

Foreword by [Brandon Watson](#), Microsoft Corporation



Essential Windows Phone 7.5

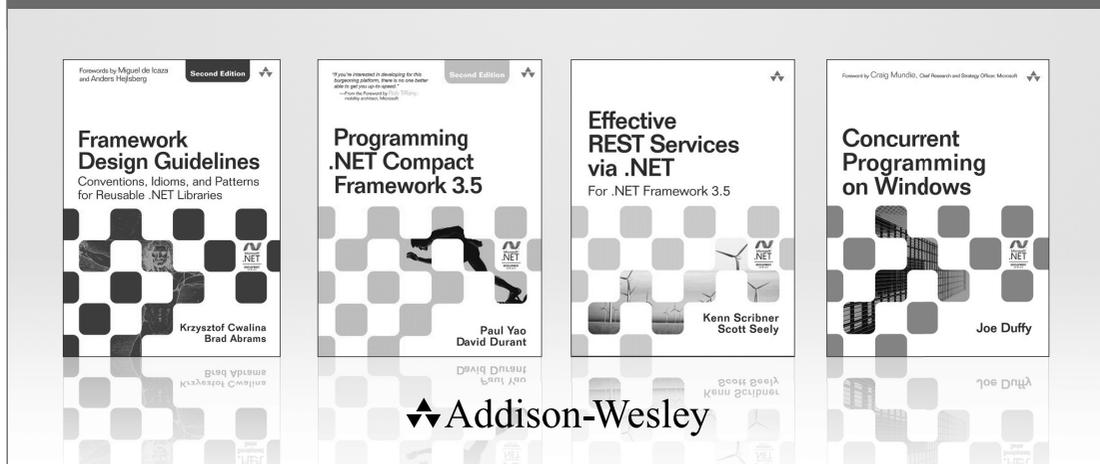
Application Development with Silverlight



Shawn Wildermuth

Essential Windows Phone 7.5

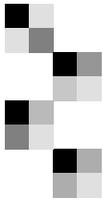
Microsoft® .NET Development Series



Visit informit.com/msdotnetseries for a complete list of available products.

The award-winning **Microsoft .NET Development Series** was established in 2002 to provide professional developers with the most comprehensive, practical coverage of the latest .NET technologies. Authors in this series include Microsoft architects, MVPs, and other experts and leaders in the field of Microsoft development technologies. Each book provides developers with the vital information and critical insight they need to write highly effective applications.





Essential Windows Phone 7.5

Application Development
with Silverlight

■ Shawn Wildermuth

◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The .NET logo is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries and is used under license from Microsoft.

Microsoft, Windows, Visual Basic, Visual C#, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries/regions.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

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

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Wildermuth, Shawn.

Essential windows phone 7.5 : application development with silverlight
/ Shawn Wildermuth.

p. cm.

Includes index.

ISBN 978-0-321-75213-0 (pbk. : alk. paper)

1. Windows phone (Computer file)
2. Silverlight (Electronic resource)
3. Operating systems (Computers)
4. Application software—Development.
5. Mobile computing—Programming. I. Title.

QA76.59.W54 2012

005.4'46—dc23

2011036842

Copyright © 2012 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-75213-0

ISBN-10: 0-321-75213-9

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, December 2011

*To my friend and mentor, Chris Sells,
without whom I would have never learned
that the story is more important than the facts.*





Contents at a Glance

Figures xvii

Tables xxv

Foreword xxvii

Preface xxix

Acknowledgments xxxi

About the Author xxxiii

- 1 Introducing Windows Phone 1**
- 2 Writing Your First Phone Application 25**
- 3 XAML Overview 61**
- 4 Controls 89**
- 5 Designing for the Phone 139**
- 6 Developing for the Phone 187**
- 7 Phone Integration 219**
- 8 Databases and Storage 305**
- 9 Multitasking 337**
- 10 Services 369**
- 11 The Marketplace 431**

Index 459



Contents

Figures xvii
Tables xxv
Foreword xxvii
Preface xxix
Acknowledgments xxxi
About the Author xxxiii

1 Introducing Windows Phone 1
A Different Kind of Phone 1
Integrated Experiences 6
Phone Specifications 7
Input Patterns 9
 Designing for Touch 10
 Hardware Buttons 11
 Keyboards 11
 Sensors 13
Application Lifecycle 14
Driving Your Development with Services 15
Live Tiles 16
The Marketplace 18
 Distributing Your Application through the Marketplace 18
 Marketplace Submissions 19
 Application Policies 20

<i>Content Policies</i>	23
Where Are We?	24
2 Writing Your First Phone Application	25
Preparing Your Machine	25
Creating a New Project	27
<i>Visual Studio</i>	27
XAML	32
Designing with Blend	36
Adding Code	43
<i>Working with Events</i>	46
<i>Debugging in the Emulator</i>	47
<i>Debugging with a Device</i>	48
<i>Using Touch</i>	52
Working with the Phone	55
Where Are We?	59
3 XAML Overview	61
What Is XAML?	61
<i>XAML Object Properties</i>	63
<i>Understanding XAML Namespaces</i>	64
<i>Naming in XAML</i>	65
Visual Containers	66
Visual Grammar	70
<i>Shapes</i>	71
<i>Brushes</i>	72
<i>Colors</i>	73
<i>Text</i>	74
Images	75
Transformations and Animations	77
<i>Transformations</i>	77
<i>Animations</i>	80
XAML Styling	82
<i>Understanding Resources</i>	83
<i>Understanding Styles</i>	84
Where Are We?	87

4 Controls	89
Controls in Silverlight	89
Simple Controls	91
Content Controls	97
List Controls	98
Phone-Specific Controls	99
Panorama Control	99
Pivot Control	102
Data Binding	105
Simple Data Binding	105
Using a DataTemplate	108
Improving Scrolling Performance	108
Binding Formatting	110
Element Binding	110
Converters	111
Data Binding Errors	113
Control Templates	114
Silverlight for Windows Phone Toolkit	119
AutoCompleteBox Control	119
ContextMenu Control	121
DatePicker and TimePicker Controls	122
ListPicker Control	124
LongListSelector Control	127
PerformanceProgressBar Control	131
ToggleSwitch Control	132
ExpanderView Control	133
PhoneTextBox Control	134
WrapPanel Layout Container	136
Where Are We?	138
5 Designing for the Phone	139
The Third Screen	139
It Is a Phone, Right?	143
Deciding on an Application Paradigm	144
Panorama	146

<i>Pivot</i>	147
<i>Simple Pages</i>	150
Microsoft Expression Blend	150
<i>Creating a Project</i>	150
<i>A Tour around Blend</i>	151
Blend Basics	159
<i>Layout</i>	159
<i>Brushes</i>	164
<i>Creating Animations</i>	169
<i>Working with Behaviors</i>	173
Phone-Specific Design	176
<i>The ApplicationBar in Blend</i>	176
<i>Using the Panorama Control in Blend</i>	179
<i>Using the Pivot Control in Blend</i>	182
Previewing Applications	185
Where Are We?	185
6 Developing for the Phone	187
Application Lifecycle	187
<i>Navigation</i>	190
<i>Tombstoning</i>	195
The Phone Experience	200
<i>Orientation</i>	201
<i>Designing for Touch</i>	203
<i>Application Client Area</i>	211
<i>Application Bar</i>	213
<i>Understanding Idle Detection</i>	215
<i>The Tilt Effect</i>	216
Where Are We?	218
7 Phone Integration	219
Using Vibration	219
Using Motion	220
<i>Emulating Motion</i>	223
Using Sound	226
<i>Playing Sounds with MediaElement</i>	226

<i>Using XNA Libraries</i>	227
<i>Playing Sounds with XNA</i>	228
<i>Adjusting Playback</i>	229
<i>Recording Sounds</i>	230
Contacts and Appointments	233
<i>Contacts</i>	233
<i>Appointments</i>	238
Alarms and Reminders	240
<i>Creating an Alarm</i>	242
<i>Creating a Reminder</i>	244
<i>Accessing Existing Notifications</i>	245
Using Tasks	246
<i>Launchers</i>	248
<i>Choosers</i>	257
Media and Picture Hubs	266
<i>Accessing Music</i>	266
<i>Playing Music</i>	268
<i>Accessing Pictures</i>	270
<i>Storing Pictures</i>	272
<i>Integrating into the Pictures Hub</i>	274
<i>Integrating into the Music+Videos Hub</i>	276
Working with the Camera	280
<i>Using the PhotoCamera Class</i>	280
<i>Raw Hardware Access</i>	284
The Clipboard API	287
Live Tiles	288
<i>Main Live Tile</i>	289
<i>Secondary Tiles</i>	290
<i>Dual-Sided Live Tiles</i>	292
Location APIs	293
<i>Location Permission</i>	293
<i>Accessing Location Information</i>	294
<i>Emulating Location Information</i>	300
Where Are We?	303

8 Databases and Storage 305

Storing Data 305

Isolated Storage 306

Serialization 308

Local Databases 314

Getting Started 314

Optimizing the Context Class 320

Associations 324

Using an Existing Database 330

Schema Updates 332

Database Security 334

Where Are We? 335

9 Multitasking 337

Multitasking 337

Background Agents 338

Periodic Agent 340

Resource-Intensive Agent 348

Audio Agent 350

Background Transfer Service 360

Requirements and Limitations 360

Requesting Transfers 362

Monitoring Requests 363

Where Are We? 368

10 Services 369

The Network Stack 370

The WebClient Class 370

Accessing Network Information 373

Consuming JSON 376

Using JSON Serialization 377

Parsing JSON 379

Web Services 383

Consuming OData 387

How OData Works 388

<i>The URI</i>	389
<i>Using OData on the Phone</i>	398
<i>Generating a Service Reference for OData</i>	398
<i>Retrieving Data</i>	399
<i>Updating Data</i>	401
Using Push Notifications	403
<i>Push Notification Requirements</i>	404
<i>Preparing the Application for Push Notifications</i>	405
<i>Setting Up the Server for Push Notifications</i>	407
<i>Raw Notifications</i>	410
<i>Sending Toast Notifications</i>	419
<i>Creating Live Tiles</i>	423
<i>Handling Push Notification Errors</i>	427
Where Are We?	429
11 The Marketplace	431
What Is the Marketplace?	431
<i>How It Works</i>	432
<i>Charging for Apps</i>	435
<i>Getting Paid</i>	438
Submitting Your App	439
<i>Preparing Your Application</i>	439
<i>The Submission Process</i>	445
<i>After the Submission</i>	451
Modifying Your Application	453
Dealing with Failed Submissions	454
Using Ads in Your Apps	457
Where Are We?	458
<i>Index</i>	459



Figures

- FIGURE 1.1 *Windows Phone Start screen* 3
- FIGURE 1.2 *Phone screen real estate* 3
- FIGURE 1.3 *The application bar in action* 4
- FIGURE 1.4 *Panorama application* 5
- FIGURE 1.5 *Last pane of a panorama application* 5
- FIGURE 1.6 *Using Metro chrome, or not* 6
- FIGURE 1.7 *Seven points of input* 8
- FIGURE 1.8 *Metro's interactive element sizes* 10
- FIGURE 1.9 *Default keyboard* 12
- FIGURE 1.10 *Contextual keyboards* 12
- FIGURE 1.11 *Application lifecycle (tombstoning)* 15
- FIGURE 1.12 *A tile in the hub* 17
- FIGURE 1.13 *Updating tiles* 17
- FIGURE 1.14 *Marketplace application submission process* 19

- FIGURE 2.1 *Microsoft Visual Studio 2010 Express for Windows Phone* 28
- FIGURE 2.2 *New Project dialog* 29
- FIGURE 2.3 *Picking the phone version to target* 29
- FIGURE 2.4 *The Visual Studio user interface* 30
- FIGURE 2.5 *Enabling the toolbar* 31
- FIGURE 2.6 *Using the emulator* 31
- FIGURE 2.7 *The emulator* 31

- FIGURE 2.8 *Using the Visual Studio XAML design surface* 33
- FIGURE 2.9 *Location of the Properties window* 34
- FIGURE 2.10 *Contents of the Properties window* 34
- FIGURE 2.11 *The changed property* 35
- FIGURE 2.12 *Opening Blend directly in Visual Studio* 36
- FIGURE 2.13 *The Blend user interface* 37
- FIGURE 2.14 *Selecting an object in Blend* 38
- FIGURE 2.15 *Selecting an object to edit in the Properties pane* 38
- FIGURE 2.16 *Updating a property in Blend* 39
- FIGURE 2.17 *Drawing in a container* 40
- FIGURE 2.18 *Rounding the corners* 40
- FIGURE 2.19 *Editing brushes* 41
- FIGURE 2.20 *Picking a color* 41
- FIGURE 2.21 *Inserting a TextBlock* 42
- FIGURE 2.22 *Centering the TextBlock* 42
- FIGURE 2.23 *Changing the text properties* 43
- FIGURE 2.24 *Naming an element in the Properties window* 45
- FIGURE 2.25 *Running the application* 46
- FIGURE 2.26 *Using the Visual Studio debugger* 48
- FIGURE 2.27 *Connected device* 49
- FIGURE 2.28 *Your phone connected to the Zune software* 49
- FIGURE 2.29 *Registering your device* 50
- FIGURE 2.30 *Successfully registered developer phone* 51
- FIGURE 2.31 *Changing the deployment to use a development phone* 51
- FIGURE 2.32 *Running on a device* 52
- FIGURE 2.33 *Dragging the ellipse* 55
- FIGURE 2.34 *The SearchTask in action* 57
- FIGURE 2.35 *Choosing a contact to retrieve an email address via the
EmailAddressChooserTask* 59
- FIGURE 2.36 *Showing the selected email in a MessageBox* 59
-
- FIGURE 3.1 *Path explained* 72
- FIGURE 3.2 *Image stretching* 77
- FIGURE 3.3 *Transformations in action* 78
- FIGURE 3.4 *Entire container transformed* 79

FIGURE 4.1	<i>TextBox control example</i>	90
FIGURE 4.2	<i>Software input panel (SIP)</i>	92
FIGURE 4.3	<i>Special SIP keys</i>	92
FIGURE 4.4	<i>Long-hold keys</i>	93
FIGURE 4.5	<i>Chat input scope</i>	94
FIGURE 4.6	<i>Simple button with simple content</i>	97
FIGURE 4.7	<i>Button with XAML content</i>	97
FIGURE 4.8	<i>List box</i>	98
FIGURE 4.9	<i>Panorama application</i>	99
FIGURE 4.10	<i>Panorama explained</i>	100
FIGURE 4.11	<i>Landscape sections</i>	101
FIGURE 4.12	<i>Pivot control</i>	103
FIGURE 4.13	<i>Pivot control in action</i>	104
FIGURE 4.14	<i>Looping pivot sections</i>	104
FIGURE 4.15	<i>Simple data binding</i>	105
FIGURE 4.16	<i>Changes in the source</i>	107
FIGURE 4.17	<i>Output window</i>	113
FIGURE 4.18	<i>Binding error shown in the Output window</i>	114
FIGURE 4.19	<i>Conversion error shown in the Output window</i>	114
FIGURE 4.20	<i>TemplatePart attribute</i>	116
FIGURE 4.21	<i>TemplateVisualState attribute</i>	118
FIGURE 4.22	<i>AutoCompleteBox example</i>	120
FIGURE 4.23	<i>ContextMenu example</i>	121
FIGURE 4.24	<i>Date picking user interface</i>	123
FIGURE 4.25	<i>Setting icons as “Content”</i>	124
FIGURE 4.26	<i>Time picking user interface</i>	125
FIGURE 4.27	<i>ListPicker example (closed)</i>	125
FIGURE 4.28	<i>ListPicker example (opened)</i>	126
FIGURE 4.29	<i>ListPicker example (full screen)</i>	126
FIGURE 4.30	<i>LongListSelector with groups</i>	128
FIGURE 4.31	<i>LongListSelector’s pop-up groups</i>	128
FIGURE 4.32	<i>ToggleSwitch example</i>	132
FIGURE 4.33	<i>ToggleSwitch components</i>	132
FIGURE 4.34	<i>ExpanderView in action</i>	133
FIGURE 4.35	<i>PhoneTextBox with the Hint and ActionIcon shown</i>	134

- FIGURE 4.36 *PhoneTextBox's length indication support* 135
- FIGURE 4.37 *PhoneTextBox's AcceptReturn functionality* 136
- FIGURE 4.38 *Buttons in a StackPanel* 137
- FIGURE 4.39 *Buttons in a WrapPanel* 137
- FIGURE 4.40 *Buttons in a vertical WrapPanel* 138
-
- FIGURE 5.1 *Foursquare.com* 140
- FIGURE 5.2 *Phone-sized app* 141
- FIGURE 5.3 *Panorama application* 142
- FIGURE 5.4 *A sample Foursquare on Windows Phone* 142
- FIGURE 5.5 *Sample application navigation* 145
- FIGURE 5.6 *Single-page Windows Phone application* 145
- FIGURE 5.7 *Sample panorama application* 146
- FIGURE 5.8 *Panorama in the emulator* 146
- FIGURE 5.9 *Pivot example* 148
- FIGURE 5.10 *Pivot pages* 149
- FIGURE 5.11 *Blend New Project dialog* 150
- FIGURE 5.12 *Blend user interface* 152
- FIGURE 5.13 *Blend toolbar* 153
- FIGURE 5.14 *Projects panel* 154
- FIGURE 5.15 *Assets panel* 155
- FIGURE 5.16 *Objects and Timeline panel* 155
- FIGURE 5.17 *Artboard* 157
- FIGURE 5.18 *Item Tools panel* 158
- FIGURE 5.19 *Searching in the Properties panel* 159
- FIGURE 5.20 *Dragging a new control* 160
- FIGURE 5.21 *Margin and alignment layout* 160
- FIGURE 5.22 *Column and row gutters* 161
- FIGURE 5.23 *Splitting the grid into rows* 162
- FIGURE 5.24 *Modifying row/column properties* 163
- FIGURE 5.25 *Sizing across rows* 163
- FIGURE 5.26 *Sizing across rows with RowSpan* 164
- FIGURE 5.27 *Brushes in the Properties panel* 164
- FIGURE 5.28 *Converting a color to a resource* 167
- FIGURE 5.29 *Creating a color resource* 168

- FIGURE 5.30 *Applying a color resource* 168
- FIGURE 5.31 *Creating a brush resource* 169
- FIGURE 5.32 *Applying a brush resource* 169
- FIGURE 5.33 *Storyboard basics* 169
- FIGURE 5.34 *Creating a storyboard* 170
- FIGURE 5.35 *Objects and Timeline panel with animation* 170
- FIGURE 5.36 *Picking the animation point* 171
- FIGURE 5.37 *Animation mode on the artboard* 171
- FIGURE 5.38 *The ellipse animated* 172
- FIGURE 5.39 *Animation values in the Objects and Timeline panel* 172
- FIGURE 5.40 *RenderTransform in an animation* 173
- FIGURE 5.41 *Closing a storyboard* 173
- FIGURE 5.42 *Behaviors in the Assets panel* 174
- FIGURE 5.43 *Applying a behavior* 175
- FIGURE 5.44 *Changing behavior properties* 175
- FIGURE 5.45 *Multiple behaviors* 176
- FIGURE 5.46 *ApplicationBar explained* 177
- FIGURE 5.47 *Adding an ApplicationBar* 178
- FIGURE 5.48 *Adding items to the ApplicationBar* 178
- FIGURE 5.49 *Selecting a built-in icon for an ApplicationBar icon* 179
- FIGURE 5.50 *New panorama application* 180
- FIGURE 5.51 *PanoramaItems in the Objects and Timeline panel* 180
- FIGURE 5.52 *Panorama control user interface* 181
- FIGURE 5.53 *PanoramaItem selection* 181
- FIGURE 5.54 *Adding a PanoramaItem* 182
- FIGURE 5.55 *Creating a pivot application* 183
- FIGURE 5.56 *A pivot application* 183
- FIGURE 5.57 *Pivot control user interface* 184
- FIGURE 5.58 *Editing a PivotItem* 184
- FIGURE 5.59 *Changing device properties* 185

- FIGURE 6.1 *Important files in a new project* 188
- FIGURE 6.2 *Page navigation explained* 191
- FIGURE 6.3 *URI mapping to the files in the project* 192
- FIGURE 6.4 *How tombstoning works* 196

- FIGURE 6.5 *Portrait orientation* 201
- FIGURE 6.6 *Landscape left orientation* 201
- FIGURE 6.7 *Landscape right orientation* 202
- FIGURE 6.8 *Application client area* 212
- FIGURE 6.9 *Untilted* 216
- FIGURE 6.10 *Tilted* 216
-
- FIGURE 7.1 *Accelerometer axes* 221
- FIGURE 7.2 *Showing the Accelerometer window in the emulator* 224
- FIGURE 7.3 *The Accelerometer window* 225
- FIGURE 7.4 *An alarm* 240
- FIGURE 7.5 *A reminder* 241
- FIGURE 7.6 *Stacked notifications* 242
- FIGURE 7.7 *Media player controls* 253
- FIGURE 7.8 *PhoneCallTask confirmation* 255
- FIGURE 7.9 *Allowing photo cropping* 262
- FIGURE 7.10 *Music library objects* 267
- FIGURE 7.11 *Displaying the albums and pictures* 272
- FIGURE 7.12 *The apps in the Pictures hub* 274
- FIGURE 7.13 *Tile layers* 288
- FIGURE 7.14 *Opening the emulator's Additional Tools sidebar* 300
- FIGURE 7.15 *Selecting the Location tab* 301
- FIGURE 7.16 *Location tab of the Additional Tools dialog* 301
- FIGURE 7.17 *Using pins to create waypoints* 302
- FIGURE 7.18 *Saving recorded data* 302
-
- FIGURE 8.1 *The SQL query* 319
- FIGURE 8.2 *SQL Server Compact Edition database as Content* 330
-
- FIGURE 9.1 *Relationship between application and scheduled task* 339
- FIGURE 9.2 *Adding a new Scheduled Task Agent project* 341
- FIGURE 9.3 *Picking the Windows Phone Scheduled Task Agent* 341
- FIGURE 9.4 *Adding a reference to the Scheduled Task Agent project* 344
- FIGURE 9.5 *The PeriodicTask's description in the management user interface* 346

FIGURE 9.6	<i>The Universal Volume Control (UVC) in action</i>	351
FIGURE 9.7	<i>Adding an audio agent to your project</i>	352
FIGURE 9.8	<i>Making a reference to the audio agent project</i>	353
FIGURE 10.1	<i>Adding a service reference</i>	383
FIGURE 10.2	<i>The Add Service Reference dialog</i>	384
FIGURE 10.3	<i>Service files displayed</i>	385
FIGURE 10.4	<i>Adding a service reference to an OData feed</i>	399
FIGURE 10.5	<i>Adding a using statement to the data service</i>	400
FIGURE 10.6	<i>Push notification message flow</i>	404
FIGURE 10.7	<i>Debugging push notifications</i>	420
FIGURE 10.8	<i>A toast message</i>	420
FIGURE 10.9	<i>Tile layers</i>	424
FIGURE 11.1	<i>The Marketplace</i>	432
FIGURE 11.2	<i>The Marketplace in Zune</i>	432
FIGURE 11.3	<i>Submission process</i>	433
FIGURE 11.4	<i>The App Hub</i>	434
FIGURE 11.5	<i>Capability detection results</i>	442
FIGURE 11.6	<i>Works in the dark theme</i>	444
FIGURE 11.7	<i>Does not work in the light theme</i>	444
FIGURE 11.8	<i>Accessing your “dashboard”</i>	445
FIGURE 11.9	<i>Starting the submission process</i>	446
FIGURE 11.10	<i>Step 1 of the submission process</i>	446
FIGURE 11.11	<i>Filling in the descriptive fields</i>	448
FIGURE 11.12	<i>Pricing your app</i>	449
FIGURE 11.13	<i>Publish and testing options</i>	450
FIGURE 11.14	<i>Submission confirmation</i>	450
FIGURE 11.15	<i>Application lifecycle page</i>	451
FIGURE 11.16	<i>My Apps page</i>	452
FIGURE 11.17	<i>Deep link</i>	453
FIGURE 11.18	<i>Application actions</i>	454
FIGURE 11.19	<i>A failure report</i>	455



Tables

TABLE 1.1	<i>Integrated Experiences</i>	7
TABLE 1.2	<i>Hardware Specifications</i>	8
TABLE 1.3	<i>Hardware Inputs</i>	9
TABLE 1.4	<i>Sample Keyboard Layouts</i>	13
TABLE 1.5	<i>Sensors</i>	13
TABLE 1.6	<i>Microsoft Phone Services</i>	16
TABLE 2.1	<i>Windows Phone Developer Tools Requirements</i>	26
TABLE 3.1	<i>Visual Containers</i>	67
TABLE 3.2	<i>Grid Row and Column Sizing</i>	69
TABLE 3.3	<i>Brush Types</i>	73
TABLE 3.4	<i>Transformation Types</i>	79
TABLE 4.1	<i>Common InputScope Values</i>	94
TABLE 4.2	<i>RichTextBox Markup Tags</i>	96
TABLE 4.3	<i>Data Binding Modes</i>	107
TABLE 5.1	<i>New Project Types in Blend</i>	151
TABLE 5.2	<i>Row/Column Sizing Icons</i>	162
TABLE 5.3	<i>Brush Editors</i>	165
TABLE 5.4	<i>Blend Behaviors</i>	174





TABLE 6.1	<i>Manipulation Events</i>	207
TABLE 6.2	<i>UIElement Touch Events</i>	211
TABLE 7.1	<i>FilterKind Enumeration</i>	234
TABLE 7.2	<i>Launchers</i>	246
TABLE 7.3	<i>Choosers</i>	247
TABLE 7.4	<i>MediaPlayerPlaybackControls Enumeration</i>	253
TABLE 9.1	<i>Scheduled Task Limitations</i>	340
TABLE 10.1	<i>OData HTTP Verb Mappings</i>	388
TABLE 10.2	<i>OData Query Options</i>	391
TABLE 10.3	<i>\$filter Operators</i>	393
TABLE 10.4	<i>\$filter Functions</i>	394
TABLE 10.5	<i>Push Notification Response Headers</i>	414
TABLE 10.6	<i>Response Codes and Header Status Codes</i>	415
TABLE 10.7	<i>ChannelErrorType Enumeration</i>	429
TABLE 10.8	<i>ChannelPowerLevel Enumeration</i>	429
TABLE 11.1	<i>International Pricing Example</i>	436
TABLE 11.2	<i>Application Images</i>	443
TABLE 11.3	<i>Advertising Vendors for the Phone</i>	457



Foreword

When Shawn asked me to write a foreword for his Windows Phone development book, I had a couple of reactions. First, that they must really be scraping the bottom of the barrel if they have asked me to write anything. There are so many people who actually help bring the product to market who never really get the credit they deserve. While I am honored that I was asked to write this, based in part on my public role on the team, the engineering team that designed and built this amazing product are the real heroes. The product itself is amazing, but the right application platform, which enables the amazing Metro apps and games to be built, is a developer's playground. I do this to honor them.

My second reaction was to think about the huge value Shawn has in the Microsoft ecosystem. As an eight-time MVP and Silverlight Insider, Shawn's contributions are highly valued both for their content as well as for their reach. When Shawn speaks, you know that he has the developer in mind: He is a developer's developer. Without individuals like Shawn, it would be tough (if possible at all) for Microsoft to have built our developer ecosystem over the last three decades. I do this to honor him.

My last reaction was one of panic. I have never written a foreword before, so I was at a bit of a loss as to what I should say. I figure if you are buying this book, you did so of your own volition, and not on the strength of what I have to say here. However, if you are reading the foreword with



an eye toward confirming your belief that Windows Phone is where it's at, well, for that I can be accommodating. I do this to honor you.

With the initial release of Windows Phone, and the subsequent pairing with Nokia, Microsoft is investing in building the third ecosystem for mobile developers. The canvas with which mobile developers can work on Windows Phone is unlike any other platform, whereby developers can create simply gorgeous apps with more focus on the user experience than tinkering with the innards of a convoluted framework. Metro apps come alive on the screen, and you will be able to build deeply engaging applications using Live Tiles.

Windows Phone 7.5 is an updated release, codenamed “Mango,” and carries with it the tagline “Put people first.” We think the same way about the developer platform. We aim to put developers first. The book you are holding might be your first step on your journey to building Windows Phone apps. It may be a refresher course. Either way, with Shawn's guidance, we know that you will come away from this experience feeling great about your prospects of building amazing mobile experiences for Windows Phone, and a firm belief that Microsoft puts the developers first when we think about Windows Phone. Every developer matters. Every. Single. One.

—Brandon Watson
Microsoft Corporation



Preface

I have never owned a PalmPilot. But I have owned palmtops and smartphones. I dived into writing software for a plethora of different devices but never got very far. My problem was that the story of getting software onto the phones was chaotic and I didn't see how the marketing of software for phones would lead to a successful product. In the intervening years, I got distracted by Silverlight and Web development. I didn't pay attention as the smartphone revolution happened. I was happily neck-deep in data binding, business application development, and teaching XAML.

The smartphone revolution clearly started with the iPhone. What I find interesting is that the iPhone is really about the App Store, not the phone. It's a great device, but the App Store is what changed everything, providing a simple way to publish, market, and monetize applications for these handheld powerhouses that everyone wanted. Of course, Apple didn't mean to do it. When the original iPhone shipped, Apple clearly said that Safari (its Web browser) was the development environment. With the pressure of its OS X developer community, Apple relented and somewhat accidentally created the app revolution.

When it was clear that I had missed something, I dived headlong into looking at development for phones again. I had an Android phone at the time, so that is where I started. Getting up to speed with Eclipse and Java wasn't too hard, but developing for the phone was still a bit of a chore. The development tools just didn't seem to be as easy as the development I was



used to with Visual Studio and Blend. In this same time frame, I grabbed a Mac and tried my hand at Objective-C and Xcode to write something simple for the iPhone. That experience left me bloodied and bandaged. I wanted to write apps, but since it was a side effort, the friction of the tool sets for Android and iPhone left me wanting, and I put them aside.

Soon after my experience with iPhone and Android, Microsoft took the covers off its new phone platform: Windows Phone 7. For me, the real excitement was the development experience. At that point I'd been teaching and writing about Silverlight since it was called WPF/E, so the ability to marry my interest in mobile development to my Silverlight knowledge seemed like a perfect match.

I've enjoyed taking the desktop/Web Silverlight experience I have and applying the same concepts to the phone. By being able to use Visual Studio and Blend to craft beautiful user interface designs and quickly go from prototype to finished application, I have found that the workflow of using these tools and XAML makes the path of building my own applications much easier than on other platforms.

In the middle of this learning process Microsoft continued to mature the platform by announcing and releasing Windows Phone 7.5 (code-named Mango). I was left questioning whether to finish my Windows Phone 7 book or rush forward and mold all the new features of Windows Phone 7.5 into a book for this next version of the phone. Obviously you know the answer to that question.

It has been a long road to get the right story for this book, and to help both beginners and existing Silverlight developers to learn from the book. My goal was always to allow readers to get started writing apps quickly, while also including the information that leads to great apps. Because of the relative size of these minicomputers we keep in our pockets, knowing when to pull back is often the key to a great application. As you will see throughout this book, my goal has been to help you build great apps, not rich applications. This means I will try to hold your hand as you read the book, but I will also challenge your assumptions about how you approach the process of building applications for the phone.



Acknowledgments

Writing a book is a team sport. Anyone who thinks for a moment that writing a book requires that you sit in a dark room and craft words that magically get bound into Amazon currency hasn't been through the sausage factory that is book writing. The fact is that I may have the skills to get words down on virtual paper, but I am not good at much of the rest of the process. It takes a strong editor who knows how to dole out praise and pressure in equal amounts. It takes technical reviewers who aren't afraid to ruffle your feathers. It takes production people to take the mess of Visio ramblings you call figures and create something the reader will understand. Finally, it takes an army of people to listen to your questions about the ambiguity of writing a book based on a beta version of a product . . . and who will not stop responding to your constant pestering. So I'd like to thank my army of people by acknowledging their real contributions (in no particular order).

First and foremost, I want to thank my editor at Addison-Wesley, Joan Murray. I am not an easy author to work with, and she's been a trouper in getting me to stick to deadlines and coercing me to make the right decisions, not just the easy ones. The rest of the people at Addison-Wesley that I've had the pleasure to work with are all great, too. Of special note, Christopher Cleveland did a great job picking up the role of developmental editor in the middle of the book, and has been great through the whole process.



To the litany of people on the Silverlight Insiders Mailing List and the Windows Phone 7 Advisors Mailing List, I would like to thank you for your patience as I pestered the lists with endless questions and hyperbolic rants. You all helped shape this book, even if you didn't realize it.

During this process, my blog's readers and my followers on Facebook and Twitter remained a consistent sounding board. My polls and open questions helped me shape what is and isn't in this book. For that I am indebted to you.

I also want to thank my terrific technical reviewers, Jeremy Likeness, Ambrose Little, and Bruce Little. Not only did they help me find the tons of places I just plain got it wrong, but they also helped me when the story got off track and I missed that key piece of the puzzle. Of particular note, I want to thank Ambrose for his tenacious adherence to the designer's voice. He helped me make sure I wasn't coddling the developers into bad user experience design.

To anyone else I forgot to mention, I apologize.

—Shawn Wildermuth

November 2011

<http://wildermuth.com>

@shawnwildermuth



About the Author

During his twenty-five years in software development, **Shawn Wildermuth** has experienced a litany of shifts in software development. These shifts have shaped how he understands technology. Shawn is a nine-time Microsoft MVP, a member of the INETA Speaker's Bureau, and an author of several books on .NET. He is also involved with Microsoft as a Silverlight Insider and a Data Insider. He has spoken at a variety of international conferences, including TechEd, MIX, VSLive, OreDev, SDC, WinDev, DevTeach, DevConnections, and DevReach. He has written dozens of articles for a variety of magazines and websites including *MSDN*, DevSource, InformIT, *CoDe Magazine*, ServerSide.NET, and MSDN Online. He is currently teaching workshops around the United States through his training company, AgiliTrain (<http://agilitrain.com>).

■ 2 ■

Writing Your First Phone Application

While the press might have you believe that becoming a phone-app millionaire is a common occurrence, it's actually pretty rare, but that doesn't mean you won't want to create applications for the phone. Hopefully the days of cheap and useless but popular phone apps are over, and we can start focusing on phone-app development as being a way to create great experiences for small and large audiences. Microsoft's vision of three screens is becoming a reality, as the phone is joining the desktop and the TV as another vehicle for you to create immersive experiences for users.

Although understanding Windows Phone capabilities and services is a good start, you are probably here to write applications. With that in mind, this chapter will walk you through setting up a machine for authoring your very first Windows Phone Silverlight application.

Preparing Your Machine

Before you can start writing applications for the phone, you must install the Windows Phone Developer Tools. Go to <http://create.msdn.com> to download the tools called Windows Phone SDK 7.1. This website is the

starting point for downloading the tools as well as accessing the forums if you have further questions about creating applications.

Windows Phone Versioning Confusion

At the time of this writing there is a difference in how the phone and the underlying operating system are named. The phone itself is marketed as “Windows Phone 7.5” but the operating system is called “Windows Phone OS 7.1” and the development tools are called “Windows Phone SDK 7.1.” This can be confusing, but if you remember the phone is “7.5” and all the software is “7.1” it can help.

To install the Windows Phone SDK 7.1 you must meet the minimum system requirements shown in Table 2.1.

Once you meet the requirements, you can run the `vm_web.exe` file that you downloaded from the website to install the Windows Phone SDK 7.1. The SDK installer includes Microsoft Visual Studio 2010 Express for Windows Phone, Microsoft Blend Express for Windows Phone (the Express version of Microsoft Expression Blend), and the Software Development Kit (SDK). Visual Studio Express is the coding environment for Windows Phone. Blend Express is the design tool for phone applications. And the SDK is a set of libraries for creating phone applications and an emulator for creating applications without a device.

TABLE 2.1 Windows Phone Developer Tools Requirements

Requirement	Description
Operating system	Windows 7, x86 or x64 (all but Starter Edition); or Windows Vista SP2, x86 or x64 (all but Starter Edition)
Memory	3GB RAM
Disk space	4GB free space
Graphics card	DirectX 10 capable card with a WDDM 1.1 driver

Tip

The Windows Phone SDK 7.1 does not work well in a virtual machine (e.g., Virtual PC, VMware, etc.) and is not officially supported. The emulator is a virtual machine of the phone, so running a virtual machine in a virtual machine tends to cause problems, especially slow performance.

Visual Studio is the primary tool for writing the code for your phone applications. Although the Windows Phone SDK 7.1 installs a version of Visual Studio 2010 Express specifically for phone development, if you already have Visual Studio 2010 installed on your machine the phone tools will also be integrated into this version of Visual Studio. The workflow for writing code in both versions of Visual Studio is the same. Although both versions offer the same features for developing applications for the phone, in my examples I will be using Visual Studio Express Edition for Windows Phone. In addition, I will be using Blend Express, not the full version of Blend (i.e., Expression Blend).

Creating a New Project

To begin creating your first Windows Phone application you will want to start in one of two tools: Visual Studio or Expression Blend. Visual Studio is where most developers start their projects, so we will begin there, but we will also discuss how you can use both applications for different parts of the development process.

Visual Studio

As noted earlier, when you install the Windows Phone SDK 7.1 you get a version of Visual Studio 2010 Express that is used to create Windows Phone applications only. When you launch Visual Studio 2010 Express you will see the main window of the application, as shown in Figure 2.1.

Click the New Project icon on the Start page and you will be prompted to start a new project. Visual Studio 2010 Express only supports creating applications for Window Phone. In the New Project dialog (see Figure 2.2)

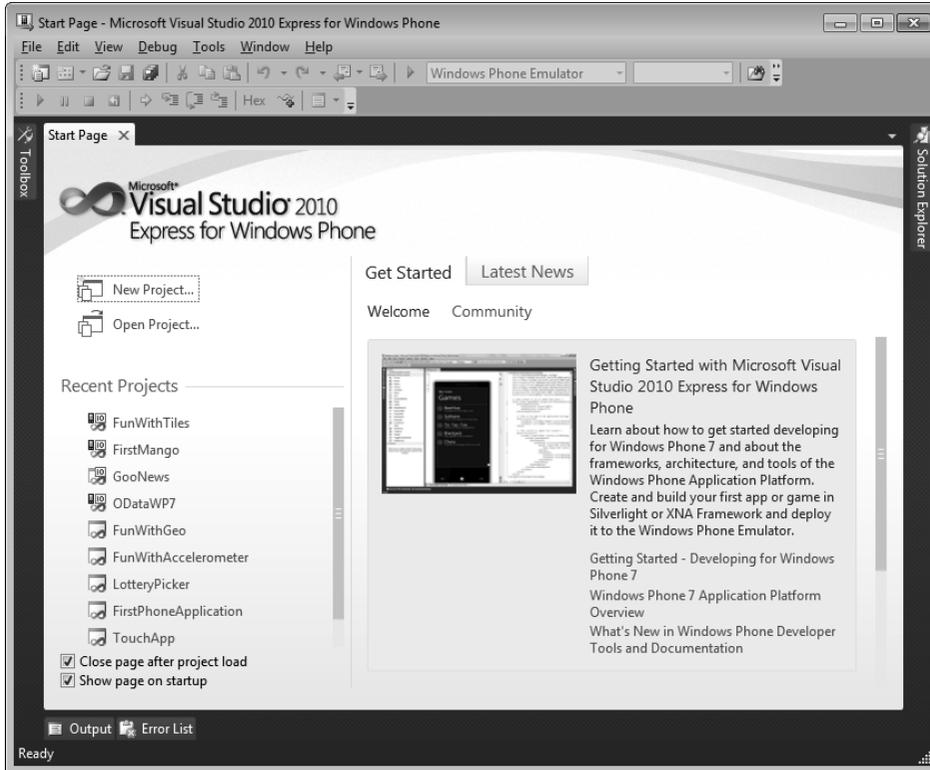


FIGURE 2.1 Microsoft Visual Studio 2010 Express for Windows Phone

you will notice that only Silverlight and XNA project templates are shown. For our first project we will start with a new Windows Phone Application template and name it “HelloWorldPhone”.

When you click the OK button to create the project, Visual Studio will prompt you with a dialog where you can pick what version of the phone to target (version 7.0 or 7.1), as seen in Figure 2.3.

Once Visual Studio creates the new project, you can take a quick tour of the user interface (as shown in Figure 2.4). By default, Visual Studio shows two main panes for creating your application. The first pane (labeled #1 in the figure) is the main editor surface for your application. In this pane, every edited file will appear separated with tabs as shown. By default, the MainPage.xaml file is shown when you create a new Windows Phone application; this is the main design document for your new application.

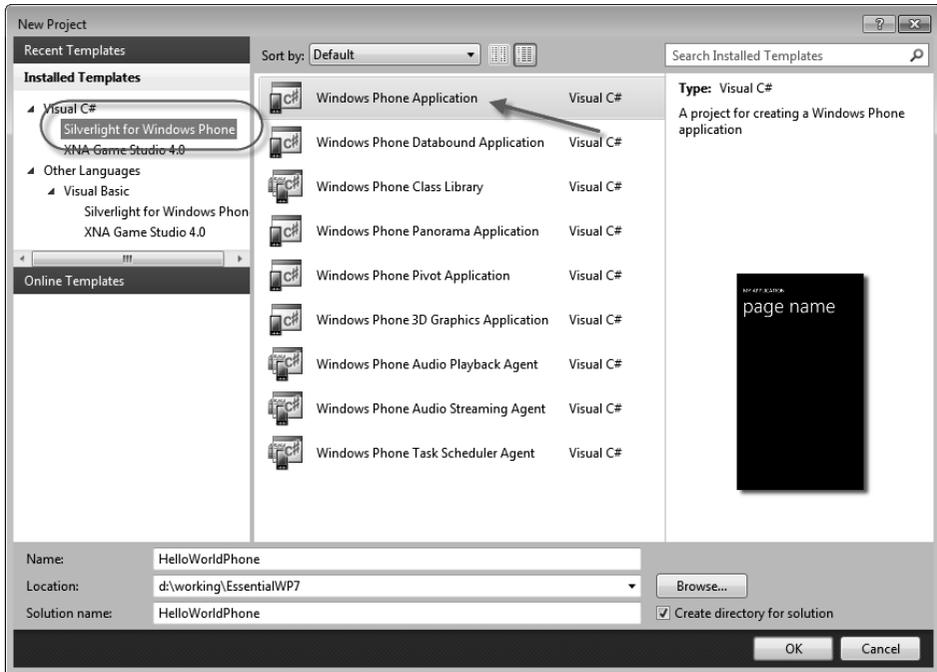


FIGURE 2.2 New Project dialog

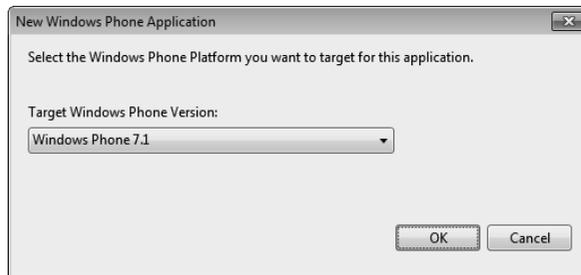


FIGURE 2.3 Picking the phone version to target

The second pane (#2 in the figure) is the Solution Explorer pane and it displays the contents of the new project.

Another common pane that you will use is the toolbar, and it is collapsed when you first use Visual Studio 2010 Express for Windows Phone. On the left side of the main window you will see a Toolbox tab that you can click to display the Toolbox, as shown in Figure 2.5.

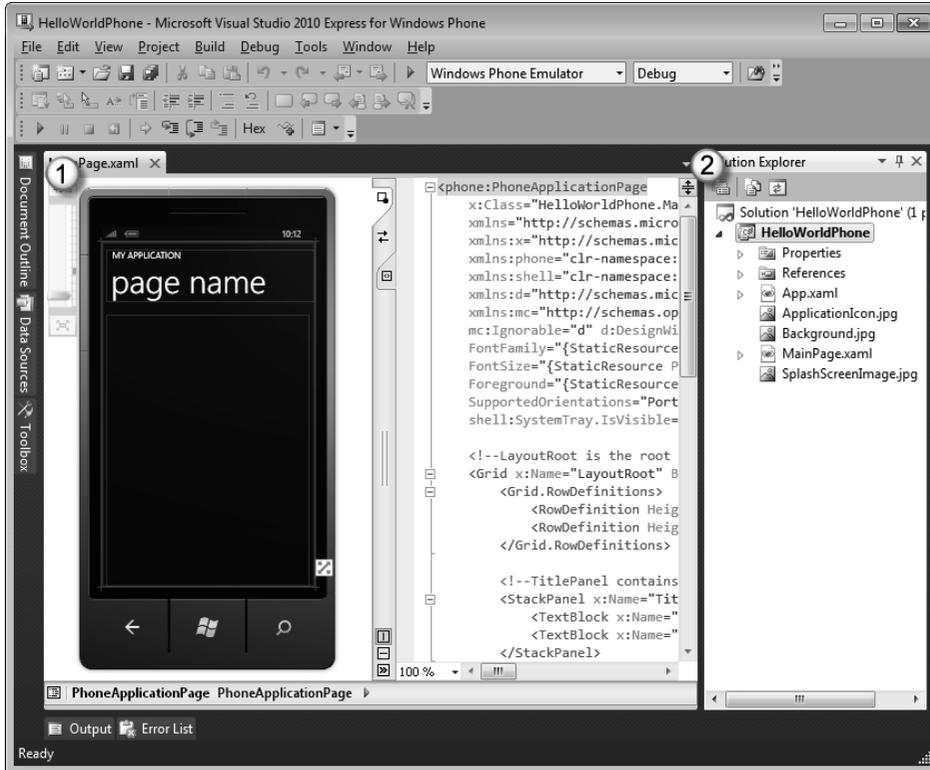


FIGURE 2.4 The Visual Studio user interface

You may also want to click the pin icon to keep the toolbar shown at all times (as highlighted in Figure 2.5).

Before we look at how to create the application into something that is actually useful, let's see the application working in the device. You will notice that in the toolbar (not the Toolbox) of Visual Studio there is a bar for debugging. On that toolbar is a drop-down box for specifying what to do to debug your application. This drop down should already display the words "Windows Phone Emulator" as that is the default when the tools are installed (as shown in Figure 2.6).

At this point, if you press the F5 key (or click the triangular play button on the debugging toolbar), Visual Studio will build the application and start the emulator with our new application, as shown in Figure 2.7.

This emulator will be the primary way you will debug your applications while developing applications for Windows Phone. Our application

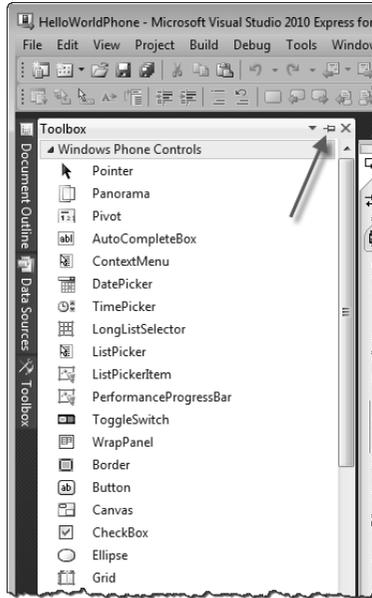


FIGURE 2.5 Enabling the toolbar

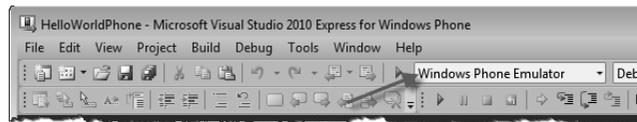


FIGURE 2.6 Using the emulator

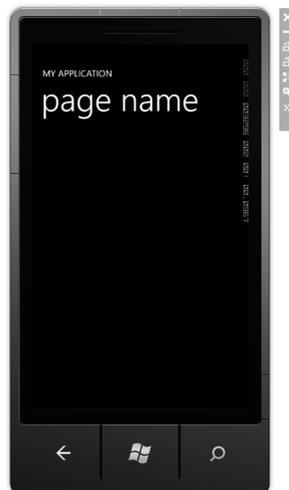


FIGURE 2.7 The emulator

does not do anything, so you can go back to Visual Studio and click the square stop button on the debugging toolbar (or press Shift-F5) to end your debugging session. You should note that the emulator does not shut down. It is meant to stay running between debugging sessions.

XAML

In Silverlight, development is really split between the design and the code. The design is accomplished using a markup language called **XAML** (rhymes with camel). XAML (or eXtensible Application Markup Language) is an XML-based language for representing the look and feel of your applications. Since XAML is XML-based, the design consists of a hierarchy of elements that describe the design. At its most basic level, XAML can be used to represent the objects that describe the look and feel of an application.¹ These objects are represented by XML elements, like so:

```
<Rectangle />

<!-- or -->

<TextBox />
```

You can modify these XML elements by setting attributes to change the objects:

```
<Rectangle Fill="Blue" />

<!-- or -->

<TextBox Text="Hello World" />
```

Containers in XAML use XML nesting to imply ownership (a parent-child relationship):

```
<Grid>
  <Rectangle Fill="Blue" />
  <TextBox Text="Hello World" />
</Grid>
```

Using this simple XML-based syntax you can create complex, compelling designs for your phone applications. With this knowledge in hand, we

1. This is an oversimplification of what XAML is. Chapter 3, XAML Overview, will explain the nature of XAML in more detail.

can make subtle changes to the XAML supplied to us from the template. We could modify the XAML directly, but instead we will start by using the Visual Studio designer for the phone. In the main editor pane of Visual Studio the `MainPage.xaml` file is split between the designer and the text editor for the XAML. The left pane of the `MainPage.xaml` file is not just a preview but a fully usable editor. For example, if you click on the area containing the words “page name” on the design surface, it will select that element in the XAML, as shown in Figure 2.8.

Once you have that element selected in the designer, the properties for the element are shown in the Properties window (which shows up below the Solution Explorer). If the window is not visible, you can enable it in the View menu by selecting “Properties window” or by pressing the F4 key. This window is shown in Figure 2.9.

The Properties window consists of a number of small parts containing a variety of information, as shown in Figure 2.10.

The section near the top (#1 in Figure 2.10) shows the type of object you have selected (in this example, a `TextBlock`) and the name of the object, if any (here, “PageTitle”). This should help you ensure that you have selected the right object to edit its properties. The next section down (#2) contains a Search bar where you can search for properties by name, as well as buttons for sorting and grouping the properties. The third section (#3) is a list of the properties that you can edit.

Note

You can also use the Properties window to edit events, but we will cover that in a later chapter.



FIGURE 2.8 Using the Visual Studio XAML design surface

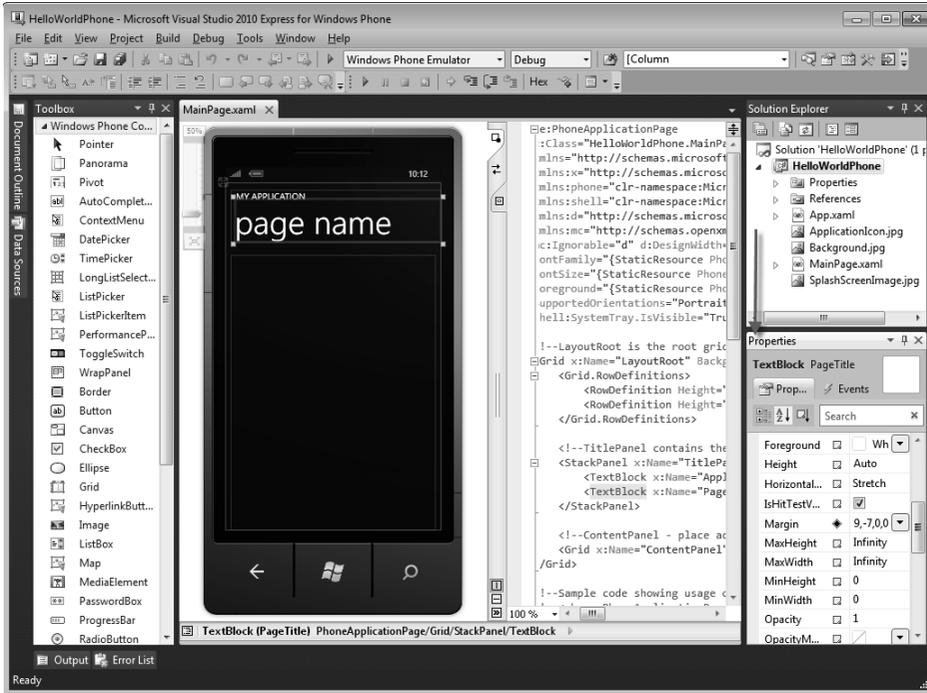


FIGURE 2.9 Location of the Properties window

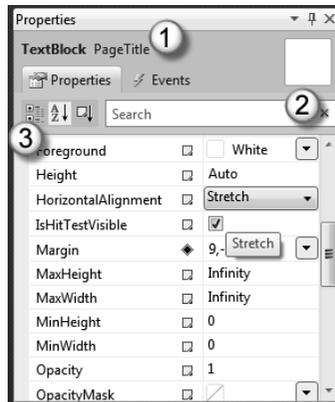


FIGURE 2.10 Contents of the Properties window

From the Properties window you can change the properties of the selected item. For example, to change the text that is in the `TextBlock`, you can simply type in a new value for the `Text` property. If you enter “hello world” in the `Text` property and press Return, the designer will

change to display the new value. Changing this property actually changes the XAML in the MainPage.xaml file. The design surface is simply reacting to the change in the XAML. If you look at the XAML the change has been affected there as well, as shown in Figure 2.11.

You can edit the XAML directly as well if you prefer. If you click on the `TextBlock` above the `PageTitle` (the one labeled “ApplicationTitle”), you can edit the `Text` attribute directly. Try changing it to “MY FIRST WP7 APP” to see how it affects the designer and the Properties window:

```
...
<TextBlock x:Name="ApplicationTitle"
    Text="MY FIRST WP7 APP"
    Style="{StaticResource PhoneTextNormalStyle}" />
...
```

Depending on their comfort level, some developers will find it easier to use the Properties window while others will be more at ease editing the XAML directly. There is no wrong way to do this.

Although the Visual Studio XAML designer can create interesting designs, the real powerhouse tool for designers and developers is Blend. Let’s use it to edit our design into something useful for our users.

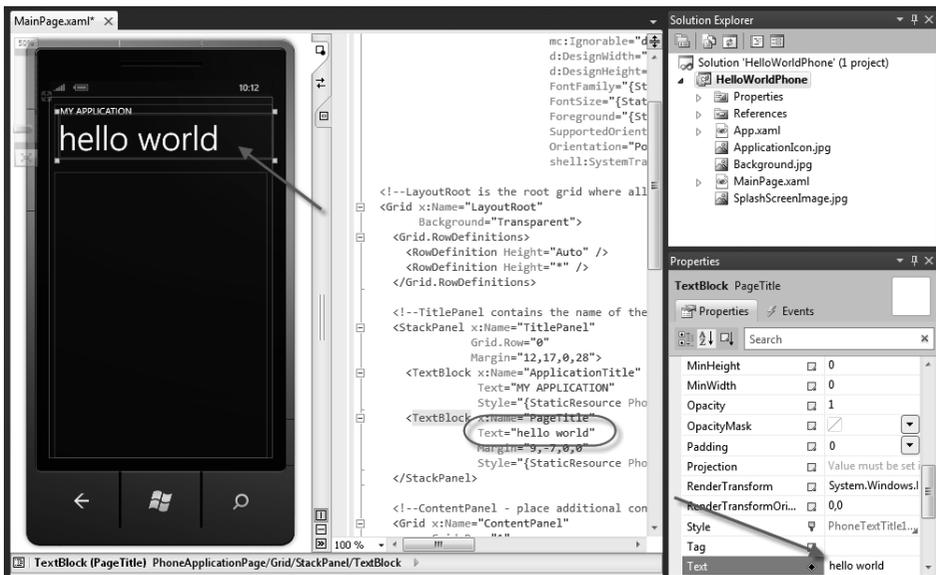


FIGURE 2.11 The changed property



FIGURE 2.13 The Blend user interface

Note

Blend and Visual Studio both open entire solutions, not just files. This is a significant difference from typical design tools.

The first pane (labeled #1 in Figure 2.13) contains multiple tabs that give you access to several types of functionality. By default, the first tab (and the one in the foreground) is the Projects tab (though you could be showing a different tab by default). This tab shows the entire solution of projects. The format of this tab should look familiar; it's showing the same information as the Solution Explorer in Visual Studio. The next pane (#2) is the editor pane. This pane contains tabs for each opened file (only one at this point). MainPage.xaml should be the file currently shown in the editor. Note that the editor shows the page in the context of the phone so that you can better visualize the experience on the phone. On the right-hand side of the Blend interface is another set of tabs (#3) that contain information about selected items in the design surface. The selected tab should be the Properties tab.

This tab is similar to the Properties window in Visual Studio but is decidedly more designer-friendly. As you select items on the design surface, you'll be able to edit them in the Properties tab here. Finally, the Objects and Timeline pane (#4) shows the structure of your XAML as a hierarchy.

Let's make some changes with Blend. First (as shown in Figure 2.14) select the "hello world" text in the designer.

Once it's selected, you can see that eight squares surround the selection. These are the handles with which you can change the size or location of the `TextBlock`. While this object is selected, the Objects and Timeline pane shows the item selected in the hierarchy; as well, the item is shown in the Properties tab so that you can edit individual properties (as shown in Figure 2.15).



FIGURE 2.14 Selecting an object in Blend

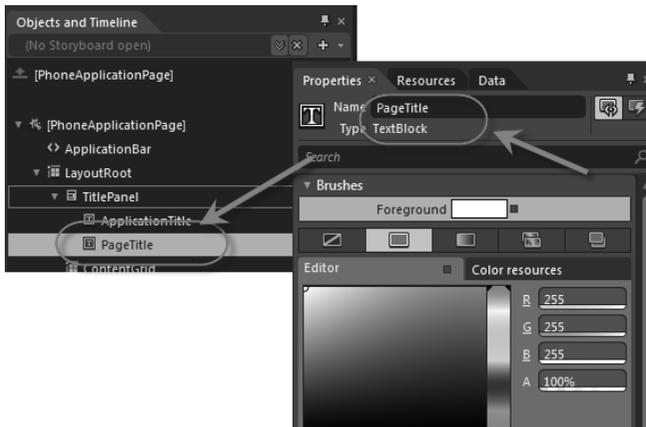


FIGURE 2.15 Selecting an object to edit in the Properties pane

If you type “text” into the search bar of the Properties pane, the properties that have that substring in them will appear (to temporarily reduce the number of properties in the Properties pane). You can change the title by changing the `Text` property, as shown in Figure 2.16.

Once you’re done changing the text, you may want to click the “X” in the Search bar to clear the search criteria. This will remove the search and show all the properties of the `TextBlock` again.

Selecting items and changing properties seems similar to what you can do in Visual Studio, but that’s just where the design can start. Let’s draw something. Start by selecting a container for the new drawing. In the Objects and Timeline pane, choose the `ContentPanel` item. This will show you that it is a container that occupies most of the space below our “hello world” text on the phone’s surface.

We can draw a rectangle in that container by using the left-hand toolbar. On the toolbar you’ll see a rectangle tool (as shown in Figure 2.17). Select the tool and draw a rectangle in the `ContentPanel` to create a new rectangle (also shown in Figure 2.17). If you then select the top arrow tool (or press the `V` key) you’ll be able to modify the rectangle.



FIGURE 2.16 Updating a property in Blend

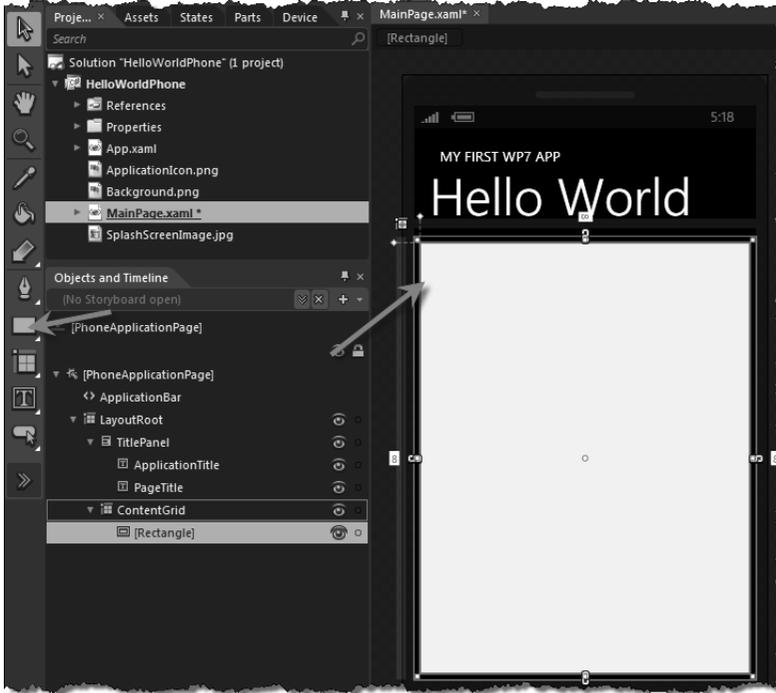


FIGURE 2.17 Drawing in a container

The rectangle you created has eight control points (the small squares at the corners and in the middle of each side). In addition, the rectangle has two small control points in the upper-left side (outside the surface area of the rectangle). These controls are used to round the corners of rectangles. Grab the top one with your mouse and change the corners to be rounded slightly, as shown in Figure 2.18.

Now that you have rounded the corners you can use the Properties pane to change the colors of the rectangle. In the Properties pane is a Brushes section showing how the different brushes for the rectangle are painted.

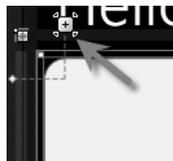


FIGURE 2.18 Rounding the corners



FIGURE 2.19 Editing brushes

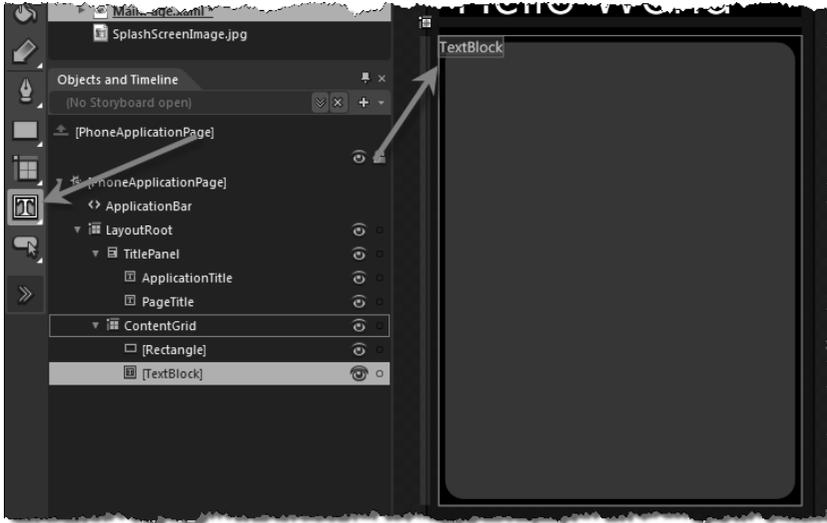
The rectangle contains two brushes: a fill brush and a stroke brush. Selecting one of these brushes will allow you to use the lower part of the brush editor to change the look of that brush. Below the selection of brush names is a set of tabs for the different brush types, as shown in Figure 2.19.

The first four tabs indicate different options for brushes. These include no brush, solid color brush, gradient brush, and tile brush. Select the stroke brush, and then select the first tab to remove the stroke brush from the new rectangle. Now select the fill brush, and change the color of the brush by selecting a color within the editor, as shown in Figure 2.20.

Now let's put some text in the middle of our design to show some data. More specifically, let's put a `TextBlock` on our design. Go back to the toolbar and double-click the `TextBlock` tool (as shown in Figure 2.21). Although we drew our rectangle, another option is to double-click the toolbar, which will insert the selected item into the current container (in



FIGURE 2.20 Picking a color

FIGURE 2.21 Inserting a `TextBlock`

this case, the `ContentPanel`). The inserted `TextBlock` is placed in the upper left of our `ContentPanel`, as also shown in Figure 2.21.

Once the new `TextBlock` is inserted, you can simply type to add some text. Type in “Status” just to have a placeholder for some text we will place later in this chapter. You should use the mouse to click on the Selection tool (the top arrow on the toolbar) so that you can edit the new `TextBlock`. You could use the mouse to place the `TextBlock` exactly where you like, but you could also use the Properties pane to align it. In the Properties pane, find the Layout section and select the horizontal center alignment and vertical bottom alignment, as shown in Figure 2.22. You might need to

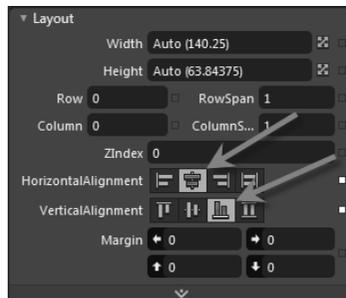
FIGURE 2.22 Centering the `TextBlock`



FIGURE 2.23 Changing the text properties

set your margins to zero as well to achieve the effect (as Blend may put a margin on your item depending on how you draw it).

Next you can edit the font and size of the `TextBlock` using the `Text` section of the Properties pane. You will likely need to scroll down to reach the `Text` section. From there you can change the font, font size, and text decoration (e.g., bold, italic, etc.). Change the font size to 36 points and make the font bold, as shown in Figure 2.23.

At this point our application does not do much, but hopefully you have gotten your first taste of the basics of using Blend for design. To get our first application to do something, we will want to hook up some of the elements with code. So we should close Blend and head back to Visual Studio.

When you exit Blend you will be prompted to save the project. Upon returning to Visual Studio your changes will be noticed by Visual Studio; allow Visual Studio to reload the changes.

Tip

Blend is great at a variety of design tasks, such as creating animations, using behaviors to interact with user actions, and creating transitions. In subsequent chapters we will delve much further into using those parts of the tool.

Adding Code

This first Windows Phone application is not going to do much, but we should get started and make something happen with the phone. Since this is your first Windows Phone application, let's not pretend it is a desktop application but instead show off some of the touch capabilities.

First, if you look at the text of the XAML you should see that the first line of text shows the root element of the XAML to be a `PhoneApplicationPage`. This is the basic class from which each page you create will derive. The `x:Class` declaration is the name of the class that represents the class. If you open the code file, you will see this code was created for you.

```
<phone:PhoneApplicationPage x:Class="HelloWorldPhone.MainPage"  
...
```

Note

The “phone” alias is an XML alias to a known namespace. If you’re not familiar with how XML namespaces work we will cover it in more detail in Chapter 3.

You will want to open the code file for the XAML file. You can do this by right-clicking the XAML page and picking View Code or you can simply press F7 to open the code file. The initial code file is pretty simple, but you should see what the basics are. The namespace and class name match the `x:Class` definition we see in the XAML. This is how the two files are related to each other. If you change one, you will need to change the other. You should also note that the base class for the `MainPage` class is the same as the root element of the XAML. They are all related to each other.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Net;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Documents;  
using System.Windows.Input;  
using System.Windows.Media;  
using System.Windows.Media.Animation;  
using System.Windows.Shapes;  
using Microsoft.Phone.Controls;  
  
namespace HelloWorldPhone  
{  
    public partial class MainPage : PhoneApplicationPage  
    {
```

```
// Constructor
public MainPage()
{
    InitializeComponent();
}
}
```

These two files (the .xaml and the code files) are closely tied to each other. In fact, you can see that if you find an element in the XAML that has a name it will be available in the code file. If you switch back to the .xaml file, click on the `TextBlock` that you created in Blend. You will notice in the Properties window that it does not have a name (as shown in Figure 2.24).

If you click where it says “<no name>” you can enter a name. Name the `TextBlock` “theStatus”. If you then switch over to the code file, you will be able to use that name as a member of the class:

```
...
public partial class MainPage : PhoneApplicationPage
{
    // Constructor
    public MainPage()
    {
        InitializeComponent();

        theStatus.Text = "Hello from Code";
    }
}
...
```

At this point, if you run the application (pressing F5 will do this) you will see that this line of code is being executed as the `theStatus` `TextBlock` is changed to show the new text (as seen in Figure 2.25).

There is an important fact you should derive from knowing that named elements in the XAML become part of the class: The job of the XAML is to

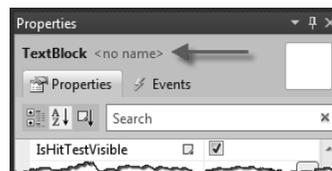


FIGURE 2.24 Naming an element in the Properties window



FIGURE 2.25 Running the application

build an object graph. The hierarchy of the XAML is just about creating the hierarchy of objects. At runtime, you can modify these objects in whatever way you want.

When you stop your application the emulator will continue to run. You can leave the emulator running across multiple invocations of your application. You should not close the emulator after debugging your application.

Working with Events

Since you are building a phone application, let's show how basic events work. You can wire up events just as easily using standard language (e.g., C#) semantics.² For example, you could handle the `MouseButtonUp` event on `theStatus` to run code when the text is tapped:

```
...
public partial class MainPage : PhoneApplicationPage
{
    // Constructor
    public MainPage()
    {
        InitializeComponent();
    }
}
```

2. For Visual Basic, you would just use the `handles` keyword instead of the C# event handler syntax.

```
theStatus.Text = "Hello from Code";

theStatus.MouseLeftButtonUp +=
    new MouseButtonEventHandler(theStatus_MouseLeftButtonUp);
}

void theStatus_MouseLeftButtonUp(object sender,
    MouseButtonEventArgs e)
{
    theStatus.Text = "Status was Tapped";
}
}
...
```

When you tap on `theStatus` the `MouseLeftButtonUp` event will be fired (which is what causes the code in the event handler to be called). All events work in this simple fashion, but the number and type of events in Silverlight for Windows Phone vary widely.

Debugging in the Emulator

If clicking the user interface was not working the way we would like, it might help if we could stop the operation during an event to see what was happening during execution. We can do this by debugging our operation. We can use the debugger to set breakpoints and break in code while using the emulator. Place the text cursor inside the event handler and press F9 to create a breakpoint. When you run the application (again, press F5) you can see that when you click on the `theStatus` `TextBlock` the debugger stops inside the event handler. You can hover your mouse over specific code elements (e.g., `theStatus.Text`) to see the value in a pop up (as shown in Figure 2.26).

Pressing the F5 key while stopped at a breakpoint will cause the application to continue running. There are other ways to walk through the code, but for now that should be sufficient to get you started. Using the emulator is the most common way you will develop your applications, but there are some interactions that are difficult to do with the emulator (e.g., multi-touch, using phone sensors, etc.) for which debugging directly on a device would be very useful. Luckily, debugging on the device is supported and works pretty easily.

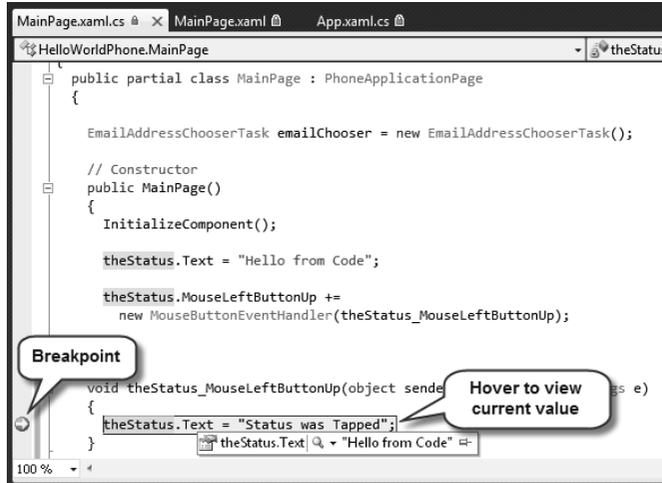


FIGURE 2.26 Using the Visual Studio debugger

Debugging with a Device

If you have a phone with which you want to do your development, you will want to be able to deploy and debug directly on the phone itself. First you need to connect your phone to your development machine. All communication with the phone is routed through the Zune software. By connecting your device to your computer, the Zune software should start automatically. If it does not start up, you can run it manually. Although the device will continue to operate normally when connected to Zune, several functions are disabled to allow Zune to synchronize media (music, photos, and videos) to the device. Consequently, these functions are using your media on the phone. Once connected, the device looks like it would normally (as seen in Figure 2.27).

All the communication that you will do to your phone (e.g., debugging, deploying, and registration) is done while Zune is running. Once you've connected your phone to your computer and run Zune, you should be able to see the phone attached, as shown in Figure 2.28.

Now that your device is connected, you can use it to sync your music, videos, and photos to the phone. However, before you can use a phone as a development device, you will need to register the phone for development.



FIGURE 2.27 Connected device

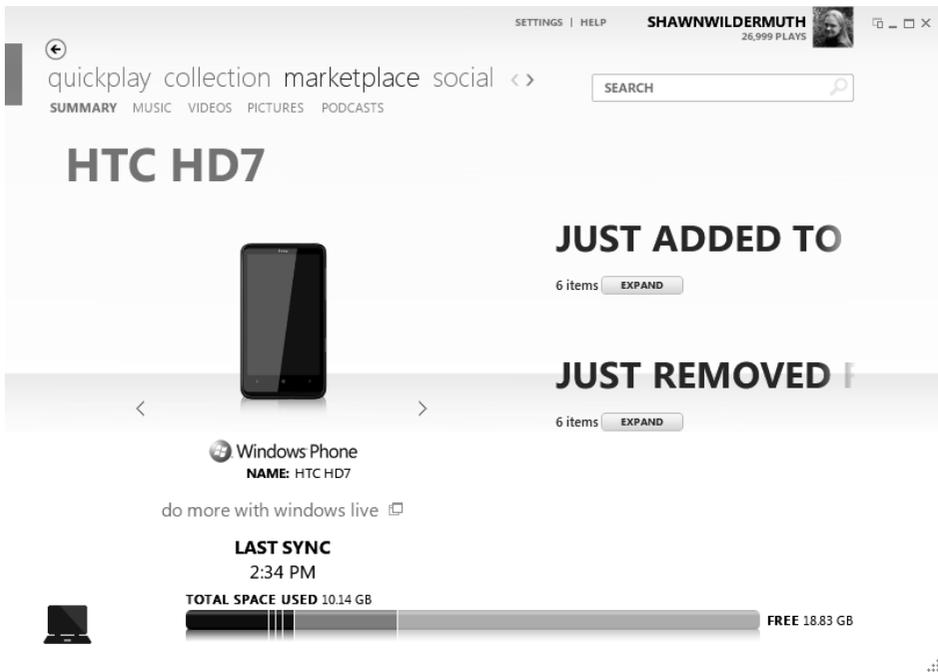


FIGURE 2.28 Your phone connected to the Zune software

This lifts the requirements that applications be signed by Microsoft, and allows you to deploy your applications directly to the phone so that you can debug applications.

Before you can enable your phone as a developer phone, you will need to have an account at the Windows Phone developer portal (<http://developer.windowsphone.com>). Once you have done that, you can enable your phone to be used for development. To do this you will need the Windows Phone Developer Registration tool, which is installed when you install the Windows Phone SDK 7.1. When you run this application it will ask you for your Windows Live ID that you used to register with the developer portal, as shown in Figure 2.29.

If your phone is successfully attached to your computer, the Status area will tell you that it is ready to register your device for development. At this point, just click the Register button to register with the developer portal. Once it registers the phone, it changes the status to show you that the phone is ready (as shown in Figure 2.30).

Now that you've registered your device, you can deploy and debug your applications using Visual Studio. The key to using the device instead



FIGURE 2.29 Registering your device



FIGURE 2.30 Successfully registered developer phone

of the emulator is to change the deployment using the drop-down list of deployment options. There are only two:

- Windows Phone Emulator
- Windows Phone Device

The drop down is located in the toolbar of Visual Studio, as shown in Figure 2.31.

Once you change the deployment target you can debug just like you did with the emulator. When you run the application, it will deploy your application to the device and run it so that you can debug it in the same way as you did with the emulator. Figure 2.32 shows the application running on a device.

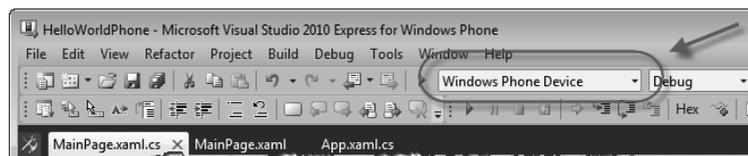


FIGURE 2.31 Changing the deployment to use a development phone

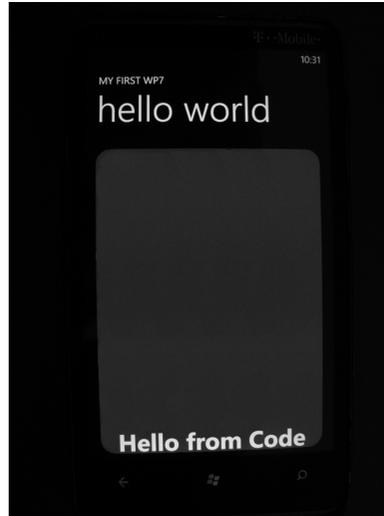


FIGURE 2.32 Running on a device

Using Touch

Even though the touch interactions do fire mouse events, there are other events that allow you to design your application for touch. Since touch is so important to how applications on the phone work, this first application should give you a taste of that experience. To show touch working, let's add an ellipse to the application that the user can move around by dragging it with her finger. To get started, you should open the MainPage.xaml file and add a new ellipse in the center of the page. To do this find the TextBlock called theStatus and place a new Ellipse element after it, like so:

```

...
<Grid x:Name="ContentGrid"
    Grid.Row="1">
    <Rectangle Fill="#FF7E0505"
        Margin="8"
        RadiusY="24"
        RadiusX="24" />
    <TextBlock HorizontalAlignment="Center"
        TextWrapping="Wrap"
        Text="Status"
        VerticalAlignment="Bottom"
        FontSize="48"
        FontWeight="Bold"
        Name="theStatus" />

```

```
<Ellipse x:Name="theEllipse"
        Fill="White"
        Width="200"
        Height="200">
</Ellipse>
</Grid>
...
```

We want to be able to move the ellipse (named `theEllipse`) as the user drags it. To allow us to do this we will need to use something called a **transform**. In Silverlight, a transform is used to change the way an object is rendered without having to change properties of the ellipse. While we could change the margins and/or alignments to move it around the screen, using a transform is much simpler. You should use a `TranslateTransform` to allow this movement. A `TranslateTransform` provides `X` and `Y` properties, which specify where to draw the element (as a delta between where it originally exists and where you want it). You can specify this transform by setting the `RenderTransform` property with a `TranslateTransform` (naming it in the process):

```
...
<Ellipse x:Name="theEllipse"
        Fill="White"
        Width="200"
        Height="200">
    <Ellipse.RenderTransform>
        <TranslateTransform x:Name="theMover" />
    </Ellipse.RenderTransform>
</Ellipse>
...
```

Now that we have a way to move our ellipse around the page, let's look at dealing with touch. In Silverlight, there are two specific types of touch interactions that are meant to allow the user to change on-screen objects. These are when the user drags her finger on the screen and when she uses a pinch move to resize objects. These types of interactions are called **manipulations**. Silverlight has three events to allow you to use this touch information:

- `ManipulationStarted`
- `ManipulationDelta`
- `ManipulationCompleted`

These events let you get information about the manipulation as it happens. For example, let's handle the `ManipulationDelta` event to get information about when the user drags on the screen. This event is called as the manipulation happens, and it includes information about the difference between the start of the manipulation and the current state (e.g., how far the user has dragged her finger):

```
...
public partial class MainPage : PhoneApplicationPage
{
    // Constructor
    public MainPage()
    {
        InitializeComponent();

        theStatus.Text = "Hello from Code";

        theStatus.MouseLeftButtonUp +=
            new MouseButtonEventHandler(theStatus_MouseLeftButtonUp);

        theEllipse.ManipulationDelta +=
            new EventHandler<ManipulationDeltaEventArgs>(
                theEllipse_ManipulationDelta);
    }

    void theEllipse_ManipulationDelta(object sender,
                                     ManipulationDeltaEventArgs e)
    {
        // As a manipulation is executed (drag or resize), this is called
        theMover.X = e.CumulativeManipulation.Translation.X;
        theMover.Y = e.CumulativeManipulation.Translation.Y;
    }

    ...
}
...
```

The event is fired while the user either pinches or drags within the `theEllipse` element. In this case the code is only concerned with the dragging. In the event handler for `ManipulationDelta`, the `ManipulationDeltaEventArgs` object contains information about the extent of the manipulation. In the `CumulativeManipulation` property of the event args, there is a property called `Translation`, which contains the extent of the drag operation (the complete delta). We are just changing

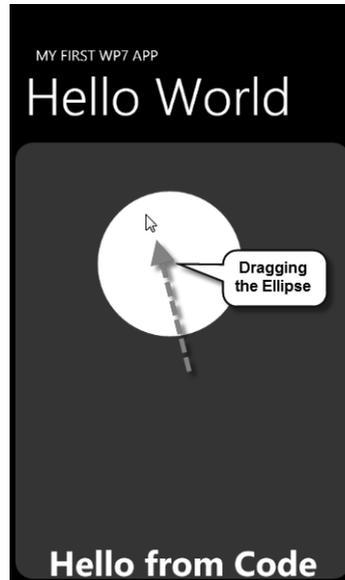


FIGURE 2.33 Dragging the ellipse

`theMover`'s properties to match the manipulation. This means we can now drag the `theEllipse` element around and see it change position under our dragging, as seen in Figure 2.33.

Working with the Phone

This first application is a program that can be pretty self-sufficient, but not all applications are like that. Most applications will want to interact with the phone's operating system to work with other parts of the phone. From within your application you may want to make a phone call, interact with the user's contacts, take pictures, and so on. The Windows Phone SDK 7.1 calls these types of interactions **tasks**. Tasks let you leave an application (and optionally return) to perform a number of phone-specific tasks. Here is a list of some of the most common tasks:

- `CameraCaptureTask`
- `EmailAddressChooserTask`
- `EmailComposeTask`

- PhoneCallTask
- SearchTask
- WebBrowserTask

These tasks allow you to launch a task for the user to perform. In some of these tasks (e.g., `CameraCaptureTask`, `EmailAddressChooserTask`), once the task is complete the user expects to return to your application; while in others (e.g., `SearchTask`) the user may be navigating to a new activity (and may come back via the Back key, but may not).

Let's start with a simple task, the `SearchTask`. Add a using statement to the top of the code file for `Microsoft.Phone.Tasks` to make sure the `SearchTask` class is available to our code file. Next, create an event handler for the `MouseLeftButtonUp` event on the `Ellipse`. Then, inside the handler for the `MouseLeftButtonUp` event, you can create an instance of the `SearchTask`, set the search criteria, and call `Show` to launch the task:

```
...
using Microsoft.Phone.Tasks;
...
public partial class MainPage : PhoneApplicationPage
{
    // Constructor
    public MainPage()
    {
        ...

        theEllipse.MouseLeftButtonUp += new
            MouseButtonEventHandler(theEllipse_MouseLeftButtonUp);
    }

    void theEllipse_MouseLeftButtonUp(object sender,
        MouseButtonEventArgs e)
    {
        SearchTask task = new SearchTask();
        task.SearchQuery = "Windows Phone";
        task.Show();
    }

    ...
}
```



FIGURE 2.34 The SearchTask in action

If you run your application, you'll see that when you tap on the `theEllipse` element it will launch the phone's Search function using the search query you supplied (as shown in Figure 2.34). The results you retrieve for the search query may vary as it is using the live version of Bing for search.

While this sort of simple task is useful, the more interesting story is being able to call tasks that return to your application. For example, let's pick an email address from the phone and show it in our application. The big challenge here is that when we launch our application, we may get tombstoned (or deactivated). Remember that, on the phone, only one application can be running at a time. In order to have our task wired up when our application is activated (remember, it can be deactivated or even unloaded if necessary), we have to have our task at the page or application level and wired up during construction. So, in our page, create a class-level field and wire up the `Completed` event at the end of the constructor for it, like so:

```
public partial class MainPage : PhoneApplicationPage
{
    EmailAddressChooserTask emailChooser =
    new EmailAddressChooserTask();
}
```

```

// Constructor
public MainPage()
{
    ...

    emailChooser.Completed += new
        EventHandler<EmailResult>(emailChooser_Completed);
}

...
}

```

In the event handler, we can simply show the email chosen using the `MessageBox` API:

```

...
void emailChooser_Completed(object sender, EmailResult e)
{
    MessageBox.Show(e.Email);
}
...

```

Now we need to call the task. To do this, let's hijack the event that gets called when the `theEllipse` element is tapped. Just comment out the old `SearchTask` code and add a call to the `emailChooser`'s `Show` method, like so:

```

...
void theEllipse_MouseLeftButtonUp(object sender,
                                    MouseButtonEventArgs e)
{
    //SearchTask task = new SearchTask();
    //task.SearchQuery = "Windows Phone";
    //task.Show();

    // Get an e-mail from the user's Contacts
    emailChooser.Show();
}
...

```

Once you run the application, a list of contacts will be shown and you will be able to pick a contact (and an address, if there is more than one), as shown in Figure 2.35. The emulator comes prepopulated with several fake contacts to test with.

Once the user picks the contact, the phone returns to your application. You will be returned to your application (and debugging will continue).

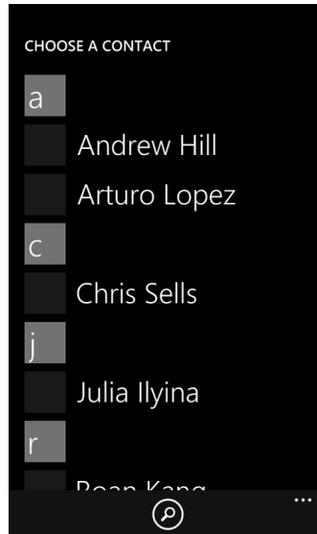


FIGURE 2.35 Choosing a contact to retrieve an email address via the `EmailAddressChooserTask`

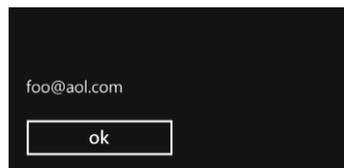


FIGURE 2.36 Showing the selected email in a `MessageBox`

The event handler should be called when it is returned to the application, as shown in Figure 2.36.

Where Are We?

You have created your first Windows Phone application using Silverlight. Although this example has very little real functionality, you should have a feel for the environment. Taking this first stab at an application and applying it to your needs is probably not nearly enough. That's why, in subsequent chapters, we will dive deep into the different aspects of the phone application ecosystem. You will learn how to build rich applications using Silverlight, services, and the phone itself.



Index

Symbols

\$expand, 391, 396–397
\$filter, 391–396
\$orderby, 391–392
\$select, 391, 397–398
\$skip, 391–392
\$stop, 391–392

A

.aac (Unprotected ISO Advanced Audio Coding) files, 227
Abort, 342–343, 358
Accelerometer, 220–225
Accept headers, 389
AcceptsReturn, 136
Access
 to alarms, 245–246
 to appointments, 233
 to camera, 284–287
 to contacts, 233
 to location information, 294–295
 to music, 266
 to network information, 373–376
 to notifications, 245–246
 Read-only, 233
 to reminders, 245–246
Accidental circular navigation, 194
Accounts of users, 237
ActionIcon, 134–135
Actions, defined, 174
Activated state, 196–200
Add notifications, 245
Add Service Reference dialogs, 383–384
Additional Tools dialog, 300–303

AddressChooserTask, 258–259
Advertising, 457–458
Age requirements, 456
Agents
 audio. *see* Audio agents
 background, 338–340
 debugging, 347
 periodic background, 340–348
 resource-intensive, 348–350
 specifying type of, 344
A-GPS (Assisted Global Positioning System), 293–294
Alarms
 accessing existing, 245–246
 creating, 242–244
 generally, 240
Albums, 266–267, 270–272
Alignment, 163
Altitude, 297
Animations. *see also* Motion
 in Expression Blend, 169–173
 in XAML, 77, 80–82
App class, 187–189
App Hub membership, 433–435
App Submission Wizard, 445–450
Appdata, 317
Appdata monikers, 331
Apple phones, 1
Application bars
 for app development, 213–215
 for application client areas, 211–212
 in Blend, 156, 176–179
 introducing, 2–5
Application class, 187–190, 197



Application idle detection mode, 216
 Application level, in brush editors, 167
 Applications. *see* Designing phone apps;
 Developing phone apps
 ApplicationSettings, 313
 Appointments, 233, 238–240
 App.xaml class, 187, 197–198
 Artboard
 animations in, 171–172
 behaviors in, 175
 in Blend, 151, 156–158
 Artist objects, 266–267
 ASP.NET, 407, 419
 Asset tools, 153–155
 Assets panel, 174
 Assisted Global Positioning System
 (A-GPS), 293–294
 Association attribute, 328
 Associations, 324–330, 333–334
 Asynchronous programming
 AsyncCallback, 402–403
 network stacks in, 370–372
 searches in, 234
 Atom Publishing Protocol (AtomPub), 387–390
 Audio agents
 changing audio tracks with, 358–360
 creating, 351–354
 playing audio with, 354–358
 types of, 351
 Universal Volume Control and, 350–351
 Audio paths, 357
 AudioSink, 285–287
 Auto sizing, 162
 AutoCompleteBox, 119–121
 AutoPlay, 226
 Axes of phone's position, 220–221, 224

B

Back buttons
 on keyboards, 11
 navigating with, 193
 requirements for, 456
 Back sides of tiles, 292
 Back stacks, 193
 Background
 agents, 338–340, 344, 347
 BackgroundAudioPlayer, 353–359
 BackgroundCreation, 109
 BackgroundServiceAgent, 344
 BackgroundTransferRequest. *see*
 Background Transfer Service (BTS)
 in panorama, 101
 URLs. *see* BackgroundImageURI
 Background Transfer Service (BTS)
 generally, 360
 monitoring requests via, 363–368
 requesting transfers via, 362–363
 requirements and limitations of, 360–361
 BackgroundImageURI
 pointing to files, 441
 requirements for, 427
 in tile layers, 424
 Bandwidth, 375
 Banking, 140, 438
 BeginSaveChanges, 402–403
 BeginTime, 243
 Behaviors, 173–176
 Beta testing, 447
 Binary streams, 372
 Binding data. *see* Data binding
 BindToShellTime, 424
 Bind
 BingMapsDirectionsTask, 249–250
 BingMapsTask, 248–249
 introducing, 120
 launchers for, 248–250
 BitmapImage, 108–109
 Blend Express. *see also* Expression Blend
 defined, 26–27, 36
 designing with, 35–43
 Objects and Timeline pane in, 38
 Properties tabs in, 37–39
 Brushes
 in Blend Express, 40–41
 in Expression Blend, 152–153, 164–169
 image, 101, 166
 resources for, 168–169
 in XAML, 72–73
 BTS (Background Transfer Service). *see*
 Background Transfer Service (BTS)
 BufferReady, 231–232
 Buffers, 284
 Built-in mouse events, 204
 Buttons
 back. *see* Back buttons
 Button, 97–98
 CameraButtons, 283–284
 hardware, 11
 HyperlinkButton, 98, 192
 icon, 177–179
 monochrome, 179
 MouseLeftButtonDown, 175

- MouseLeftButtonUp, 46–47
 - RadioButton, 98
- C**
- Cameras
 - CameraButtons, 283–284
 - CameraCaptureTask, 259–260, 280
 - generally, 280
 - PhotoCamera class, 280–284
 - for photographs. *see* Pictures
 - raw hardware access for, 284–287
 - for videos. *see* Videos
 - CanDeserialize, 310–311
 - Capability Detection, 441–442
 - Capture tasks
 - CaptureCameraTask, 272–273
 - CaptureDeviceConfiguration, 287
 - CaptureSource, 284–286, 287
 - CaptureThumbnailAvailable, 281
 - Cell-phone tower triangulation, 293
 - CellularMobileOperator, 373
 - Certification of apps
 - failing, 455–457
 - status updates during, 450–452
 - testing for, 20
 - Channels
 - alpha, 74
 - audio, 232–233
 - ChannelErrorType, 428–429
 - ChannelPowerLevel, 428–429
 - ChannelUriUpdated, 406–407
 - HttpNotificationChannel, 405–407
 - push notification, 404–407
 - URIs for, 406–410
 - Charging for apps, 435–437
 - Chat input scope, 93–95
 - CheckBox, 98, 131–132
 - Choosers
 - AddressChooserTask, 258–259
 - CameraCaptureTask, 259–260
 - EmailAddressChooserTask, 260
 - generally, 246–248, 257–258
 - PhoneNumberChooserTask, 260–261
 - PhotoChooserTask, 261–262
 - SaveContactTask, 263
 - SaveEmailAddressTask, 263–264
 - SavePhoneNumberTask, 264
 - SaveRingtoneTask, 264–265
 - Chrome, 6
 - Click events, 213–214
 - Client areas, 211–213
 - Client-side models, 398
 - Clipboard API, 287–288
 - Closing, 198
 - Code
 - adding to first app, 43–46
 - for application bar, 214
 - for attached properties, 217
 - background, 338–339
 - behaviors vs., 176
 - for containers, 66–70
 - debugging with devices, 48–52
 - debugging with emulator, 47–48
 - for events, 46–47
 - for geolocation, 297
 - for launchers, 248
 - launching, 122
 - libraries of, 154
 - for naming, 65–66
 - for new projects, 187–189
 - for panorama background images, 101
 - for playing sound, 226–227, 233
 - poorly written, 337
 - for ringtones, 265
 - for SearchTask, 56–58
 - for serialization, 312
 - for Storyboard, 80–81
 - for styling, generally, 82
 - for touch interactions, 52–55, 204–206
 - CodePlex, 119
 - Color resources, 167
 - Colors in XAML, 73–74
 - Columns
 - adding, 333–334
 - for associations, 326–329
 - attributes of, 315–316
 - in Blend, 161–164
 - version, 323
 - Command binding, 122
 - Common panel, 153
 - CommonStates, 117
 - Complex schema changes, 334
 - Computers, user experience of, 139
 - Constructors
 - for databases, 329–330
 - for isolated storage, 317
 - in XML serialization, 309
 - ConsumerID, 274
 - Consuming services
 - JSON. *see* JSON (JavaScript Object Notation)
 - OData. *see* OData (Open Data Protocol)
 - Visual Studio and, 369
 - Contacts, 233–238, 263

- Containers, 156
- ContainsText, 287
- Content
 - of alarm tiles, 243
 - ContentGrid, 161
 - ContentType, 426
 - controls for, 97–98
 - of databases, 330–331
 - policies for, 23–24
 - requirements for, 457
- Context classes
 - accessing data and, 399
 - creating, 319–320
 - optimizing, 320–324
 - saving, 326
- ContextMenu, 121–122
- Contextual keyboards, 12
- Control Library, 151
- Controls
 - binding formatting, 110
 - content, 97–98
 - converter, 111–113
 - data binding, 104–105, 113–114
 - element binding, 110–111
 - generally, 89
 - keyboard, 91–96
 - list, 98
 - Panorama, 99–102
 - phone-specific, generally, 99
 - Pivot, 102–104
 - in Properties panel, 159
 - scrolling performance, 108–110
 - in Silverlight, generally, 89–91, 119
 - simple, 91–96
 - simple data binding, 105–108
 - summary of, 138
 - templates for, 114–118
- ControlStoryboardAction, 175
- Convert to New Resource dialog, 167
- Converter controls, 111–113
- Course heading, 297
- Create, Read, Update, Delete (CRUD), 318, 388
- CreateDatabase, 318
- CreateFile, 307–308
- CreateImageSource, 238
- CreationOptions, 108
- Credit card payments, 439
- CRUD (Create, Read, Update, Delete), 318, 388
- CumulativeManipulation, 207–208

- Current property, 188, 198–199
- CurrentValueChanged, 222

D

- Dark theme, 444
- Dashboard in Marketplace, 445, 451–453
- Data binding
 - controls for, generally, 104–105
 - converters, 111–113
 - data binding errors, 113–114
 - element binding and, 110–111
 - errors in, 113–114
 - formatting, 110
 - JSON objects and, 382–383
 - scrolling performance, 108–110
 - in searching for appointments, 239
 - in searching for contacts, 235–238
 - simple, 105–108
 - templates for, 114–118
- Data panel, 151
- Databases
 - .sdf files for, 314, 329–330
 - .xap files in, 317
 - associations in, 324–330
 - Context class in, 320–324
 - copying to isolated storage, 331–332
 - CreateDatabase, 318
 - DatabaseExists, 318
 - DatabaseSchemaUpdater, 332–334
 - defined, 305–306
 - existing, 318, 330–332
 - local. *see* Local databases
 - relational, 324
 - security of, 334–335
 - updating, 332–334
- Databound apps, 151
- DataContext
 - in element binding, 111
 - for isolated storage, 316–318
 - passwords in, 334–335
 - in simple data binding, 106–108
- DataContractJsonSerializer, 311–312
- DataContractXmlSerialization, 311
- DataServiceCollection, 399–402
- DataTemplate, 108, 129–130
- DateConverter, 113
- DatePicker, 122–125
- Deactivation, 195–199
- Debugging
 - with devices, 48–52
 - with emulator, 30–32, 47–48

- Music+Videos hub, 277
 - push notifications, 419–420
 - Deep links, 453
 - DefaultTask, 189
 - DelayCreation, 109
 - DeleteAllOnSubmit, 319–320
 - DeleteOnSubmit, 319–320
 - Description fields, 447–448
 - Deserializing, 310–311, 378–379
 - Designing phone apps
 - AppBar in, 176–179
 - Blend for. *see* Expression Blend generally, 139
 - panorama apps, 146–147
 - Panorama control in, 179–182
 - panorama vs. pivot, 149
 - paradigms for, 144–145
 - phone-specific factors in, 143–144, 176
 - Pivot control in, 182–184
 - pivot-style apps, 147–149
 - previewing apps, 185
 - screens, 139–143
 - simple pages, 150
 - summary of, 185–186
 - with XAML. *see* XAML (Extensible Application Markup Language)
 - Developer Registration Tool, 50
 - Developing phone apps
 - application bars in, 213–215
 - client areas in, 211–213
 - generally, 187
 - idle detection mode in, 215–216
 - lifecycle of apps in, 187–190
 - location APIs. *see* Location APIs
 - navigation in, 190–195
 - orientation types, 200–203
 - SDK for. *see* Windows Phone SDK 7.1
 - Silverlight for. *see* Silverlight
 - summary of, 218
 - tilt effect in, 216–217
 - tombstoning in, 195–200
 - tools for, 25–26
 - touch interactions in, 203–211
 - user experience, generally, 200
 - Device panel, 185
 - DeviceExtendedProperties, 143
 - DeviceNetworkInformation, 373–376
 - Devices, debugging with, 48–52
 - Disabled, 117, 296
 - "Discovery" user experience type, 149
 - Dispatcher, 427
 - DispatcherTimer, 227–228, 230
 - DisplayedMaxLength, 135
 - Dispose, 266, 365
 - Distribution of apps, 18
 - Document tabs, 157
 - Dormant state, 195–196, 199–200
 - Double-tap interactions, 11, 211
 - DownloadString, 370–372, 377
 - Drawing, 39–40, 159–160
 - Dynamic containers, 160–164
- ## E
- EBay, 398
 - Edit links, 390–391
 - EINs (Employer Identification Numbers), 438
 - Element binding controls, 110–111
 - Ellipse element, 52–58
 - Ellipses in animations, 172
 - Email
 - EmailAddressChooserTask, 55–59, 260
 - EmailComposeTask, 250
 - SaveEmailAddressTask, 263–264
 - Employer Identification Numbers (EINs), 438
 - Emulator
 - Accelerometer tab in, 223–225
 - debugging with, 47–48
 - isolated storage in, 308
 - for motion, 224–226
 - VibrateController in, 220
 - viewing sample apps in, 146–147
 - Encryption, 334
 - Endpoints, 386, 388–390
 - EndSaveChanges, 403
 - Entity Framework, 408
 - EntityRef, 326–328
 - EntitySet, 328–330
 - Entry, 390
 - Enumerations
 - CaptureDeviceConfiguration, 287
 - ChannelErrorType, 428–429
 - ChannelPowerLevel, 428–429
 - FilterKind, 234–236
 - MediaLocationType, 243
 - MediaPlaybackControls, 253–254
 - NetworkInterfaceSubType, 375
 - PlayState, 358
 - RecurrenceType, 243
 - SupportedPageOrientation, 202
 - TaskResult, 257–258
 - TransferPreferences, 363
 - TransferStatus, 367

Errors. *see also* Debugging
 in data binding, 113–114
 ErrorOccurred, 427
 in Odata transactions, 403
 in push notifications, 427–429

Ethernet network connections, 376

Events. *see also specific events*
 Click, 213–214
 coding first, 46–47
 Initialized, 280–281
 mouse, 204
 notification, 419
 touch, 205, 211
 Unobscured, 213

Existing databases, 330–332

ExpanderView, 133–134

Expansion, 401

Experience of users. *see* User experience

ExpirationTime, 243

Expression Blend. *see also* Blend Express
 animations in, 169–173
 ApplicationBar in, 176–179
 basics of, generally, 159
 behaviors in, 173–176
 brushes in, 164–169
 creating new projects in, 150–151
 defined, 26, 36
 designing with, generally, 150
 layout in, 159–164
 orientation types in, 203
 Panorama control in, 179–182
 Pivot control in, 182–184
 tour of, 151–159

ExtendedTask, 343–344

Extensible Application Markup Language (XAML). *see* XAML (Extensible Application Markup Language)

Extensible Markup Language (XML). *see* XML (Extensible Markup Language)

Extensions, 274–275

F

Facebook, 237

Failed submissions, 454–457

FallbackValue, 110

Feed locations, 398–399

Fill, 165

Filtering data, 234–237

FinalVelocities, 209–210

Find method, 405–406

Find notifications, 245

Fixed sizing, 162

Flash, 281–282

Focus, 117, 282–283

FontSizeSlider, 111

Forbidden APIs, 340

Forbidden headers, 361

Foreign keys, 328

Foursquare website, 140–142

FrameReported, 204–205

FrameworkDispatcher, 227–229, 268

Free version of apps, 436–437

FromUri static method, 269

G

Games
 databases and, 328–329
 Game class, 318–321, 326
 loops, 227–228, 231
 XNA for. *see* XNA game developers toolset

Genre objects, 266–267

GeoCoordinateWatcher class, 294–299

Geolocation, 295–303

GeoTrust, 435

Get requests
 GetActions, 245
 GetImage, 270–271
 GetPicture, 238
 GetPictureFromToken, 276
 GetPrimaryTouchPoint, 205–206
 GetResponse, 412–413
 GetText, 287–288
 GetThumbnailImage, 270–271
 GetTouchPoints, 205–206
 GetUserStoreForApplication, 307
 in OData, 391

Global Positioning System (GPS), 293

GoBack, 192–194

Google phones, 1–2

Google searches, 120

GPS (Global Positioning System), 293

Gradient brushes, 166

Gravity, 220

Grid, 156, 161–163

Groups, 129–131

GUIDs for alarms, 243

Gutters, 161–163

Gyroscopes, 220

H

HandleResponse, 413–414

Haptic feedback, 219, 226

Hard-copy tax invoices (HCTIs), 439

- Hardware
 - access for cameras, 284–287
 - buttons, 11
 - inputs, 6–9
- HCTIs (hard-copy tax invoices), 439
- Header, 104, 133
- Header status codes, 415–418
- Hierarchy of objects, 156
- Hint, 134
- History in Music+Videos hub, 276–278
- Hold touch event, 211
- Horizontal Accuracy, 297
- HTTP (HyperText Transfer Protocol)
 - HttpNotificationChannel, 405–406
 - HttpNotificationEventArgs, 419
 - HttpNotificationReceived, 419
 - HTTPWebRequest, 410
 - HttpWebRequest, 372
 - HTTPWebResponse, 410, 412
 - HttpWebResponse, 372
 - OData over, 388–389
 - status codes, 414–418
- Hubs
 - App, 433–435
 - HubType="1," 277
 - introducing, 4–5
 - Live Tiles on, 16–17
 - Marketplace as, 251, 431–432
 - Music+Videos, 274–280
 - picture, 266, 274–276
- HyperlinkButton, 98, 192
- HyperText Transfer Protocol (HTTP). *see* HTTP (HyperText Transfer Protocol)
- I**
- Icons
 - ActionIcon, 134–135
 - buttons for, 177–179
 - IconPath, 441
 - in.xap files, 123–124
- IDictionary<string, object>, 198
- IDisposable, 266, 307, 365
- Idle detection mode, 215–216
- IEnumerable
 - introducing, 98
 - MediaLibrary and, 267
 - for playing songs, 270
 - searching for contacts with, 235
- IGrouping<T,T>, 130–131
- IList, 98
- Images
 - background generally, 101
 - backgroundImageURI, 424, 427, 441
 - BitmapImage, 108–109
 - brushes for. *see* Brushes
 - CaptureImage, 281
 - CaptureImageAvailable, 281–282
 - GetImage, 270–271
 - GetThumbnailImage, 270–271
 - ImageSource, 109, 238
 - ImageStream, 278
 - for Marketplace submissions, 442–443
 - performance of, 109
 - in XAML, 75–77
- Implicit styles, 86–87
- Indexes, 316, 333–334
- Individual Taxpayer Identification Numbers (ITINs), 438
- Inertia, 209–210
- Initialized events, 280–281
- Initializing status, 296
- INotifyPropertyChanged, 107, 320–323
- INotifyPropertyChanging, 320–323
- Input patterns
 - generally, 9
 - for hardware buttons, 11
 - for keyboards, 11–13
 - for sensors, 13–14
 - for touch interfaces, 10–11
- Input scope, 93–95
- Instance, 353
- Integrating with phones
 - alarms, 240–246
 - appointments, 233, 238–240
 - cameras, 280–287
 - choosers for tasks, 257–265
 - clipboard APIs, 287–288
 - contacts, 233–238
 - generally, 219
 - launchers for tasks, 248–257
 - Live Tiles, 288–293
 - media, generally, 266
 - motion, 220–226
 - music, 266–270
 - Music+Videos hub, 276–280
 - notifications, 245–246
 - picture hubs, 266, 274–276
 - pictures, 270–273
 - reminders, 240–246
 - sound, 226–233
 - summary of, 303
 - tasks, generally, 246–247
 - vibration, 219–220
- Integration of user experience. *see* User experience
- Interfaces, 375

Internal Revenue Service (IRS), 438
 International pricing and taxes, 435–436
 Internet services. *see* Services
 IRS (Internal Revenue Service), 438
 IsApplicationInstancePreserved property, 199
 IsCellularDataEnabled, 373
 IsForeignKey, 326–328
 IsNetworkAvailable, 373
 ISO MPEG-1 Layer III (.mp3 files), 227
 Isolated storage
 copying databases to, 331–332
 databases in, 305–308
 IsolatedStorageFile, 307–308
 Isostore, 317
 NotifyComplete writing to, 342
 settings for, 312–314
 Isostore moniker, 332
 IsScheduled, 245
 IsTrial, 437
 IsUnknown, 297
 IsVersion = true, 323
 IsWifiEnabled, 373
 Item Tools, 158
 ItemsSource
 in AutoCompleteBox, 120
 introducing, 98
 in LongListSelector, 130
 specifying collections with, 134
 text changes and, 120–121
 ItemTemplate, 108, 127
 ITINs (Individual Taxpayer Identification Numbers), 438
 IValueConverter, 111

J

JPEG files, 278
 JSON (JavaScript Object Notation)
 generally, 376–377
 Json.NET, 377–382
 parsing, 379–383
 serialization, 311–312, 377–379
 Web, 383–387

K

Key strings, 279–280
 Keyboards, 11–13, 91–96

L

Lambda expressions, 371
 Landscape orientations, 101–102, 201–202
 Latitude, 297

Launchers
 BingMapsDirectionsTask, 249–250
 BingMapsTask, 248–249
 EmailComposeTask, 250
 generally, 246–248
 LaunchForTest, 347, 349
 Launching by users, 198
 MarketplaceDetailTask, 250–251
 MarketplaceHubTask, 251
 MarketplaceReviewTask, 252
 MarketplaceSearchTask, 252
 MediaPlaybackControls, 253–254
 MediaPlayerLauncher, 252–253
 PhoneCallTask, 254–255
 SearchTask, 255
 ShareLinkTask, 255–256
 ShareStatusTask, 256
 SmsComposeTask, 256–257
 WebBrowserTask, 257
 Launching requirements, 456
 Layout, 156, 159–164
 Legal usage policies, 21
 Length indicators, 135–136
 LicenseInformation, 437
 Lifecycles
 in app submissions, 451
 of apps, 14–15, 187–190
 Light theme, 444
 LinearVelocity, 210
 LINQ
 introducing, 130–131
 MediaLibrary and, 267–268
 OData queries using, 401
 parsed JSON and, 381–383
 searching for contacts with, 235–236
 to SQL, 314–315, 319–320, 323
 List controls, 98
 ListBox
 customizing, 108
 in data binding, 113
 in element binding, 111
 introducing, 98
 in searching for appointments, 239
 in searching for contacts, 236
 ListPicker, 125–127
 Lite version of apps, 436–437
 Live Tiles
 dual-sided, 292–293
 generally, 288–289
 introducing, 2, 16–17
 main, 289–290
 push notifications and, 423–427

- secondary, 290–292
 - updating with push notifications, 404
- Local databases
- associations in, 324–330
 - Context class in, 320–324
 - defined, 305–306
 - existing, 330–332
 - generally, 314
 - schema updates for, 332–334
 - security of, 334–335
 - starting, 314–320
 - summary of, 335
- Location APIs
- accessing location information, 294–295
 - emulating location information, 300–303
 - generally, 293
 - one-time geolocation, 295–297
 - permissions, 293–294
 - tracking geolocation changes, 297–299
- Locations
- of animation objects, 172–173
 - APIs for. *see* Location APIs
 - requirements for, 456
 - using emulator for, 224
- Lock screens, 215
- Locked, running while phone is, 456
- Logic, 176
- Logical clients, 2–4, 211–212
- Long-hold keys, 92–93
- Longitude, 297
- LongListSelector, 127–131
- Looping, 104, 230
- M**
- MainPage.xaml, 187–189
- Managed objects, 382–383
- Manifest files, 189–190
- ManipulationDelta, 207–209
- Manipulations, 53–55, 206–210
- Maps, 300–303
- Margins, 160–164
- Marketplace
- advertising, 457–458
 - App Hub membership in, 433–435
 - charging for apps in, 435–437
 - dashboard of, 451–453
 - defined, 431–432
 - distribution by, 18
 - failed submissions to, 454–457
 - generally, 18
 - MarketplaceDetailTask, 250–251
 - MarketplaceHubTask, 251
 - MarketplaceReviewTask, 252
 - MarketplaceSearchTask, 252
 - modifying apps on, 453–454
 - payment by, 438
 - policies of, 20–24
 - preparing apps for, 439–445
 - submitting apps to, 19–20, 439, 445–451
 - summary of, 458
 - tax information, 438–439
 - workings of, 432–435
- Marquee in Music+Videos hub, 276–277
- Master-detail interfaces, 149
- Media
- integrating with phones, generally, 266
 - Media Library.SavePicture, 272–273
 - MediaElement, 270
 - MediaHistoryItem, 277–280
 - MediaHistoryNew, 279
 - MediaLibrary, 266–268, 270–275
 - MediaPlaybackControls, 253–254
 - MediaPlayer, 268–270
 - MediaPlayerLauncher, 252–253
 - MediaSource, 267
- Memory
- background tasks using, 340
 - consumption of, 143
 - MemoryStream, 230–232
 - requirements for, 456
 - storage and, 198
- Menus, 121–122, 177–179
- Metro
- application bars in, 2–5
 - chrome, 6
 - hubs in, 4–5
 - introducing, 2
 - keyboards in, 12–13
 - Live Tiles in, 2
 - logical clients in, 2–4
 - monochrome buttons in, 179
 - panorama apps in, 5–6
 - Start screens in, 2–3
 - styles of apps in, 150
 - system trays in, 2–4
 - touch interactions in, 10–11
- Microphones, 230–233
- Microsoft
- Blend Express by. *see* Blend Express
 - Expression Blend by. *see* Expression Blend
 - on hardware inputs, 6–9
 - Marketplace of. *see* Marketplace
 - Microsoft. Xna.Framework.dll, 228
 - Microsoft.Device.Sensors.dll, 221

- Microsoft (*continued*)
 - Microsoft.Phone.Tasks, 246
 - Microsoft.Phone.Controls, 99
 - Microsoft.Phone.Shell, 211
 - phone services by, 15–16
 - on process execution, 14
 - Push Notification Service of, 23, 403–404, 410–411
 - SDK (Software Developer Kit) by. *see* Windows Phone SDK 7.1
 - "three screens and a cloud" idea of, 139
 - Windows Media Audio (.wma) files, 227
 - Windows Phone by, 1
 - XNA by. *see* XNA game developers toolset
- Mobile operator billing, 439
- Modifying published apps, 453–454
- Motion. *see also* Animations
 - adding recorded data to emulator, 225–226
 - emulating, 223–225
 - integrating, 220–223
- Mouse events
 - in Blend, 204
 - MouseLeftButtonDown, 175
 - MouseLeftButtonUp, 46–47
 - .mp3 (ISO MPEG-1 Layer III) files, 227
- Multitasking
 - audio agents for. *see* Audio agents
 - background agents for, 338–340
 - background transfers and. *see* Background Transfer Service (BTS)
 - in developing phone apps, 196–197
 - introducing, 14, 337–338
 - periodic background agents for, 340–348
 - resource-intensive agents for, 348–350
 - summary of, 368
- Multitouch interactions, 10
- Music
 - accessing, 266
 - emulator, 266–268
 - picker, 147–149
 - playing, 268–270
- Music+Videos hub
 - debugging, 277
 - generally, 266, 276–277
 - History section of, 278
 - introducing, 7
 - launching, 279–280
 - New section of, 279
 - Now Playing section of, 277–278
- N**
- Names in XAML
 - assigning, 64–66
 - changing, 189
 - namespaces, 64–65, 99, 103
 - Panorama control and, 99
 - Pivot control and, 103
- Navigation
 - in developing phone apps, 190–195
 - Framework, 200
 - Navigate, 192
 - Navigate(), 194
 - NavigationContext, 195
 - NavigationContext property, 275
 - NavigationPage, 189
 - NavigationService, 190–195
 - NavigationUri, 244–245
 - NavigationUri property, 291–292
 - OnNavigatedFrom, 193–195
 - OnNavigatedTo, 193–194, 279
 - OnNavigatingFrom, 194
 - page-based, 190–191
 - styles of apps and, 144–147
- .NET open source library
 - ASP.NET, 407, 419
 - Json.NET, 377–382
 - network stacks and, 372–373
 - serialization and, 377
- Network stacks
 - accessing network information, 373–376
 - generally, 370
 - WebClient class, 370–372
- NetworkAvailabilityChanged, 374–375
- NetworkInterfaceInfo, 375
- New Project dialogs, 28–29, 150–151
- New section, in Music+Videos hub, 276, 279
- NoData status, 296
- Notifications
 - accessing existing, 245–246
 - alarms as, 240
 - NotifyComplete, 342–343, 358
 - reminders as, 241–242
- Now Playing section, 277–278
- O**
- Object properties in XAML, 63–64
- Object tools in Blend, 153
- Objects and Timeline panel
 - adding ApplicationBar, 177–179
 - animations in, 169–172
 - behaviors in, 175
 - in Blend, generally, 151, 155–156
 - introducing, 38
 - Panorama control in, 180–182
 - Pivot control in, 183
- ObjectTrackingEnabled, 323–324

- Obscuring apps, 213
- OData (Open Data Protocol)
 - \$expand, 391, 396–397
 - \$filter, 391, 392–396
 - \$select, 391, 397–398
 - \$skip, 391–392
 - \$top, 391–392
 - consuming, generally, 387–388
 - retrieving data, 399–401
 - service references for, 398–399
 - updating data, 401–403
 - URIs in, 389–392
 - using on phones, 398
 - workings of, 388–389
- OnCaptureStarted, 285–286
- OnCaptureStopped, 285–286
- OneTime data binding, 107
- One-time geolocation, 295–297
- OneWay data binding, 107
- OnFormatChange, 285–286
- OnPlayStateChanged, 357–358
- OnSample, 285–286
- OnUserAction, 355
- Opacity, 172
- Open Data Protocol (OData). *see* OData (Open Data Protocol)
- OpenFile, 308
- OpenRead, 372
- OpenWrite, 372
- Operating systems, 26
- OperationResponse, 403
- Options page, 194
- Orientation
 - emulating motion and, 224
 - introducing, 137
 - of pages, 202–203
 - types of, 200–203
- OtherKey, 328–329
- Output windows, 113–114
- P**
- Page, 156
- Page-based navigation, 190–191
- Panels, 151
- Panorama apps
 - construction of, 146–147
 - Foursquare example of, 142
 - introducing, 5–6
 - as project type, 151
- Panorama control
 - in Blend, 179–182
 - introducing, 99–102
 - Pivot vs., 149
 - user interface, 181–182
- Paradigms for designing apps, 144–145
- Payment for apps, 432, 438
- PCM WAV (.wav) files, 227–228
- PerformanceProgressBar, 131–132
- Periodic background agents, 340–348
- Permissions, 293–294
- Personal information requirements, 457
- Phone apps
 - designing. *see* Designing phone apps
 - developing. *see* Developing phone apps
- Phones
 - Apple, 1
 - controls specific to. *see* Phone-specific controls
 - Google, 1–2
 - integrating functions on. *see* Integrating with phones
 - Windows. *see* Windows Phone 7.5
- Phone-specific controls
 - binding formatting, 110
 - converters, 111–113
 - data binding, 104–105, 113–114
 - element binding, 110–111
 - generally, 99
 - Panorama, 99–102
 - Pivot, 102–104
 - scrolling performance, 108–110
 - simple data binding, 105–108
 - templates for, 114–118
- Pictures. *see also* Cameras
 - of contacts, 238
 - generally, 270–272
 - hub for, 266, 274–276
 - PhotoCamera class, 280–284
 - PhotoChooserTask, 261–262
 - PictureAlbum class, 271–272
 - Pictures property, 270
 - resolution of, 281
 - storing, 272–273
- Pinch gestures, 209
- Pinned tiles, 423
- Pitch, 230
- Pivot control
 - in Expression Blend, 182–184
 - introducing, 102–104
 - as project type, 151
 - styles of apps and, 145

- Pivot-style apps, 147–149
- Platforms, 24
- PlayCurrentTrack, 356
- PlayerContext property, 278–279
- Playlist objects, 266–267
- PlayState, 357–359
- PNG (Portable Network Graphics) files, 179
- PNS (Push Notification Service). *see* Push Notification Service (PNS)
- Policies for apps, 20–24
- Portable Network Graphics (PNG) files, 179
- Portrait orientation, 201, 224
- PositionChanged, 296–299
- PowerLevelChanged, 428–429
- Prepping phone for apps, 25–27
- PresentationContainer, 116
- Preview buffers, 284
- Previewing apps, 185
- Pricing apps, 435, 449, 454
- Primary keys, 315, 323–326
- Private beta testing, 447
- ProgressBar, 91, 131–132
- Project menu, 185
- Projects panel, 153–154
- Properties panel
 - behaviors in, 175
 - in Blend, 151, 158–159
 - sizing objects in, 163–164
- Properties tab in Blend, 37–39
- Properties window in Visual Studio, 33–35
- PropertyChanging, 330
- Publication, 449–450, 452
- Publisher, 325–328
- Push Notification Service (PNS)
 - generally, 403–404
 - server setups and, 410–411
 - Windows Phone and, 23
- Push notifications
 - debugging, 419–420
 - error handling, 427–429
 - generally, 403–404
 - Live Tiles and, 423–427
 - preparing apps for, 405–407
 - raw, 410–419
 - requirements for, 404–405, 456
 - server setup for, 407–410
 - toast notifications vs., 419–423

Q

- Quality assurance, 20
- Query options, 391
- Quota for storage, 306

R

- RadioButton, 98
- RaisePropertyChanged, 323
- RaisePropertyChanging, 323
- Raw hardware access, 284–287
- Raw notifications, 404, 410–419
- ReadObject, 312
- Read-only access, 117, 233
- Ready status, 296, 299
- RecentPlay, 278
- Recorded data, 224–226
- Rectangle, 165
- RecurrenceType, 243
- Red-eye reduction, 281
- References for services, 384–387
- RefreshBinding, 364–366
- Registration, 50–51, 349
- Related entity links, 391
- Relational databases, 324, 387
- Relative to UIElement, 205–206
- Release build, 444
- Reminders
 - accessing existing, 245–246
 - creating, 244–245
 - generally, 240–242
- Removing
 - back entries, 193
 - completed requests, 366–367
 - notifications, 245
- RenderTransform, 173
- Replace notifications, 245
- Requests, 361, 363–368
- Re-sizing objects, 160–164
- Resource-intensive agents, 348–350
- Resources
 - brushes, 167–169
 - for colors, 167–168
 - dictionary option, 167
 - panel, 151
 - sharing in XAML, 83–84
- Response codes, 415–418
- REST
 - AtomPub and, 388
 - Json.NET and, 378
 - in services, 369
- Retrieving data, 399–401
- RichTextBox, 95–96
- Ringtones, 264–265
- Roaming, 375
- Rodriguez, Jamie, 203
- RootFrame, 189
- RootPictureAlbum property, 270–271

- Rows, 161–164
 - Rowspan, 163–164
 - RSS feeds, 290–291
 - Rules.xml files, 442
 - Running state, 195–196
- S**
- Saving
 - SaveContactTask, 238, 263
 - Saved state, 196
 - SavedPictures, 273
 - SavedPictures property, 270
 - SaveEmailAddressTask, 263–264
 - SavePhoneNumberTask, 264
 - SaveRingtoneTask, 264–265
 - tombstone state, 199
 - ScaleTransform, 209
 - Schedule Task Agent projects, 340–345,
349–350
 - ScheduledActionService, 243–245, 349
 - Scheduling background tasks, 340
 - Schema updates, 332–334
 - Screens, 139–143
 - Scrolling performance controls, 108–110
 - .sdf (standard database format) files, 314,
329–330
 - SDK (Software Developer Kit). *see* Windows
Phone SDK 7.1
 - SearchAsync, 239
 - Searching
 - for contacts, 234–237
 - SearchAsync, 234–235
 - SearchTask, 55–57, 255
 - Security, 334–335
 - SelectedItem, 113
 - Selection tools, 152–153
 - Sensors, 13–14, 221–226
 - Separator, 121
 - Serialization
 - of JSON, 376–379
 - of objects, 198
 - XML, 308–314
 - Server setup, 407–410
 - Server-side code, 407
 - Service references, 398–399
 - Services. *see also* Web
 - for consuming JSON, 383–387
 - for consuming OData. *see* OData (Open
Data Protocol)
 - driving development, 15–16
 - generally, 369
 - network stacks, 370–376
 - push notifications. *see* Push notifications
 - summary of, 429
 - Setters, 321–323, 328
 - SetText, 287
 - Shake, 225–226
 - Shapes in XAML, 71–72
 - Share pickers, 274
 - ShareLinkTask, 255–256
 - ShareStatusTask, 256
 - ShellTile, 289–292
 - ShellToast, 347
 - ShellToastNotificationReceived, 421
 - Shutter keys, 283–284
 - Silverlight. *see also* .xap (Silverlight
Application) files
 - AutoCompleteBox in, 119–121
 - Blend Express for, 35–43
 - code for, generally, 43–46
 - ContextMenu in, 121–122
 - controls in, 89–91, 119
 - DatePicker in, 122–125
 - debugging with devices, 48–52
 - debugging with emulator, 47–48
 - events in, 46–47
 - ExpanderView in, 133–134
 - generally, 25, 59, 119
 - ListPicker in, 125–127
 - LongListSelector in, 127–131
 - navigation in, 190
 - PerformanceProgressBar in, 131–132
 - for Phone Toolkit. *see* Windows Phone
Toolkit
 - phone-specific tasks and, 55–59
 - PhoneTextBox in, 134–136
 - prepping phone for, 25–27
 - for SDK. *see* Windows Phone SDK 7.1
 - testing apps in, 185
 - TimePicker in, 122–125
 - ToggleSwitch in, 132–133
 - touch interactions, 52–55, 91
 - Visual Studio 2010 Express and, generally,
27–32
 - WrapPanel in, 136–138
 - XAML for, generally, 32–35, 44–46
 - Simple controls, 91–96
 - Simple data binding controls, 105–108
 - Simple pages, 150
 - Single touch interactions, 10
 - Single-page apps, 145
 - Sinks, 285–287

- SIPs (Software input panels), 91–93
 - Size
 - of animation objects, 172–173
 - of devices, 141–143
 - of objects, 160–164
 - policies on, 361
 - of .xap files, 456
 - Slider, 91, 111
 - SmsComposeTask, 256–257
 - Snooze function, 240–241
 - Social Security Numbers (SSNs), 438
 - SOCKS proxies, 414
 - Software Developer Kit (SDK). *see* Windows Phone SDK 7.1
 - Software input panels (SIPs), 91–93
 - Solid color brushes, 165
 - Songs
 - integrating on phones, 266–267
 - MediaPlayer for, 269–270
 - playing, 356
 - URI for, 357
 - Sound
 - adjusting playback, 229–230
 - of alarms, 243
 - integrating, generally, 226
 - playing with MediaElement, 226–227
 - playing with XNA, 228–229
 - recording, 230–233
 - XNA libraries for, 227–228
 - SoundEffect class
 - adjusting playback with, 229–230
 - playing sounds with, 228–229, 232–233
 - Source property, 227
 - Specifications of Windows Phone 7.5, 7–9
 - Speed, 297
 - SplashScreenImage.jpg, 442
 - SQL Server Compact Edition, 314–317, 330
 - SSNs (Social Security Numbers), 438
 - Stability requirements, 456
 - Stacked notifications, 242
 - StackPanel
 - in element binding, 111
 - in simple data binding, 106
 - WrapPanel and, 136–137
 - StandardTileData, 290, 292
 - Star sizing, 162
 - Start screens, 2–3
 - States, 173, 198–199
 - Status of submissions, 451
 - StatusChanged, 295–298
 - Storage
 - of data, generally, 198, 305–306
 - isolated, 305–308, 312–314
 - JSON serialization for, 311–312
 - serialization for, generally, 308
 - summary of, 335
 - XML serialization for, 308–311
 - Storyboards, 169–173
 - Streams
 - binary, 372
 - for images, 238
 - for isolated storage, 308
 - memory, 230
 - Stretch alignment, 164
 - StringFormat, 110
 - Stroke, 165
 - Styling in XAML. *see also* XAML (Extensible Application Markup Language)
 - generally, 82–83
 - Implicit styles, 86–87
 - resource sharing, 83–84
 - Style object default properties, 84–86
 - SubmitChanges, 318–320
 - Submitting apps to Marketplace. *see* Marketplace
 - Supported networks, 361
 - SupportedPageOrientation, 202
 - Suspended state, 195–196, 199–200
 - System trays, 2–4, 211–212
 - System.Device, 293–294
 - System.IO stack, 306–307
 - System.Runtime.Serialization, 311
 - System.Xml.Serialization, 309
- ## T
- Tables, 324–329, 333
 - Tap touch event, 211
 - TargetNullValue, 110
 - Tasks
 - apps for managing, 337
 - choosers, 257–265
 - defined, 55–56
 - generally, 246–247
 - launchers, 248–257
 - in manifest files, 189
 - TaskID, 274
 - TaskResult, 257–258
 - Tax information, 438–439
 - Templates, 114–118
 - Testing apps, 185
 - Text
 - on Clipboard, 287–288
 - properties, 43
 - TextBlock, 38–43, 47, 52–53

- TextChanged, 120–121
 - in XAML, 74–75
 - TextBox
 - binding to Name property, 105–106
 - in element binding, 111
 - introducing, 89–90
 - as simple control, 91
 - visual state managers in, 117
 - Third screen, 139–143
 - This document option, 167
 - ThisKey, 328–329
 - "Three screens and a cloud," 139
 - Thumbnails, 270–271
 - Tiles. *see* Live Tiles
 - Tilt, 216–217, 221–222
 - TimeBetweenUpdates, 222
 - Timeline, 169–172
 - TimePicker, 122–125
 - Timestamps, 223–224
 - Titles, 100, 244
 - Toast notifications, 404, 419–423
 - Toasts, 347
 - ToggleSwitch, 132–133
 - Tokens, 275–276, 441
 - Tombstoning
 - in developing phone apps, 195–200
 - introducing, 14–15
 - multitasking and, 338
 - Toolbars, 151–153
 - Toolbox, 29–31
 - Touch interactions
 - in Blend, 203–211
 - generally, 10–11
 - in Silverlight, 52–55, 91
 - Touch class, 204
 - TouchFrameEventArgs class, 205
 - TouchPoint, 205–206
 - Track, 359
 - TrackEnded, 358
 - Tracking geolocation changes, 297–299
 - TrackReady, 358
 - TransferProgressChanged, 366
 - Transferring
 - Background Transfer Service for, 362–368
 - locations, 361
 - preferences, 363
 - protocols for, 361
 - Transformations in XAML, 77–80
 - Transforms, 207–209
 - TranslateTransform, 52–53, 207–209
 - Transparency, 172, 179
 - Trial version of apps, 437
 - TVs, 139
 - Twitter, 376, 378–379, 398
 - TwoWay data binding, 107
- ## U
- U. S. (United States). *see* United States (U.S.)
 - UI thread, 223, 224
 - UIElement, 205–206, 210–211
 - Uniform Resource Identifiers (URIs). *see* URIs (Uniform Resource Identifiers)
 - Uniqueness of Windows Phone, 1–6
 - United States (U.S.)
 - advertising in, 457–458
 - charging for apps in, 435–436
 - taxes in, 438
 - Universal Volume Control (UVC), 350–351
 - Unlocking phones, 434
 - Unobserved events, 213
 - Unprotected ISO Advanced Audio Coding (.aac) files, 227
 - Updating
 - in CRUD, 318
 - data, 401–403
 - databases, 332–334
 - push notifications in, 404
 - schema, 332–334
 - tiles, 290–293, 404
 - TimeBetweenUpdates, 222
 - Uploading
 - with Background Transfer Service, 338, 360–368
 - Marketplace submissions, 433, 443, 447
 - text to servers, 372
 - toast messages, 423
 - URIs (Uniform Resource Identifiers)
 - backgroundImageURI, 424, 427, 441
 - BackgroundTransferRequest and, 362
 - channel, 406–410
 - fsong, 357
 - isolated storage, 317
 - NavigationUri, 244–245, 291–292
 - OData, 389–392
 - push notification, 406–410
 - User experience
 - application bars in, 213–215
 - client areas in, 211–213
 - generally, 200
 - idle detection mode in, 215–216
 - integration of, 6–7
 - interfaces for, 152
 - orientation types, 200–203
 - tilt effect in, 216–217

User experience (*continued*)
 touch interactions in, 203–211
 UserPreferences, 310, 312
 User idle detection mode, 215
 Using instances, 399–400
 UTF-8, 411
 UVC (Universal Volume Control), 350–351

V

Value method, 379
 Valued Added Tax (VAT) identification
 numbers, 438–439
 Vector3 class, 221–222
 Velocity, 209–210
 Verb mappings, 388
 Vertical Accuracy, 297
 Vertical control stacks, 137–138
 Vibration, 219–220
 Videos. *see also* Cameras
 hub for. *see* Music+videos hub
 MediaPlayerLauncher for, 252–253
 VideoBrush, 73, 280, 285
 VideoSink, 285–287
 Zune for, 48
 View tools, 152–153
 Visibility of controls, 159
 Visual containers, 66–70
 Visual grammar
 animations, 77, 80–82
 brushes, 72–73
 colors, 73–74
 generally, 70
 images, 75–77
 shapes, 71–72
 text, 74–75
 Transformations, 77–80
 Visual state managers (VSMs), 117–118, 173
 Visual Studio 2010 Express
 Blend and, 36–37, 43
 coding apps with, generally, 26–27
 creating new projects in, 27–32
 debugging in, 48–51, 319–320
 emulator in, 308
 hotkeys in, 185
 isolated storage in, 308
 launching projects in, 185
 OData service references in, 398–399
 Output windows, 113
 previewing apps in, 185
 Web services and, 383–387
 XAML designer in, 35
 Visual Studio Professional, 369, 383, 387

Volume, 230
 VSMS (visual state managers), 117–118, 173

W

.wav (PCM WAV format) files, 227, 228
 Waypoints, 301–302
 WCF (Windows Communication
 Foundation), 407
 Weather apps, 384–387
 Web. *see also* Services
 user experience of, 139–140, 144
 WebBrowser, 116
 WebBrowserTask, 257
 WebClient class, 370–372
 WebClient class, 376–377
 WiFi Positioning System, 293
 Windows Communication Foundation
 (WCF), 407
 Windows Live, 369
 Windows Phone 7.5
 app lifecycles and, 14–15
 coding apps for. *see* Visual Studio 2010
 Express
 controls specific to. *see* Phone-specific
 controls
 designing apps for, generally. *see*
 Designing phone apps
 designing apps with XAML. *see* XAML
 (Extensible Application Markup
 Language)
 developing apps for, generally. *see*
 Developing phone apps
 developing apps with Silverlight. *see*
 Silverlight
 factors specific to, 143–144
 hardware inputs on, 6–9, 11
 integrating functions on. *see* Integrating
 with phones
 introducing, 1–2
 keyboards on, 11–13
 Live Tiles on. *see* Live Tiles
 Marketplace for apps on. *see* Marketplace
 numbers for, 264
 PhoneApplication Phone class, 214
 PhoneApplicationFrame, 192
 PhoneApplicationPage, 193–195, 202
 PhoneApplicationService, 197–199
 PhoneApplicationService class, 215–216
 PhoneCallTask, 254–255
 PhoneNumberChooserTask, 260–261
 PhoneTextBox, 134–136
 as platform, 24

- polices for apps on, 20–24
 - prepping for apps, 25–27
 - sensors in, 13–14
 - services driving development for, 15–16
 - software for apps on. *see* Windows Phone SDK 7.1
 - specifications of, 7–9
 - tasks specific to, 55–59
 - touch interfaces on, 10–11
 - uniqueness of, 1–6
 - user experiences on, 6–7
 - windows in, 190
 - Windows Phone Application Certification Requirements, 143
 - Windows Phone Developer Registration tool, 50
 - Windows Phone Developer Tools, 25
 - Windows Phone Marketplace. *see* Marketplace
 - Windows Phone SDK 7.1
 - accelerometer API and, 221
 - advertising and, 457
 - appointments in, 233
 - background transfers in. *see* Background Transfer Service (BTS)
 - Blend in. *see* Expression Blend
 - CapabilityDetection.exe in, 441–442
 - choosers in, 246–247, 257–265
 - contacts in, 233
 - content controls in, 96–98
 - controls in, generally, 90–91
 - DatabaseSchemaUpdater class in, 332–334
 - DataContractJsonSerializer class in, 311
 - DeviceNetworkInformation class in, 373–376
 - downloading, 25–27
 - entry points in, 233
 - installing, 26–27
 - launchers in, 246–247, 248–257
 - list controls in, 98
 - location permission and, 293–294
 - Marketplace submissions and, 433
 - phone-specific controls in, 99–104
 - raw hardware access in, 284–287
 - simple controls in, 91–96
 - tasks in, generally, 55–59, 246
 - Windows Phone Toolkit
 - AutoCompleteBox in, 119–121
 - ContextMenu in, 121–122
 - controls in, generally, 119
 - DatePicker in, 122–124
 - ExpanderView in, 133–134
 - ListPicker in, 124–127
 - LongListSelector in, 127–131
 - PerformanceProgressBar in, 131–132
 - PhoneTextBox in, 134–136
 - tilt effect in, 216–217
 - TimePicker in, 122–124
 - ToggleSwitch in, 132–133
 - WrapPanel in, 136–138
 - .wma (Windows Media Audio) files, 227
 - WManifest.xml files. *see also* XML (Extensible Markup Language)
 - audio agents in, 352
 - completing, 440–441
 - Live Tiles in, 288–289, 423–424
 - Music+Videos hub in, 277
 - new scheduled agents in, 343–344
 - picture hubs in, 274–275
 - push notifications in, 404–405
 - WPConnect.exe tool, 277
 - WrapPanel, 136–138
 - WriteAcquiredItem, 279
 - WriteObject, 312
 - WriteRecentPlay, 278
- ## X
- XAML (Extensible Application Markup Language)
 - animations in, 77, 80–82
 - in Blend, 157
 - brushes in, 72–73
 - colors in, 73–74
 - defined, 61–62
 - generally, 61
 - images in, 75–77
 - implicit styles in, 86–87
 - introducing, 32–35, 44–46
 - namespaces in, 64–65
 - naming in, 65–66
 - object properties in, 63–64
 - resource sharing in, 83–84
 - shapes in, 71–72
 - styling in, 82–86
 - summary of, 87
 - text in, 74–75
 - transformations in, 77–80
 - visual containers in, 66–70
 - visual grammar in, generally, 70
 - .xap (Silverlight Application) files. *see also* Silverlight
 - alarm sounds in, 243
 - audio in, 357
 - background agents for, 339, 347

- .xap (Silverlight Application) files
 - (*continued*)
 - databases in, 317
 - defined, 19–20
 - icons in, 123
 - installing manually, 431
 - media in, 252–253
 - panoramas and, 101
 - searching in, 192
 - size of, 456
 - songs in, 269
 - submitting to Marketplace, 19–20, 433, 447
 - X-axis, 220–221
 - X-DeviceConnectionStatus, 414
 - XML (Extensible Markup Language). *see also*
 - WMAAppManifest.xml files
 - AtomPub as, 387, 389
 - creating recorded data with, 225–226
 - Live Tiles in, 424–426
 - Rules.xml files, 442
 - serialization in, 308–311
 - toast messages in, 421–423
 - XNA game developers toolset
 - accelerometer API and, 221
 - file extensions for, 19
 - libraries in, 227–233
 - as platform, 24
 - playback adjustments in, 229–230
 - playing sounds with, 226, 228–229
 - project templates in, 28
 - recording sounds in, 230–233
 - X-NotificationClass, 411, 414, 426
 - X-SubscriptionStatus, 414
 - X-WindowsPhone-Target, 423, 426
- ## Y
- Y-axis, 220–221
- ## Z
- Z-axis, 220–221
 - Zoom gestures, 209
 - Zune
 - in Marketplace, 431–432, 453
 - in Music+Videos hub, 276–277
 - software, 48–49, 147