
 - Fluent APIs, 665-668
 - insert operations, 646-647
 - instantiating, 645
 - modifying, 645-652
 - optimistic concurrency, handling, 648-650
 - overview of, 629-630
 - stored procedures, mapping, 654-657
 - update operations, 648
 - validating data, 650-652
- enumerations, 305, 315-321
 - applying, 315
 - CLS, 320-321
 - generating, 437
 - namespaces, 284
 - RefreshMode members, 616
 - return methods, applying as from methods, 319-320
 - System.Enum class, applying methods from, 316-319
 - values as bit flags, 320
- environments
 - instrumentation, debugging, 180-192
 - My.Application property, 480-482
 - variables, retrieving, 481
- Environment tab, 14
- Environment Variables element (InstallShield), 1242
- equality (=) operator, 126, 164, 313, 570
- Equals method, 309, 314, 328
- ErrorCode property, 1205
- ErrorDialog property, 1058
- Error Lists, 24
 - projects, compiling, 40
 - windows, navigating, 30-31

errors

- Code Analysis, 1314
- correction options, 112
- FormatException, 100
- handling, 207
- messages, 30
- Microsoft code analysis rules, 1313
- runtime, debugging, 53-54
- type conversion, 103
- warnings, treating as, 44

evaluating expressions, 190

event, Navigated, 965

Event-based Asynchronous Pattern. *See* EAP, 1104-1106

event-based asynchrony, asynchronous programming, 1116-1120

Event keyword, 386

EventLogTraceListener, 196

events, 379

- Activated, 509
- animations, 787-789
- applications, 509-510
- caches, 856-857
- CollectionChanged, 816
- contracts, 1355
- customizing, 389-391
- Deactivated, 509
- deserialization, 1047
- DispatcherUnhandledException, 509
- Exit, 510
- FragmentNavigation, 510
- garbage collection, registering, 281
- handling, 383-385
 - ASP.NET, 860-861
 - ContentFrame_Navigated event handlers, 910
- implementing, 385-389
- Init, 856
- IntelliTrace, 1331-1333
- Load, 856
- LoadCompleted, 510
- MediaFailed, 798
- My.Settings property, 495-496
- Navigated, 510
- Navigating, 510, 965
- NavigationFailed, 510, 965

NavigationProcess, 510

NavigationStopped, 510, 965

NetworkAvailabilityChanged, 509

NonSerialized, 1042

pages, 856

passing, 387-389

postback, 856-857

PreRender, 856

ProgressChanged, 1106, 1134

PropertyChanged, 387, 495

registering, 383-384

routing, 707-708

serialization, 1047-1048

SessionEnding, 510

SettingChanging, 495

SettingsLoaded, 495

SettingsSaving, 495

ShutDown, 509

Silverlight applications, handling, 899

Startup, 509-510

StartupNextInstance, 509

TextChanged, 857

UnhandledException, 509

Windows Event Log, 199

WPF

- applications, 706-709

- handling, 694

EventSchemaTraceListener, 196

Event viewer, 199

examples

- of LINQ, 551-552

- Visual Basic 2012, debugging, 179-180

example tag (XML comments), 1214

Excel (Microsoft), exporting code metrics to, 1317

exceptions

- AggregateException, 1076

- AppDomainUnloadedException, 1147

- ArgumentException, 359, 459, 614

- ArgumentNullException, 218-219, 223, 459, 615

- CannotUnloadAppDomainException, 1147

- ChangeConflictException, 616

- COM, catching, 1195

- DataServiceException, 1031

- DbEntityValidationException, 651
- DbUpdateConcurrencyException, 649
- directories, handling, 459
- DirectoryNotFoundException, 459
- DispatcherUnhandledException, 509
- FaultException, 1007-1008
- FileNotFound, 160, 209, 219
- FormatException, 100, 209
- handling, 207-224
 - Async pattern, 1127
 - files, 463
 - parallel computing, 1075-1077
 - P/Invoke, 1205-1206
- hierarchies, 212-213
- IndexOutOfRangeException, 153, 209, 626
- inheritance, customizing, 344-346
- IntelliTrace, 1331-1333
- InvalidCastException, 123-125, 373, 626
- InvalidOperationException, 252, 584, 612, 1091
- IOException, 455, 459
- MethodAccessException, 1152
- MissingLastNameException, 345-346
- naming conventions, 345
- NotImplementedException, 436
- NullReferenceException, 626
- OperationCanceledException, 1078, 1134
- overview of, 207-208
- pathnames, handling, 459
- PathTooLongException, 459
- PLINQ, handling, 1100-1101
- rethrowing, 219
- SecurityException, 463
- SEHException, 1205
- serialization, handling, 1037
- SerializationException, 1037
- SqlClient.SqlException, 606
- System.Exception class, 209
- System.Net.Exception, 488
- System.Security.SecurityException, 1150
- task-specific, 220
- throwing, 218-220
- Try..Catch..Finally blocks, 209
- UnauthorizedAccessException, 459
- UnhandledException, 509

- variables, catching without, 223-224
- Visual Basic 6 migration, 209
- WCF, handling, 1007-1008
- exception tag (XML comments), 1214
- Except method, 519
- executable (.EXE) files, profiling, 1327
- ExecuteAssembly method, 1146
- ExecuteCommand method, 617
- Execute (Of T) method, 1026
- ExecuteReader method, 543
- executing
 - actions with methods, 235-247
 - applications
 - debugging, 49
 - managing, 4
 - Windows Phone, 984-985
 - assemblies, 1145-1147
 - data operations, 660-663
 - debugging, 180-182
 - deferred execution (LINQ), 565-568
 - processes, 5
- existing settings, importing, 1273
- existing toolbars, customizing, 1268
- Exit event, 510
- exiting
 - iterators, 418
 - methods, 246-247
- Exit Try statements, 218
- Expander control, 734
- explicit bounds, arrays, 150
- exponentiation (^) operator, 155
- Export Diagram as Image dialog box, 424, 432
- exporting
 - code metrics to Excel, 1317
 - diagrams (Visual Studio Class Designer), 432
 - extension methods, 523-524
 - images, 432
 - services, metadata, 999
 - templates, 1261-1265
- Export Settings Wizard, 1271-1273
- exposing
 - COM objects, 1197-1199
 - custom types, 232
 - EDMs, 1001

- generics in WCF, 1001
- LINQ to SQL, 1001
- serializable objects from WCF services, 1050
- shared members, 259
- Expression Blend
 - applying, 775
 - for Windows Phone, 957
- Expression combo box, 191
- expressions, 89
 - embedding, applying, 682-685
 - evaluating, 190
 - lambda, 526-533
 - arguments, 518
 - asynchronous programming, 1136-1137
 - creating threads with, 1060
 - lexical closures, 532-533
 - LINQ, 552
 - local type inference, 529
 - multiline, 530-531
 - Sub, 531-532
 - ternary If operators, 533-534
 - queries, 551
 - regular, 251-252, 999
- extending
 - data types, 90, 517-524
 - LINQ, 554
 - My.Application property, 504-505
 - My.Computer property, 505-506
 - My namespace, 502-506
 - My.Resources namespace, 506
 - My.Settings property, 506
- extensibility. *See also* extensions
 - Visual Studio 2012, 1287
 - building packages, 1289-1299
 - deploying, 1299-1302
 - extending code editors, 1304-1307
 - managing, 1302-1303
 - optimizing add-ins, 1304
 - overview, 1287-1289
- XML. *See* XML
- eXtensible Application Markup Language.
 - See* XAML
- Extensible Hypertext Markup Language.
 - See* XHTML, 855, 860

- Extensible Markup Language. *See* XML
- Extension and Updates tool, 1302
- extension methods, 125, 517-524
 - behavior, 521
 - built-in, 519
 - customizing, 521-523
 - Data Services (WCF), 1025
 - exporting, 523-524
 - LINQ, 552
 - LINQ to DataSets, 624-627
 - members, 403
 - overloading, 523
- extensions
 - Binding markup, 812-813
 - FrontPage, 883
 - installing, 1301
- external documentation files, 1217-1218
- external executable (.EXE) files, profiling, 1327
- External Tools dialog box, 1268

F

- Fail method, 194-195, 1341
- FaultException, 1007-1008
- feeds, RSS, 14-15
- Fiddler, 1359
- Field (Of T) method, 626-627
- fields
 - classes, 227-229
 - generating, 437
- File class, 461
- File.Delete method, 197
- FileNotFoundException, 160, 209, 219
- files
 - Application Files (ClickOnce), 1251
 - Application Files group (InstallShield),
 - Application.myapp, 75-76
 - assemblies. *See* assemblies
 - AssemblyInfo.vb, 76-77
 - audio, playing, 485
 - BAML, 844

- binary
 - reading, 465
 - writing, 465
- classes, diagrams, 425
- code
 - Visual Basic 2012 projects, 72-83
 - Visual Studio 2012 packages, 1293
- configuration files
 - listener settings, 200-202
 - naming, 493-495
- CSS, 856
- exceptions, handling, 463
- executable (.EXE), profiling, 1327
- external documentation, 1217-1218
- Help, generating, 1220
- images, exporting, 432
- I/O file operations, asynchronous programming, 1137-1141
- modifying, 453, 460-463
- overwriting, 197-199
- permissions, 463
- Resources.resx (Visual Basic 2012 projects), 77-81
- searching, 3
- Settings.settings, 83
- Solution Explorer windows, 28-30
- System.IO.File class, 460-461
- System.IO.FileInfo class, 462-463
- text
 - reading, 464-465
 - writing, 464-465
- .Vbproj file (project files), 16
- web.config (WCF), 995
- XAP, 988
- XML, 16, 44
- FileStream class, 68
- FileSystem property, 482-484
- file systems
 - accessing, 926
 - ClickOnce, publishing to, 1247
 - My.Computer property, 483-484
- File Transfer Protocol. *See* FTP, 884
- FileUpload control, 859
- filtering
 - adding, 873-874
 - data with PivotViewer control, 920-923
 - IntelliTrace, 1331-1333
- Finalize method, 271-272, 276, 328
- finalizers, comparing reference types/value, 110
- Finally block, unlocking resources, 211
- FindAll method, 405
- FindResource method, 720
- Firefox, 852
- Fiddler, 1359
- firewalls, deploying web applications, 885
- First method, 519, 1025
- FirstNode property, 674
- FirstOrDefault method, 519
- First property, 410
- flags, enumeration values as bit, 320
- Flags attribute, 320
- floating-point numbers, 156
- floating windows, creating projects, 24
- FlowDocument objects, 799
- FlowDocumentReader control, 799, 802
- flow documents
 - implementing, 799
 - viewing, 803
- Fluent APIs, Entity Framework (ADO.NET), 665-668
- FlushAsync method, 1138
- Flush method, 194
- folders. *See also* files
 - formatting, 793
 - permissions, configuring, 881
- For Each loops, 150, 168-169, 358, 1059
- ForeignKey attribute, 664
- foreign keys, support, 633
- FormatException, 100, 209
- formatting
 - applications
 - clients, 1022-1027
 - domains, 1145-1147
 - WPF, 693. *See also* WPF
 - arrays, 152-155
 - classes, portable, 440-451
 - clients, 1002

- CSS, 856
- DataSets, 543-547
- dates, 139-142
- EDMs from existing databases, 631
- folders, 793
- generics, 368-375
- hit counts, 185
- master-details forms, 826-830
- MSDeploy, 889-888
- multicast delegates, 382
- multiple diagrams (Visual Studio Class Designer), 431
- passwords, 1226
- projects
 - InstallShield, 1232-1242
 - TDD, 1344-1345
 - Visual Basic 2012, 16-18
 - Visual Studio 2010, 16-27
- shortcuts (InstallShield), 1238
- Silverlight applications with Visual Basic 2012, 895-897
- strings, 128-129, 835-840
- symbols, 129
- tabular data forms, 819-825
- tasks, 1073-1075
- text, aligning, 802
- threads, 1060-1061
- toolbars, customizing, 1269-1270
- unit tests, 1338-1349
- Visual Basic 2012 (ASP.NET applications), 864-862
- Windows Phone applications (Visual Basic 2012), 959-985
- XML
 - documents, 672-674
 - schemas, 689
- forms
 - adding, 868-869
 - ASP.NET applications, 855-857
 - master-details, formatting, 826-830
 - tabular data, formatting, 819-825
 - Windows Forms
 - adding, 1193
 - application frameworks, 507
 - compatibility, 75
 - localizing applications, 842-843
- formulas, 157. *See also* arithmetic operators, 155-157
- For..Next loops, 166-168
- Forth, 7
- Fortran, 7
- FragmentNavigation event, 510
- Frame control, 734-735
- frameworks
 - ADO.NET Entity Framework, 629. *See also* ADO.NET
 - applications
 - My.Application property, 478
 - Windows Forms, 507
 - WPF, 508
 - Managed Extensibility Framework, 1288
 - navigation (Windows Phone applications), 963-966
 - .NET Framework. *See* .NET Framework
 - Silverlight Navigation Applications, 908-910
 - WCF
 - Data Services, 1013
 - RIA Services, 911-923
 - XNA (Windows Phone), 959
- Friend qualifiers, 308
- inheritance, 327
- interfaces, 348
- modifier, 225, 234
- From keyword, 553
- FrontPage, 883
- FTP (File Transfer Protocol), publishing web applications, 884
- FullName method, 325, 333, 435
- Function keyword, 527
- functions
 - CBool, 121
 - CByte, 121
 - CChar, 121
 - CDate, 121
 - CDbl, 121
 - CInt, 121
 - CLng, 121
 - CObj, 121
 - CShort, 121
 - CSng, 121
 - CStr, 121

- CUInt, 121
- CULong, 121
- CUShort, 121
- importing, auto-generating, 657
- return values, 45
- fundamental types
 - applying, 125-155
 - arrays, 148-155
 - dates, 137-143
 - GUIDs, 147-148
 - strings, 125-137
 - time, 143-144
 - TimeZone, 144-147
 - TimeZoneInfo, 144-147

G

- GAC (Global Assembly Cache), 84, 1144, 1221
 - assemblies
 - installing/uninstalling, 1223-1224
 - signing with strong names, 1224-1226
 - DLLs, troubleshooting, 1221-1222
 - InstallShield, 1237
 - overview of, 1222-1227
- Garbage Collection, 8, 106
 - advanced, 279-281
 - ASP.NET, 853
 - events, registering, 281
 - Finalize method, 272
 - finalizers, 110
 - generations, 280-281
 - interacting with, 279-280
 - objects, managing, 270-271
 - operation modes, 280-281
 - structures, allocating memory, 309
- GC.ReRegisterForFinalize method, 279
- GC.SuppressFinalize method, 278
- Generate Database from Model command, 657
- Generate from Usage feature, 433-439
- Generate New Type dialog box, 1346
- Generate New Type window, 438
- Generate Other command, 434
- generating
 - code, System.Reflection.Emit namespace, 1171-1177
 - complex objects, 437-439
 - databases, Code First approach, 660-663
 - debug information, 47
 - Dependency Graphs, 1334-1335
 - Help files, 1220
 - methods, stubs, 436
 - on-the-fly objects, Generate from Usage feature, 433-439
 - property stubs, 1347
 - serialization assemblies, 48
 - shared members, 436
 - types, 437
 - XML files, 44
- generations
 - garbage collection, 280-281
 - operators, 574-575
- generics, 244, 367
 - collections, 394
 - applying, 403-412
 - optimizing, 395
 - serialization, 1049
 - constraints, 372-374
 - delegates, 380
 - formatting, 368-375
 - IComparable(Of T) interface, 359
 - methods, implementing, 371-372
 - nullable types. *See* nullable types
 - overview of, 367-368
 - types, 8, 368
 - variances, 535-537
 - WCF, exposing in, 1001
 - XML comments, 1219
- geospatial data in WCF Data Services, 1015
- GetAssembly method, 1160
- GetAudioStream method, 484
- GetCallingAssembly method, 1160
- GetCreationTime method, 455
- GetCurrentDirectory method, 458
- GetCustomerQuery method, 825
- GetCustomers method, 502
- GetData method, 484

- GetDataObject method, 484
- GetEntryAssembly method, 1160
- GetEnumerator method, 420
- GetExecutingAssembly method, 1160
- GetExportedTypes method, 1162
- GetExtension method, 454
- GetFileDropDownList method, 484
- GetFiles method, 456
- GetHashCode method, 309, 314, 328
- GetILGenerator method, 1175
- GetImage method, 484
- GetLogicalDrives method, 457
- GetLowerBound method, 153
- GetModules method, 1162
- GetName method, 317
- GetNames method, 316-317
- GetObjectData method, 1046
- GetOrders method, 916
- GetRange method, 396
- Get Started tab (Visual Studio 2010), 14
- GetSystemTimeZones method, 145
- GetText method, 484
- getting/setting attributes, 456
- GetType method, 119, 328
- GetTypes method, 1163
- GetUpperBound method, 153
- GetValidationErrors method, 651
- GetValue method, 155
- GetValues method, 316-317
- Global Assembly Cache. See GAC
- Global keyword, namespaces, 295-299
- GoBack method, 965
- GoForward method, 965
- Go to Definition command, 58
- graphics
 - 3-D (Silverlight applications), 908
 - animations, 782-790
- graphs, Dependency Graphs, 1334-1335
- greater than (>) operator, 164, 313
- greater than or equal to (>=) operator, 164, 313
- Grid panel, 709-711, 1261
- GridView control, 859
- GroupBox control, 735

- GroupBy method, 519
- GroupJoin method, 519
- Group Members command, 424
- groups (InstallShield)
 - Application Files,
 - Application Information, 1233
 - Application Shortcuts, 1238-1239
 - Installation Interview, 1241
 - Installation Requirements, 1234-1236
 - operators, 577-579
- Guid.NewGuid method, 147
- GUIDs (globally unique identifiers), 147-148
- GZip identifiers, 68
- GZipStream objects, 467

H

- handlers in ASP.NET, 8
- Handles clause, 385
- handling
 - errors, 207
 - events, 383-385
 - animations, 787-789
 - ASP.NET, 860-861
 - ContentFrame_Navigated event handlers, 910
 - Silverlight applications, 899
 - WPF, 694, 706-709
- exceptions, 207-224
 - Async pattern, 1127
 - directories, 459
 - files, 463
 - parallel computing, 1075-1077
 - pathnames, 459
 - P/Invoke, 1205-1206
 - PLINQ, 1100-1101
 - Try..Catch..Finally blocks, 209
 - WCF, 1007-1008
- optimistic concurrency, 616, 648-650
- serialization exceptions, 1037
- HasExtension method, 454
- HashTable collection, 398-399

Heap

- boxing, 114
- reference types, 90, 108

Help

- generating, 1220
- navigating, 56
- properties, 214

HiddenField control, 859

hierarchies

- exceptions, 212-213
- Logical Tree, 704
- method calls, 60
- shortcuts, 1238
- of types, 90

highlights, applying, 805

historical debugging (with IntelliTrace), 1310, 1328-1334

Hit Count command, 185

hit counts, formatting, 185

hosting

- data services, 1016
- WCF in IIS, 1008-1009

HTML (Hypertext Markup Language), 8, 852

- controls, 858-860
- Silverlight applications, 894
- support, 8

HtmlAnchor control, 858

HtmlButton control, 858

HtmlForm control, 858

HtmlGenericControl control, 858

HtmlInputButton control, 858

HtmlInputCheckBox control, 858

HtmlInputFile control, 858

HtmlInputHidden control, 858

HtmlInputImage control, 858

HtmlInputRadioButton control, 858

HtmlInputText control, 858

HtmlTableCell control, 858

HtmlTable control, 858

HtmlTableRow control, 858

HtmlTextArea control, 858

HTTP (Hypertext Transfer Protocol), 991

- applications, 8
- GET requests, 1030
- requests, querying data via, 1014-1015

HybridDictionary collection, 400

HyperLink control, 859

Hypertext Markup Language. *See* HTMLHypertext Transfer Protocol. *See* HTTP

I

IAsyncResult.AsyncState property, 1107

IBookService contract, 998

ICloneable interface, 115, 118, 355

ICollection interface, 394

ICommand interface, 908

IComparer interface, 359-360

IComparable interface, 109, 355, 358-359, 373

icons, 77

ApplicationIcon.png, 986

My Project, 34

IConvertible interface, 355, 360-363

IDataErrorInfo interface, 616

IDEs (Integrated Development Environments), 11

advanced features, 1261

code snippets, reusing, 1275-1283

events, generating handlers, 707

NuGet, managing libraries, 1283-1286

Silverlight applications, 895

tabs, navigating, 50

templates, exporting, 1261-1265

tool windows, docking, 27

user settings, managing, 1271-1273

Visual Studio 2012, 11, 1267-1270.
See also Visual Studio 2010

WPF projects, 699

identifiers

assigning, 72

CLS naming conventions, 265

constants, 176

GUIDs, applying, 147-148

GZip, 68

MyCollectionsUtils, 502

naming conventions, 72

reserved words, applying as, 72

root namespaces, 292

- IDeserializationCallback interface, 1041
- IDictionary interface, 1049
- IDisposable interface, 404
 - templates, 1264
- IDisposable interface, 347, 355
 - implementing, 273-278
 - polymorphism, 352
- IDocument interface, 348
 - polymorphism, 352
- IDs, threads, 1074
- IEnumerable interface, 355-358, 394, 420, 551
- IEnumerator interface, 356
- If operators
 - LINQ, 553
 - ternary, 533-534
- IFormattable interface, 355, 363-365
- If statements, 181
- If..Then code block, 159
- If..Then..Else code block, 172-173
- If operator, 173-174
- IInvoice interface, 353
- IIS (Internet Information Services), 852, 883
 - Data Services (WCF), deploying, 1021
 - WCF, hosting in, 1008-1009
 - web applications, publishing, 885
- IL (Intermediate Language), 263
 - metadata, 1158
- ILGenerator class, 1175
- IList interface, 352, 1049
- ImageBrush, 758, 762-765
- ImageButton control, 859
- Image control, 736, 793-795, 858-859
- ImageMap control, 859
- images, 77
 - backgrounds (Windows Phone), 986
 - exporting, 432
 - Pictures Hub (Windows Phone), 982-984
 - viewing, 793-795
 - Visual Studio 2012 built-in image editors, 498
- implementing
 - annotations, services, 804
 - custom awaiters, 1141
 - custom contracts, 997
 - Data Services (WCF), 1013, 1016-1021
 - destructors, 275
 - Dispose interfaces, 273-278
 - Dispose method, 276
 - DoubleAnimation, 905
 - events, 385-389
 - Finalize method, 276
 - flow documents, 799
 - generics, methods, 371-372
 - ICloneable interface, 118
 - IDisposable interface, 273-278
 - interfaces, 348-352
 - abstract classes, 428
 - comparing reference types/value types, 109-110
 - structures, 309
 - Iterator class, 419-422
 - IValueConverter interfaces,
 - media players, 900
 - nested namespaces, 288-291
 - permissions, 1149
 - services, operations, 1027-1029
 - spell check, 808
 - Tool window, 1296
 - WCF services, 993-1001
 - XML comments, 1210-1219
- Implements keyword, 309, 348
- implicit conversions, 45, 111-113. *See also* converting
- implicit line continuation, 552, 1182
 - LINQ, 559-561
 - Visual Basic 2012 projects, 70-71
- implicit types, 45
- importing
 - COM objects, 1192-1196
 - existing settings, 1273
 - functions, auto-generating, 657
 - XML namespaces, 293, 689
- Imports directives, 17
 - adding, 286
 - multithreading, 1060
 - namespaces, 292-295
 - Visual Basic 2012 projects, 68-69

Imports System.GlobalizedString directive, 843

Include method, 653, 667

include tag (XML comments), 1214

Inconclusive method, 1341

increaseCounter method, 384

Indent method, 194

indexers, 234

indexes, Code Metrics, 1315

IndexOf method, 153, 396

IndexOutOfRangeException exception, 153, 209

Field (Of T) method, 626

inequality (<>) operator, 104, 164, 313, 570

inference, local types, 511-514

lambda expressions, 529

LINQ, 552

Open Infer directives, 513

scope, 514

Infer XML Schema Set from XML dialog box, 687

Info property, 483, 488-490, 720

information messages, 30

infrastructure (WCF), 6

inheritance, 323-324

abstract classes, 337

applying, 324-327

base classes, accessing members, 337-341

classes, preventing, 335

CLS, 327

conditioning, 334-337

constraints, 374

constructors, 341-342

defining, 1187

Dispose interfaces, 275

exceptions, customizing, 344-346

interfaces, 353-354

members, 327, 331-334

modifiers, 428

modules, 303

polymorphism, 329-331

serialization, 1047

shadowing, 342-343

shared members, overriding, 343-344

structures, limitations of, 309

styles, 773

System.Object class, 328-329

types, comparing reference/value, 108-109

Visual Basic 2012 projects, 65-66

Inherits keyword, 66, 303, 309, 354

Init event, 856

initializers

collections, 405-406

objects, 258-259, 437

applying, 558-559

LINQ, 553

InitializeService method, 1018

InitializeWithWindowsUser method, 500

initializing

arrays, 150

fields, 228

ink notes, adding, 806

inline initialization

arrays, 150

fields, 228

InnerException property, 214, 1101

INotifyPropertyChanged interface, 387, 1042, 1178

Insert breakpoint command, 51

Insert method, 395

insert operations

databases, 541

DataSets, 546

Entity Framework (ADO.NET), 646-647

LINQ to SQL, 605-608

InsertRange method, 395

inspecting

arrays, 152-155

assemblies, 1161

objects with debugger visualizers, 192-193

strings, 131-133

Install ApplicationName onto This Computer command, 924

Installation Interview group (InstallShield), 1241

installing

ClickOnce, 1229

extensions, 1301-1303

GAC assemblies, 1223-1224

InstallShield Installation Requirements group, 1234-1236

local installation options, running out-of-browser applications, 925

.NET Framework, 2-3

- SDKs, 7
- templates, 19
- Visual Studio 2012 tools (Windows Azure), 931-932
- web applications, packaging for, 888-891
- Windows Installer, 1230
- InstallShield, 1229
 - Application Files group,
 - Application Information group, 1233
 - Application Registry, 1239
 - downloading, 1231
 - Environment Variables element, 1242
 - groups. *See* groups
 - Installation Interview group, 1241
 - Installation Requirements group, 1234-1236
 - overview, 1231
 - packages, deploying, 1244
 - projects
 - creating, 1232-1242
 - formatting, 1242
- instances
 - clients, 606
 - declaring (LINQ to SQL), 600
 - variables, 45
- instantiating
 - Entity Framework (ADO.NET), 645
 - proxy classes, 1003
 - windows at runtime, 718
- instrumentation, debugging, 180-192
- integer division (\) operator, 155, 313
- Integer keyword, 95
- integers, removing overflow checks, 47
- Integrated Development Environment. *See* IDE
- integration, media, 696
- IntelliSense
 - code editors, 25-27
 - collections, 405
 - enumerations, 315
 - events
 - customizing, 389
 - generating handlers, 706
 - exception handling, 210
 - extension methods, 518
 - generics, 368-370
 - members, overriding, 332
 - methods, overloading, 244
 - namespaces, 292
 - partial classes, 250
 - uncommented members, 1208
 - With..End With statements, 176
 - XAML code editors, 701
 - XML
 - comments, 1210. *See also* XML comments
 - importing namespaces, 689
- IntelliTrace, 52, 1310, 1328-1334
- interacting
 - with garbage collectors, 279-280
 - with operating systems, 4
- interceptors
 - change, 1032
 - queries, 1030-1033
- Interface..End Interface block, 348
- interfaces, 11, 347
 - abstract classes, implementing, 428
 - accessing, 348-352
 - APIs. *See* APIs
 - CLS, 354-355
 - defining, 347-348
 - Dispose, implementing, 273-278
 - Fiddler, 1359
 - Fluent APIs, Entity Framework (ADO.NET), 665-668
 - generating, 437
 - generics, defining, 370
 - ICloneable, 115, 118, 355
 - ICollection, 394
 - ICommand, 908
 - IComparable, 109, 355, 358-359
 - IComparer, 359-360
 - IConvertible, 355, 360-363
 - IDataErrorInfo, 616
 - IDeserializationCallback, 1041
 - IDictionary, 1049
 - IDisposable, 404, 1264
 - IDispose, 347, 355
 - implementing, 273-278
 - polymorphism, 352

- IDocument, 348, 352
- IEnumerable, 355-358, 394, 420, 551
- IEnumerator, 356
- IFormattable, 355, 363-365
- Invoice, 353
- IList, 352, 1049
- implementing, 109-110, 348-352
- inheritance, 353-354
- INotifyPropertyChanged, 387, 1042, 1178
- IPerson, 427
- ISerializable, 1046
- IService1, 996
- IValueConverter, implementing, 835-838
- media players, 796
- members, adding, 426
- method arguments, passing as, 351
- method constraints, 373
- Modern style (Windows Phone), 956
- modules, 303
- namespaces, 284
- nested classes, 348
- .NET Framework, 355-365
- polymorphism, 352-353
- scope, 348
- selecting, 348
- structures, implementing, 309
- System.IProgress (Of T), 1134
- Tool window, implementing, 1296
- Visual Studio 2012, 439
- Windows Phone, 12
- WPF, 694, 721-724. *See also* WPF
- XBAP, 722
- XML schemas, 685-690
- Intermediate Language. *See* IL, 263, 1158
- Internet code samples, searching, 21
- Internet Explorer, 852
 - Fiddler, 1359
 - XML serialization, 1043
- Internet Information Services. *See* IIS
- Interop.AssemblyName.dll, 86
- interoperability
 - COM assemblies, registering for, 1197
 - between COM/.NET architectures, 86
- Interop.WMPLib.dll, 87
- Intersect method, 519
- Into clause, 570
- InvalidCastException, 123-125, 373
 - Field (Of T) method, 626
- InvalidOperationException, 252, 584, 612, 1091
- invariants, contracts, 1354
- Invoke method, 380
- invoking
 - AsParallel, 1097
 - constructors, 254
 - destructors, 270
 - dynamic code, 1169-1171
 - members, 29, 1003-1007
 - methods, 236-237, 308
 - Object Browsers from code editors, 57
- I/O (input/output)
 - asynchronous programming, 8, 1137-1141
 - Windows Phone, 959
- IOException, 455, 459
- IPerson interface, 427
- IronPython, 6-7
- IronRuby, 6-7
- IsAddingCompleted property, 1091
- IsAuthenticated method, 500
- IsCompleted property, 1075, 1091
- ISerializable interface, 1046
- IService1 interface, 996
- IsFalse method, 1341
- IsFalse operator, 158
- IsInRole method, 500
- IsInstanceOfType method, 1341
- IsNetworkAvailable property, 487
- IsNetworkDeployed property, 480
- IsNotInstanceOfType method, 1341
- IsNotNull method, 1341
- IsNot operator, 164-165, 570
- IsNull method, 1341
- Is operator, 164-165, 570
- IsRunningOutOfBrowser property, 924
- IsTrue method, 1341
- IsTrue operator, 158
- Item method, 396

- items
 - projects, adding, 30
 - properties, configuring, 31
 - references, specifying in, 1266
 - templates, exporting, 1263-1265
- ItemSource property, 823
- iterations, 166-170
- Iterator class, implementing, 419-422
- iterators, 169, 393
 - anonymous, 419
 - exiting, 418
 - LINQ, applying, 562-565
 - simple, 417-418
 - Try..Catch..Finally blocks, 418-419
 - Visual Basic 2012, 414-422
 - XML literals, 685
- IValueConverter interfaces, implementing, 835-838

J

- jagged arrays, 151-152, 516
- JavaScript, 8, 894
- JavaScript Object Notation. *See* JSON, 1015, 1053
- JIT (Just-In-Time), 5
 - multicore JIT compilation, 853
 - optimizing, 47
- Join method, 519
- JSON (JavaScript Object Notation), 1015
 - serialization, 1053
- JustDecompile, 1359
- Just-In-Time. *See* JIT, 5, 47, 853
- Just My Code
 - debugging, 182-184
 - disabling, 1078

K

- KeepChanges, 616
- KeepCurrentValues, 616
- Key attribute, 664
- Keyboard property, 482, 485

- keyboard shortcuts, Debug menu (Visual Studio 2012), 181
- keys, foreign, 633
- keywords
 - From, 553
 - Of, 368
 - (pointer to address in memory), 95
 - AddHandler, 383-384
 - Await, 1109, 1120-1122
 - Boolean, 95
 - ByRef, 237-242
 - Byte, 95
 - ByVal, 237-242
 - Char, 95
 - Decimal, 95
 - Default, 233
 - Delegate, 380
 - Dim, 308
 - Double, 95
 - Event, 386
 - Function, 527
 - GetType, 119
 - Global, namespaces, 295-299
 - Implements, 309, 348
 - Inherits, 66, 303, 309, 354
 - Integer, 95
 - Let, 572
 - Long, 95
 - Me, 229
 - MustInherit, 303, 336
 - MustOverride, 336
 - MyBase, 332, 337-339
 - MyClass, 339-341
 - New, 67, 136, 252
 - constraints, 373
 - structures, 438
 - NotInheritable, 303, 335-336
 - NotOverridable, 334
 - Optional, 240-241
 - Overloads, 255, 334
 - Overrides, 332
 - Partial, 614
 - ReadOnly, 231
 - ReDim, 150-151
 - RemoveHandler, 383-384

- reserved (Visual Basic 2012 projects), 72
- SByte, 95
- Select, 553
- Shared, 259
- Short, 95
- Single, 95
- Structure, 374
- Throw, 218-220
- UInteger, 95
- ULong, 95
- unreserved (Visual Basic 2012 projects), 74
- UShort, 95
- value types, 94
- When, 222-223
- Where, 553
- WithEvents, 384-385
- Kill method, 1059

L

- Label controls, 736, 859
- labels, editing breakpoints, 184
- lambda expressions, 526-533
 - arguments, 518
 - asynchronous programming, 1136-1137
 - lexical closures, 532-533
 - LINQ, 552
 - local type inference, 529
 - multiline, 530-531
 - Sub, 531-532
 - ternary If operators, 533-534
 - threads, creating with, 1060
- languages
 - anonymous types, 524-525
 - Array literals feature, 515-516
 - BAML, 844
 - CIL, 5
 - CLS, 263-267
 - Common Type System, 89-93
 - CSDL, 634
 - design, 12
 - exceptions, 207-208, 224
 - extension methods, 517-524
 - features, 511
 - generic variances, 535-537
 - lambda expressions, 526-533
 - LINQ, support, 552-553
 - local type inference, 511-514
 - Mapping Definition Language, 637
 - .NET Framework, 6-7
 - relaxed delegates, 526
 - SSDL, 636
 - ternary If operators, 533-534
 - WDSL, 991
 - XAML. *See* XAML
 - XHTML, 855
 - XML. *See* XML
- LastEdit property, 1185
- Last-In, First-Out. *See* LIFO, 398
- Last method, 519
- LastNameChangedEventArgs class, 388
- LastName method, 325
- LastName property, 534
- LastNode property, 674
- LastOrDefault method, 519
- Last property, 410
- late binding, 45, 115
 - reflection, 1176
- Latest News tab (Visual Studio 2010), 14-15
- launchers (Windows Phone), 967-720
- Launch Performance Wizard, 1320
- layers
 - architecture (LINQ), 554
 - .NET architecture, 2-3
 - User32 (WPF), 696
- Layout Diagram command, 424
- learning resources (Windows Phone), 958
- left-shift (<<) operator, 162, 313
- Length method, 464
- less than (<) operator, 164, 313
- less than or equal to (<=) operator, 164, 313
- Let keyword, 572
- levels
 - scope (Visual Basic 2012), 234
 - Transparency Level 2, 1150-1152
 - trust for roles, 935
- lexical closures, lambda expressions, 532-533

libraries

- BCL, 2, 5-6, 57
- classes, 226
- Code Contract, 1350-1355
- COM, adding references, 85-86
- DirectX (WPF), 696
- DLLs, 2
- extensions, testing, 524
- Microsoft Core Library (MsCorlib.dll), 6
- MSDN Library, 48
 - navigating, 56
 - publishing web applications, 890
- NuGet, managing, 1283-1286
- Open Data Protocol (OData), 1015
- portable, 423
- Portable Class Libraries, 7, 440-451
- TPL, 413

licenses (NuGet), 1285

lifetimes

- objects
 - finalizers, 110
 - managing, 269
- pages, 856

LIFO (Last-In, First-Out), 398

limitations

- of ClickOnce, 1246
- of drag'n'drop data-binding, 818
- of inheritance, structures, 309
- of LINQ, 550
- of XML serialization, 1043

LinearGradientBrush, 758-761

LineHeight property, 899

lines

- grid, viewing, 711
- implicit line continuation, 70-71, 552, 559-561, 1182

Line Stacking Strategy, 899

LinkButton control, 859

LinkedList (Of T) collection, 410-412

links, specifying resources to, 1219

LINQ, 6, 549

- architecture, 554
- to DataSets, 621
 - extension methods, 624-627
 - queries, 621-624

deferred execution, 565-568

examples of, 551-552

extending, 554

iterators, applying, 562-565

language support, 552-553

limitations of, 550

to Objects, 557

overview of, 557-558

querying in memory objects, 558-568

overview of, 549-551

parallel, 1069

partial methods, 251

PLINQ, 1092

providers, 553-554

queries

anonymous types, 524

measuring performances, 1093-1095

standard query operators, 568-585

LINQPad, 1359

LINQ to Data Services, 1027

LINQ to Entities, querying with EDMs, 652-653

LINQ to SQL, 587

advanced, 613-617

applying logs, 613

classes, 589-599

deleting entities, 609

EDMs, exposing, 1001

inserting entities, 605-608

mapping stored procedures, 610-613

overview of, 588-599

queries, 600-604

SQL Server Compact Edition 3.5, 617-619

updating entities, 608-607

LINQ to XML

interface schemas, 685-690

modifying, 671

overview of, 672-677

querying, 676-677

ListBox control, 736, 859

List (Of T) collection, 403-405

ListDictionary collection, 399

listeners

configuration file settings, 200-202

trace, applying, 196-202

lists

- data-binding views, 830-835
- Error Lists, 24
- tags (XML comments), 1214
- XML comments, 1218

ListView control, 738

literals

- Array Literals, 151
- Array literals feature, 515-516
- XML
 - LINQ, 553
 - writing XML markup in, 677-685

literal type characters, 98

Live Tiles, 956

LNG function, 121

LoadCompleted event, 510

LoadComponent method, 720

LoadedAssemblies property, 479

LoadedBehavior property, 797

Load event, 856

LoadFile method, 1160

LoadFrom method, 1160

loading

- assemblies, 1160
- RSS feeds, 1116
- XML documents, 674

Load method, 1160

LoadVideoAsync method, 1133

local data access (LINQ to SQL), 618

local data storage (Windows Phone), 980-981

Local DB (SQL Server 2012) system requirements, 539-540

local file systems, accessing, 926

local installation options, running out-of-browser applications, 925

localizing applications, 841

- .NET Framework, 841-842
- Windows Forms, 842-843
- WPF, 844-850

Local property, 667

Local Storage, 930

Locals window, 187

local type inference, 511-514

- lambda expressions, 529
- LINQ, 552

Open Infer directives, 513

scope, 514

local variables

- ambiguities, avoiding, 228-229
- local type inference, 511-514
- unused, 45

locations. *See also* storage

- assemblies, 1144
- breakpoints, 185
- ClickOnce deployments, 1247
- GAC, 1223
- IronPython, 6-7
- IronRuby, 6-7
- memory, data types, 90
- .NET Framework, 2
- projects, searching, 24
- SDKs, 7
- services (Windows Phone), 959
- templates, 20

LocBaml.exe, 844-845

locks, read/write, 1065-1066

logical operators, 158-159

Logical Tree (WPF), 704-705

logs

- Activity log (Windows Azure), 947
- applications
 - managing, 199-200
 - writing entries to, 481-482
- Caller Information, 1178
- compilation, 40
- IntelliTrace, 1333-1334
- SQL, applying, 613
- Windows Event Log, 199

LongCount method, 519

Long keyword, 95

loops, 95-172

- Do..Loops, 170-171
- For Each, 150, 168-169, 358, 1059
- For..Next, 166-168
- iterators, 417
- local type inference, 513
- parallel computing, 1080-1086
- scaling, 1080
- While..End While, 171-172

loss of precision, 122-123

M

- Main method, 24, 106, 434, 720
- MainWindow property, 720
- MainWindow.xaml.vb code file, 29
- managed code
 - ClickOnce, accessing, 1257-1258
 - writing, 4
- Managed Extensibility Framework, 8, 442, 1288
- Managed Heap, 90, 106
- Management Portal, 946
 - Windows Azure, 949-952
- Manage NuGet Packages dialog box, 658
- managing
 - add-ins, 1304
 - applications
 - logs, 199-200
 - .NET Framework, 2. *See also* .NET Framework
 - breakpoints, 184
 - Code Snippet Manager, 1278
 - concurrent operations, 1129
 - Configuration Manager, 42
 - controls, 702, 709-716
 - extensions (Visual Studio 2012), 1302-1303
 - Help contents, 56
 - indexes, 233
 - keyboards, 485
 - libraries (NuGet), 1283-1286
 - Memory Manager, 106
 - namespaces
 - Global keyword, 295-299
 - overview of, 283-284
 - objects
 - advanced garbage collection, 279-281
 - allocating memory, 269-270
 - Finalize method, 271-272
 - garbage collection, 270-271
 - implementing Dispose/IDisposable interfaces, 273-278
 - lifetimes, 269
 - resurrecting, 278-279
 - permissions, 881
 - PLINQ queries, 1097-74
 - processes, 1058-1060
 - projects, items, 29
 - Reference Manager dialog box, 84-85
 - ResourceManager class, 80, 500
 - Solution Explorer windows, 28-30
 - state, 861-864
 - structures, 310
 - tasks, 1073-1074
 - templates, 18
 - types within namespaces, 283-295
 - user settings, 1271-1273
 - windows, 716-719
 - Windows Azure Management Tool, 1360
- manifest options, ClickOnce, 1254
- manual installation, packaging for web applications, 888-891
- mapping
 - classes, 1335
 - EDMs, 638-640, 1335
 - resources, 500
 - stored procedures, 610-613, 654-657
- Mapping Definition Language, 637
- Mapping Details window, 640
- Marketplace (Windows Phone)
 - applications (apps), submitting, 987-989
 - registering, 956
- MarketPlaceDetailsTask, 967
- MarketPlaceHubTask, 967, 974
- MarketPlaceReviewTask, 967
- MarketPlaceSearchTask, 967
- markup
 - BAML, 844
 - Binding markup extension, 812-813
 - XAML. *See* XAML
 - XML, writing, 677-685
- master-details forms, formatting, 826-830
- master pages, 855-857, 865-868
- MaxLength attribute, 664
- Max method, 519
- MaxValue property, 101, 137, 143-144
- measuring performance
 - LINQ queries, 1093-1095
 - PLINQ queries, 1095-1085

media

- animations, 782-790
- images, viewing, 793-795
- integration, 696
- playing, 795-798
- Silverlight applications, playing, 900-905
- Windows Phone, 959
- WPF, 793

MediaElement control, 739, 795, 900

MediaFailed event, 798

MediaPlayerLauncher, 967-969

media players

- content reproduction, 901
- implementing, 900

Me keyword, 229

members

- accessing (Visual Basic 2012 projects), 67-68

Application class, 720

ArrayList collections, 396

base classes, accessing, 337-341

behavior, 29

deriving, overriding, 334

extension methods, 403

inheritance, 327, 331-334

interfaces

- adding, 426

- defining, 348

LinkedList (Of T) collection, 410

My.Application property, 478

My.Computer.Clipboard, 484

My.User property, 500

NavigationService class, 965

RefreshMode enumeration, 616

scope, 327

services, invoking from, 1003-1007

sharing, 45

- classes, 259-263

- generating, 436

- overriding, 343-344

streams, 461, 464

structures, 308

visibility, 234-235

Visual Basic 2012 projects, 63-71

Visual Studio 2010, generating based on interfaces, 350

XDocument class, 674

MemberwiseClone method, 116, 328

memory

allocating

- comparing reference types/value types, 106-108

- managing objects, 269-270

deallocating, 110

locations, data types, 90

objects, queries, 558-568

streams, 466

structures, 309

value types, 90, 94

Memory Manager, 106

Menu control, 740

menus

- Debug (Visual Studio 2012), 181

- Tools (Visual Studio 2012), customizing, 1267-1268

MessageContract, 993

Message property, 214

messages

- errors, 30

- Code Analysis, 1314

- implicit conversions, 113

- information, 30

- text, 972

- warnings, 30

metadata, 5

- assemblies, 1158-1160

- attributes, 1183

- reflection, 1158. *See also* reflection

- services, exporting, 999

MethodAccessException, 1152

methods

- Add, 370, 394, 660, 1087

- AddAfter, 410

- AddAfterSelf, 674

- AddBefore, 410

- AddBeforeSelf, 674

- AddFirst, 410, 674

- AddLast, 410

- AddMemoryPressure, 280
- AddNew, 833
- AddParticipant, 1080
- AddParticipants, 1080
- AddProduct, 606
- AddRange, 395
- Aggregate, 519
- All, 519
- Any, 519
- AppDomain.CreateDomain, 1147
- AreEqual, 1341
- AreNotEqual, 1341
- AreSame, 1341
- arguments, 237-242, 351
- AsDataView, 623
- AsEnumerable, 517-519
- AsNoTracking, 654
- Assert, 194
- Asset class, 1341
- Average, 519
- BeginRead, 464
- BeginWrite, 464
- BinarySearch, 153
- CalculateDiscount, 354
- CalculatePerimeter, 1349
- calling, 60
- CanRead, 464
- CanSeek, 464
- CanWrite, 464
- Cast, 519
- ChangeExtension, 454
- CheckMailAddress, 381
- classes, executing actions with, 235-247
- Clear, 396, 404, 410, 484
- Clone, 117, 352, 374
- Close, 194, 464
- Collect, 279
- Compare, 127
- Concat, 519
- constraints, 373
- constructors, 252-259
- Contains, 396, 410, 519
- ContainsAudio, 484
- ContainsData, 484
- ContainsFileDropList, 484
- ContainsImage, 484
- ContainsText, 484
- ConvertTimeBySystemZoneId, 147
- CopyTo, 396, 410
- CopyToAsync, 1138
- CopyToDataTable, 624-626
- Count, 307, 396, 519
- .ctor, 216
- CurrentPrincipal, 500
- Debug class, 194-195
- DefaultIfEmpty, 519
- Delete, 457
- DeleteObject, 1025
- Dequeue, 398
- Descendants, 674
- Dispose, 271, 275-276, 347, 404
- Distinct, 519
- Divide, 533
- DoCancel, 873
- DoSomething, 229
- DownloadFile, 487
- DownloadString, 1115
- Element, 674
- ElementAt, 519
- ElementAtOrDefault, 519
- EndRead, 464
- EndWrite, 1107
- Equals, 309, 314, 328
- Except, 519
- Execute (Of T), 1026
- ExecuteAssembly, 1146
- ExecuteCommand, 617
- ExecuteReader, 543
- exiting, 246-247
- extension, 517-524
 - behavior, 521
 - built-in, 519
 - customizing, 521-523
 - Data Services (WCF), 1025
 - exporting, 523-524
 - LINQ, 552
 - LINQ to DataSets, 624-627
 - members, 403
 - overloading, 523

- Fail, 194-195, 1341
- Field (Of T), 626-627
- File.Delete, 197
- Finalize, 271-272, 276, 328
- FindAll, 405
- FindResource, 720
- First, 519, 1025
- FirstOrDefault, 519
- Fluent APIs, 667
- Flush, 194
- FlushAsync, 1138
- FullName, 325, 333, 435
- GC.ReRegisterForFinalize, 279
- GC.SuppressFinalize, 278
- generating, 437
- generics
 - defining, 370
 - implementing, 371-372
- GetAssembly, 1160
- GetAudioStream, 484
- GetCallingAssembly, 1160
- GetCreationTime, 455
- GetCurrentDirectory, 458
- GetCustomerQuery, 825
- GetCustomers, 502
- GetData, 484
- GetDataObject, 484
- GetEntryAssembly, 1160
- GetEnumerator, 420
- GetExecutingAssembly, 1160
- GetExportedTypes, 1162
- GetExtension, 454
- GetFileDropDownList, 484
- GetFiles, 456
- GetHashCode, 309, 314, 328
- GetILGenerator, 1175
- GetImage, 484
- GetLogicalDrives, 457
- GetLowerBound, 153
- GetModules, 1162
- GetName, 317
- GetNames, 316-317
- GetObjectData, 1046
- GetOrders, 916
- GetRange, 396

- GetSystemTimeZones, 145
- GetText, 484
- GetType, 119, 328
- GetTypes, 1163
- GetUpperBound, 153
- GetValidationErrors, 651
- GetValue, 155
- GetValues, 316-317
- GoBack, 965
- GoForward, 965
- GroupBy, 519
- GroupJoin, 519
- Guid.NewGuid, 147
- HasExtension, 454
- Include, 653, 667
- Inconclusive, 1341
- increaseCounter, 384
- Indent, 194
- IndexOf, 153, 396
- InitializeService, 1018
- InitializeWithWindowsUser, 500
- Insert, 395
- InsertRange, 395
- Intersect, 519
- Invoke, 380
- invoking, 236-237
- IsAuthenticated, 500
- IsFalse, 1341
- IsInRole, 500
- IsInstanceOfType, 1341
- IsNotInstanceOfType, 1341
- IsNotNull, 1341
- IsNull, 1341
- IsTrue, 1341
- Item, 396
- Join, 519
- Kill, 1059
- Last, 519
- LastName, 325
- LastOrDefault, 519
- Length, 464
- Load, 1160
- LoadComponent, 720
- LoadFile, 1160
- LoadFrom, 1160

- LoadVideoAsync, 1133
- LongCount, 519
- Main, 24, 106, 434, 720
- Max, 519
- MemberwiseClone, 116, 328
- Min, 519
- Move, 455
- MoveCurrentTo, 832
- MoveCurrentToFirst, 832
- MoveCurrentToLast, 832
- MoveCurrentToNext, 832
- MoveCurrentToPrevious, 832
- Name, 500
- Navigate, 965
- New, 255, 328
- OffType, 519
- OnCreateMainForm, 76
- OnDeserialization, 1041
- OnModelCreating, 666
- OrderBy, 519
- OrderByDescending, 519
- overloading, 242-246
- Parallel.For, 1080-1083
- Parallel.ForEach, 1080, 1083-1084
- Parallel.Invoke, 1072
- Parse, 100, 139, 318-319
- partial, 251-252
- Peek, 398
- Position, 464
- Print, 194
- PrintString, 237
- QueryVideosAsync, 1132
- Read, 464
- ReadAllText, 160
- ReadAsync, 1138
- ReadByte, 464
- ReadLineAsync, 1138
- ReadToEndAsync, 1138
- ReferenceEquals, 328
- ReflectionOnlyLoad, 1160
- ReflectionOnlyLoadFrom, 1160
- Remove, 410, 660, 674
- RemoveFirst, 410
- RemoveLast, 410
- RemoveMemoryPressure, 280
- RemoveNodes, 674
- RemoveParticipant, 1080
- RemoveParticipants, 1080
- ReplaceWith, 674
- ReRegisterForFinalize, 279
- ReturnFullName, 243
- Reverse, 396, 519
- Root, 674
- rules, 267
- Run, 720
- SaveChanges, 833
- Seek, 464
- Select, 519
- SelectMany, 519
- SequenceEquals, 519
- Serialize, 1036
- SetAudio, 484
- SetCreationTime, 455
- SetCurrentDirectory, 458
- SetData, 484
- SetDataObject, 484
- SetField (Of T), 626-627
- SetImage, 484
- SetLastError, 1206
- SetText, 484
- ShowDialog, 755
- Shutdown, 720
- SimulateProcessing, 1081
- Single, 519, 1025
- SingleOrDefault, 519
- Skip, 519
- SkipWhile, 519
- Sort, 396
- Start, 1058
- StopLoading, 965
- Stream class, 1138
- structures, passing to, 308
- stubs, generating, 436
- Sub Main, 35
- Subtract, 142
- Sum, 519
- SuppressFinalize, 279
- System.Array class, 153
- System.Convert class, 123
- System.Enum class, 316-319

- System.GC.Collect, 271
- System.Object class, 92, 328
- System.Object.ToString, 66
- System.String class, 126
- Take, 519
- TakeWhile, 519
- Task.Run, 1128
- Task.Wait, 1075
- Test, 181, 187
- TestAccessFile, 222
- TestInstance, 374
- ThenBy, 519
- ThenByDescending, 519
- Thread.Start, 1061
- ToArray, 396, 519
- ToBool, 123
- ToByte, 123
- ToChar, 123
- ToDateTime, 123
- ToDecimal, 123
- ToDictionary, 519
- ToDouble, 123
- ToFileTime, 141
- ToFileTimeUtc, 141
- ToInt16, 123
- ToInt32, 123
- ToInt64, 123
- ToList, 519
- ToLocalTime, 141
- ToLongDateString, 141
- ToLongTimeString, 141
- ToLookup, 519
- ToOADate, 141
- ToSByte, 123
- ToShortDateString, 141
- ToShortTimeString, 141
- ToSingle, 123
- ToString, 121, 309
 - enumerations, 318-319
 - reflection, 1163
 - System.Object class, 328
- ToUInt16, 123
- ToUInt32, 123
- ToUInt64, 123
- ToUniversalTime, 141
- Trace.Listener.Clear, 197
- TrimToSize, 396
- TryParse, 100, 139
- Unindent, 194
- Union, 519
- UpdateProduct, 648
- Upgrade, 495
- validatebook, 998
- value types, 100-101
- Visual Basic 2012 projects, 64-65
- WaitForPendingFinalizers, 279
- WhenAll, 1129
- WhenAny, 1129-1131
- Where, 519
- WithDegreeOfParallelism, 1098
- WithMergeOptions, 1098
- Write, 194, 464
- WriteAllBytes, 460
- WriteAsync, 1138
- WriteByte, 464
- Writelf, 194
- WriteLine, 26, 194
- WriteLineIf, 194
- metrics, Code Metrics, 1309, 1315
- Metro interface. *See* Modern style interface (Windows Phone), 956
- Metro-style applications, 8. *See also* Windows 8
 - .NET for, 7
 - ObservableCollection (Of T) collection, 407
- Microsoft Access, 541
- Microsoft code analysis rules, 1311
- Microsoft Core Library (MsCorlib.dll), 6
- Microsoft Design Style, 11-12
- Microsoft Developer Network. *See also* MSDN
- Microsoft Excel, exporting code metrics
 - to, 1317
- Microsoft Expression Blend, 14
- Microsoft Silverlight. *See* Silverlight applications
- Microsoft SQL Server, LINQ to SQL, 588.
 - See also* LINQ
- migrating
 - Code First approach, 665
 - from old CAS-based code, 1155
 - Visual Basic 6, 209
 - from Visual Studio 2010 to Visual Studio 2012, 3

- MIME (Multipurpose Internet Mail Extensions), 80
- MinLength attribute, 664
- Min method, 519
- MinValue property, 101, 137, 143-144
- MissingLastNameException, 345-346
- Mixed Mode debugging, 182
- mobile phones, 955. *See also* Windows Phone
- Model Browser window, 640, 656
- Model First, 657
- models
 - adding, 868
 - APM, 1106-1107
 - ASP.NET, 851. *See also* ASP.NET
 - Code First approach, 659-660
 - COM, 1191
 - EDMs, viewing, 820
 - MVVM, 443, 811, 913
 - objects, EDMs, 630
 - programming (Windows Phone), 958-959
 - sandboxed, 1149, 1152-1154
 - security in .NET 4.5, 1148-1155
 - transparency, 1148-1149
 - views, defining, 446
- Model-View-Controller. *See* MVC, 854
- Model-View-View Model. *See* MVVM, 443, 811, 913
- Modern style interface (Windows Phone), 956
- modes
 - connection (ADO.NET), 541
 - data-binding, 813
 - declarative, XAML, 703
 - disconnected (ADO.NET), 541
 - operation, garbage collection, 280-281
 - profiling, 1320
- modifiers
 - Friend, 225, 234
 - inheritance, 428
 - Private, 225, 234
 - Protected, 225, 234
 - Protected Friend, 225, 234
 - Public, 225, 234
- modifying
 - database schemes, 664-665
 - debugging, 50
 - default values, thread pools, 1063
 - directories, 453-459
 - System.IO.Directory class, 455-458
 - System.IO.DirectoryInfo class, 458-459
 - documents, 799-808
 - elements, angles, 780
 - Entity Framework (ADO.NET), 645-652
 - files, 453, 460-463
 - permissions, 463
 - System.IO.File class, 460-461
 - System.IO.FileInfo class, 462-463
 - LINQ to XML, 671-677
 - pathnames, 453-459
 - properties, notifications, 387
 - Registry, 486-487
 - Startup objects, 35
 - streams, 453
- Mod operator, 155, 570
- modules, 301
 - classes, comparing, 303
 - namespaces, 284
 - overview of, 301-303
 - Visual Basic 2012 projects, 65
- Monitor class, synchronization, 1065
- Mouse property, 482
- MoveCurrentToFirst method, 832
- MoveCurrentToLast method, 832
- MoveCurrentTo method, 832
- MoveCurrentToNext method, 832
- MoveCurrentToPrevious method, 832
- Move method, 455
- moving
 - breakpoints, 185
 - drag'n'drop data-binding, 818-840
 - elements, 781
 - XML namespaces, 293
- MSBuild.exe, 7, 40, 847
- MsCorlib.dll, 6
- MSDeploy, 884
 - publishing, 886-891
 - settings, 889-888

MSDN (Microsoft Developer Network)

Code Gallery, 1303

Library, 48

navigating, 56

Visual Basic 2012 resources,
1357-1358

web applications, publishing, 890

MSIL. *See* CIL, 5

MsmqIntegrationBinding, 1005

multicast delegates, 382-383

multicore JIT compilation, 853

multicore processor architectures, Parallel JIT
Compilation, 5

multidimensional arrays, 151, 516

multiline lambda expressions, 530-531

multimedia animation, 782-790

multiple applications, references (GAC), 1227

multiple constraints, 374

multiple diagrams, formatting, 431

multiple interface implementations, 350

multiple modules, 302

multiple roles, adding, 936

multiple scopes, 8

multiple times, applying attributes, 1187

multiple transforms, 782

multiplication (*) operator, 155, 313

Multipurpose Internet Mail Extensions.

See MIME, 80

multi-targeting, 18

multithreading, 1057, 1060-1061

MultiView control, 859

MustInherit keyword, 303, 336

MustOverride keyword, 336

MVC (Model-View-Controller), 854

MVVM (Model-View-View Model), 443, 811

Silverlight applications, 913

My.Application property, 477

assemblies, retrieving information, 478-479

cultures, 479-480

deploying, 480-482

extending, 504-505

MyBase keyword, 332, 337-339

MyClass keyword, 339-341

MyCollectionsUtils identifier, 502

My.Computer property, 478, 482-490

Audio property, 485

Clipboard property, 484-485

extending, 505-506

file systems, 483-484

Keyboard property, 485

Name property, 488-490

Network property, 487

Registry property, 486-487

My namespace, 477

applying, 506-510

extending, 502-506

My.Application property, 478-482

cultures, 479-480

deploying, 480-482

My.Computer property, 482-490

Audio property, 485

Clipboard property, 484-485

Keyboard property, 485

Name property, 488-490

Network property, 487

Registry property, 486-487

My namespace, 478-482

My.Resources property, 497-500

My.Settings property, 490-496

application-level only settings, 492-493

events, 495-496

naming configuration files, 493-495

My.User property, 500-502

My.WebServices property, 502

overview of, 477-478

My Project, 33-39

compiling options, 41

navigating, 72-75

resources, 78

Settings tab, 82

WPF applications, 699

My.Resources namespace, 478, 506

My.Resources property, 497-500

My.Settings property, 478, 490-496

application-level only settings, 492-493

events, 495-496

extending, 506

naming configuration files, 493-495

MySQL, 541
 My.User property, 478, 500-502
 My.WebServices property, 478, 502

N

Name method, 500
 Name property, 428, 488-490
 names
 configuration files, 493-495
 conventions, 8
 CLS, 264-266
 Code Analysis, 1314
 exceptions, 209, 345
 identifiers, 72
 interfaces, 355
 value types, 94
 enumerations, 316-317
 pathnames
 modifying, 453-459
 System.IO.Path class, 454-455
 resources, 80
 strong names, signing GAC assemblies with, 1224-1226
 System.Object class, 91
 Namespace..End Namespace blocks, 284, 288
 namespaces
 CLS, 295
 compiling, 6
 Global keyword, 295-299
 Imports directives, 292-295
 My namespace. See My namespace
 My.Resources, 478, 506
 nested, implementing, 288-291
 purpose of, 287-288
 root, 34, 291-292
 scope, 291
 System.Collections, 394
 System.Collections.Concurrent, 413
 System.DataSet.DataSetExtensions, 622
 System.Diagnostics, 197

System.Globalization, 842
 System.IO, 8, 293
 System.Reflection, 1158
 System.Reflection.Emit, 1171-1177
 System.Runtime.CompilerServices, 1177
 System.Runtime.InteropServices, 310
 System.Windows.Controls, 6
 System.Xml.Linq, 672-674
 System.Xml.Serialization, 1043
 types
 managing, 283-295
 overview of, 283-284
 Visual Basic 2012 projects, 66-67
 XML, importing, 293, 689
 NameValueCollection collection, 401
 narrowing conversions, 120-125
 Navigated event, 510, 965
 Navigate method, 965
 NavigateUri property, 806, 910
 navigating
 ASP.NET applications, 864-862
 controls, adding, 874
 IDE tabs, 50
 MSDN Library, 56
 My Project, 33-39, 72-75
 Object Browser window, 56-59
 Online Help, 56
 Silverlight Navigation Applications, 908-910
 Solution Explorer windows, 28-30
 Visual Studio 2012, 11
 windows, Error Lists, 30-31
 Windows Phone applications, 963-966
 Navigating event, 510, 965
 NavigationFailed event, 510, 965
 NavigationProcess event, 510
 Navigation Properties, EDMs, 640
 NavigationService class, 965
 NavigationStopped event, 510, 965
 NestedClass class, 227
 nested classes, 226-227, 348
 nested constructor invocations, 256-257
 nested namespaces, implementing, 288-291
 nested Try..Catch..Finally blocks, 217-218
 nested types, 374

.NET Framework**ADO.NET, 539**

connecting databases, 541-543

connection modes, 541

data providers, 540-541

overview of, 540-543

anonymous types, 524

applications

ASP.NET, 851

InstallShield, 1229. *See also*

InstallShield

localizing, 841-842

architecture, 2-3

arrays, applying, 148-155

assemblies, 4-5, 1143

application domains, 1145-1147

overview of, 1143-1145

security models, 1148-1155

asynchronous programming

APM, 1106-1107

Async pattern, 1107-1112

EAP, 1104-1106

overview of, 1104

attributes

applying, 1181-1184

coding, 1181

customizing code, 1184-1188

reflection, 1189-1190

BCL, 5-6

classes, 225

CLS, 263-267

constructors, 252-259

declaring, 225-227

executing actions with methods, 235-247

fields, 227-229

partial, 248-251

partial methods, 251-252

properties, 229-234

scope, 234-235

shared members, 259-263

ClickOnce, 1229, 1245-1247

CLR, 4-5

Code Analysis, 1310

Code Contract, 1350-1355

code documentation, 1207. *See also* XML
comments

collections, 393

architecture, 394

ArrayList, 394-397

BitArray, 401

Bitvector32, 402-403

concurrent, 413

customizing, 413

Dictionary (Of TKey, TValue), 407

HashTable, 398-399

HybridDictionary, 400

initializers, 405-406

ListDictionary, 399

NameValueCollection, 401

nongeneric, 394-403, 406

ObservableCollection (Of T), 408-410

OrderedDictionary, 399

Queue, 397-398

Queue (Of T), 412

SortedDictionary (Of TKey, TValue), 408

SortedList, 400

Stack, 398

Stack (Of T), 412

StringCollection, 400

StringDictionary, 400

Common Type System, 89-93

conditional code blocks, 172-175

constants, 175-176

DataSets, 539, 543-547

data types

converting between reference
types/value, 111-119

fundamental types, 125-155

dates, applying, 137-143

delegates, 379

combining, 382-383

declaring, 380-382

overview of, 379-383

differences between 4.0/4.5, 3

documentation, 55-59

Entity Framework (ADO.NET)

EDMs, 630-643

overview of, 629-630

enumerations, 305, 315-321

error handling, 207

- events, 379
 - customizing, 389-391
 - handling, 383-385
 - registering, 383-384
- exception handling, 207-208, 224
- files
 - modifying, 460-463
 - permissions, 463
- Finalize method, 271-272
- GAC, 1221
- generics, 367
 - formatting, 368-375
 - overview of, 367-368
- GUIDs, applying, 147-148
- inheritance, 323-324
 - accessing base classes members, 337-341
 - applying, 324-327
 - conditioning, 334-337
 - constructors, 341-342
 - customizing exceptions, 344-346
 - overriding members, 331-334, 343-344
 - polymorphism, 329-331
 - shadowing, 342-343
 - System.Object class, 328-329
- interfaces, 347, 355-365
 - CLS, 354-355
 - defining, 347-348
 - implementing, 348-352
 - inheritance, 353-354
 - polymorphism, 352-353
- I/O file operations, 1137-1141
- iterations, 166-170
- iterators, 393, 414-422
- lambda expressions, 526-533
- LINQ. *See also* LINQ
 - architecture, 554
 - examples, 551-552
 - language support, 552
 - LINQ to Objects, 557
 - overview of, 549-551
 - providers, 553-554
- loops, 95-172
- methods, 64. *See also* methods
- multi-targeting, 18
- multithreading, 1057, 1060-1061
- My namespace, 477-478. *See also* My namespace
- namespaces, 283-284
- new features, 7-8
- nullable types, 367, 376-377
- objects
 - advanced garbage collection, 279-281
 - allocating memory, 269-270
 - garbage collection, 270-271
 - resurrecting, 278-279
 - structures, 65
- operators, 155-165
 - arithmetic, 155-157
 - assignment, 157-158
 - bitwise, 160-162
 - comparison, 163-165
 - concatenation, 163
 - logical, 158-159
 - shift, 162-163
 - short-circuiting, 159-160
- optimization, 272
- overview of, 1-3
- partitioning, 1085-1086
- primitive reference types, 105-106
- processes, 1057-1060
- programming languages, 6-7
- reflection, 1143-1157
- serialization, 1035
 - ADO.NET, 1053-1054
 - customizing, 1045-1048
 - objects, 1036-1042
 - WCF, 1050-1053
 - XAML, 1048-1050
- starting, 2
- streams, 464-475
- strongly typed objects, 104-105
- structures, 305
 - allocating memory, 309
 - assigning to variables, 308
 - CLS, 314
 - implementing interfaces, 309
 - inheritance limitations, 309
 - managing, 310
 - member visibility, 308

- overloading operators, 310-314
- overview of, 305-308
- passing to methods, 308
- System.IO.Directory class, 455-458
- System.IO.DirectoryInfo class, 458-459
- System.IO.DriveInfo class, 459
- System.IO.File class, 460-461
- System.IO.FileInfo class, 462-463
- System.IO.Path class, 454-455
- threads
 - pools, 1061-1063
 - synchronization, 1063-1066
- time, applying, 143-144
- TimeZone type, 144-147
- TimeZoneInfo type, 144-147
- tools, 7
- types, managing within namespaces, 283-295
- value types
 - conversion operators, 120-125
 - differences between reference types/value types, 106-111
 - primitive, 93-94
- versions, selecting, 18
- Windows 8, 8
- WinRT, 8
- With..End With statements, 176-177
- NetMsmqBinding, 1005
- NetNamedPipeBinding, 1005
- NetPeerTcpBinding, 1005
- NetTcpBinding, 1005
- NetTcpContextBinding, 1005
- NetworkAvailabilityChanged event, 509
- Network property, 482, 487
- networks
 - streams, 474
 - Visual Basic 2012 tools, 1359
- NetworkStream class, 474
- New Event Handler command, 706
- new features
 - .NET Framework, 7-8
 - Silverlight, 898
 - Silverlight applications, 928
 - Visual Studio 2010, 11-12
 - WPF, 695
- NewGuid method, 147
- New Interface dialog box, 425
- New Item command, 30
- New keyword, 67, 136, 252
 - constraints, 373
 - structures
- New method, 255
 - System.Object class, 328
- New Project dialog box, 21, 854, 1263
- New Project window, 17-19
- New Solution Configuration window, 42
- New With statement, 525
- NextNode property, 674
- nongeneric collections, 394-403, 406
- NonSerialized events, 1042
- Northwind databases, 589. *See also* databases
 - data sources, adding, 912-913
 - data validation, 614
 - LINQ to DataSets, 622
 - tables, 590
- notation, JSON, 1015
- notifications
 - properties, modifying, 387
 - push, 959
 - tiles, 959
- NotImplementedException, 436
- NotInheritable keyword, 303, 335-336
- Not operator, 158-162
- NotOverridable keyword, 334
- NuGet, managing libraries, 1283-1286
- nullable arguments, 241
- nullable types, 101, 367, 376-377, 553
- NullReferenceException, 1101
 - Field (Of T) method, 626
- null strings, checking for, 128
- numbers
 - binary, 160-162
 - dates, applying, 137-143
 - floating-point, 156
 - time
 - applying, 143-144
 - zones, 144-147

O

Object Browser tool, applying, 394

Object Browser window, 56-59

 System.Object class, 91

ObjectContext class, 916

object-oriented programming. *See* OOP

objects

 adding

 databases, 655

 Visual Studio Class Designer, 425-428

 Application, 719-721

 CollectionViewSource, 830

 COM

 applying, 1192-1196

 exposing, 1197-1199

 comparison operators, 164-165

 complex, generating, 437-439

 contexts, 645

 databases, adding, 655

 data types, 90

 DeflateStream, 467

 EDMs, 630

 events, 379. *See also* events

 FlowDocument, 799

 Generate from Usage feature, 433-439

 GZipStream, 467

 initializers, 258-259, 437

 applying, 558-559

 LINQ, 553

 JSON, 1015

 lifetimes

 finalizers, 110

 managing, 269

 LINQ to Objects, 557. *See also* LINQ

 overview of, 557-558

 querying in memory objects, 558-568

 managing

 advanced garbage collection, 279-281

 allocating memory, 269-270

 Finalize method, 271-272

 garbage collection, 270-271

 implementing Dispose/IDisposable

 interfaces, 273-278

 resurrecting, 278-279

.NET

 structures, 65

 viewing, 57

POCO, 640

polymorphism, 329-331

serialization, 1036-1042

serialization, copying with, 1038

shortcuts, 477. *See also* My namespace

Startup, modifying, 35

StringBuilder, 136

StringComparison, 127

strongly typed, 104-105

System.Timers.Timer, 383

visualizers, inspecting with debuggers,
192-193

Visual Studio Class Designer, 424-432

WithEvents keyword, declaring with,
384-385

XAttribute, 672

XCDATA, 672

XComment, 672

XContainer, 672

XDeclaration, 672

XDocument, 672

XDocumentType, 672

XElement, 672

XName, 672

XNamespace, 672

XNode, 672

XText, 672

ObservableCollection (Of T) collection, 408-410,
814-818

ObservableCollectionHelper class, 503

ODBC (open database connectivity), 540

OffType method, 519

Of keyword, 368

old CAS-based code, migrating, 1155

OleDb, 540

OnCreateMainForm method, 76

OnDeserialization method, 1041

OneTime mode, 813

one-to-many relationships, 605

OneWay mode, 813

OneWayToSource mode, 813

Online Help, navigating, 56

online templates, accessing, 19-20

OnModelCreating method, 666

on-the-fly code, 1297

- anonymous iterators, 419

- compiling, 5

on-the-fly objects, Generate from Usage feature, 433-439

OOP (object-oriented programming), 225

- polymorphism, 329-331

- types, comparing reference/value, 107-110

OpCodes, 1175

open database connectivity. See ODBC, 540

Open Data Protocol (OData), 1015, 1112

opening

- .NET Framework, 2

- Object Browser, 57

- Quick Launch tool, 59-60

- Visual Studio Class Designer, 424

operating systems

- interacting with, 4

- .NET Framework, 1

- Windows 8. See Windows 8

- Windows Phone, 955. See also Windows Phone

OperationCanceledException, 1078, 1134

operation modes, Garbage Collection, 280-281

operations, implementing services, 1027-1029

operators, 155-165

- And, 158-162, 313, 570

- addition (+), 155, 313

- aggregation, 570-572

- AndAlso, 158-159, 570

- arithmetic, 155-157

- assignment, 157-158

- binary, invoking AsParallel, 1097

- bitwise, 160-162

- comparison, 163-165

- concatenation, 163

- concatenation operators, 583

- conversion, 103, 120-125, 572-574

- division (/), 155, 313

- elements, 583-584

- equality (=), 126, 164, 313, 570

- exponentiation (^), 155, 313

- generation, 574-575

- greater than (>), 164, 313

- greater than or equal to (>=), 164, 313

- grouping, 577-579

- If

- LINQ, 553

- ternary, 533-534

- lIf, 173-174

- inequality (<>), 104, 164, 313, 570

- integer division (\), 155, 313

- Is, 164-165, 570

- IsFalse, 158

- IsNot, 164-165, 570

- IsTrue, 158

- left-shift (<<), 162, 313

- less than (<), 164, 313

- less than or equal to (<=), 164, 313

- LINQ to DataSets, 623

- logical, 158-159

- Mod, 155, 570

- multiplication (*), 155, 313

- Not, 158-162

- Or, 158-162, 313, 570

- ordering, 575-576

- OrElse, 158-159, 570

- partitioning, 584-585

- polymorphism, 330

- precedence, 165

- projection, 568-569

- recursive, 45

- restriction operators, 569-570

- return values, 45

- right-shift (>>), 162, 313

- set, 576-577

- shift, 162-163

- short-circuiting, 159-160

- standard query, 568-585

- standard query operators (LINQ to Entities), 652

- structures, overloading, 310-314

- subtraction (-), 155, 313

- TypeOf, 125, 164-165

- union, 579-582

- Xor, 158-162, 313

optimistic concurrency, 616, 648-650

Optimization tab (Visual Studio 2010), 47

- optimizing
 - collections, generics, 395
 - enabling, 47
 - IntelliTrace, 1329
 - .NET Framework, 272
 - performance (Silverlight applications), 928
 - structures, 310
 - value types, 101
- Optional keyword, 240-241
- optional nullable arguments, 241
- Option Infer directives, 513
- options
 - ClickOnce, 1253-1255
 - compiling, 41-44
 - IntelliTrace, 1329
 - item templates, 1264
 - local installation, running out-of-browser applications, 925
 - role configuration, accessing, 935
 - toolboxes, 1275
 - Visual Studio 2012, 1267-1270, 1289-1299
- Options button (ClickOnce), 1253-1255
- Options dialog box, 513
- Option Strict settings, 113
 - arrays, 149
 - LINQ to XML, 682
 - reflection, 1177
- Oracle, 540
- OrderBy clause, 916
- OrderBy method, 519
- OrderByDescending method, 519
- OrderDescription property, 309
- OrderedDictionary collection, 399
- ordering
 - operators, 575-576
 - sequences (PLINQ), 1096
- OrElse operator, 158-159, 570
- orientation, Windows Phone applications, 963-966
- Or operator, 158-162, 313, 570
- out-of-browser applications (Silverlight applications), 923-926
- out of scope objects, 271
- output
 - debugging, 198
 - to projects (InstallShield), adding, 1236

- Output window, 31-33
 - compilation process results, viewing, 40
 - debugging, 194
- overflow checks, removing integers, 47
- overhead, performance, 222
- overlapping catch blocks, 45
- overloading
 - constructors, 255-257
 - CType, 312-314
 - extension methods, 523
 - methods, 242-246
 - operators, structures, 310-314
 - parameters, 245
 - properties, 246
 - types, parameters, 375
- Overloads keyword, 255, 334
- Overrides keyword, 332
- overriding
 - members
 - deriving, 334
 - inheritance, 331-334
 - shared members, 343-344
- OverwriteCurrentValues, 616
- overwriting files, 197-199

P

- packages
 - InstallShield, 1234, 1244
 - MSDeploy, 886
 - Silverlight applications, 899
 - Visual Studio 2012 extensibility, 1289-1299
 - VSIX, 1300
 - web applications, installing, 888-891
- Page class, 856
- pages. *See also* web pages, 852, 856, 1249
 - events, 856
 - lifetimes, 856
 - master, 865-868
 - requests, 852
 - server-driving paging, Data Services (WCF), 1033-1034
 - Windows Phone applications, 963-966

Panel control, 859

panels

- Canvas, 714

- controls, managing, 709-716

- DockPanel, 714

- Grid, 709-711, 1261

- StackPanel, 711-712

- ViewBox, 716

- WrapPanel, 713-714

panorama controls, Windows Phone

- applications (apps), 974-980

Parallel class, 1071

parallel classes, 1071

parallel computing, 1069-1071

- concurrent collections, 1087-1092

- exception handling, 1075-1077

- loops, 1080-1086

- tasks

- applying, 1072-1080

- debugging, 1086

ParallelEnumerable class, 1097

Parallel.ForEach method, 1080-1084

Parallel.For method, 1080-1083

Parallel.Invoke method, 1072

Parallel JIT Compilation, 5

Parallel LINQ. *See* PLINQ

ParallelLoopState class, 1085

ParallelOptions class, 1071

Parallel Stacks window, 1087

Parallel Tasks window, 1086

ParamArray arguments, 239-240

ParameterizedThreadStart delegate, 1061

parameters

- attributes, types, 1185

- overloading, 245

- passing, 1061

- types, 233, 375

paramref tag (XML comments), 1213

param tag (XML comments), 1213

para tag (XML comments), 1213

Parse method, 100, 139

- enumerations, 318-319

parsing XML documents, 674

partial classes, 248-251

Partial keyword, 614

partial methods, 251-252

ParticipantCount property, 859

ParticipantsRemaining property, 1080

partitioning

- .NET Framework, 1085-1086

- operators, 584-585

Pascal, 7, 355

passing

- arguments by value, 238

- arrays, 239

- empty strings, 221

- events, 387-389

- interfaces as method arguments, 351

- parameters, 1061

- reference types, 111, 238

- structures

- allocating memory, 309

- APIs, 310

- values, 238

PasswordBox control, 741

passwords

- formatting, 1226

- saving, 1226

pathnames

- exception handling, 459

- modifying, 453-459

- System.IO.Path class, 454-455

PathTooLongException, 459

patterns

- Async, 1107-1112. *See also* Async pattern

- canceling, 1131-1134

- documentation, 1126

- MVVM, 443

- regular expressions, 999

Peek method, 398

performance

- ASP.NET, 852-853

- exceptions, catching, 217

- generics, 368

- iterators, 416

- Launch Performance Wizard, 1320

- LINQ queries, measuring, 1093-1095

- overhead, 222

- PLINQ queries, measuring, 1095

- Silverlight applications, 928
- types, comparing reference/value, 110
- Visual Basic 2012 tools, 1359
- permissions
 - assemblies, 1149
 - files, 463
 - folders, configuring, 881
 - implementing, 1149
 - requirements, documentation, 1219
 - Silverlight applications, 926
 - tags (XML comments), 1214
- PhoneCallTask, 967, 972
- PIAs (Primary Interoperability Assemblies), 86-87
- PictureBox control, 499
- Pictures Hub, Windows Phone applications (apps), 982-984
- P/Invoke (Platform Invoke), 4
 - COM, 1200
 - converting types to unmanaged, 1202-1203
 - handling exceptions, 1205-1206
 - encapsulating, 1201-1202
 - Interop Assistant, 1358
 - Silverlight, 928
- PivotViewer control, filtering data with, 920-923
- Plain Old CLR Object. *See* POCO, 640, 1053
- platforms, Windows Azure, 929-931. *See also* Windows Azure
- playing
 - audio, 485
 - media, 795-798
 - Silverlight applications, 900-905
 - Windows Phone, 959
- PLINQ (Parallel LINQ), 1092
 - exception handling, 1100-1101
 - queries
 - managing, 1097
 - measuring performances, 1095
 - sequences, ordering, 1096
- POCO (Plain Old CLR Object), 640, 1053
- PointAnimation, 905
- policies, CAS, 1149
- polymorphism, 329-331, 352-353
- pools, threads, 1061-1063
- portable classes, creating, 440-451
- Portable Class Libraries, 7, 226, 440-451
- portable libraries, 423
- Ports property, 483
- positioning breakpoints, 185
- Position method, 464
- postback events, 856-857
- post-conditions, contracts, 1353-1354
- PostgreSQL, 541
- Power Tools (Visual Studio 2010), 29
- precedence, operators, 165
- preconditions, contracts, 1352-1353
- PreRender event, 856
- prerequisites, LINQ to SQL, 588
- Prerequisites dialog box, 1252
- PresentationCore, 696
- PresentationFramework, 696
- PreviousNode property, 674
- Primary Interoperability Assemblies. *See* PIAs, 86-87
- primary output, adding (InstallShield), 1236. *See also* output, 198
- primitive reference types, 105-106
- primitive value types, 93-94
- Print method, 194
- PrintString method, 237
- private constructors, 257
- Private qualifiers, 225, 234, 308
- private variables, 265
- privileges (Visual Studio 2012), 933
- probing assemblies, 1144
- procedures
 - LINQ to SQL, mapping, 610-613
 - stored, mapping, 654-657
- processes, 1057
 - debugging, 180-182
 - executing, 5
 - managing, 1058-1060
 - queries, 1059-1060
- ProcessStartInfo class, 1058
- products, saving databases, 607
- Profiler, 1310
- programming
 - asynchronous, 1103
 - APM, 1106-1107
 - Async pattern, 1107-1112, 1120-1122
 - callbacks, 1116-1120

- EAP, 1104-1106
- event-based asynchrony, 1116-1120
- overview of, 1104
- synchronous approach to, 1112-1116
- task-based asynchrony, 1127-1131
- attributes, 1183. *See also* attributes
- ClickOnce, accessing, 1257-1258
- parallel computing, 1069. *See also* parallel computing
- TDD, 1344-1349
- programming languages, 6-7. *See also* languages
- programming models, Windows Phone applications (apps), 958-959
- progress, reporting Async patterns, 1134-1136
- ProgressBar control, 743
- ProgressChanged event, 1106, 1134
- projection operators, 568-569
- project-level default Imports directives, 294
- projects
 - compiling, 39-48
 - Console Application project template, 22
 - data services, adding, 1016
 - InstallShield, 1229
 - creating, 1232-1242
 - formatting, 1242
 - items, adding, 30
 - My Project. *See* My Project
 - reusing, 22
 - searching, 24
 - Silverlight
 - creating with Visual Basic 2012, 895-897
 - Windows Azure, 936-942
 - Solution Explorer windows, 28-30
 - strong names, adding, 1225
 - TDD, 1344-1345
 - templates, 17-18. *See also* templates
 - exporting, 1261-1263
 - WCF, 993
 - WPF, 698-697
 - upgrading, 16
 - Visual Basic 2012, 63
 - accessing members, 67-68
 - Application.myapp file, 75-76
 - application settings, 81-83
 - AssemblyInfo.vb file, 76-77
 - attributes, 69-70
 - classes, 64
 - code files, 72-83
 - creating, 16-18
 - implicit line continuation, 70-71
 - Imports directives, 68-69
 - inheritance, 65-66
 - methods, 64-65
 - modules, 65
 - namespaces, 66-67
 - navigating My Project, 72-75
 - properties, 64
 - references, 83-88
 - Region directives, 69
 - reserved keywords, 72
 - resources, 77-81
 - structures, 65
 - unreserved keywords, 74
 - Visual Studio 2010, applying, 16-27
 - Windows Azure, creating, 933-944
 - Windows Forms, adding, 1193
- properties
 - Access, 428
 - accessing, 45, 232
 - AllowMultiple, 1187
 - App.Current, 924
 - Audio, 482
 - auto-implemented, 229-231
 - Background, 717
 - BoundedCapacity, 1091
 - brushes, applying to, 758
 - Cancel, 496
 - CanFilter, 832
 - CanGoBack, 965
 - CanGoForward, 965
 - CanGroup, 832
 - CanSort, 832
 - CanUserAddRows, 818
 - CanUserRemoveRows, 818
 - CanUserReorderColumns, 818
 - CanUserResizeColumns, 818
 - CanUserResizeRows, 818
 - CanUserSortColumns, 818
 - Capacity, 394

- classes, 229-234
- Clipboard, 482-485
- ColumnGap, 803
- Content, 726
- contracts, configuring, 1350-1351
- controls, binding, 813
- Converter, 840
- Count, 410
- Counter, 343
- Culture, 479
- Current, 720
- CurrentCell, 818
- CurrentDeployment, 1257
- CurrentItem, 832
- CurrentPhaseNumber, 1080
- CurrentPosition, 832
- customizing, 1301
- Data, 214
- DataContext.Log, 613
- default, 233-234
- Dispatcher, 720
- ElementName, 812
- Embed Interop Types, 87
- ErrorCode, 1205
- ErrorDialog, 1058
- FileSystem, 482-484
- First, 410
- FirstNode, 674
- generating, 437
- Help, 214
- IAsyncResult.AsyncState, 1107
- Info, 483, 488-490, 720
- InnerException, 214, 1101
- IsAddingCompleted, 1091
- IsCompleted, 1075, 1091
- IsNetworkAvailable, 487
- IsNetworkDeployed, 480
- IsRunningOutOfBrowser, 924
- ItemSource, 823
- Keyboard, 482
- Last, 410
- LastEdit, 1185
- LastName, 534
- LastNode, 674
- LineHeight, 899
- LoadedAssemblies, 479
- LoadedBehavior, 797
- Local, 667
- MainWindow, 720
- MaxValue, 101, 137, 143-144
- Message, 214
- MinValue, 101, 143-144
- modifying, 387
- Mouse, 482
- My.Application, 477
 - applying, 478-482
 - cultures, 479-480
 - deploying, 480-482
 - extending, 504-505
 - retrieving information, 478-479
- My.Computer, 478, 482-490
 - Audio property, 485
 - Clipboard property, 484-485
 - extending, 505-506
 - file systems, 483-484
 - Keyboard property, 485
 - Name property, 488-490
 - Network property, 487
 - Registry property, 486-487
- My.Resources, 497-500
- My.Settings, 478, 490-496
 - application-level only settings, 492-493
 - events, 495-496
 - extending, 506
 - naming configuration files, 493-495
- My.User, 478, 500-502
- My.WebServices, 478, 502
- Name, 428
- NavigateUri, 806, 910
- Navigation Properties, EDMs, 640
- Network, 482
- NextNode, 674
- OrderDescription, 309
- overloading, 246
- ParticipantCount, 859
- ParticipantsRemaining, 1080
- Ports, 483
- PreviousNode, 674
- QuantityPerUnit, 643
- read-only, 231

- Registry, 482
- Resource, 720
- RowStyle, 818
- rules, 267
- scalar, 643
- Screen, 483
- SelectedItem, 818
- Settings, defining, 83
- ShowGridLines, 711
- ShowInTaskBar, 717
- ShutdownMode, 720
- Source, 214, 965
- StackTrace, 54, 214
- StartupUri, 720
- Stretch, 794
- stubs, generating, 1347
- System.Drawing.Bitmap, 498
- System.Exception class, 214-216
- TargetSite, 214
- Task.Result, 1074
- Test, 340
- Title, 717
- TopMost, 717
- UICulture, 479
- View, 830-835
- Visual Basic 2012, 64
- Windows, 717, 720
- Windows Phone, customizing, 986
- WindowStartupLocation, 717
- WindowState, 717
- WindowStyle, 717
- write-only, 231-232
- Properties window, 31, 642, 702
- PropertyChanged event, 387, 495
- Protected Friend members, inheritance, 327
- Protected Friend modifier, 225, 234
- Protected members, inheritance, 327
- Protected modifier, 225, 234
- protocols
 - FTP, 884
 - HTTP, 8, 991
 - OData, 1015, 1112
 - SOAP, 1039-1040
 - TCP, 991

- providers (LINQ), 553-554
- providers, .NET data, 540-541
- proxy classes (WCF), 1002
- public keys (GAC), 1224
- public modules, 303. *See also* modules
- Public qualifiers, 308
 - inheritance, 327
 - interfaces, 348
 - modifiers, 225, 234
- Public Shared Operator statement, 311
- Publish.htm web page, 1249
- publishing
 - ASP.NET applications, 883-885
 - extensions, 1302
 - MSDeploy, 886-891
 - Windows Phone Marketplace, 956
- Publish Web dialog box, 887
- Publish Windows Azure Application wizard, 944
- Publish Wizard, 1247
- purpose of namespaces, 287-288
- push notifications, 959

Q

- qualifiers
 - Friend, 308, 348
 - Private, 308
 - Public, 308, 348
 - writing, 434
- quantifiers, 582-583
- QuantityPerUnit property, 643
- queries
 - Code First models, 661
 - databases, 542
 - Data Services (WCF), 1026
 - DataSets, 547
 - EDMs
 - with LINQ to Entities, 652-653
 - with SQL, 653-654
 - expressions, 551
 - interceptors, applying, 1030-1033

- LINQ, 565. *See also* LINQ
 - anonymous types, 524
 - measuring performances, 1093-1095
 - standard query operators, 568-585
- LINQPad, 1359
- LINQ to DataSets, 621-624
- LINQ to SQL, 600-604
- LINQ to XML, 676-677, 680-682
- memory, objects, 558-568
- PLINQ
 - managing, 1097-74
 - measuring performances, 1095
 - processes, 1059-1060
 - thread pools, 1063
 - via HTTP requests, 1014-1015
- QueryInterceptor attribute, 1030
- QueryVideosAsync method, 1132
- Queue collection, 397-398
- Queue (Of T) collection, 412
- Queues Storage, 931
- Quick Launch tool, 59-60
- Quick Watch window, 191

R

- RadialGradientBrush, 758, 762
- RadioButton control, 744, 859
- RadioButtonList control, 859
- raising events, 385-389
- RangeValidator control, 859
- ReadAllText method, 160
- read-and-write operations, 229
- ReadAsync method, 1138
- ReadByte method, 464
- reading
 - binary files, 465
 - text, 464-465
- ReadLineAsync method, 1138
- Read method, 464
- ReadOnlyCollection (Of T) collection, 406-407
- ReadOnly keyword, 231
- read-only properties, 231
- ReadToEndAsync method, 1138
- read/write locks, 1065-1066
- ready-to-use themes, 14
- really simple syndication. *See* RSS
- recent templates, accessing, 19-20
- Rectangle control, 745
- recursive clones, 118
- recursive operators, 45
- ReDim keyword, 150-151
- refactoring code, 1349
- ReferenceEquals method, 328
- Reference Manager dialog box, 84-85, 936
- references
 - COM libraries, adding, 85-86
 - Data Services, adding to, 1023
 - delegates, 382
 - GAC, 1227
 - items, specifying in, 1266
 - reference types, passing, 238
 - services, adding, 1002
 - Solution Explorer, 85
 - types, 90-93
 - applying, 103-106
 - conversion operators, 120-125
 - primitive, 105-106
 - value types
 - converting between, 111-119
 - differences between, 106-111
 - Visual Basic 2012 projects, 83-88
 - Win32 API calls, 1206
- Reflection, 500
- reflection, 1143-1157
 - assemblies
 - Caller Information, 1177-1180
 - metadata, 1158-1160
 - retrieving information from, 1160-1162
 - attributes, 1189-1190
 - dynamic code, running, 1169-1171
 - late binding, 1176
 - overview of, 1157
 - security, 1169
 - System.Reflection.Emit namespace, 1171-1177
 - types, 1162-1169
- ReflectionOnlyLoad method, 1160
- ReflectionOnlyLoadFrom method, 1160

RefreshMode enumeration members, 616

regional settings, 844

Region directives (Visual Basic 2012), 69

registering

assemblies for COM interoperability, 1197

Developer Portal (Windows Azure), 931

for events, 383-384

garbage collection events, 281

Marketplace (Windows Phone), 956

Registration-Free COM (ClickOnce), 1258-1259

Registry

Application Registry (InstallShield), 1239

modifying, 486-487

Registry property, 482

regular expressions, 251-252

patterns, 999

relationships

LINQ to SQL, 593

one-to-many, 605

relaxed delegates, 526

Release configuration (Visual Studio 2012), 40-43

releasing COM objects, 1195

remarks tag (XML comments), 1214

RemoveFirst method, 410

RemoveHandler keyword, 383-384

RemoveLast method, 410

RemoveMemoryPressure method, 280

Remove method, 410, 660, 674

RemoveNodes method, 674

RemoveParticipant method, 1080

RemoveParticipants method, 1080

removing. *See* deleting, 47, 457, 1270

ReplaceWith method, 674

replacing

Default.Aspx pages, 950

Latest News tabs, 14

reporting progress, Async patterns, 1134-1136

REpresentational State Transfer.

See REST, 1014

reproduction of content, 901

requests

asynchronous, 853

HTTP

GET, 1030

querying data via, 1014-1015

pages (ASP.NET), 852

Required attribute, 650, 664

RequiredFieldValidator control, 859

requirements

classes for COM exposure, 1197

InstallShield installations, 1234-1236

LINQ to SQL, 588

memory, value types, 94

permissions, 1149, 1219

system, Local DB (SQL Server 2012), 539-540

ReRegisterForFinalize method, 279

reserved keywords

Await, 1109

My namespace. *See* My namespace

Visual Basic 2012 projects, 72

resizing

elements, 780

windows, 711

ResourceManager class, 80, 500

Resource property, 720

resources, 8

adding, 498

code, accessing by name, 500

localization, editing, 848

mapping, 500

My.Resources property, 497-500

specifying links to, 1219

unlocking, 211

Visual Basic 2012, 77-81, 1357-1358

WPF, 497-500

Resources Designer, 497

Resources.resx file (Visual Basic 2012), 77-81

REST (REpresentational State Transfer), 1014

restriction operators, 569-570

restyling windows, 777. *See also* styles

results

of build processes, viewing, 31

Code Coverage, enabling, 1343

IntelliTrace, 1331-1333

serialization, 1037

value types, assigning, 99

ResurrectDemo class, 279

resurrecting objects, 278-279

ResurrectionDemo type, 279

rethrowing exceptions, 219

- retrieving
 - command-line arguments, 482
 - environment variables, 481
 - Return instruction, 560
 - return methods, 319-320
 - returns tag (XML comments), 1214
 - Return statement, 319
 - return types, array literals, 515
 - return values, 45, 1074-1075
 - ReturnFullName method, 243
 - reusing
 - code
 - generics, 368
 - snippets, 1275-1283
 - data types, 90
 - projects, 22
 - templates, 1246
 - Reverse method, 396, 519
 - RIAs (rich Internet applications), 893. *See also* Silverlight applications; Visual Basic 2012
 - RichTextBox control, 806-808
 - right-shift (>>) operator, 162, 313
 - roles
 - configuring, 881
 - web, 934-936
 - Root method, 674
 - root namespaces, 34, 291-292
 - RotateTransform, 780
 - rotating elements, 780
 - round-tripping projects, 16
 - routing
 - events, 707-708
 - strategies, 708-709
 - RowStyle property, 818
 - RSS (really simple syndication)
 - asynchronous programming, 1112-1116
 - Latest News tab (Visual Studio 2012), 14
 - loading, 1116
 - reading view, closing, 1018
 - rules
 - arrays, 267
 - classes, 266
 - methods, 267
 - Microsoft code analysis, 1311
 - properties, 267
 - security, Transparency Level 2, 1150-1152
 - Run method, 720
 - running
 - applications, 875
 - assemblies, 4-5
 - .NET Framework, 2. *See also* .NET Framework
 - dynamic code, 1169-1171
 - out-of-browser applications, 925
 - sandboxed assemblies, 1154
 - tasks, 1073-1074
 - unit tests, 1342-1343, 1348
 - WCF RIA Services, 920
 - runtime
 - assemblies, 5
 - code, generating at, 1171
 - Code Contract, 1350-1355
 - dynamic code, creating at, 1147
 - errors, debugging, 53-54
 - windows, instantiating at, 718
 - WinRT, 8, 1191
 - Runtime Callable Wrapper, 1192
 - Run to Cursor command, 181
- ## S
- samples, searching code, 21
 - sandboxed models, 1149, 1152-1154
 - SaveChanges method, 833
 - saving
 - cookies, applying, 863
 - passwords, 1226
 - products to databases, 607
 - SByte keyword, 95
 - scalability (ASP.NET), 852-853
 - scalar properties, 643
 - ScaleTransform, 780
 - scaling loops, 1080

scheduling

tasks, customizing, 1070

Task Scheduler, 1070

schemas

EDMs, 638

XML

adding, 1112

interfaces, 685-690

schemes, modifying databases, 664-665

SciTech Memory Profiler, 1359

scope

assemblies, 1144

classes, 234-235

of fields, 228

interfaces, 348

iterators, 414

levels (Visual Basic 2012), 234

local type inference, 514

members, 327

modules, 302

multiple, 8

namespaces, 291

out of scope objects, 271

Screen property, 483

ScrollView control, 746

SDKs (Software Development Kits), 7

Visual Studio 2012, 1288-1289

Windows Phone, 957-958

searching

Bing, 972

code

clones, 1310, 1317-1319

samples, 21

files, 3

IronPython, 6-7

IronRuby, 6-7

probing, 1144

projects, 24

templates, 20

tools, 1360

SearchTask, 967, 972

sections of EDMs, 633

security

ASP.NET applications, configuring, 879-882

assemblies (.NET 4.5), 1148-1155

attributes, 1151

CAS, 1148

ClickOnce, 1255

GAC, 1224, 1227

managed code, writing, 4

permissions, 463

reflection, 1169

Silverlight applications, 926

Transparency Level 2, 1150-1152

SecurityCritical attribute, 1151

SecurityException, 463

SecurityRules attribute, 1151

SecuritySafeCritical attribute, 1151

SecurityTransparent attribute, 1151-1152

Security Zones, 4

Seealso tag (XML comments), 1214

Seek method, 464

See tag (XML comments), 1214

SEHException, 1205

Select Case statement, 174-175

SelectedItem property, 818

selecting

code editor extension templates, 1305

components (Windows Media Player), 86

default types, 493

interfaces, 348

locations (ClickOnce), 1247

.NET Framework versions, 18

objects, EDMs, 632

Silverlight applications, 721

types, customizing, 111

SelectionBrush, 758, 765-767

Select keyword, 553

SelectMany method, 519

Select method, 519

Select Tool window, 1291

Separator control, 746

SequenceEquals method, 519

sequences, ordering PLINQ, 1096

Serializable attribute, 70, 1182

- serialization, 117, 1035
 - ADO.NET, 1053-1054
 - assemblies, generating, 48
 - binary, 1036-1038
 - customizing, 1045-1048
 - events, 1047-1048
 - JSON, 1053
 - objects, 1036-1042
 - SOAP, 1039-1040
 - WCF, 1050-1053
 - XAML, 1048-1050
 - XML, 1043-1045
- SerializationException, 1037
- Serialize method, 1036
- server-driving paging, Data Services (WCF), 1033-1034
- Server Explorer, 589
- servers, 929
 - controls, 858
 - IIS, configuring, 994
 - WCF service operations, implementing, 1027-1029
- ServiceContract, 993, 996
- services
 - annotations, implementing, 804
 - cloud. *See* cloud services, 930, 946
 - Data Services (WCF)
 - consuming, 1022-1027
 - implementing, 1013, 1016-1021
 - overview of, 1013-1015
 - Domain Service Class, adding, 913-916
 - locations (Windows Phone), 959
 - members, invoking from, 1003-1007
 - metadata, exporting, 999
 - operations, implementing, 1027-1029
 - references, adding, 1002
 - Storage Account, 930
 - WCF, 991. *See also* WCF
 - consuming, 1001-1007
 - implementing, 993-1001
 - overview of, 992
 - WCF RIA Services (Silverlight applications), 911-923
 - web, 502
- SessionEnding event, 510
- Session state, 864
- SetAudio method, 484
- SetCreationTime method, 455
- SetCurrentDirectory method, 458
- SetData method, 484
- SetDataObject method, 484
- SetField (Of T) method, 626-627
- SetImage method, 484
- SetLastError method, 1206
- Set Next Statement, 181
- set operators, 576-577
- SetText method, 484
- SettingChanging event, 495
- settings. *See also* configuring
 - applications (Visual Basic 2012), 81-83
 - Boolean user-level, 491
 - cloud services, 946
 - contracts, properties, 1350-1351
 - Domain Service Class, 913
 - existing, importing, 1273
 - Export Settings Wizard, 1271-1273
 - InstallShield, 1229. *See also* InstallShield
 - listeners, configuration files, 200-202
 - manual package generation, 890
 - MSDeploy, 889-888
 - My.Settings property, 490-496
 - Option Strict, 113
 - arrays, 149
 - LINQ to XML, 682
 - reflection, 1177
 - out-of-browser, enabling, 923
 - project-level default Imports directives, 294
 - regional, 844
 - StartState, 83
 - users, managing, 1271-1273
 - Visual Studio 2012 projects, 1289-1299
 - Windows Azure subscriptions, 944
- Settings command, 433
- Settings Designer, 492
- SettingsLoaded event, 495
- Settings property, defining, 83
- SettingsSaving event, 495
- Settings.settings file, 83
- Settings tab (My Project), 82
- setup. *See* configuring; settings
- shadowing inheritance, 342-343

shallow copy, converting between reference/
value types, 116-117

shapes (Microsoft Design Style), 12

Shared keyword, 259

ShareLinkTask, 967, 973

ShareStatusTask, 967, 973

sharing

assemblies, 853

classes, 259

constructors, 262-263

fields, 259

libraries with NuGet, 1286

members, 45

classes, 259-263

generating, 436

overriding, 343-344

methods, 260-262

properties, 260

shift operators, 162-163

short-circuiting operators, 159-160

shortcuts

Application Shortcuts group (InstallShield),
1238-1239

objects, 477. *See also* My namespace

Short keyword, 95

Show Data Sources command, 917

ShowDialog method, 755

ShowGridLines property, 711

ShowInTaskBar property, 717

Show Next Statement, 182

Show Output From combo box, 33

ShutDown event, 509

Shutdown method, 720

ShutdownMode property, 720

signing

assemblies, 1145

GAC assemblies with strong names,
1224-1226

Silverlight applications, 55, 699, 893

3-D graphics, 908

controls, 726, 897-900

data sources, adding, 912-913

deploying, 899

drag'n'drop data-binding, 916-919

event handling, 899

media, playing, 900-905

new features, 898, 928

ObservableCollection (Of T) collection, 407

out-of-browser applications, 923-926

overview of, 894

packages, 899

permissions, 926

security, 926

selecting, 721

Silverlight Navigation Applications, 908-910

UI elements, animating, 905-908

views, defining, 449

Visual Basic 2012, creating with, 895-897

WCF RIA Services, 911-923

Windows Azure, building projects, 936-942

Windows Phone, 959

XAML, 700, 926-927

simple iterators, 417-418

Simple Object Access Protocol. *See* SOAP,
1039-1040

SimulateProcessing method, 1081

Single keyword, 95

Single method, 519, 1025

SingleOrDefault method, 519

single types, reflecting, 1167-1169

sizing

controls, 711

windows, 711

SkewTransform, 780

Skip method, 519

SkipWhile method, 519

SmsComposeTask, 967, 972

snippets, code

deploying, 1300

reusing, 1275-1283

SoapFormatter class, 1039

SOAP (Simple Object Access Protocol)
serialization, 1039-1040

Software Development Kit. *See* SDK

SolidColorBrush, 758-760

Solution Explorer, 24, 589

InstallShield, 1232

references, 85

windows, navigating, 28-30

solutions

Solution Explorer windows, 28-30

Visual Studio 2010, 16-27

- SortedDictionary (Of TKey, TValue)
 - collection, 408
- SortedList collection, 400
- sorting
 - arrays, 152-155
 - data with PivotViewer control, 922
- Sort method, 396
- source code, 5. *See also* code; programming
- Source property, 214, 965
- specialization, exceptions, 209
- specifying links to resources, 1219
- spell check, implementing, 808
- splash screens
 - Windows Phone, 985
- SQL (Server Query Language)
 - code, database objects, 655
 - EDMs, querying with, 653-654
 - entities, applying against, 617
 - LINQ to, 587. *See also* LINQ
 - overview of, 588-599
 - SQL Server Compact Edition 3.5, 617-619
 - logs, applying, 613
- SqlClient.SqlException, 606
- SqlCommand class, 541
- SQL Server 2012 (Local DB), 882
 - system requirements, 539-540
- SQL Server Object Explorer window, 59, 662
- SSDL (Store Schema Definition Language), 636
- Stack, 90
 - boxing, 114
 - reference types, addresses, 108
 - value types, 90, 106-108
- Stack collection, 398
- Stack (Of T) collection, 412
- StackPanel panel, 711-712
- StackTrace property, 54, 214
- standard query operators
 - aggregation operators, 570-572
 - concatenation operators, 583
 - conversion operators, 572-574
 - element operators, 583-584
 - generation operators, 574-575
 - grouping, 577-579
 - Let keyword, 572
 - LINQ, 568-585
 - LINQ to DataSets, 623
 - LINQ to Entities, 652
 - ordering operators, 575-576
 - partitioning operators, 584-585
 - projection operators, 568-569
 - quantifiers, 582-583
 - restriction operators, 569-570
 - set operators, 576-577
 - union operators, 579-582
- starting
 - Data Service (WCF), 1017
 - .NET Framework, 2
 - Object Browser, 57
 - Quick Launch tool, 59-60
 - Visual Studio Class Designer, 424
 - Windows Phone applications, 963-964
- Start method, 1058
- Start Pages (Visual Studio 2010), 12-14
- StartState setting, 83
- Startup event, 509-510
- StartupNextInstance event, 509
- Startup Object field (My Project), 35
- StartupUri property, 720
- state
 - Application, 861
 - Cache, 862-863
 - Context, 579
 - managing, 861-864
 - Session, 864
 - ViewState, 864
- statements
 - breakpoints, 186
 - Catch, 210
 - Class..End Class, 225
 - Console.WriteLine, 181
 - Exit Try, 218
 - If, 181
 - New With, 525
 - Public Shared Operator, 311
 - Return, 319
 - Select Case, 174-175
 - Set Next Statement, 181
 - Show Next Statement, 182
 - SyncLock..End SyncLock, 1064

- Using..End Using, 275
- With..End With, 176-177
- Yield, 419
- Static Checker, 1351
- StaticResource, 772
- status, Windows Azure deployment, 947
- StatusBar control, 747
- status bars (Visual Studio 2012), 12
- Step Into command, 181
- Step Out command, 181
- Step Over command, 181
- StopLoading method, 965
- storage. *See also* cloud computing; Windows Azure
 - tools, 7
 - value types, 90, 96
 - Windows Phone, 980-981
- Storage Account service, 930
- Storage Emulator, 944
- Store (Windows 8) applications, 1111
- stored procedures
 - LINQ to SQL, mapping, 610-613
 - mapping, 654-657
- Store Schema Definition Language.
 - See* SSDL, 636
- Storyboard events, 787
- strategies, routing, 708-709
- Stream class methods, 1138
- streams, 464-475
 - data, compressing with, 467-474
 - members, 461, 464
 - memory, 466
 - modifying, 453
 - networks, 474
 - strings, applying, 467
 - Windows Phone, 960-963
- StreamWriter class, 464
- Stretch property, 794
- StringBuilder objects, 136
- StringCollection collection, 400
- StringComparison object, 127
- StringDictionary collection, 400
- StringLength attribute, 664
- strings, 77
 - applying, 125-137
 - comparing, 126-128
 - comparison operators, 164
 - concatenating, 136-137
 - connection, writing, 619
 - copying, 131
 - dates, converting to, 138-139
 - editing, 106, 133-136
 - formatting, 128-129, 835-840
 - inspecting, 131-133
 - streams, applying, 467
 - stringToPrint argument, 238
 - strongly typed data controls, 875-862
 - strongly typed objects, 104-105
 - strong names, signing GAC assemblies with, 1224-1226
- StructLayout attribute, 310, 1203-1204
- Structure..End Structure block, 305
- Structure keyword, 374
- structures, 305
 - APIs, passing, 310
 - ClickOnce deployments, 1250-1251
 - CLS, 314
 - declaring, 306
 - generating, 437
 - generics, defining, 370
 - inheritance limitations, 309
 - interfaces, implementing, 309
 - managing, 310
 - members, visibility, 308
 - memory allocation, 309
 - metadata, 1158
 - methods, passing to, 308
 - namespaces, 284
 - operators, overloading, 310-314
 - overview of, 305-308
 - variables, assigning, 308
 - Visual Basic 2012 projects, 65
- stubs
 - methods, generating, 436
 - properties, generating, 1347
- Style class, 771
- styles
 - CSS, 856
 - inheritance, 773
 - Microsoft Design Style, 11-12
 - triggers, 773-775
 - WPF, 770-775

- Sub lambda expressions, 531-532
- Sub Main method, 35
- submitting applications (apps) to Marketplace (Windows Phone), 987-989
- subscriptions (Windows Azure), 945
- Substitution control, 859
- subtracting dates, 142-143
- subtraction (-) operator, 155, 313
- Subtract method, 142
- summary tag (XML comments), 1214
- Sum method, 519
- support
 - 3-D graphics, 908
 - 64-bit browsers, 928
 - arrays, 8
 - classes, 301
 - COM Automation, 926
 - delegates, generics, 380
 - foreign keys, 633
 - HTML5, 8
 - IntelliSense, XAML code editors, 701
 - languages, LINQ, 552-553
 - .NET operating systems, 1
 - P/Invoke (Silverlight applications), 928
 - Windows Azure, 928
 - ZIP archives, 8
- SuppressFinalize method, 279
- Suppress Message, 1314
- switching threads, 1128-1129
- symbols
 - dates, formatting, 140
 - format, 129
- synchronization
 - Monitor class, 1065
 - read/write locks, 1065-1066
 - threads, 1063-1066
- synchronous calls, 1108
- SyncLock..End SyncLock statement, 1064
- System.Array class, 105, 152-155
- System.Attribute class, 69
- System.Boolean class, 95
- System.Byte class, 95
- System.Char class, 95
- System.Collections class, 394
- System.Collections.Concurrent namespace, 413
- System.Convert class methods, 123
- System.DataSet.DataSetExtensions namespace, 622
- System.DateTime class, 95, 139
- System.Decimal class, 95
- System.Delegate class, 380
- System.Diagnostics namespace, 197
- System.Diagnostics.Process class, 1058
- System.Double class, 95
- System.Drawing.Bitmap property, 498
- System.Enum class methods, 316-319
- System.Exception class, 105, 209, 214-216, 344
- System.GC class, 270
- System.GC.Collect method, 271
- System.Globalization namespace, 842
- System.Guid class, 95
- System.Int16 class, 95
- System.Int32 class, 95
- System.Int64 class, 95
- System.IntPtr class, 95
- System.IO.Directory class, 455-458
- System.IO.DirectoryInfo class, 458-459
- System.IO.DriveInfo class, 459
- System.IO.IOException, 209
- System.IO.File class, 293, 460-461
- System.IO.FileInfo class, 462-463
- System.IO namespace, 8, 293
- System.IO.Path class, 454-455
- System.IO.Stream class, 105, 464
- System.IProgress (Of T) interface, 1134
- System.Math class, 157
- System.Net.Exception, 488
- System.Numerics.BigInteger class, 95
- System.Object class, 90, 105
 - implicit conversions, 111
 - inheritance, 325-329
 - methods, 92
 - naming, 91
 - WPF, 696
- System.Object.ToString method, 66
- System.Reflection.Emit namespace, 1171-1177
- System.Reflection namespace, 1158

- system requirements, Local DB (SQL Server 2012), 539-540
- System.Runtime.CompilerServices namespace, 1177
- System.Runtime.InteropServices namespaces, 310
- System.SByte class, 95
- System.Security.SecurityException, 1150
- System.SerializableAttribute class, 70
- System.ServiceModel.dll, 6
- System.Single class, 95
- System.String class, 105, 126
- System.Threading.ThreadPool class, 1062
- System.Timers.Timer object, 383
- System.TimeSpan class, 95, 142
- System.TimeZone class, 95
- System.UInt16 class, 95
- System.UInt32 class, 95
- System.UInt64 class, 95
- System.ValueType class, 91-92
- System.Windows.ContentElement class, 57
- System.Windows.Controls namespace, 6
- System.Windows.Media.MediaCommands enumeration, 902
- System.Windows.RoutedEventArgs type, 707
- System.Xml.Linq namespace, 672-674
- System.Xml.Serialization namespace, 1043

T

- TabControl control, 748
- Table attribute, 664
- Table control, 859
- tables, Northwind databases, 590
- Tables Storage, 930
- tabs
 - Application (My Project), 34-39
 - Get Started (Visual Studio 2010), 14
 - IDE, navigating, 50
 - Latest News (Visual Studio 2010), 14-15
 - Optimization (Visual Studio 2010), 47
 - Settings (My Project), 82
- tabular data forms, formatting, 819-825
- tags
 - Grid panel, 1261
 - XML comments, 1214
- Take method, 519
- TakeWhile method, 519
- target CPU, 44
- Target CPU combo box, 42
- Target Framework (My Project), 39
- targets, multi-targeting, 18
- TargetSite property, 214
- task-based asynchrony, 1127-1131
- TaskFactory class, 1071
- Task Parallel Library. *See* TPL, 413, 1070
- Task.Result property, 1074
- Task.Run method, 1128
- tasks
 - canceling, 1077-1078
 - debugging, 1086
 - parallel computing, 1072-1080
 - return values, formatting, 1074-1075
 - running, 1073-1074
 - scheduling, customizing, 1070
- Task Scheduler, 1070
- TaskScheduler class, 1071
- task-specific exceptions, 220
- Task.Wait method, 1075
- TCP (Transmission Control Protocol), 991
- TDD (Test Driven Development), 1344-1349
- templates
 - accessing, 19-20
 - ASP.NET, 854-855
 - code editor extension, selecting, 1305
 - Console Application project, 22
 - Domain Service Class, 913
 - exporting, 1261-1265
 - items, exporting, 1263-1265
 - projects, 17-18, 993
 - searching, 20
 - Silverlight applications, 895
 - Visual Studio 2012 extensibility, 1289
 - WCF RIA Service, 912
 - Windows Phone, 957

- WPF
 - controls, 775-778
 - projects, 698-697
 - XML to Schema item, 686
- ternary If operators, 533-534
- LINQ, 553
- TestAccessFile method, 222
- Test class, 227
- Test Driven Development. *See* TDD, 1344-1349
- Test Explorer window, 1342
- testing
 - applications (Windows Azure), 942-944
 - code, 1337
 - applying Code Contracts, 1350-1355
 - TDD, 1344-1349
 - custom extension libraries, 524
 - IntelliTrace, 1334
 - unit tests, 1337-1349
- TestInstance method, 374
- Test method, 181, 187
- Test property, 340
- text
 - aligning, 802
 - customizing, 898
 - Data Annotations (Code First approach), 663-665
 - messages, 972
 - modifying, 799-808
 - output, debugging, 198
 - reading, 464-465
 - RichTextBox control, 806-808
 - spell check, implementing, 808
 - writing, 464-465
- TextBlock control, 749
- TextBox control, 750, 812, 857-859
- TextChanged event, 857
- TextRange class, 807
- TextWriterTraceListener, 196-197
- themes
 - CSS, 856
 - Visual Studio 2012, 13-14
- ThenByDescending method, 519
- ThenBy method, 519
- third-party programming languages, 7
- threads
 - Async pattern/Await keyword, 1110
 - formatting, 1060-1061
 - IDs, 1074
 - localization, 842
 - pools, 1061-1063
 - switching, 1128-1129
 - synchronization, 1063-1066
- thread-safe collections, 1087
- Thread.Start method, 1061
- Threads window, 191-192
- throwing exceptions, 218-220
- Throw keyword, 218-220
- time
 - adding time to, 142-143
 - applying, 143-144
 - zones, 144-147
- TimeZoneInfo type, 144-147
- TimeZone type, 144-147
- Title property, 717
- ToArray method, 396, 519
- ToBool method, 123
- ToByte method, 123
- ToChar method, 123
- ToDateTime method, 123
- ToDecimal method, 123
- ToDictionary method, 519
- ToDouble method, 123
- ToFileTime method, 141
- ToFileTimeUtc method, 141
- ToInt16 method, 123
- ToInt32 method, 123
- ToInt64 method, 123
- ToList method, 519
- ToLocalTime method, 141
- ToLongDateString method, 141
- ToLongTimeString method, 141
- ToLookup method, 519
- ToDate method, 141
- ToolBar control, 750
- toolbars
 - configuring, 1271
 - customizing, 1268-1270
 - deleting, 1270

Toolbox, adding controls, 701

toolboxes, customizing, 1275

tools, 3

- Add-in Manager, 1304

- ASP.NET Administration, 879

- Binary Rewriter, 1351

- Call Hierarchy, 60

- Class View window, 432-433

- code contracts, 1351-1352

- Code Snippet Manager, 1278

- code snippets, generating, 1283

- columns, adding, 710

- design

 - EDMs, 640

 - XAML, 699

- Document Outline, 704

- Extension and Updates, 1302

- Generate from Usage feature, 433-439

- InstallShield. *See* InstallShield

- LocBaml, 841-845

- MSDeploy, 884-891

- My Project, 33-39

- .NET Framework, 7

- NuGet, 1283-1286

- Object Browser, applying, 394

- Parallel Tasks window, 1086

- Quick Launch, 59-60

- searching, 1360

- Server Explorer, 589

- Silverlight applications (Visual Studio 2012), 894

- Solution Explorer, 589

- Static Checker, 1351

- visual, 423

- Visual Basic 2012, 1357

 - coding tools, 1358

 - data access, 1359

 - debugging, 180-192

 - developer tools, 1358

 - diagnostics/performance, 1359

 - JustDecompile, 1359

 - networks, 1359

 - Windows Azure Management Tool, 1360

- Visual Basic Compiler (vbc.exe), 2

Visual Studio 2010

- applying windows, 27-33

- Code Metrics, 1309, 1315

Visual Studio 2012, 1309

- Code Analysis, 1309, 1310-1315

- Code Clone Detection, 1310, 1317-1319

- generating Dependency Graphs, 1334-1335

- IntelliTrace, 1310, 1328-1334

- overview of, 1309-1310

- Profiler, 1310, 1319-1328

- Windows Azure, 931-932

Visual Studio Class Designer, 424-432

Visual Studio Historical Debuggers, 52

windows

- Error List, 30-31

- Output, 31-33

- Properties, 31

- Solution Explorer, 28-30

Windows Phone, 957-958

WPF, 757

- brushes, 757-771

- styles, 770-775

Tools menu (Visual Studio 2012), customizing, 1267-1268

Tool window, implementing, 1296

TopMost property, 717

ToSByte method, 123

ToShortDateString method, 141

ToShortTimeString method, 141

ToSingle method, 123

ToString method, 121, 309

- enumerations, 318-319

- reflection, 1163

- System.Object class, 328

ToUInt16 method, 123

ToUInt32 method, 123

ToUInt64 method, 123

ToUniversalTime method, 141

TPL (Task Parallel Library), 413, 1070. *See also* parallel computing

Trace class, 195

Trace.Listener.Clear method, 197

trace listeners, applying, 196-202

trace points, applying, 184-187

tracing, enabling, 1010

- tracking (IntelliTrace), 1331-1333
- training resources (Windows Phone), 958
- transformations (WPF), 778-782
- TranslateTransform, 781
- Transmission Control Protocol. *See* TCP, 991
- Transparency Level 2, 1150-1152
- transparency models, 1148-1149
- trees (WPF), 704-705
- TreeView control, 751
- triggers, styles, 773-775
- TrimToSize method, 396
- troubleshooting
 - Code Analysis, 1315
 - DLLs, 1221-1222
 - exceptions, 207-208
 - return types, array literals, 515
 - System.Diagnostics namespace, 197
 - unit tests, 1342
 - Visual Basic 2012
 - compiler fixes, 169-170
 - tools, 1359
- trust levels
 - ClickOnce, 1255
 - for roles, 935
- TryCast, 123-125
- Try..Catch..Finally blocks, 1127
 - exception handling, 209
 - iterators, 418-419
- TryParse method, 100, 139
- tunneling strategies, 708
- TwoWay mode, 813
- Type Metadata, 1158
- typeof operator, 125, 164-165
 - polymorphism, 330
- typeparam tag (XML comments), 1214
- types
 - anonymous, 524-525
 - building with LINQ to DataSets, 623
 - LINQ, 552
 - LINQ to SQL, 603
 - applications (My Project), 34
 - arrays, applying, 148-155
 - assemblies, 1144
 - attribute parameters, 1185
 - in BCL, 6
 - CLS, 264
 - COM libraries, adding references, 85-86
 - Common Type System, 89-93
 - conditional code blocks, 172-175
 - constants, 175-176
 - constraints, 373
 - converting between reference types/value, 111-119
 - customizing
 - exposing, 232
 - selecting, 111
 - data, 89. *See also* data types
 - dates, applying, 137-143
 - default, selecting, 493
 - fundamental types, 125-155
 - generating, 437
 - generics, 8, 368. *See also* generics
 - GUIDs, applying, 147-148
 - IEnumerable interface, 358
 - implicit, 45
 - iterations, 166-170
 - literal type characters, 98
 - local
 - inference, 511-514
 - LINQ, 552
 - loops, 95-172
 - MIME, 80
 - namespaces
 - managing, 283-295
 - overview of, 283-284
 - nested, 374
 - nullable, 101, 376-377. *See also* nullable types
 - LINQ, 553
 - OOP, comparing reference/value, 107-110
 - operators, 155-165
 - arithmetic, 155-157
 - assignment, 157-158
 - bitwise, 160-162
 - comparison, 163-165
 - concatenation, 163
 - logical, 158-159
 - shift, 162-163
 - short-circuiting, 159-160

- parameters, 233, 375
- reference types, 90-93
 - applying, 103-106
 - passing references, 238
 - primitive, 105-106
- reflection, 1162-1169
- ResurrectionDemo, 279
- strongly typed data controls, 875-862
- strongly typed objects, 104-105
- System.Windows.RoutedEventArgs, 707
- time, applying, 143-144
- TimeZone, 144-147
- TimeZoneInfo, 144-147
- unmanaged code, converting, 1202-1203
- values, enumerations. *See* enumerations
- value types, 90-93
 - analyzing, 100
 - applying, 92-103
 - assigning, 98-99
 - conversion operators, 120-125
 - customizing, 103
 - differences between reference types/
value types, 106-111
 - methods, 100-101
 - .NET Framework, 93-94
 - optimizing, 101
 - passing values, 238
- visibility, 234-235
- Visual Basic 2012 projects, 63-71
- With..End With statements, 176-177

U

- UAC (User Account Control), 36-39
- UI (user interface), 11, 551. *See also* interfaces
 - data-binding, 812. *See also* data-binding
 - data validation and, 615
 - elements
 - adding to Silverlight applications, 897
 - animating, 905-908
 - localization, 841-850
 - Logical Tree, 704
 - UICulture property, 479
 - UInteger keyword, 95
 - ULong keyword, 95
 - UnauthorizedAccessException, 459
 - unboxing types, converting reference/value,
114-115
 - underscore (_) character, 70, 265, 355, 552
 - UnhandledException, 509
 - Unindent method, 194
 - uninstalling GAC assemblies, 1223-1224
 - Union method, 519
 - union operators, 579-582
 - unit tests, 1337-1344
 - creating, 1345-1349
 - IntelliTrace, 1334
 - unlocking resources, 211
 - unmanaged code
 - COM, 1199
 - types, converting, 1202-1203
 - writing, 4
 - unreserved keywords (Visual Basic 2012), 74
 - unused local variables, 45
 - update operations
 - databases, 542
 - DataSets, 546
 - Entity Framework (ADO.NET), 648
 - LINQ to SQL, 608-607
 - UpdateProduct method, 648
 - updating
 - ClickOnce, 1252-1253
 - Extension and Updates tool, 1302
 - Solution Explorer, references, 85
 - Upgrade method, 495
 - Upgrade Wizard, 16
 - upgrading
 - projects, 16
 - from Visual Basic 6, 94
 - URLs (uniform resource indicators), 993, 1014
 - server-driving paging (WCF), 1033-1034
 - User32 layers (WPF), 696
 - User Account Control. *See* UAC, 36-39
 - user interface. *See* UI

- users
 - configuring, 881
 - My.User property, 500-502
 - settings, managing, 1271-1273
- UShort keyword, 95
- Using..End Using statement, 275

V

- ValidateBook method, 998
- validating
 - clients, 1004
 - customizing, 614-616
 - Entity Framework (ADO.NET), 650-652
- values
 - Boolean operators, 164
 - converting, 835-840
 - dates, adding to, 142
 - default for optional arguments, 241
 - enumerations as bit flags, 320
 - IValueConverter interfaces,
 - passing, 238
 - resources, 80
 - return, 45, 319-320
 - System.Windows.Media.MediaCommands
 - enumeration, 902
- value tags (XML comments), 1214
- value types, 90-93
 - analyzing, 100
 - applying, 92-103
 - assigning, 98-99
 - conversion operators, 120-125
 - customizing, 103
 - enumerations. *See* enumerations
 - literal type characters, 98
 - memory requirements, 94
 - methods, 100-101
 - optimizing, 101
 - primitive, 93-94
 - reference types
 - converting between, 111-119
 - differences between, 106-111
 - values, passing, 238
- variables
 - assigning, 45
 - declaring, storing value types, 96
 - environments, retrieving, 481
 - exceptions, catching without, 223-224
 - instances, 45
 - local
 - avoiding ambiguities, 228-229
 - local type inference, 511-514
 - private, 265
 - structures, assigning, 308
 - unused local, 45
- variances, generics, 535-537
- VBFixedString attribute, 1205
- .Vbproj file (project files), 16
- versions
 - assemblies, 1144
 - GAC, 1227
 - .NET Framework, selecting, 18
- video, playing, 798. *See also* media
- View All Files, enabling, 638
- View control, 859
- View Detail window, 54
- ViewBox panel, 716
- viewing
 - arrays, 152-155
 - BCL, 57
 - build process results, 31
 - code snippets, 1279
 - debugger visualizers, 193
 - default constructors, 253
 - EDMs, 820
 - Event viewer, 199
 - flow documents, 803
 - grid lines, 711
 - hierarchies, Logical Tree, 704
 - images, 793-795
 - MyClass keyword, 340
 - panorama controls (Windows Phone), 974-980
 - server-driving paging, Data Services (WCF), 1033-1034
 - settings, 82
 - strings, 131-133
 - System.Object class, 91

- unit test results, 1344
- XPS documents, 808-809
- View property, 830-835
- views
 - data-binding, 830-835
 - models, defining, 446
 - Silverlight applications, defining, 449
 - WPF, defining, 448
- ViewState state, 864
- View Windows Settings (My Project), 36-39
- VirtualizingStackPanel control, 714
- virtual machines. *See* VMs, 930, 936
- visibility
 - members, 234-235, 308
 - types, 234-235
- Visual Basic 6
 - migration, 209
 - upgrading from, 94
- Visual Basic 2010, Array literals feature, 515
- Visual Basic 2012, 6
 - animations, 789-790
 - applications, debugging, 50
 - Array literals feature, 515
 - ASP.NET applications, 864-862
 - bindings, creating, 814
 - compiler fixes, 169-170
 - conversion functions, 121
 - covariance, 535
 - debugging, 179
 - applying breakpoints/trace points, 184-187
 - Autos window, 192
 - Call Stack window, 188-189
 - in code, 193-206
 - Command window, 187-188
 - implementing tools, 180-192
 - inspecting objects, 192-193
 - Just My Code debugging, 182-184
 - Locals window, 187
 - Mixed Mode debugging, 182
 - preparing examples for, 179-180
 - Threads window, 191-192
 - Watch windows, 189-191
 - documentation, 55-59
 - extension methods, 517-524

- iterations, 98-170
- iterators, 414-422
- keywords, value types, 94
- modules, 301
 - comparing classes and, 303
 - overview of, 301-303
- projects, 63
 - accessing members, 67-68
 - Application.myapp file, 75-76
 - application settings, 81-83
 - AssemblyInfo.vb file, 76-77
 - attributes, 69-70
 - classes, 64
 - code files, 72-83
 - creating, 16-18
 - implicit line continuation, 70-71
 - Imports directives, 68-69
 - inheritance, 65-66
 - methods, 64-65
 - modules, 65
 - namespaces, 66-67
 - navigating My Project, 72-75
 - properties, 64
 - references, 83-88
 - Region directives, 69
 - reserved keywords, 72
 - resources, 77-81
 - structures, 65
 - unreserved keywords, 74
- resources, 1357-1358
- scope levels, 234
- Silverlight applications, creating with, 895-897
- SyncLock..End SyncLock statement, 1064
- tools, 1357
 - coding tools, 1358
 - data access, 1359
 - developer tools, 1358
 - diagnostics/performance, 1359
 - JustDecompile, 1359
 - networks, 1359
 - searching, 1360
 - Windows Azure Management Tool, 1360
- WindowsFormsLocalization, 842

- Windows Phone applications (apps), 959-985
- Zip archives, 472-474
- Visual Basic Compiler (vbc.exe), 2, 7
- VisualBrush, 758, 767-768
- Visual C# 5.0, 6
 - compiler (Csc.exe), 7
- Visual C++ 2012, 6
- Visual F# 2012, 6
- visualizers, inspecting objects with debuggers, 192-193
- Visual Studio 2010, 3
 - add-ins (Windows Phone), 957
 - Call Hierarchy, 60
 - code editor, applying, 24-27
 - debugging, 48-55
 - Error List windows, 30-31
 - exceptions, 208
 - Get Started tab, 14
 - GUIDs, creating, 148
 - Latest News tab, 14-15
 - members, generating based on interfaces, 350
 - My Project, 33-39
 - My.Settings property, 495
 - new features, 11-12
 - online code samples, searching, 21
 - Output window, 31-33
 - projects. *See also* projects
 - applying, 16-27
 - compiling, 39-48
 - Properties window, 31
 - Quick Launch tool, 59-60
 - Start Pages, 12-14
 - tools, applying windows, 27-33
- Visual Studio 2012
 - analysis tools, 1309
 - Code Analysis, 1309-1315
 - Code Clone Detection, 1310, 1317-1319
 - Code Metrics, 1309, 1315
 - generating Dependency Graphs, 1334-1335
 - IntelliTrace, 1310, 1328-1334
 - overview of, 1309-1310
 - Profiler, 1310, 1319-1328
 - applications
 - deploying, 883
 - WPF, 697-699
 - built-in image editors, 498
 - code snippets, reusing, 1275-1283
 - columns, formatting, 823
 - COM components, importing, 1192-1194
 - customizing, 1267-1270
 - Debug configuration, 40-43
 - Express for Web, 932
 - extensibility, 1287
 - building packages, 1289-1299
 - deploying, 1299-1302
 - extending code editors, 1304-1307
 - managing, 1302-1303
 - optimizing add-ins, 1304
 - overview, 1287-1289
 - Generate from Usage feature, 433-439
 - IDE, 11
 - InstallShield, 1229. *See also* InstallShield
 - interfaces, 439
 - NuGet, managing libraries, 1283-1286
 - parallel loops/tasks, debugging, 1086
 - privileges, 933
 - Release configuration, 40-43
 - SDKs, 1288-1289
 - Silverlight application tools, 894
 - templates, exporting, 1261-1265
 - unit tests, 1337-1344
 - Visual Studio Class Designer, 424-432
 - Windows Azure, downloading/installing tools, 931-932
 - Windows Installer, 1230
- Visual Studio Express for Windows Phone, 957
- Visual Studio Gallery
 - Extension Manager, 1307
 - extensions, publishing, 1302
- Visual Studio Historical Debuggers, 52
- Visual Studio Output Selector dialog box, 1236
- visual tools, 423
- Visual Tree (WPF), 704-705
- VMs (virtual machines), 930
 - Windows Azure, 936
- Vsi Builder 2012, 1358
- VSIX packages, 1300

W

WaitForPendingFinalizers method, 279

waiting for tasks to complete, 1075

warnings

configurations, 44-46

messages, 30

Watch windows, 189-191

WCF (Windows Communication Foundation), 6

binding, 1005

clients, configuring, 1004

contracts, 993, 997-1001

Data Services

applying query interceptors, 1030-1033

consuming, 1022-1027

implementing, 1013, 1016-1021

overview of, 1013-1015

querying data, 1026

server-driving paging, 1033-1034

exception handling, 1007-1008

Fiddler, 1359

generics, exposing in, 1001

IIS, hosting in, 1008-1009

overview of, 992

portable libraries, creating, 443

project templates, 993

SDKs, 7

serialization, 1050-1053

Service Configuration Editor, 1009-1010

service operations, implementing,
1027-1029

services, 991

consuming, 1001-1007

implementing, 993-1001

WCF RIA Services

controls, data-binding, 916-919

running, 920

Silverlight applications, 911-923

web applications, 8

ASP.NET, 851-855

data services, hosting, 1016

deploying, 887-888

publishing, 884-885

WebBrowser control, 735, 753

WebBrowserTask, 967-968

WebClient class, 1105

web.config files (WCF), 995

web forms

adding, 868-869

ASP.NET applications, 855-857

Web Forms Application template

(ASP.NET), 855

Web Form template (ASP.NET), 855

WebGet attribute, 1028

WebHttpBinding, 1005

WebInvoke attribute, 1028

web pages

events, 856

lifetimes, 856

Publish.htm, 1249

requests, 852

web roles, 934-936

Web-service Definition Language.

See WSDL, 999

web services, 502

websites, Windows Azure, 930

WhenAll method, 1129

WhenAny method, 1129-1131

When Hit command, 186

When keyword, 222-223

Where keyword, 553

Where method, 519

While..End While loops, 171-172

widening conversions, 120

Win32 API calls, references to, 1206

Window class, 928

windows

Advanced Compiler Settings, 46-47

Assembly Information (My Project), 35-36

Autos, 192

Breakpoints, 184

Call Stack, 188-189

Class View, 432-433

Code Metrics Result tool, 1317

Command, 187-188

Data Source, 820

Document Outline, 704

Edit Breakpoints Labels, 184

Error Lists, 30-31

- floating, creating projects, 24
- Generate New Type, 438
- Locals, 187
- managing, 716-719
- Mapping Details, 640
- Model Browser, 640, 656
- My Project, 33-39
- New Project, 17-19
- New Solution Configuration, 42
- Object Browser, 56-59, 91
- Output, 31-33, 194
- Parallel Stacks, 1087
- Parallel Tasks, 1086
- Properties, 31, 642, 702
- properties, 717
- Quick Watch, 191
- resizing, 711
- restyling, 777
- runtime, instantiating at, 718
- Select Tool, 1291
- Solution Explorer, navigating, 28-30
- SQL Server Object Explorer, 59, 662
- Test Explorer, 1342
- Threads, 191-192
- Tool, implementing, 1296
- tools
 - Call Hierarchy, 60
 - Visual Studio 2010, 27-33
- View Detail, 54
- Watch, 189-191
- Windows 8, 956
 - .NET Framework, 8
 - Store apps, 1111
 - unit tests, 1338
 - WPF and, 695
- Windows Azure
 - Activity log, 947
 - applications, 929
 - creating demo projects, 933-944
 - deploying, 944-952
 - testing, 942-944
 - Developer Portal, registering, 931
 - drives, 931
 - Management Portal, 949-952
 - overview of, 929-931

- Silverlight applications, building projects, 936-942
- Storage Explorer, 932
- support, 928
- Visual Studio 2012, downloading/installing tools, 931-932
- Windows Azure Management Tool, 1360
- Windows Communication Foundation. *See* WCF
- Windows Event Log, 199
- Windows Explorer (GAC), 1223
- Windows Forms
 - adding, 1193
 - applications
 - frameworks, 507
 - localizing, 842-843
 - compatibility, 75
- WindowsFormsHost control, 753, 1196
- Windows Installer, 1230. *See also* InstallShield
- Windows Media Player, 86-87
- Windows Performance Toolkit, 1359
- Windows Phone
 - applications, 955
 - AppBar class, 982
 - customizing, 985-987
 - debugging, 963-964
 - executing, 984-985
 - local data storage, 980-981
 - pages, 963-966
 - panorama controls, 974-980
 - Pictures Hub, 982-984
 - programming models, 958-959
 - starting, 963-964
 - submitting to Marketplace, 987-989
 - Visual Basic 2012, 959-985
 - Developer Center, accessing, 987
 - Emulator, 964
 - interfaces, 12
 - launchers, 967-720
 - LINQ to SQL, 618
 - Marketplace, registering, 956
 - overview of, 955-956
 - Registration tool, 957
 - thread pools, 1062
 - tools, 957-958

Windows Presentation Foundation. See WPF

Windows property, 720

Windows Runtime. See WinRT, 8, 1191

Windows SDK (Software Development Kit), 7

WindowStartupLocation property, 717

WindowState property, 717

WindowStyle property, 717

WinRT (Windows Runtime), 8, 1191

WithDegreeOfParallelism method, 1098

With..End With statement, 176-177

WithEvents keyword, 384-385

WithMergeOptions method, 1098

Wizard control, 859

wizards

 Data Source Configuration Wizard, 544

 Entity Data Model Wizard, 630

 Export Settings Wizard, 1271-1273

 Launch Performance Wizard, 1320

 Publish Windows Azure Application Wizard, 944

 Publish Wizard, 1247

 Upgrade Wizard, 16

WMPLib.dll, 86

WPF (Windows Presentation Foundation), 6

 animations, 782-790

 applications

 Application object, 719-721

 arranging controls with panels, 709-716

 Browser Applications, 721-724

 contra variance, 536

 creating, 693

 frameworks, 508

 handling events, 706-709

 localizing, 844-850

 Logical Tree/Visual Tree, 704-705

 managing windows, 716-719

 Visual Studio 2012, 697-699

 XAML, 699-704

 architecture, 696-697

 brushes, 757-771

 code editors, 1308

 common dialogs, 754-755

 COM objects, calling from, 1196

 controls, 725

 Border, 727

 Button, 728

 Calendar, 728-729

 CheckBox, 729-730

 ComboBox, 730-732

 ContentControl, 726-727

 DataGrid, 731

 DatePicker, 733

 DocumentViewer, 733

 Expander, 734

 features, 725-726

 Frame, 734-735

 GroupBox, 735

 Image, 736

 Label, 736

 ListBox, 736

 ListView, 738

 MediaElement, 739

 Menu, 740

 PasswordBox, 741

 ProgressBar, 743

 RadioButton, 744

 Rectangle, 745

 ScrollView, 746

 Separator, 746

 StatusBar, 747

 TabControl, 748

 templates, 775-778

 TextBlock, 749

 TextBox, 750

 ToolBar, 750

 TreeView, 751

 WebBrowser, 735, 753

 WindowsFormsHost, 753

 data-binding, 811

 converting values, 835-840

 drag'n'drop, 818-840

 formatting strings, 835-840

 overview of, 811-818

 views, 830-835

 documents, 793, 799-808

 Error List windows, 31

 events, routing, 707-708

 images, viewing, 793-795

- media, 793-798
- My namespace, 478
- new features, 695
- ObservableCollection (Of T) collection, 407
- overview of, 694-695
- projects
 - creating, 24
 - templates, 17
- resources, 497-500
- Silverlight applications, 897. *See also* Silverlight applications
- styles, 770-775
- tools, 757
- transformations, 778-782
- views, defining, 448
- Visual Studio 2012
 - new features, 11
 - Start Pages, 14
- Windows Forms, adding, 1193
- XPS documents, viewing, 808-809
- WrapPanel panel, 713-714
- Write method, 194, 464
- WriteAllBytes method, 460
- WriteAsync method, 1138
- WriteByte method, 464
- Writelf method, 194
- WriteLinelf method, 194
- WriteLine method, 26, 194
- write-only properties, 231-232
- writing
 - binary files, 465
 - connection strings, 619
 - custom attributes, 1184
 - Data Annotations (Code First approach), 663-665
 - debug information to Output windows, 195
 - entries to application logs, 481-482
 - enumerations, 315
 - Fluent APIs, 665-668
 - managed code, 4
 - qualifiers, 434
 - read-and-write operations, 229
 - reserved keywords, 72
 - text, 464-465

- unmanaged code, 4
- XAML code, 703
- XML markup, 677-685
- WSDL (Web-service Definition Language), 999
- WSDualHttpBinding, 1005
- WSFederationHttpBinding, 1005
- WSHttpBinding, 1005
- WSHttpContextBinding, 1005

X

- X.509 certificates, 926
- XAML (eXtensible Application Markup Language), 699
 - Binding markup extension, 812-813
 - Browser Applications (XBAP), 721-724
 - code editors, 536
 - controls, 725
 - media players, implementing, 900
 - MSBuild.exe, localization, 847
 - RichTextBox control, 806
 - serialization, 1048-1050
 - Silverlight applications, debugging, 926-927
 - WPF, 696, 699-704
- XamlServices class, 1048
- XAP files, 988
- XAttribute object, 672
- XBAP (XAML Browser Applications), 721-724
- XCDATA object, 672
- XComment object, 672
- XContainer object, 672
- XCopy deployment, 1144, 1222
- XDeclaration object, 672
- XDocument class
 - members, 674
 - objects, 672
- XDocumentType object, 672
- XElement object, 672
- XHTML (Extensible Hypertext Markup Language), 855
 - event handling, 860

- XML (Extensible Markup Language), 16
 - Application.myapp file, 75-76
 - comments, 1207
 - generating Help files, 1220
 - implementing, 1210-1219
 - overview of, 1208-1209
 - files, generating, 44
 - LINQ, literals, 553
 - LINQ to XML
 - modifying, 671
 - overview of, 672-677
 - querying, 676-677
 - markup, writing, 677-685
 - namespaces, importing, 293, 689
 - schemas
 - adding, 1112
 - interfaces, 685-690
 - serialization, 1043-1045
 - viewing, 82
- Xml control, 859
- XmlRoot attribute, 1045
- XmlSerialization class, 1043
- XmlSerializer class, 48
- XmlWriterTraceListener class, 196-197
- XNA framework (Windows Phone), 959
- XNA Game Studio, 957
- XName object, 672
- XNamespace object, 672
- XNode object, 672
- Xor operator, 158-162, 313
- XpsDocument class, 808
- XPS documents, viewing, 808-809
- XText object, 672

Y-Z

- Yield statement, 419
- zero-based arrays, 149
- Zip archives
 - support, 8
 - Visual Basic 2012, 472-474
- zones, time, 144-147
- Zoom command, 424
- zooming code, 25