Dear Windows Phone 7.5 Unleashed reader,

Welcome to the ranks of the many developers trying to achieve richness or fame in this new and exciting platform. It has been a year since Windows Phone 7 launched (and 1.5 years since it was announced). You are joining the ranks of thousands of registered developers who delivered more than 30,000 apps within the first year after launch. Windows Phone developers are having a lot fun because we have the best tools and, (especially) with this Windows Phone “Mango” release, one of the best platforms. Don’t worry if you are just getting started; the business opportunity is still in its infancy. You are still on time (if you hurry), and you are in luck because you now have in your hands (or on your screen) one of the best tools for delivering compelling, well-architected, maintainable Windows Phone applications. That tool is this book: Windows Phone 7.5 Unleashed.

I have known Daniel, your author, for about four years. We first met as part of the WPF Disciples, an elite group of XAML experts that spent countless hours passionately discussing and sharing patterns and best practices for .NET and Silverlight application development. Within the group, Daniel is well respected for his knowledge and recognized for his always optimistic attitude toward raising the bar (both on quality and experience) on projects he works on. As usual, he has raised the bar with this epic 16-month writing effort.

Windows Phone 7.5 Unleashed is much more than a great introduction to Windows Phone development. Besides providing a comprehensive view of the platform, where this book excels is in striking the perfect balance between great samples, good and detailed explanations, insightful tips, and—most importantly—real-world experiences and real-world best practices. While other Windows Phone books simply introduce a concept like the application bar, Daniel provides great wrappers or abstractions to improve the development experience. Other books might oversimplify a sample to keep a chapter focused on the topic, but Daniel delivers a useful, real-world example that is more interesting and comprehensive. I can’t say he keeps it short—at about 1,200 pages this is not a brief book—but it is an easy read because of its focus and carefully editorialized relevance. I could go on citing examples of Daniel’s thoroughness (maybe one more since I am confident few books will have unit testing or input validation chapters), but instead I will let you get to the task at hand: building your Windows Phone apps.

Enjoy the journey!

Happy Windows Phone coding!

Jaime Rodriguez
Principal Evangelist, Microsoft
P.S. To Daniel (and the Sams team):

I know you spent more than a year writing this book and when facing tough decisions, such as what to cut or whether to hold off the book to cover Mango instead of releasing a book that was going to be out-of-date at print, you always made the right calls for the readers. The book shows it.

Congratulations!
Preface

Windows Phone OS is Microsoft’s new mobile phone operating system. It is a substantial departure from Microsoft’s previous Windows Mobile technology, as it provides the capability to develop applications for the mobile phone using either Silverlight or XNA. Silverlight, in particular, is a technology that has seen rapid adoption for multimedia web development and desktop line of business applications. Extending the Silverlight development experience to the phone was welcome news to many Silverlight developers because it opened up many exciting opportunities. Silverlight’s key advantage is that it brings increased productivity to developers.

Windows Phone offers developers the opportunity not only to target the phone OS but also to build cross-platform games and applications that run on the desktop and in the browser: on Windows, Linux, and the Mac. In addition, Windows Phone supports integration with XBox Live and Zune services.

Many features make Windows Phone compelling, such as the phone's built-in services for geographic location, multitouch capabilities, and hubs, which combine both local and online content. Most importantly, the tooling and platform make it fun to develop for.

Windows Phone has a friendlier tiled interface than its predecessor and has been designed around the Metro UI philosophy. The Metro UI philosophy had its origins in the ill-fated Zune media devices. The large typography and fluid scrolling lists make Metro beautiful and highly suited to mobile devices. We now see the Metro language making it to the tablet and desktop environment with Windows 8. Likewise, the application marketplace paradigm will also accompany the release of Windows 8, and like Windows Phone, Windows 8 Metro apps will operate in a sandboxed environment, with fewer capabilities than their fully trusted desktop counterparts. The sandboxing of Windows Phone and Windows 8 Metro apps ensures that the user never has a bad experience with an app. What does this have to do with Windows Phone, you may ask. It is important to recognize the evolution of these various technologies, to have an eye on the future, and to prepare for the likely convergence of the technologies. A total convergence of Windows Phone and Windows desktop OSs may happen at some point. Yet, until it does, both OSs will continue to drive innovation in the other. New features of the Windows Metro UI will undoubtedly make their way to the phone OS over time, and vice-versa. There was, and continues to be, some uncertainty about Silverlight, especially on the browser. Silverlight on Windows Phone, however, is alive and well, and with the upcoming release of Windows 8 and the incorporation of the Metro UI in Windows 8, demand for developers with skills in this area is set to rise.

Over the course of writing this book I witnessed the initial release of the Windows Phone OS 7.0 and the subsequent release of Windows Phone OS 7.5 (Mango). Development of Windows Phone apps using Silverlight is a large topic, and one that increased broadly with version 7.5. Many new features (more than 500 according to Microsoft) were added to the OS, and these had to be covered for the release of the book, adding several more months of writing time. The new features of Mango are, however, compelling, and they unlock many new and exciting opportunities on the platform.
Scope of This Book

This book targets Windows Phone 7.5 Mango. While you see some examples incorporating XNA, this book’s focus is squarely on Silverlight for Windows Phone. The book covers all main areas of the topic in a deep, yet easily comprehensible way, using practical examples with a real-world context. The goal is to provide you with the concepts and techniques that will help you to design and develop well-engineered and robust Windows Phone apps.

Throughout this book you see a small number of techniques and custom code applied to make developing phone apps easier. It is not the intention to make what you learn in the book harder to reach, but, on the contrary, the techniques are tried and tested approaches that, once familiar, will help you build more testable and maintainable apps. The competition between apps on the platform is set to intensify as the number of apps in the Windows Phone marketplace increases. This competition will not only bring a “long tail,” where independent developers find evermore niche categories to create apps for, but will also require apps competing in the more popular categories to increase their feature sets. As apps become more complex, maintainability comes to the forefront, and greater attention to managing complexity is required.

This book is not a Silverlight beginner’s book. Although there is considerable reference material for some essential Silverlight infrastructure, included within its chapters are also advanced topics such as the Model-View-ViewModel design pattern (MVVM). In fact, most sample apps follow the MVVM pattern. The concepts and techniques used throughout the book are described in Chapter 2, “Fundamental Concepts in Silverlight Development for Windows Phone.” Do not worry if some of these approaches seem foreign to you; by the end of the book they will be second nature.

Wherever possible you are provided with tips and techniques that go beyond the topic, and you will frequently find content not easily found elsewhere. A lot of custom code is provided that extends the Windows Phone SDK to support real app scenarios.

Assumptions about the Reader

If you are an experienced developer who has basic experience in Silverlight or WPF concepts, looking to transfer your skills to Windows Phone, this book is for you. It is assumed that you are familiar with C#, XAML, and Visual Studio.

Book Structure

The book is divided into five parts:

- Part I, “Windows Phone App Development Fundamentals”
- Part II, “Essential Elements”
- Part III, “Windows Phone App Development”
- Part IV, “Building Windows Phone Data Driven Applications”
- Part V, “Multitasking”
Most chapters have sample apps. Chapter 2 is required reading to understand the techniques used throughout the book and the samples.

Some chapters, in particular Chapter 9, “Silverlight Toolkit Controls,” are not intended to be read from end to end but rather are intended as references that you may refer back to when you need to learn about a particular topic within the chapter.

**Code Samples**

To demonstrate each concept, this book contains more than 100 samples. The sample code for this book can be downloaded from http://informit.com/title/9780672333484.

All code is in C#. The project structure is divided into topic areas. To view a particular sample, you can run the main solution and select the sample page from the index (see Figure P.1).

![Figure P.1 The sample code index page](image)

In the downloadable sample code there are several solutions. In most cases, the DanielVaughan.WindowsPhone7Unleashed.sln is used. Some topics, however, have been separated into separate solutions due to technical constraints.

Much of the infrastructure code presented in the book has been consolidated into the Calcium open-source project. You can find more information about the Calcium SDK at http://calciumsdk.net.

A note about the code snippets printed in the book: Occasionally, when a line runs too long for the printed page, a code-continuation arrow (➥) has been used to indicate the line continuation.
About the Author

Daniel Vaughan is cofounder and president of Outcoder, a Swiss software and consulting company dedicated to creating best-of-breed user experiences and leading-edge back-end solutions, using the Microsoft stack of technologies—in particular Silverlight, WPF, WinRT, and Windows Phone.

Daniel is a technical advisory board member of PebbleAge, a Swiss finance company specializing in business process management.

He is a Microsoft MVP for Client Application Development, with more than a decade of commercial experience across a wide range of industries including finance, e-commerce, and multimedia.

He is a Silverlight and WPF Insider, a member of the elite WPF Disciples group, and three-time CodeProject MVP. Daniel is also the creator of a number of open-source projects, including Calcium and Clog.

Daniel blogs at http://danielvaughan.org, where he publishes articles and software prototypes.

He has a degree in Computer Science from UNE, where he received various awards including the Thomas Arnold Burr Memorial Prize in Mathematics, and twice the annual School of Mathematics, Statistics and Computer Science Prize.

With his wife, Daniel runs the Windows Phone Experts group on LinkedIn at http://linkd.in/jnFoqE

Originally from Australia and the UK, Daniel is based in Zurich, Switzerland, where he lives with his wife, Katka.
Dedication

To my wonderful wife, Katka.

Acknowledgments

Brennon Williams, for first suggesting to Sams Publishing that I write the book and for his inspiration and encouragement along the way.

Jaime Rodriguez, for answering my numerous questions about the SDK and the future of the platform.

My friends Sacha Barber and Peter O’Hanlon, for technically reviewing some chapters.

Katka Vaughan, for endless advice, proofreading, and formatting. Your contribution and unending patience made this book possible.

Laurent Bugnion, for being my “author buddy” and for answering my numerous authoring-related questions.

The great folks at PebbleAge—in particular Olivier Parchet and Christian Kobel—for their encouragement and goodwill.

Microsoft, for answering questions and building great tools.

The terrific team at Sams, especially Neil Rowe, for guidance throughout the entire process, technical editor J. Boyd Nolan, for going over my code with a fine-toothed comb, and Mark Renfrow and Betsy Harris.

For inspiration and support (in no particular order):

David Anson, Cliff Simpkins, John Papa, Davide Zordan, Marlon Grech, Glenn Block, Charles Petzold, Erik Mork, Jeremy Likness, Rene Schulte, Josh Smith, Corrado Cavalli, Colin Eberhardt, Jeff Wilcox, all the WPF Disciples.
We Want to Hear from You!

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

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

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

E-mail: feedback@samspublishing.com
Mail: Neil Rowe
      Executive Editor
      Sams Publishing
      800 East 96th Street
      Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
This page intentionally left blank
Understanding the events within the life cycle of a Windows Phone application is critical to providing an optimal user experience on the phone. The phone’s single application process model means that your app may be interrupted and terminated at any time. It is your responsibility to maintain the appearance of continuity across interruptions, to save your app’s state whenever an interruption occurs, and, if necessary, restore the state when the user returns to your app.

While the 7.5 release of Windows Phone OS includes support for fast application switching, where your app is kept in memory and its threads suspended, you still need to preserve the state of your app because there are no guarantees that an app will not be terminated if the device memory runs low.

Like localizability, app state preservation is an aspect of development that should not be deferred and is likely one of the biggest challenges you will face as a Windows Phone developer.

Traditionally, developers of Windows desktop applications have not been overly concerned with persisting runtime state. There is usually no need to maintain state for a desktop application, since the application remains in execution and resident in memory while it is open. This is in stark contrast to Windows Phone apps, where an application may be stopped and started many times during a single session.

Seasoned ASP.NET developers who recall the days before AJAX may feel slightly more at home developing for Windows Phone, with ASP.NET’s reliance on view state to...
overcome the transient nature of page state in which the lifespan of a web page is limited to the period before a postback occurs. This is not too dissimilar to the state model of the phone, although Silverlight has nothing like the view state system built into ASP.NET. For that you need to roll your own, and you see how to build an automated state preservation system in Chapter 25, “Isolated Storage and State Preservation.”

There is no doubt that the single application process model of the phone presents some challenge for developers, but it may also lead to better designed and more robust applications, with an emphasis on decoupling visual elements from their state so that it can be more readily preserved.

This chapter begins with an overview of the application execution model and examines the various application life cycle events, which are used to coordinate state persistence and restoration.

You see how to enable an app to run under the lock screen. You also look at page navigation and how to optimize the user experience by using a splash screen or a loading indicator.

Finally, the chapter delves into the sample application and looks at image caching, design-time data, and consuming a simple WCF service.

**Exploring the Execution Model**

The execution model of Windows Phone is designed to make the phone as responsive as possible and to maximize the battery life of the device. One way that this is achieved is by limiting the phone to a single running application. Multiple applications running in the background risk slowing the foreground application and may tie up the processor and cause the phone to consume more power.

**NOTE**

While the phone’s execution model is limited to a single app being in execution at any time, Windows Phone allows the use of background tasks, which run periodically and are independent of your foreground app. These are explored in Chapter 27, “Scheduled Actions.”

In addition to greater responsiveness and extended battery life, the execution model provides users with a consistent navigation experience between applications. On Windows Phone, users are able to launch applications from the App List screen or from a tile on the Start Experience. The hardware Back button allows users to navigate backward, through the pages of a running application or through the stack of previously running applications.

The goal of transient state preservation and restoration is to provide the user with a simulated multiple application experience, where it seems to the user that your application was left running in the background, even though it may have been terminated by the operating system.
**Application State**

There are two types of application state: persistent and transient. Persistent state exists when an application launches. It is saved to a private storage area called isolated storage and may include data such as configurable settings or files.

Transient state is discarded when an application is closed. It is stored at the application level in the `Microsoft.Phone.Shell.PhoneApplicationService.State` dictionary or at the page level in the `PhoneApplicationPage.State` dictionary.

There is a single `PhoneApplicationService` instance for the entire app, and its state dictionary should be used only by objects running in the context of the application as a whole. A unique state dictionary is created for each page in your app, and you should use it rather than the `PhoneApplicationService.State` dictionary whenever possible.

**NOTE**

The `PhoneApplicationPage.State` dictionary is accessible only during or after the `OnNavigatedTo` method is called, or during or before the `OnNavigatedFrom` method is called. If you attempt to access it too early or too late an exception is raised.

The `PhoneApplicationPage.State` dictionary is limited to 2MB for each page and 4MB for the entire app. You should, therefore, not use it for excessive storage.

Transient state may include results from web service calls, or data from partially completed forms (see Figure 3.1).

**Life Cycle Events**

The `Microsoft.Phone.Shell.PhoneApplicationService` exposes four life cycle related CLR events, which provide an application with the opportunity to save or load state (see Figure 3.2).
CHAPTER 3  Application Execution Model

FIGURE 3.2  Application life cycle

**Launching Event**
When a user selects an application from the App List screen, or from a tile on the Start Experience, or when the application is being debugged, the application moves from the stopped state, to the running state. This represents a cold start. Use the **Launching** event to restore any persistent state from isolated storage that is not page specific. This event occurs after the App class is instantiated, but before the main page of an application is created.

**NOTE**
The Windows Phone 7 Certification Requirements state that an application must render the first screen within 5 seconds after launch and activation, and be fully responsive within 20 seconds. A splash screen can be used to offset startup delay. Later in this chapter you see how to create a splash screen for an application.

The **Launching** and **Activated** events are mutually exclusive. That is, exactly one of these two events occurs when the application is being started. Likewise, the **Deactivated** and
Closing events are also mutually exclusive; only one of these events occurs when the application is exiting.

**Subscribing to Life Cycle Events Using the PhoneApplicationService**

The PhoneApplicationService allows your app to be notified of the various life cycle events. The PhoneApplicationService is, by default, initialized in XAML by the application’s App instance, as shown in the following example:

```xml
<Application.ApplicationLifetimeObjects>
  <!--Required object that handles lifetime events for the application-->
  <shell:PhoneApplicationService
      Launching="Application_Launching" Closing="Application_Closing"
      Activated="Application_Activated" Deactivated="Application_Deactivated"/>
</Application.ApplicationLifetimeObjects>
```

The PhoneApplicationService is a singleton class, which exposes an instance of the class via a static property called Current. Code to handle the Launching, Closing, Activated, and Deactivated events can be placed in the App class, for which handlers are created when a new project is created, or directly in code by using the Current property of the PhoneApplicationService.

**NOTE**

Subscription to the PhoneApplicationService.Activated event must occur via the App class. If subscription to the event is done in a UI element, the handler will not be called. This is because the PhoneApplicationService itself subscribes to the System.Windows.Application.Current's Startup event. When the application raises the Startup event, the PhoneApplicationService notifies the operating system that it is ready to receive execution model events (that is, Launching, Closing, Activated, and Deactivated). This causes the PhoneApplicationService.Activated event (or the Launched event in the case of a non-tombstoned application) to occur almost immediately. Therefore, subscription to the event after this point has no effect. Moreover, event subscription is lost when an application is tombstoned; the application has, after all, terminated at that point. Thus, subscription to the Activated or Launched events from, for example, the MainPage constructor, will occur after the event has already been raised.

There may be times when it is tempting to promote certain kinds of transient state to persistent state. When launching your app, however, the user should feel like he is not resuming your app, but rather that it is indeed a new instance, a clean slate.

Persistent state is stored in isolated storage, while transient state is stored in the PhoneApplicationService.State dictionary, an IDictionary<string, object> that is maintained while the application is tombstoned, but abandoned when the application moves from the tombstoned state to the not running state.

**Deactivation Event and Tombstoning**

On entering the running state, an application must contend with being interrupted. Each interruption causes the PhoneApplicationService.Deactivated event to be raised. The app
is then placed in a dormant state, where it remains in memory but its threads are suspended.

If an app is reactivated after being in the dormant state, there is no need for your app to restore its transient state, reducing its load time.

Detecting whether an app is returning from a dormant state can be done within the PhoneApplicationService.Activated event handler using the IsApplicationInstancePreserved property of the ActivatedEventArgs, as shown in the following excerpt:

```csharp
void Application_Activated(object sender, ActivatedEventArgs e)
{
    if (e.IsApplicationInstancePreserved)
    {
        /* Application was placed in the dormant state. */
    }
    else
    {
        /* Application state should be restored manually. */
    }
}
```

When the device's memory usage reaches a minimum threshold, the operating system may decide to tombstone your app.

**NOTE**

When an application is tombstoned, it is terminated.

When tombstoned, the operating system is aware that the application may be reactivated. If an application moves from being tombstoned back to the running state, it will be from a cold start, and all objects must be instatiated and persistent and transient state must be restored. The only differences between the tombstoned state and the closed state are that when tombstoned, the operating system retains the transient state dictionary for the app along with an identifier for the app, so that if activated, chooser and launcher events can be resubscribed. Launchers and choosers perform common tasks, such as sending email. You learn more about choosers and launchers in Chapter 12, “Launchers and Choosers.”

An application is deactivated when it is no longer the foreground application. The following is a list of causes for deactivation:

- The user presses the start button.
- The phone’s lock screen is engaged without having enabled running under the lock screen. Enabling your app to run under the lock screen is discussed in the section “Running Under the Lock Screen” later in the chapter.
A launcher or a chooser is shown.

The user selects a toast notification, which launches another application.

**Saving Transient State**

The Deactivated event provides an application with the opportunity to save its transient and persistent state.

**NOTE**

Saving the transient state of a PhoneApplicationPage should be performed in its OnNavigatedFrom method, as you see later in the chapter.

The goal is to enable restoration of the application to its prior state before being tombstoned. It should be assumed, however, that when the Deactivated event occurs, the application is going to be closed, moving to the closed state. The user may, after all, opt not to resume the application, or may use the Start Experience to relaunch the application, rather than using the hardware Back button to return to the application. Moreover, if the user launches many other apps, your app may get bumped off the end of the Back button application stack.

The Visual Studio new project templates place an empty handler for the Deactivated event in the App class. See the following excerpt:

```csharp
void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
    /* Save transient state like so:
        * PhoneApplicationService.Current.State["DataContractKey"]
        * = DataContract;
    */

    /* Save persistent state like so:
        * IsolatedStorageSettings.ApplicationSettings["Key"] = someObject; */
}
```

You can also subscribe to the Deactivated event elsewhere, in the following manner:

```csharp
PhoneApplicationService.Current.Deactivated += OnDeactivated;

void OnDeactivated(object o, DeactivatedEventArgs args)
{
    ...
}
```
CAUTION
The operating system gives an app 10 seconds when the PhoneApplicationService.Closing event occurs, before it is forcibly terminated. If the time required to save your app’s state exceeds this amount, then its state should be saved periodically, and perhaps incrementally, while it is running.

NOTE
The Windows Phone emulator terminates an application if it takes longer than 10 seconds to display its first visual. Therefore, when debugging an Activated or Launched event, if this time is exceeded the application exits and the debugger detaches before any UI elements can be shown.

Transient State Requirements
All objects to be stored in the PhoneApplicationService.State property must meet one of the following requirements:

- It is a primitive type.
- It is a known serializable reference type including decimal, string, or DateTime, with a matching System.Convert.ToString method signature.
- It is capable of being serialized using a DataContractSerializer. To achieve this, it must be decorated with a System.Runtime.Serialization.DataContract attribute. Each property or field intended for serialization must be decorated with theDataMember attribute, and in turn each serializable property or field type must be decorated with theDataContract attribute.

Storing an application’s transient state can be difficult because objects containing state often have event subscriptions to or from other objects that are not serialized. Also, types from third-party libraries are usually not decorated with the DataContract attribute, preventing serialization.

Restoring Transient State
When an app transitions from being tombstoned or dormant, back to the running state, the PhoneApplicationService.Activated event is raised. This provides an opportunity to restore the transient and persistent state of the app.

NOTE
Restoring the transient state of a PhoneApplicationPage should be performed in its OnNavigatedTo method, as you see later in the chapter.

Restoring the transient state involves taking the user to the point where she was when the Deactivated event occurred, and may involve restoring the positions of UI elements,
repopulating viewmodel properties, and so on. The goal is to provide the user with a seamless experience, and to emulate a multiple application-like environment, so that to the user the application appears as though it was left running in the background.

The following code demonstrates handling of the `PhoneApplicationService.Activated` event, to restore transient and persistent state:

```csharp
void Application_Activated(object sender, ActivatedEventArgs e)
{
    /* Restore persistent state like so: */
    someObject = IsolatedStorageSettings.ApplicationSettings["Key"];  

    /* Restore transient state like so: */
   DataContract = PhoneApplicationService.Current.State["DataContractKey"];  
}
```

**Saving Persistent State**

Persistent state is usually stored whenever transient state is stored. In addition, your app should save its persistent state when it is closing, by subscription to the `PhoneApplicationService.Closing` event. Persistent state may include files or application settings, as shown in the following excerpt from the `App` class:

```csharp
void Application_Closing(object sender, ClosingEventArgs e)
{
    System.IO.IsolatedStorage.
    IsolatedStorageSettings.ApplicationSettings["someObject Key"]
    = someObject;
}
```

**NOTE**

Transient state should not be retained when the Closing event occurs.

---

**Running Under the Lock Screen**

Users expect some kinds of apps to run under a lock screen. These apps include music players, mapping apps, stop watches, and so on.

In the first release of the Windows Phone OS, running under the lock screen was favorable to apps that wanted to avoid being tombstoned. These were apps that were slow to load or relied on complex state models. This was alleviated, however, with the introduction of fast application switching in Windows Phone 7.5. Now apps are placed in a dormant state and remain in memory.
Running under the lock screen is not an alternative to implementing efficient transient state persistency. Recall that an app may still be tombstoned if the phone runs low on memory.

The following steps outline how to enable your app to run under the lock screen:

2. Detect when the lock screen is engaged, or disengaged, by handling the `PhoneApplicationFrame.Obscured` and `Unobscured` events, respectively.
3. When the lock screen is engaged your app should reduce its processing to a bare minimum to minimize CPU usage and thus battery consumption.
4. When the lock screen is disengaged your app should resume from where it left off.
5. Optionally, you should prompt the user to allow him or her to opt-in to running under the lock screen, and/or provide an options setting for enabling or disabling running under the lock screen.

**Lock Screen Management**

I created a reusable class called `LockScreenManager` that makes it easy to manage your app’s lock screen policy. The class implements a custom `ILockScreenManager` interface that has the following three properties:

- **RunningUnderLockScreen**—Gets a value indicating whether the app is running under the lock screen
- **RunningUnderLockScreenEnabled**—Allows you to set whether the app is allowed to run under the lock screen
- **UserPrompted**—Allows your app to remember whether the user has been prompted to allow running under the lock screen

At your app’s first launch you query the `UserPrompted` property. If false, you present a dialog asking the user whether it is okay to run under the lock screen, and you set the `RunningUnderLockScreenEnabled` property accordingly. You subscribe to the `PropertyChanged` event of the `LockScreenManager`, and when the `RunningUnderLockScreen` property changes, it indicates that the lock screen has been either engaged or disengaged. `LockScreenManager` is a singleton and subclasses `NotifyPropertyChangedBase` for property change notification (see Listing 3.1). The private constructor attempts to retrieve the `UserPrompted` and `RunningUnderLockScreenEnabled` property values from isolated storage settings. It then subscribes to the `PhoneApplicationFrame.Obscured` and `Unobscured` events.

When the `Obscured` event is raised the `RunningUnderLockScreen` property is set.
The `LockScreenManager` class is located in the Shell directory of the WindowsPhone7Unleashed project.

**LISTING 3.1 LockScreenManager Class (excerpt)**

```csharp
public class LockScreenManager : NotifyPropertyChangeBase, ILockScreenManager
{
    static readonly string promptedKey = "UserPromptedToAllowRunningUnderLockScreen";
    static readonly string runningEnabledKey = "RunningUnderLockScreenEnabled";

    LockScreenManager()
    {
        IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;
        bool prompted;
        if (settings.TryGetValue(promptedKey, out prompted))
        {
            UserPrompted = prompted;
        }
        bool enabledValue;
        if (settings.TryGetValue(runningEnabledKey, out enabledValue))
        {
            RunningUnderLockScreenEnabled = enabledValue;
        }

        var frame = (PhoneApplicationFrame)Application.Current.RootVisual;
        frame.Obscured += (o, args) => RunningUnderLockScreen = args.IsLocked;
        frame.Unobsurred += (o, args) => RunningUnderLockScreen = false;
    }
}
```

When either of the `UserPrompted` or `RunningUnderLockScreenEnabled` properties is set, its new value is saved to isolated storage settings using a `SaveSetting` method, as shown:

```csharp
void SaveSetting(string key, object value)
{
    IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;
    settings[key] = value;
    settings.Save();
}
```
When the `RunningUnderLockScreenEnabled` property is enabled the idle detection mode is disabled, which allows the app to run under the lock screen. If disabled, the app must be restarted or deactivated before the idle detection mode can be enabled or an `InvalidOperationException` is raised. This is a limitation of the phone OS. See the following excerpt:

```csharp
public bool RunningUnderLockScreenEnabled
{
    get
    {
        return runningUnderLockScreenEnabled;
    }
    set
    {
        var result = Assign(() => RunningUnderLockScreenEnabled,
            ref runningUnderLockScreenEnabled, value);

        if (result == AssignmentResult.Success)
        {
            if (runningUnderLockScreenEnabled)
            {
                PhoneApplicationService.Current.ApplicationIdleDetectionMode
                    = IdleDetectionMode.Disabled;
            }
/* Idle detection mode cannot be enabled
   until the application is restarted. */

            SaveSetting(runningEnabledKey, runningUnderLockScreenEnabled);
        }
    }
}
```

The `LockScreenView` page and its associated `LockScreenViewModel` class demonstrate the use of the `LockScreenManager`, and are located in the ExecutionModel directory of the WindowsPhone7Unleashed.Examples project. The `LockScreenViewModel` uses the `MessageService` to ask the user whether she wants to opt-in to running under the lock screen. When the manager's `RunningUnderLockScreen` property changes, a string is written to the Visual Studio Output view (see Listing 3.2).

**LISTING 3.2  LockScreenViewModel Class**

```csharp
public class LockScreenViewModel : ViewModelBase
{
    public LockScreenViewModel() : base('lock screen settings')
    {
        LockScreenManager manager = LockScreenManager.Instance;
        ```
LISTING 3.2  Continued

```csharp
if (!manager.UserPrompted)
{
    bool allow = MessageService.AskYesNoQuestion(
        "Is it OK to run under the phone’s lock screen?"
    );
    manager.RunningUnderLockScreenEnabled = allow;
    manager.UserPrompted = true;
}

manager.PropertyChanged += (o, args) =>
{
    if (args.PropertyName == "RunningUnderLockScreen")
    {
        Debug.WriteLine("RunningUnderLockScreen: 
            + manager.RunningUnderLockScreen);
    }
};

public bool RunningUnderLockScreenEnabled
{
    get
    {
        return LockScreenManager.Instance.RunningUnderLockScreenEnabled;
    }
    set
    {
        LockScreenManager.Instance.RunningUnderLockScreenEnabled = value;
    }
}
```

The LockScreenView XAML has a Silverlight Toolkit ToggleSwitch control that is bound to the RunningUnderLockScreenEnabled viewmodel property, as shown in the following excerpt:

```xml
<StackPanel x:Name="ContentPanel">
    <toolkit:ToggleSwitch
        Header="run under lock screen"
        IsChecked="{Binding RunningUnderLockScreenEnabled, Mode=TwoWay}" />
</StackPanel>
```

Figure 3.3 shows the ToggleSwitch located on the LockScreenView page with the Run Under Lock Screen setting enabled.
Running under the lock screen is the only way to allow your foreground app to run while the phone is idle. It should, however, be used with caution, because if an app continues to consume the device CPU, it may rapidly flatten the device battery.

**Page Navigation**

Windows Phone navigation in Silverlight is based on the Silverlight for the browser navigation model. The navigation class model looks a little different in the phone SDK however. Rather than Silverlight 4’s `Frame` and `Page` controls, Silverlight for Windows Phone apps use the subclasses `PhoneApplicationFrame` and the `PhoneApplicationPage` (see Figure 3.4).

![Diagram of page navigation classes](image)

**NOTE**

`Frame` and `Page` must not be used directly in your app. They are prohibited because of underlying differences in the way the Silverlight infrastructure interacts with the OS within the constrained environment of the phone.

Page navigation in Silverlight for Windows Phone works in much the same way as page navigation in a web browser. `PhoneApplicationFrame` is analogous to the web browser, coordinating page transitions within your app.

Figure 3.5 depicts the display elements of a Silverlight for Windows Phone app.
FIGURE 3.5 Display elements of a Silverlight for Windows Phone app

The PhoneApplicationFrame is the host for PhoneApplicationPages and reserves space for the system tray and the application bar. The PhoneApplicationPage consumes all remaining space after the system tray and the application bar.

NOTE
There can be only one PhoneApplicationFrame for an application. Attempting to place a PhoneApplicationFrame within a PhoneApplicationPage causes the content to become infinitely nested at runtime, as the page will be forced inside the frame, the frame inside the page, and so on.

Navigation Using Unmapped URIs

There are numerous ways of allowing the user to perform page navigation. This section looks at using Buttons with code-beside to open external URLs, and at HyperlinkButtons, which can rely solely on XAML. In subsequent chapters you explore other techniques to perform navigation including the use of commands and a custom navigation service.

Internal URIs

When navigating to PhoneApplicationPages within an application, URIs either must be relative to the root directory of the project, and use the relative path syntax, as shown in the following excerpt:

```csharp
Uri uri = new Uri("/DirectoryName/PageName.xaml", UriKind.Relative);
```
or they must use the relative component URI format, such as that used in the following example:

```csharp
Uri uri = new Uri("/AssemblyName;component/PageName.xaml", UriKind.Relative);
```

The assembly name segment must be the name of an assembly that is locatable at runtime. The name of a project’s output assembly can be found in the project properties editor by right-clicking on the project node in the Solution Explorer and selecting Properties, or by pressing Alt+Enter.

The **HyperlinkButton** control can be used to allow the user to navigate directly to a page within your application, as shown:

```xml
<HyperlinkButton NavigateUri="/Directory/PageName.xaml" Content="Internal Page" />
```

**External Navigation Using the Button Control**

The **Button** control is a flexible way for determining user intent. A **Button** can be used for navigation by subscribing to its **Click** event, as shown in the following excerpt from the **ProductDetailsView.xaml** in the downloadable sample code:

```xml
<Button Click="Button_EXTERNallink_Click"
    Tag="{Binding Product.ExternalUrl}"
    Content="External Page" />
```

The **WebBrowserTask** allows you to navigate to external URIs using the phone’s built-in web browser: Internet Explorer. This causes your app to be deactivated while the user views the page. You explore tasks in more detail in Chapter 12.

To provide the **WebBrowserTask** with the location of the web page, use the button’s **Tag** property. The **Click** event handler, which initiates the **WebBrowserTask**, is shown in the following excerpt:

```csharp
void Button_EXTERNallink_Click(object sender, RoutedEventArgs e)
{
    FrameworkElement button = sender as FrameworkElement;
    if (button == null || button.Tag == null)
    {
        return;
    }

    WebBrowserTask task = new WebBrowserTask
    {
        URL = button.Tag.ToString()
    };
    task.Show();
}
```
External Navigation Using the HyperlinkButton Control

The disadvantage of using a Button control for links to external content is that it does not provide the familiar look and feel of a hyperlink. The HyperlinkButton control provides an easier way for navigating to pages within an application. There is a trick to using the HyperlinkButton with external URIs. Set its TargetName property to _blank, as shown in the following example:

```xml
<HyperlinkButton TargetName="_blank" NavigateUri="http://create.msdn.com"
                  Content="http://create.msdn.com" />
```

**NOTE**
Failing to set the HyperlinkButton.TargetName to _blank, when using an external URI, raises the Frame.NavigationFailed event when the button is tapped.

Hosting Web Content Within an App

An alternative to using the phone’s built-in Internet Explorer app is to host the content in a Microsoft.Phone.Controls.WebBrowser control.

The following excerpt from the WebBrowserView.xaml page, in the downloadable sample code, shows a WebBrowser placed within the main content grid of a page:

```xml
<Grid x:Name="ContentGrid" Grid.Row="1">
    <phone:WebBrowser Source="{Binding Url}"

</Grid>
```

Here, the Source property of the WebBrowser is bound to the Url property of the view-model. The WebBrowser control is discussed in greater detail in Chapter 7, “Media and Web Elements.”

A dedicated web browser page can be used in your app to host all external content. To launch the dedicated web browser page, a relative URI can be constructed using the Binding.StringFormat property, as shown in the following excerpt:

```xml
<HyperlinkButton
    NavigateUri="{Binding ExternalUrl, StringFormat=/WebBrowser/{0}}"
    Content="External Page" />
```

Backslashes are used to escape the curly brackets in the StringFormat value.

The StringFormat property transforms the HyperlinkButton’s binding expression into the following:

```csharp
string.Format("/WebBrowser/{0}", ExternalUrl);
```

URI mapping is used to pass the external URL as a query string parameter. This is explored further in a later section.
Passing Page Arguments Using Query Strings

Query strings allow for key value pairs to be embedded in a URL and passed to a page. Just like HTML web applications, Silverlight uses query string parameters for interpage communication.

The `PhoneApplicationPage.NavigationContext` property, which is initialized after the page is created, is used to retrieve the query string. Its `QueryString` property is an `IDictionary` of string key and value pairs. The following excerpt from the `WebBrowserView.xaml`, in the downloadable sample code, demonstrates how to retrieve a query string value:

```csharp
void OnLoaded(object sender, RoutedEventArgs e)
{
    string url;
    if (NavigationContext.QueryString.TryGetValue("url", out url))
    {
        ViewModel.LoadPage(url);
    }
}
```

Navigation History Stack

The Silverlight navigation infrastructure maintains a history of pages that have been loaded. Each time an app navigates to a different page, the current page's `OnNavigatedFrom` method is called and the page is placed on the history stack (see Figure 3.6).

![Figure 3.6](image)

While on the history stack, the page remains in memory unless the application is tombstoned or closed. This means that subscribing to the `PhoneApplicationService.Deactivated` event provides the page with the opportunity to save its transient state. It is, however, preferable to use the page's `OnNavigatedFrom` method for the saving of page state.

Using the `Deactivate` event to save page state runs the risk of slowing down deactivation when all pages on the history stack are saving their state simultaneously.
BEST PRACTICE

Use the `OnNavigatedFrom` and `OnNavigatedTo` methods of `PhoneApplicationPage` to save both transient and persistent state.

NOTE

Unlike Silverlight for the browser, in Windows Phone the page’s `NavigationCacheMode` property is not assignable, and it is set to `Disabled` by default. This means that internal caching of pages does not occur, and when navigating to a page that does not exist on the history stack, the page is always instantiated.

Restoration of transient state should occur with the page’s `OnNavigatedTo` method. The `OnNavigatedTo` method is called when the `PhoneApplicationFrame` navigates to the page. This is triggered by the following actions:

- Navigation to a specified page URI occurs using one of various navigation methods such as the `PhoneApplicationFrame.Navigate` method.
- The `NavigationService.GoBack` method is called.
- The user presses the hardware Back button.
- The page is the current page when the app moves from the tombstoned or dormant state to the running state.
- The page is the app’s start page and the app is launched.

NOTE

When an application is activated, the frame navigates to the page that was active when the phone was dormant or tombstoned. The `NavigationContext.QueryString` is preserved, which means that there is no need to store this in the `PhoneApplicationService.State` dictionary.

URI Mapping

Relying on URIs that include the full path to each page in your app can make your app brittle and makes it harder to change the physical location of individual pages. If a page is moved, all references to that file must be updated. This can lead to maintainability issues as the size of the project grows.

The URI mapping system of Silverlight allows requests for a URI to be routed to another URI, and uses a single configuration point for the management of page URIs. Mapped URIs can be made shorter and are, thus, less subject to typographical errors. They also allow the exclusion of technology specific information, such as the .xaml page file extension, making it easier to retarget business logic for different platforms.
To use URI mapping, you must assign a `System.Windows.Navigation.UriMapper` instance to the `UriMapper` property of an app's `PhoneApplicationFrame`. This can be done in XAML, as shown in the following excerpt from the App.xaml file in the downloadable sample code:

```xml
<Application
    x:Class="DanielVaughan.WindowsPhone7Unleashed.Examples.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <Application.RootVisual>
        <phone:PhoneApplicationFrame x:Name="RootFrame">
            <phone:PhoneApplicationFrame.UriMapper>
                <navigation:UriMapper>
                    <navigation:UriMapper.UriMappings>
                        <navigation:UriMapping
                            Uri="/ProductDetails/{productId}"
                            MappedUri="/Navigation/ProductDetailsView.xaml?productId={productId}" />
                        <navigation:UriMapping
                            Uri="/WebBrowser/{url}"
                            MappedUri="/WebBrowser/WebBrowserView.xaml?url={url}" />
                    </navigation:UriMapper.UriMappings>
                </navigation:UriMapper>
            </phone:PhoneApplicationFrame.UriMapper>
        </phone:PhoneApplicationFrame>
    </Application.RootVisual>

    <!-- Content omitted. -->

</Application>
```

The `UriMapping` class contains a `Uri` property and a `MappedUri` property. When navigation is requested from the `Uri` value, it is rerouted to the `MappedUri` property.

By using the curly brace syntax, as shown in the previous excerpt, a substring of the requested URI can be transplanted into the rerouted `MappedUri` value. This is especially useful when you want to target the same page using different URIs and allows the query string to be used to convey the action to be undertaken by the page.

In the previous excerpt you see a `UriMapping` for the `ProductDetailsView` page. The `ProductDetailsView` displays detailed information for a particular product, identified by a query string parameter. When navigating to the ProductDetails page, if the requested URL is `/ProductDetails/2`, this is rerouted to `/Navigation/ProductDetailsView.xaml?productId=2`.

If you were to request the `ProductDetailsView` page using the `NavigationService`, as shown in the following example, the request would be rerouted accordingly:

```csharp
NavigationService.Source = new Uri("/ProductDetails/2", UriKind.Relative);
```

The product to be displayed can then be determined in the `ProductDetailsView` by reading the `productId` query string parameter, as demonstrated in the following excerpt:
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    string productIdString = NavigationContext.QueryString["productId"];
    int productId = int.Parse(productIdString);
    ViewModel.LoadProduct(productId);
}

You see later in this chapter how the viewmodel uses the product ID to retrieve the product information from a WCF service.

**Navigation Using the NavigationService**

The PhoneApplicationPage class exposes a public **NavigationService** property, which allows direct control over navigation.

**NOTE**

The **NavigationService** cannot be used to launch Internet Explorer to view an external URL. Instead, use either the **WebBrowserTask**, or open the page within the app using the **WebBrowserControl**. See the previous sections on external navigation using the **Button** and **HyperlinkButton** controls.

The **NavigationService.Navigate** method causes the frame to load the specified PhoneApplicationPage, like so:

```
NavigationService.Navigate(
    new Uri("/DirectoryName/PageName.xaml", UriKind.Relative));
```

The URI must be either a path relative to the project’s root directory, as shown in the previous example, or a relative component URI such as in the following example:

```
NavigationService.Navigate(
    new Uri("/AssemblyName;component/PageName.xaml", UriKind.Relative));
```

**TIP**

The **NavigationService** cannot be used within the constructor of the PhoneApplicationPage because it is assigned after the page’s constructor has been called. Therefore, wait until the page’s **OnNavigatedTo** method is called or the **Loaded** event has occurred before using the **NavigationService**.

The **NavigationService.Source** property allows you to retrieve the URI of the current page. Setting the **Source** property performs the same action as using the **Navigate** method; the frame loads the page at the specified URI. See the following example:
Routing is also enabled for the NavigationService, which means that mapped URIs can be used instead of relative URIs.

If you examine the API of the NavigationService, you will likely wonder what the difference is between the CurrentSource property and the Source property. The answer is that the CurrentSource property does not change until navigation has completed. Conversely, the Source property changes as soon as navigation is initiated.

NOTE
If the GoBack method is called, and the history stack is empty because the current page is the app’s start page, then an InvalidOperationException is raised. To determine whether the NavigationService is able to go back, query its CanGoBack property.

Forward Navigation
Unlike Silverlight for the browser, the GoForward method of the NavigationService does not allow forward navigation and raises an InvalidOperationException when called. Consequently, the CanGoForward property always returns false.

NOTE
Forward navigation using the NavigationService is not supported.

Handling Page Navigation
The PhoneApplicationPage extends the System.Windows.Controls.Page class, which has a number of virtual methods called when the page is brought into view or removed from view by the PhoneApplicationFrame (see Figure 3.7).

The OnNavigatingFrom method is called before a page is removed from view by the PhoneApplicationFrame, and the OnNavigatedFrom method is called after navigation occurs. Conversely, the OnNavigatedTo method is called when the frame brings the page into view.
Cancelling Navigation

The OnNavigatingFrom method offers the opportunity to cancel navigation using the NavigatingCancelEventArgs parameter.

NavigatingCancelEventArgs has the following properties:

- **NavigationMode**—An enum value that indicates the type of navigation. This value may be Back, Forward, New, or Refresh.
- **Uri**—The destination URI.
- **Cancel**—Setting this value to true cancels the navigation.

The NavigationCancelEventArgs class subclasses CancelEventArgs, which provides the Cancel property. By setting this property to true, the page can prevent the navigation from occurring, as shown in the following excerpt:

```csharp
protected override void OnNavigatingFrom(System.Windows.NavigatingCancelEventArgs e)
{
    base.OnNavigatingFrom(e);
    MessageBoxResult boxResult = MessageBox.Show(
        "Leave this page?", "Question", MessageBoxButton.OKCancel);
    if (boxResult != MessageBoxButton.OK)
    {
        e.Cancel = true;
    }
}
```
NOTE

You should not attempt to cancel navigation in the OnNavigatingFrom method when the hardware Back button is pressed. Instead, override the OnBackKeyPress method to cancel the back key, which prevents navigation.

Cross-Page Communication

Once navigation has occurred, there remains an opportunity for the previous page to interact with the current page from the previous page's OnNavigatedFrom method. This is achieved using the Content property of the NavigationEventArgs, which provides the destination PhoneApplicationPage object.

To see this in action, place a breakpoint in the OnNavigatedFrom method. When the breakpoint is hit, notice that the page being navigated to has already been instantiated and is provided in the Content property of the NavigationEventArgs (see Figure 3.8).

![Figure 3.8](image)

FIGURE 3.8 The Content property of the NavigationEventArgs contains the page being navigated to.

The Uri property of the NavigationEventArgs contains the URI of the destination page, including any query string that may be present.

NOTE

If navigating to an external URI, or when the app is being deactivated, the Uri property of the OnNavigatedFrom method's NavigationEventArgs is equal to app://external/.

Page Redirection

The OnNavigatingFrom method allows you to intercept a navigation event and to even cancel the navigation if needed. Additionally, there may be times when you want to redirect the user to a different URI based on some conditional logic.
The **NavigationService**, however, does not support overlapping navigation. That is, you are unable to cancel an existing navigation and immediately commence another.

You can, however, cancel navigation and schedule navigation to a different **Uri** using the page’s **Dispatcher** property, as shown in the following excerpt:

```csharp
protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
    if (e.Uri.ToString().Contains("RequestedUrl"))
    {
        e.Cancel = true;
        /* Perform the redirect on the UI thread. */
        Dispatcher.BeginInvoke(() => NavigationService.Navigate(
            new Uri("RedirectUrl", UriKind.Relative)));
    }
    base.OnNavigatingFrom(e);
}
```

By using the **Dispatcher** to invoke the lambda expression, which performs the call to the **NavigationService**, you allow the current navigation to complete first. This works because the UI thread can be thought of as a queue of prioritized delegates, all waiting in turn to be executed. Once all the **Navigating** event handlers have been serviced, the delegate represented by the lambda expression will be taken out of the queue and performed by the UI thread. This technique, of using the **Dispatcher** to enqueue an action, is also useful when working with some UI controls, whose event handlers may be called before the control is finished reacting to a user action.

### Hardware Back Button

The hardware Back button is analogous to the Back button on a web browser. However, when the user presses the Back button, past the first page of a phone app, the app is closed. This is in contrast to the phone’s hardware start button, which merely causes an app to be deactivated.

**NOTE**

The hardware Back button should not be used for application-specific behavior. It is only for navigation, and if used otherwise, may cause your app to fail Windows Phone Marketplace certification.

To determine whether navigation is occurring because the hardware Back button was pressed or the navigation was initiated by a call to **NavigationService.GoBack**, use the **NavigatingEventArgs.NavigationMode** property as shown:

```csharp
protected override void OnNavigatingFrom(NavigatingEventArgs e)
{
    base.OnNavigatingFrom(e);
}
```
if (e.NavigationMode == NavigationMode.Back)
{
    // Back button pressed.
}

The Back key button can also be cancelled by overriding the
PhoneApplicationPage.OnBackKeyPress, as shown in the following excerpt:

protected override void OnBackKeyPress(CancelEventArgs e)
{
    base.OnBackKeyPress(e);

    e.Cancel = true;
}

OnBackKeyPress is called before OnNavigatedFrom, and if the Back button is cancelled, then
OnNavigatedFrom is not called at all.

NOTE

The Windows Phone Marketplace certification requirements forbid cancelling the Back
button in most cases. To maintain a consistent user experience, the Back button must
only be used for backward navigation in the application. The following four certification
requirements relate to use of the Back button:

- Pressing the Back button must return the application to the previous page or return
to any previous page within the back stack.
- Pressing the Back button from the first screen of an application must close the
application.
- If the current page displays a context menu or a dialog, the pressing of the Back
button must close the menu or dialog and return the user to the screen where the
context menu or dialog box was opened.
- For games, when the Back button is pressed during gameplay, the game can choose
to present a pause context menu or dialog or navigate the user to the prior menu
screen. Pressing the Back button again while in a paused context menu or dialog
closes the menu or dialog.

For more information see section 5.2.4 of the Technical Certification Requirements at

Creating an Application Splash Screen

Windows Phone Silverlight projects have baked-in support for application splash screens.
To create a splash screen it is simply a matter of placing a jpg image called
SplashScreenImage.jpg, with the dimensions of 480 by 800 pixels, in the root directory of
your project. Ensure that its Build Action is set to Content (see Figure 3.9).
Using an image for a splash screen does not, however, prevent an application from being closed by the OS if the first page takes longer than 10 seconds to load. If your application’s first page takes longer than this to load, it is best to overlay the content with a loading indicator and perform the time consuming initialization on a background thread. Once loading is complete, the indicator can be dismissed.

The ProductsView and ProductsViewModel classes, located in the Navigation directory of the WindowsPhone7Unleashed.Examples project in the downloadable sample code, demonstrate this principle (see Figure 3.10).

The ProductsView page uses a StackPanel to present an indeterminate progress bar to the user while the viewmodel is loading, as shown in the following excerpt:

```xml
<StackPanel Grid.Row="1"
            Visibility="{Binding Loaded, Converter=(StaticResource BooleanToVisibilityConverter), ConverterParameter=Collapsed}"
            Height="150">
    <TextBlock Text="Loading...", Style="{StaticResource PhoneTextTitle2Style}"
               HorizontalAlignment="Center" Margin="20"/>
    <toolkit:PerformanceProgressBar IsIndeterminate="true" />
</StackPanel>
```
The Visibility property of the StackPanel is assigned via a binding to the viewmodel's Loaded property. To convert the boolean loaded property to a Visibility type, a custom IValueConverter called BooleanToVisibilityConverter is used (see Listing 3.3). The class is located in the ValueConverters directory of the WindowsPhone7Unleashed project, in the downloadable sample code.

LISTING 3.3 BooleanToVisibility Class

```csharp
public class BooleanToVisibilityConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string paramValue = (string)parameter;

        if (value == null || (bool)value)
        {
            return paramValue == "Collapsed"
        }

        return paramValue == 'Collapsed'
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string paramValue = (string)parameter;
        if (value == null || (Visibility)value == Visibility.Visible)
        {
            return paramValue == "Collapsed"
        }
    }
}
```
LISTING 3.3  Continued

```csharp
{ 
    return paramValue != "Collapsed";
}

return paramValue == "Collapsed";
```

The `ConverterParameter` attribute determines what value to assign to the `Visibility` property if the binding value is `true`. If the `Loaded` property of the viewmodel is `true`, then the `Visibility` property will set to `Visibility.Visible`.

To hide the rest of the content during loading, the same technique is employed for the main content control.

```xml
<StackPanel Grid.Row="1" Margin="10"
    Visibility="{Binding Loaded,
        Converter={StaticResource BooleanToVisibilityConverter},
        ConverterParameter=Visible}"
    >
    <ScrollViewer>
        <!-- Content omitted. -->
        <ScrollViewer>
    </ScrollPanel>
</StackPanel>
```

Here the `ConverterParameter` attribute is set to `Visible`, so that its `Visibility` is set to `Visible` when the viewmodel’s `Loaded` property is `true` and `Collapsed` when it is `false`.

The code listings for the `ProductsView` page and associated files are provided in the following section.

**Walking Through the Bookshop Sample Application**

This chapter’s sample app provides the beginnings of a simple data driven e-commerce app that demonstrates the use of navigation, transient and persistent state, image caching, and WCF services. It allows the user to select from a list of books, retrieved from a WCF service, and to view each item’s details on a separate details page.

The `ProductsViewModel` class retrieves a list of `Product` objects from a WCF service. Each product has various properties such as a description, price, and an image URI.

The `ProductsViewModel` saves and restores its own transient state consisting of the list of products it retrieves from the WCF service (see Listing 3.4).

The code for this section resides in the Navigation directory of the `WindowsPhone7Unleashed.Examples` project in the downloadable sample code.
The viewmodel’s constructor determines whether transient state exists for itself. If so, it restores the list of Products or else it requests the list of products from the WCF using the BookshopServiceClient. The call occurs asynchronously, and the products list is populated once the call completes.

The ViewModelBase class subclasses the NotifyPropertyChangedBase class, which implements INotifyPropertyChanged. The source for NotifyPropertyChangedBase is located in the downloadable sample code, and was discussed in Chapter 2, “Fundamental Concepts in Silverlight Development for Windows Phone.”

LISTING 3.4 ProductsViewModel Class (excerpt)

```csharp
public class ProductsViewModel : ViewModelBase
{
    readonly IDictionary<string, object> transientStateDictionary;
    const string transientStateKey = "ProductsViewModel_Products";

    public ProductsViewModel(
        IDictionary<string, object> transientStateDictionary)
    {
        this.transientStateDictionary = ArgumentValidator.AssertNotNull(
            transientStateDictionary, "transientStateDictionary");

        LoadTransientState();
        if (products != null)
        {
            return;
        }

        BookshopServiceClient client = new BookshopServiceClient();
        client.GetProductsCompleted += (sender, args) =>
        {
            if (args.Error != null)
            {
                MessageService.ShowError("Unable to retrieve products.");
                return;
            }

            Products = args.Result;
            Loaded = true;
        };
        client.GetProductsAsync();
    }

    ObservableCollection<Product> products;
```
public ObservableCollection<Product> Products
{
    get
    {
        return products;
    }
    private set
    {
        Assign(() => Products, ref products, value);
    }
}

bool loaded;

public bool Loaded
{
    get
    {
        return loaded;
    }
    private set
    {
        Assign(() => Loaded, ref loaded, value);
    }
}

public void SaveTransientState()
{
    transientStateDictionary[transientStateKey] = products;
}

public void LoadTransientState()
{
    object transientState;
    if (transientStateDictionary.TryGetValue(
        transientStateKey, out transientState))
    {
        products = transientState as ObservableCollection<Product>;
        if (products != null)
        {
            Loaded = true;
        }
    }
}
Within the OnNavigatingTo method of the ProductsView page, a ProductsViewModel is instantiated and assigned to the page’s DataContext. The ProductsViewModel is passed the transient state dictionary for the page (see Listing 3.5).

The OnNavigatingTo and OnNavigatedFrom methods are used to inform the viewmodel when to save its state.

LISTING 3.5 ProductsView Class

```csharp
public partial class ProductsView : PhoneApplicationPage
{
    public ProductsView()
    {
        InitializeComponent();
    }

    ProductsViewModel ViewModel
    {
        get
        {
            return (ProductsViewModel)DataContext;
        }
    }

    bool loaded;

    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        Debug.WriteLine("ProductsView OnNavigatedTo");
        base.OnNavigatedTo(e);

        if (!loaded)
        {
            DataContext = new ProductsViewModel(State);
            loaded = true;
        }

        ViewModel.LoadTransientState();
    }

    protected override void OnNavigatedFrom(NavigationEventArgs e)
    {
        base.OnNavigatedFrom(e);
        Debug.WriteLine("ProductsView OnNavigatedFrom");
        ViewModel.SaveTransientState();
    }
}
```
Displaying the Product List

The list of products exposed by the `ProductsViewModel.Products` property is displayed using a `ListBox` control in the `ProductsView` page. The `ListBox`'s `ItemTemplate` has various controls that are used to display the details of each `Product`, as shown in the following excerpt:

```xml
<StackPanel Grid.Row="1" Margin="10"
    Visibility="{Binding Loaded,
        Converter={StaticResource BooleanToVisibilityConverter},
        ConverterParameter=Visible}'
    <ScrollViewer>
        <ListBox ItemsSource="{Binding Products}" Height="610">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal">
                        <Image Source="{Binding SmallImageUri}"
                            MaxWidth="150" MaxHeight="150"
                            Margin="0,0,10,10" />
                        <StackPanel Margin="5">
                            <TextBlock Text="{Binding Title}"
                                TextWrapping="Wrap" />
                            <TextBlock Text="{Binding Price,
                                StringFormat="\{0:C\}"}" />
                            <HyperlinkButton
                                NavigateUri="{Binding Id,
                                    StringFormat=/ProductDetails/{\{0\}}}"
                                Content="View Details"
                                HorizontalAlignment="Left" Margin="0,10,0,0" />
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </ScrollViewer>
</StackPanel>
```

An `Image` control displays a thumbnail of the product, using the `SmallImageUri` property of the `Product`.

A string format is used to convert the `Price` property, which is a double value, to a currency formatted string using the format `{0:C}. Similarly, a link is provided for the product details page, using the format `/ProductDetails/{0}`, and the value of the `Product`'s `Id` is substituted for the `{0}` placeholder. The `UriMapping` for this product details URI causes the application to reroute to the full URI of the `ProductDetailsView.xaml` page and includes the `productId` query string parameter.

Figure 3.11 shows the `ProductsView` displaying a list of books.
When the user presses the HyperlinkButton, he is directed to the ProductDetailsView.xaml page. This page displays the various properties of the product and includes a link for an external website, where the user can find more information about the product (see Figure 3.12).

When navigating to the ProductDetailsView the page attempts to retrieve the productId query string parameter from the NavigationContext (see Listing 3.6).

LISTING 3.6  ProductDetailsView Class (excerpt)

```csharp
public partial class ProductDetailsView : PhoneApplicationPage
{
    public ProductDetailsView()
    {
        InitializeComponent();

        DataContext = new ProductDetailsViewModel(
            PhoneApplicationService.Current.State);
    }

    ProductDetailsViewModel ViewModel
    {
        get
        {
            return (ProductDetailsViewModel)DataContext;
        }
    }
```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    string productIdString = NavigationContext.QueryString["productId"];
    int productId = int.Parse(productIdString);
    ViewModel.LoadProduct(productId);
}

protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    ViewModel.SaveTransientState();
    base.OnNavigatedFrom(e);
}

FIGURE 3.12 View a product’s details.

The view then passes the parameter along to the ProductDetailsViewModel class, which handles the loading of the specified product (see Listing 3.7). The LoadProduct method first tests for the existence of the product in transient state. If not present, it retrieves the product using the service client.
public class ProductDetailsViewModel : ViewModelBase
{
    const string transientStateKey = "ProductDetailsViewModel_Product";
    readonly IDictionary<string, object> transientStateDictionary;
    public ProductDetailsViewModel(IDictionary<string, object> transientStateDictionary)
    {
        this.transientStateDictionary = ArgumentValidator.AssertNotNull(
            transientStateDictionary, "transientStateDictionary");
    }

    public void LoadProduct(int productId)
    {
        object transientState;
        if (PhoneApplicationService.Current.State.TryGetValue(
            transientStateKey, out transientState))
        {
            Product = transientState as Product;
            if (Product != null && Product.Id == productId)
            {
                return;
            }
        }

        BookshopServiceClient client = new BookshopServiceClient();
        client.GetProductByIdCompleted += (sender, args) =>
        {
            if (args.Error != null)
            {
                throw args.Error;
            }
            Product = args.Result;
        }
        client.GetProductByIdAsync(productId);
    }

    public Product Product
    {
        get
        {
            return product;
        }
    }
}
When navigating away from the page, the viewmodel’s `SaveTransientState` method is called, which places the product in the state dictionary.

The `ProductDetailsView.xaml` page presents the product details via the viewmodel’s `Product` property (see Listing 3.8).

LISTING 3.8  ProductDetailsView.xaml (excerpt)

```xml
<StackPanel Grid.Row="1"
            Style="{StaticResource PageContentPanelStyle}"
            DataContext="{d:DesignData Source=ProductDetailsViewSampleData.xaml}"
            d:DataContext="{d:DesignData Source=ProductDetailsViewSampleData.xaml}"
            />

<TextBlock Text="{Binding Product.Title}" TextWrapping="Wrap"
            Style="{StaticResource PhoneTextTitle2Style}" />
<StackPanel Orientation="Horizontal">
    <Image Source="{Binding Product.LargeImageUri, Converter="{StaticResource ImageCacheConverter}""
               MaxWidth="250" MaxHeight="250" Margin="10,10,0,10" />
    <StackPanel>
        <TextBlock Text="{Binding Product.Author}" TextWrapping="Wrap"
                    Style="{StaticResource PhoneTextTitle3Style}" />
        <TextBlock Text="{Binding Product.Price, StringFormat="\0:C\"}"
                    Style="{StaticResource PhoneTextTitle3Style}" />
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="ISBN"
                        Style="{StaticResource PhoneTextTitle3Style}" />
            <TextBlock Text="{Binding Product.Isbn13}"
                        TextWrapping="Wrap"
                        Style="{StaticResource PhoneTextNormalStyle}" />
        </StackPanel>
    </StackPanel>
</StackPanel>
```
The StackPanel includes a `d:DataContext` attribute that defines a design-time data context object, discussed in the next section.

**Design-Time Data**

It can be difficult and time consuming constructing a page or control without knowing how the content will appear at runtime. The dimensions of images can disturb the layout, as can the length of text and text wrapping settings. The `d:DataContext` markup extension, which exists in the http://schemas.microsoft.com/expression/blend/2008 namespace, allows you to simulate the runtime `DataContext` of a control with a design-time object (see Figure 3.13).
Here a design-time instance of the `ProductDetailsViewModel` class presents some sample data to improve the design-time experience of the developer or designer.

The content `StackPanel` includes a `d:DataContext` attribute, which causes a `ProductDetailsViewModel` instance to be loaded from a sample data file, as shown in the following excerpt:

```xml
<StackPanel Grid.Row="1"
    Style="{StaticResource PageContentPanelStyle}"
    d:DataContext="{d:DesignData Source=ProductDetailsViewSampleData.xaml}"
>
    ...  
</StackPanel>
```

You can see that the `d:DesignData` markup extension has its `Source` property set to the location of a sample data file, `ProductDetailsViewSampleData.xaml`. The sample data file defines the property values of the viewmodel (see Listing 3.9). The design-time environment of Visual Studio or Expression Blend instantiates the sample viewmodel at design-time.

**LISTING 3.9  ProductDetailsViewSampleData.xaml**

```xml
<local:ProductDetailsViewModel
    xmlns:local="clr-namespace:DanielVaughan.WindowsPhone7Unleashed.
    Examples.Navigation"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:BookshopServiceReference="clr-namespace:DanielVaughan.
    WindowsPhone7Unleashed.Examples.BookshopServiceReference">

<BookshopServiceReference:Product
    Id="1"
    Title="Windows Phone 7 Unleashed"
    Author="Daniel Vaughan"
    Description="The complete guide to programming..."
    Price="31.49"
    ISBN10="0672333481"
    ISBN13="978-0672333484"
    SmallImageUri="/DanielVaughan.WindowsPhone7Unleashed.
    Examples.Silverlight;component/Navigation/Images/Product01Small.jpg"
    LargeImageUri="/DanielVaughan.WindowsPhone7Unleashed.
    Examples.Silverlight;component/Navigation/Images/Product01Large.jpg"
    Daniel-Vaughan/dp/0672333481/"
/>

</local:ProductDetailsViewModel.Product>
</local:ProductDetailsViewModel>
```
Notice the relative component URIs of the images. The design-time environment will fail to resolve the image location unless relative component URIs are used and the Build Action of the image is set to Resource.

**Image Caching**

While the viewmodel saves the result of the WCF service call, which allows the app to restore its state after being tombstoned, downloaded images are not saved in the state dictionary, but rather, the app relies on some custom image caching.

A custom `IValueConverter`, called `ImageCacheConverter`, is used to download the image from the specified image URI, as shown in the following excerpt:

```xml
<Image Source="{Binding Product.LargeImageUri, Converter={StaticResource ImageCacheConverter}}" />
```

By using the `ImageCacheConverter`, images can be downloaded once and stored in isolated storage for an arbitrary period. Once that period has elapsed, the image will be downloaded again. This allows the application to work offline (see Listing 3.10).

**LISTING 3.10  ImageCacheConverter Class**

```csharp
public class ImageCacheConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        if (EnvironmentValues.DesignTime)
        {
            return value;
        }

        string url = value as string;
        if (url != null)
        {
            try
            {
                return ImageCache.GetImage(new BitmapImage(new Uri(url)));
            }
            catch (IsolatedStorageException e)
            {
                Console.WriteLine(e);
                return value;
            }
        }

        BitmapImage bitmapImage = value as BitmapImage;
        if (bitmapImage != null)
```

```csharp
```
LISTING 3.10  Continued

    
    return ImageCache.GetImage(bitmapImage);
    
    return value;

public object ConvertBack(object value, Type targetType,
    object parameter, CultureInfo culture)
{
    throw new NotImplementedException();
}

The ImageCacheConverter can be used in conjunction with a URL or a BitmapImage. In
the sample code, it is used with a URL, supplied by the product’s SmallImageUri and
LargeImageUri properties. The ImageCache class maintains a dictionary of URI keyed cached
images. It stores the dictionary in isolated storage, and when an image is requested, it
attempts to locate it in the dictionary. If found it checks to ensure that the image has not
expired, and then returns the image.

Many thanks to Peter Nowak (http://winphonedev.de/) for his image cache code, which I
have adapted, with his permission, for use in the downloadable sample code.

The ImageCache class, in the downloadable sample code, maintains a list of ImageCacheItem
objects, which represent cached images. The ImageCache.GetImage method is used to
retrieve an image from the cache. If the image is not located in the cache, it is scheduled
to be downloaded by the static ImageDownloader class.

The ImageDownloader coordinates an asynchronous download of the image file. It uses an
HttpWebRequest to retrieve the image from a remote server, and then stores the down-
loaded file in isolated storage. Once downloaded, the Source property of the original
image is assigned, which means that, if it is present in the UI, the image will appear
(see the ImageDownloader class, located in the Data/ImageCache directory of the
WindowsPhone7Unleashed project, in the downloadable sample code, for details).

Overview of the Sample Bookshop WCF Service

The Bookshop demo application includes a server-side component, which is used by both
the ProductsViewModel and ProductDetailsViewModel classes, providing the application
with a set of products to display. The server-side component is fairly arbitrary and is
presented here merely for the sake of completeness.

The WCF service is called BookshopService and resides in the WindowsPhone7Unleashed.
Web project of the downloadable sample code (see Listing 3.11).
LISTING 3.11 BookshopService Class

```csharp
[AspNetCompatibilityRequirements(
    RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
public class BookshopService : IBookshopService
{
    public IEnumerable<Product> GetProducts()
    {
        return ProductManager.Products;
    }

    public Product GetProductById(int productId)
    {
        return ProductManager.GetProductById(productId);
    }
}
```

The `BookshopService` exposes static methods of the `ProductManager` class, shown in Listing 3.12. The `ProductManager` class creates an `XDocument` instance, using an XML file, to populate a list of Products.

LISTING 3.12 ProductManager Class

```csharp
public static class ProductManager
{
    static readonly List<Product> products = new List<Product>();

    public static IEnumerable<Product> Products
    {
        get
        {
            return products;
        }
    }

    static ProductManager()
    {
            '~/Services/Bookshop/Products.xml');
        XDocument document = XDocument.Load(path);
        foreach (XElement element in document.Element("Products").Elements("Product"))
        {
            var product = new Product(element);
            product.SmallImageUri = ServerUtility.ResolveServerUrl(product.SmallImageUri);
        }
    }
}
```
LISTING 3.12  Continued

```csharp
public static Product GetProductById(int id)
{
    if (id < 0 || id > products.Count)
    {
        throw new ArgumentOutOfRangeException("id");
    }
    return products[id - 1];
}
```

The `Product` class contains the properties that are used to display each book’s details, such as Title and Author. The `Product` class also knows how to populate itself from an `XElement`. The explicit casting operators of the `XElement` class make it easy to extract the values to the `Product` properties, as can be seen in the following excerpt from the `Product` class:

```csharp
public Product(XElement element)
{
    if (element == null)
    {
        throw new ArgumentNullException("element");
    }
    Id = (int)element.Element("Id");
    Title = (string)element.Element("Title");
    Author = (string)element.Element("Author");
    Description = (string)element.Element("Description");
    SmallImageUri = (string)element.Element("SmallImageUri");
    LargeImageUri = (string)element.Element("LargeImageUri");
    Price = (double)element.Element("Price");
    Isbn10 = (string)element.Element("ISBN-10");
    ExternalUrl = (string)element.Element("ExternalUrl");
}
```
Summary

Maintaining the appearance of continuity after an application has been tombstoned is one of the key challenges facing Windows Phone developers.

There are two types of application state: persistent and transient. Persistent state exists across application launches and is saved using isolated storage. Transient state is discarded when an application is closed and is stored in the `Microsoft.Phone.Shell.PhoneApplicationService.State` dictionary.

Transient page state should be stored when the `Page.OnNavigatedFrom` method is called and restored when the `OnNavigatedTo` method is called. Transient application state can be stored when the `PhoneApplicationService.Deactivated` event occurs and restored when the `PhoneApplicationService.Activated` event occurs.

Persistent state should be saved when transient state is saved and also when the application is closing using the `PhoneApplicationService.Closing` event.

In this chapter you saw an overview of the application execution model and examined the various application life cycle events, which are used to coordinate state persistence and restoration.

You saw how to enable an app to run under the lock screen and then looked at page navigation and how to optimize the user experience by using a splash screen or a loading indicator.

Finally, the chapter delved into the sample application and discussed image caching, design-time data, and consuming a simple WCF service.
Symbols

- (difference) symbol, 718
> (greater than) symbol, 159
* (intersection) symbol, 718
< (less than) symbol, 159
! (not) operator, 717
( ) (parentheses), 718
+ (union) symbol, 718
3D XNA model, displaying, 683-684
10% of storage space remaining notification, 820

A

A-GPS (Assisted Global Positioning System), 531
Abort method, 925
absolute URIs, 187
accelerometer
  axis values, 498
calibrating, 503
compass orientation, 514
g's, 497
overview, 497
readings
  monitoring, 498-499
  smoothing, 500-503
  stationary, 497
sample view, 503-506
shake detection, 507-508
simulating, 499-500
updates, receiving, 499
Accelerometer class, 498-499
AccelerometerView class, 504-506
AccelerometerViewModel class, 503-504
Accounts property
  Appointments class, 447
  Contacts class, 442
  accuracy (geographic locations), 530, 534
Action property, 356
Add method, 909
Add New Project dialog, 4
Add Service Reference dialog, 800
AddAssociation method, 891-893
AddColumn method, 888
AddIndex method, 890-891
adding
  Bing Maps controls, 556-557
  capabilities, 27
  context menus, 267
  Extras applications to Extras menu, 619
grayscale video effect
  ARGB color components, extracting, 661
  converting back to color, 661
  grayscale conversion, 661
  previewing, 659-660
  processing, 659
  result, 661
  turning on/off, 658
headers, 277
photo share apps, 632-633
Pivot controls, 325-326
Rx, 546
Sprite Fonts, 17
TiltEffect control, 309-310
toggle switches, 312
web page behaviors, 219-221
AddorUpdateItem method, 929
addPushpinCommand, 572
AddressChooserTask, 425-427
AddressChooserTaskViewModel class, 426
addresses
  email
    saving, 397-400
    selecting, 394-396
  street
    resolution from geographic coordinates, 545
    selecting, 425-427
AddTable method, 889-890
AddValidationProperty method, 768
agents
  audio player
    creating, 970-971
    foreground app actions, handling, 975-977
    local media file playback, 972
    play state, toggling, 973
    playlist, defining, 972
    track numbers, managing, 973
    track state changes, responding, 974-975
audio streaming, 196, 983-986
  app assembly audio files, playing, 985
  calling over built-in streaming, 984
  creating, 984
  decoding tracks, 985
  registering, 984
  track information, receiving, 984
  Web files, 986
background, 921
  audio player, 919
  audio. See background audio
  common resources, accessing, 946-949
  feedback to users, providing, 946
  properties, 924
  registering, 920
  shell tiles, updating, 942-944
  types, 919
  scheduled task, 919, 924-925
Alarm class, 908
alarms
   AppBar button, 912
   begin times, 911
   creating, 910-911
   properties, 910
   recurrence, 911
   registering, 909-911
   set alarm view, 912
   sounds, 908-909
   time, setting, 909
AlarmSetCommand, 910
AlarmViewModel
   AlarmSetCommand, 910
   properties, 910
   RecurrenceIntervals property, 911
   SetAlarm method, 910-911
Album property, 970
AlbumArt property, 970
alien spacecraft model website, 684
AllItemsAreInstancesOfType method, 725
AllItemsAreNotNull method, 725
AllItemsAreUnique method, 725
AllowDownloading property, 210
alternate filter modes, 260
Angle property, 373-375
AnimateOrientationChangesFrame class, 110
AnimateTo method, 306
animations
   borders, 380-382
   composition thread, 30
   page orientation changes
      duration, 111
      entire pages, 109-112
      fades, 110
      rotations, 110
      speed, 111-112
   UI elements, 108-109
   page transitions, 112
      adding, 113-114
      navigation events, 112
      style, 115-116
Silverlight
   threads, 29-30
   visibility, 31
Anson, David, 110
APIs
   application bar, 227
   BackgroundTransferRequest class, 953
   Bing Maps key, registering, 554-555
   camera, 645
   isolated storage, 820
   launchers and choosers, 385-386
   MultiScaleImage element, 208
   scheduled task unsupported, 945
   web queries, 862-863
   webcam, 669
      audio formats, 670
      CaptureSource class, 669
      CaptureSourceViewModel Start method, 670
      image collections/formats, 670-672
      playing videos, 674-676
      recording, 673-674
      resolution settings, 670
      saving images, 671
      user authorization, 669
App class, 9-10
AppBar, 243. See also application bar
   background/foreground colors, 234-235, 241-242
   buttons/menu items
      alarm, 912
      controls, 233
      picture taking, 675
      visibility, 234, 245
      todo list, 935
      toggling, 236
      track, 566
      video playback, 675
   click events, 237
   external URLs, navigating, 236
   hiding, 239-241
   instantiating, 236-237

How can we make this index more useful? Email us at indexes@samspublishing.com
MediaView class, 201-202
menu, disabling, 238
minimizing, 238-239
navigation support, 246-247
opacity, 239
pivot support, 329-331
populating, 243-245
sample code, 234-237
system tray appearance, 243
AppBar class, 233, 243-244
AppBarHyperlinkButton class, 234, 246
AppBarHyperlinkMenuItem control, 234
AppBarIconButton control, 233
AppBarMenuItem control, 233
AppBarToggleButton control, 233
AppBarToggleMenuItem control, 233
appearance
application bar, 230
pushpins, 568
system tray, 243
application bar. See also AppBar
API, 227
appearance, 230
ApplicationBar class, 228
Bing Maps items, 558
buttons, 228
background audio foreground app page, 981
backing up/restoring local databases, 965
displaying in trial mode, 741-743
visibility, monitoring, 245
collections, 229
expanding, 229
foreground/background colors, 241-242
hiding, 229, 239-241
hosting multiple with pivots, 329-331
icon buttons
images, 231-232
retrieving at runtime, 232-233
text, 230-231
images, saving, 629-631
markup, 228
menu items, 228
limiting, 229
retrieving, 232-233
text, 230-231
visibility, monitoring, 245
minimizing, 230
navigation support, 246-247
photo upload button, 639
size, 230
system tray appearance, 243
Application Deployment tool, 25
Application List, 628
AppBar class, 228
appearance properties, 230
collections, 229
markup, 228
ApplicationBarIconButton class, 230-232
ApplicationBarMenuItem class, 230-231
ApplicationBarModeToBooleanConverter class, 238-239
ApplicationBarViewModel class, 234
ApplicationCurrentMemoryUsage property, 39
ApplicationMemoryUsageLimit property, 39-40
ApplicationPeakMemoryUsage property, 39
ApplicationSettings property, 824
ApplicationStateType enum, 827
appointments
capability, 26
retrieving, 446-450
accounts available for searching, 447
querying, 446
sample code, 447-450
search results, displaying, 446
Appointments class, 446-450
Accounts property, 447
instantiating, 446
properties, 447
sample code, 447-450
AppointmentsViewModel class, 448-449
AppointmentsView.xaml, 449

**Apps**

- Bing Maps
  - accounts, creating, 554
  - adding, 556-557
  - API key registration, 554-555
  - application bar items, 558
  - Bing logo, hiding, 560
  - BingMapsView.xaml, 557
  - center location, setting, 561
  - copyright text, hiding, 560
  - driving directions, retrieving, 388-391
  - itineraries, displaying, 590-593
  - location tracking, 564-566
  - locations, displaying, 392
  - map modes, 558-560
  - panning, 563
  - pushpins, 567-573
  - route calculation. See Bing Maps, route calculation
  - SOAP services, 574-582
  - viewable area, setting, 561
  - zooming, 562-564
- bookshop sample, 85
  - design-time data, 94-96
  - image caching, 96-97
  - product details, displaying, 90-93
  - product list, displaying, 89-94
  - ProductsView class, 88
  - ProductsViewModel class, 85-87
  - WCF service, 97-99
- Camera, 428-430
- eBay search
  - automatic state preservation, 827-828
  - eBay logo, displaying, 813
  - EbayClient class, 804
  - network connection changes, monitoring, 807
  - OData wrappers, creating, 804-806
  - proxy results class, 805-806
  - records, retrieving, 807-808
  - results, displaying, 808-812
  - search queries, entering, 810
  - view page, 806-813
  - force quitting, 64
  - geographic location features, testing
    - code-driven, 539-541
    - location simulator, 538-539
  - helium voice, 692
    - play command, disabling, 694
    - playback, 695-697
    - recording audio, 693-694
    - saving recordings, 694
    - stopping recording, 694
    - view page, 697-698
- hybrid
  - 3D XNA model, 683-684
  - components, 678
  - game loops, 681-682
  - immediate mode XNA rendering, allowing, 679-681
  - pop-up controls, 688
  - portability, 677
  - project templates, 678-679
  - rendering modes, 679
  - Silverlight elements, rendering in XNA, 684-688
  - XNA content manager, initializing, 682-683
  - XNA gestures, processing, 688-690
- icons, customizing, 467
- life cycle
  - deactivating, 61-62
  - launching, 60-61
  - subscribing, 61
  - tombstoning, 62
- Maps, 390
- Marketplace
  - details page, 400-403
  - launching, 404
  - reviewing, 405
  - searching, 405-406
- Marketplace list, 290-298
  - categories array, 291
  - grouping, 292-298

*How can we make this index more useful? Email us at indexes@sams-publishing.com*
instantiating/populating, 291
item templates, 296
MarketplaceApp class, 291
prices array, 291
viewmodel, 292
media player, 407-411
Messaging, 423-424
messaging with pivot items, 331-339
IChatService, implementing, 332
multiple AppBars, 336-338
PivotViewModel class, 331
sending messages, 333-334, 339
viewing received messages, 334-338
multiple, running, 454
Panorama Bookshop Service sample, 346-350
Photo Location, 617
Photo Picker
cropping photos, 434-436
displaying photos, 437
image streams, 434
launching, 432-433
photo share, 632
certification requirement, 634
converting images to byte arrays, 638-639
creating, 633-641
displaying images, 634, 639-641
dummy images, 633
eulator, 633-635
retrieving images, 633-634
sending images, 635-637
service references, adding, 637
Share menu, adding, 633
Picture Hub reliant, debugging, 617
picture viewer. See picture viewer
pivot, creating, 327
product IDs, retrieving, 402
properties, customizing
Silverlight, 7
XNA, 13

ringtones, 437-440
running under lock screens, 65-66
settings, storing, 826
Silverlight
App class, 9-10
executing, 9
MainPage code-beside, 10-11
MainPage.xaml, 11-12
new projects, creating, 4
properties, 7
running in emulator, 6-7
startup pages, customizing, 7
titles, 8
XAML design view, 6
sketch page sample, 192-195
startup pages, customizing, 7
state, 59
dictionary, 59
fast application switching, 57
loading, 830
persistent, 59, 65, 824-826
transient, 59, 63-65
saving, 830
visibility, 123
stock ticker, 480
channel subscription, 482-484
cloud service subscription, 484-487
images, 489
input controls, 481
server side implementation, 481
stock quotes, receiving, 487-488
storing information, 489
unregistering, 492-493
Yahoo! stock data format, 488
storage space, minimizing, 820
todo list. See todo list app
trial versions, 741-743
Web Browser, 424-425
XNA, 12
execution, 13-16
fonts, 17-20
game loops, 16
Game template, 13
Game Thumbnail, 13
new projects, creating, 12
properties, 13
startup types, 13
arbitrary JavaScript, executing, 220
architecture (geographic locations), 532-533
AreEqual method, 725-726
AreEquivalent method, 726
AreNotEqual method, 726
AreNotEquivalent method, 726
AreNotSame method, 725
AreSame method, 725
arguments
manipulation events, 360
validating, 50-51
ArgumentValidator class, 50-51
Artist property, 970
AskQuestion method, 52
AskYesNoQuestion method, 565
assemblies, testing, 727
AssemblyAudioStreamingAgent class, 985
AssemblyCleanup attribute, 720
AssemblyInitialize attribute, 720
Assert class, 725
assertions, 725-726
AssertLessThan method, 51
AssertNotNull method, 50
AssertNotNullAndOfType method, 51
AssertNotNullOrWhiteSpace method, 51
Assisted Global Positioning System (A-GPS), 531
Association attribute, 853
asynchronous attribute, 723
asynchronous validation, 770-772
all properties at once, 775-779
chat client app, 736-737
DataErrorNotifier class, 767
properties, registering, 768-769
sending email example, 786-789
string properties example, 779-781
whitespace string properties example, 780
AsyncValidationView class, 779-780
AsyncValidationViewModel class, 780
AtEndCommand property, 813
Attendees property, 447
Attitude property, 524
AttitudeReading properties, 524
attributes. See properties
audio
background
agent, creating, 970-972
audio streaming agents, 983-986
classes, 968
controlling from foreground apps, 978-983
foreground app actions, handling, 975-977
local media file playback, 972
play state, toggling, 973
player, 968-970
track information, 970
track numbers, managing, 973
track state changes, responding, 974-975
capturing, 669
CaptureSource class, 669
formats, 670
playing, 674-676
starting/stopping, 673-674
user authorization, 669
FM radio
displaying, 706-708
FMRadio class, 698
frequencies, 698, 701-705
onscreen menu, 705
power modes, 700-701
regions, 699, 702
signal strength, 699, 705
turning on/off, 698
view page, 708
media viewer page sample code, 196-202
AppBar control, 201-202
binding properties to viewmodel, 199

How can we make this index more useful? Email us at indexes@samspublishing.com
MediaView class, 197-199
muting/unmuting playback, 197
position control, 199
scroll viewer, 200-201
video, playing, 200
MediaElement, 195
output, 196
playing
agents, 919
controls, 196
media player, 407-411
recording with microphone
helium voice app, 692-698
Microphone class, 691-692
ringtones, 437-440
sound effects, 195, 202-204
sources, 196
streaming, 196, 983-986
app assembly audio files, playing, 985
creating, 984
decoding tracks, 985
registering, 984
track information, receiving, 984
Web files, 986
volume, 196
AudioPlayerAgent class, 968
OnError method, 977-978
OnPlayStateChanged method, 974-975
OnUserAction method, 975-977
AudioStreamingAgent class, 968, 984
AudioTrack class, 970
authentication, 480
authorization, 669
AutoCompleteBox control
custom filters, 260-262
custom template website, 256
data binding, reapplying, 266
filter modes
alternate, 260
availability, 259
selecting, 260
values, 257-259
overview, 255
suggestion list
bug, 256
displaying, 256
dynamic population, 262-264
styles, 264-266
viewmodel control, 256, 259
AutoCompleteFilterMode enum values, 257-259
autofocus (camera), 655
AutoFocusCompleted event, 649
automated testing
coded UI tests, 711
integration tests, 711
unit tests, 710
automatic state preservation, 826
binary serialization, 834-837
dictionaries, 831-834
eBay search app example, 827-828
Lambda expressions, unwinding, 837-838
methods, 827
property accessor delegates, creating,
838-839
state
loading, 830
restoring, 834
saving, 830
stateful properties, identifying, 830,
836-837
type required, 827
viewmodels, customizing, 828-829
Automation framework, 737-738
AutomationPeer objects, 738
AutoPlay property, 196
AutoSync property, 849
availability
camera controls, 651
cultures, 604
filter modes, 259
isolated storage space, 822
motion sensor, 523
network connections, checking, 793
AvailableFreeSpace property, 822
AverageAcceleration property, 502
axis values (accelerometer), 498

background agents, 921
  audio. See background audio
  background tasks, compared, 921
  common resources, accessing, 946-949
  feedback to users, providing, 946
  properties, 924
  registration, 920
  scheduled task, 919, 924-925
  shell tiles, updating, 942-944
  types, 919

background audio
  agent
    creating, 970-971
    foreground app actions, handling, 975-977
    local media file playback, 972
    play state, toggling, 973
    playlist, defining, 972
    track numbers, managing, 973
    track state changes, responding, 974-975
  audio streaming agents, 983-986
    app assembly audio files, playing, 985
    calling over built-in streaming, 984
    creating, 984
    decoding tracks, 985
    registering, 984
    track information, receiving, 984
    Web files, 986
  classes, 968
  controlling from foreground apps, 978
    application bar buttons, 981
    playback progress, 982-983
    slider control, 983
    testing, 978
  player, 968-970
  track information, 979-981
  updating, 979
  user opt-in requirement, 978
  viewmodel, 979, 982
  visual state, 980

background file transfer requests
  app terminations, handling, 956
  BackgroundTransferRequest class, 952
  copying files to temporary directory, 960
  current state, identifying, 955
  deleting, 952
  existing, retrieving, 956
  local databases, backing up, 960-963
  progress, monitoring, 954, 961-962
  queue limits, 952
  requirements, 952
  restoring local databases, 963-965
  re-subscribing, 956
  results, displaying, 962
  saving files to Backups directory, 956-957
  status changes, monitoring, 954
  submitting, 952
  todo list viewmodel example, 960
  upload, 953, 961
  URL rerouting, 957-958, 961
  Windows Live anonymous IDs, retrieving, 958-960

background layer (Panorama control), 344-346
Background property, 230
background tasks, 906, 921
BackgroundAudioPlayer class, 968-970
  behaviors, 969
  methods, 969
  PlayTrack method, 973
  properties, 969
  testing, 978
BackgroundAudioPlayerAgent, 972
BackgroundServiceAgent properties, 924
BackgroundTransferRequest class, 952
  API, 953
  TransferPreferences property, 952
  TransferProgressChanged event, 954
  TransferStatusChanged event, 954
  UploadLocation property, 953
BackgroundTransferService class
  Find method, 956
  Remove method, 952
BackgroundWorker class
  operation cancellation, 152
  progress bars, updating, 149-153
  RunWorkerCompleted method, 152
  WorkerReportsUpdates property, 152
backing up local databases, 960-963
BackupDatabase method, 960-963
BackupService class, 956-957
backward navigation, 78, 112
bank accounts example
  BankAccount class, 895-900
  BankingDatabaseUtility class, 898
  BankingDataContext class, 897-898
  CheckingAccount class, 897
  conflict detection, 900-901
  version property adding, 899-900
BankAccount class, 895-900
BankingDatabaseUtility class, 898
BankingDataContext class, 897-898
BankingDataContextTests class, 900
BarOpacity property, 239
batching intervals (notifications), 478-479
begin times (alarms), 911
BeginGetPropertyErrorsFromValidator method, 772
BeginTime property, 908-910
BeginValidation method, 769-772
binary serialization, 834-837
BindableChangeNotifier class, 601
BindableResources class, 599
BindingExpression class, 751, 760
BindToShellTile method, 459
BindToShellToast method, 459
Bing Maps
  accounts, creating, 554
  adding, 556-557
  API key registration, 554-555
  application bar items, 558
  Bing logo, hiding, 560
  BingMapsView.xaml, 557
  center location, setting, 561
  copyright text, hiding, 560
  driving directions, retrieving, 388-391
  itineraries, displaying, 590-593
  locations
    displaying, 392
    tracking, 564-566
  map modes, 558-560
  panning, 563
  pushpins, 567
    appearance, 568
    creating, 572
    displaying, 573
    icon images, 571-572
    populating, 567-569
    presenting, 570
    template, 570
    user prompt for naming, 573
  route calculation
    completed route, displaying, 590
    Geocode/Route services, 576-578
    searching for routes, 584-585
    user provided to and from addresses, 582-584
    visual states, 586-590
SOAP services
  client configuration file, 575
  coordinating calls with Rx, 579-582
  listing of, 574
  reference, adding, 575
  viewable area, setting, 561
  zooming, 562-564
BingMapsDirectionsTask, 388-391
BingMapsDirectionsTaskViewModel class, 389-390
picture taking, 675
todo list, 935
toggling, 236
track, 566
video playback, 675
visibility, 234, 245
application bar, 228
  background audio foreground app page, 981
  backing up/restoring local databases, 965
  displaying in trial mode, 741-743
  visibility, monitoring, 245
Bing Maps zoom, 563-564
camera shutter, monitoring, 650
check boxes, 133-140
  data binding to SelectedItems property, 138-140
  data binding to viewmodel collection, 134-137
  example, 134
  states, 133
clicking, 125
calendar appointments
  querying, 446
  retrieving, 446-450
Calibrate event, 515
Calibrate() method, 503
calibrating
  accelerometer, 503
  compass, 515-517
  motion sensors, 526
camera
  APIs, 645
  audio, capturing, 669
    CaptureSource class, 669
    formats, 670
    playing, 674-676
    recording, starting/stopping, 673-674
    user authorization, 669
capabilities, 26, 646
calculating. See measuring
calculator, 96-97
calculation, 246
Cached data, retrieving from local databases, 863-864
cached timelines, retrieving, 869-870
caching images, 96-97
Calcium project website, 233
CalciumSDK repository, 55
CalculateAsync method
  IRouteCalculator interface, 576
  RouteCalculator class, 578
calculating. See measuring
calendar appointments
  querying, 446
  retrieving, 446-450
Calibrate event, 515
Calibrate() method, 503
calibrating
  accelerometer, 503
  compass, 515-517
  motion sensors, 526
camera
  APIs, 645
  audio, capturing, 669
    CaptureSource class, 669
    formats, 670
    playing, 674-676
    recording, starting/stopping, 673-674
    user authorization, 669
capabilities, 26, 646
calculating. See measuring
calendar appointments
  querying, 446
  retrieving, 446-450
Calibrate event, 515
Calibrate() method, 503
calibrating
  accelerometer, 503
  compass, 515-517
  motion sensors, 526
camera
  APIs, 645
  audio, capturing, 669
    CaptureSource class, 669
    formats, 670
    playing, 674-676
    recording, starting/stopping, 673-674
    user authorization, 669
capabilities, 26, 646
configuring, 647
controls available, 651
flash modes, 655
focus
  autofocus, 655
  settings, 649
front facing, 646
gray-scale video effect
  ARGB color components, extracting, 661
  converting back to color, 661
  grayscale conversion, 661
  previewing, 659-660
  processing, 659
  result, 661
  turning on/off, 658
image filenames, 651
PhotoCamera class, 646
photos
  displaying, 657
  taking, 428-430, 649-651
resolution settings, 649
resources, unsubscribing, 653
saving to media library, 652
shutter button, monitoring, 650
state, 657
thumbnails
  displaying, 655, 662-668
  saving, 653
video
  capturing. See video, capturing
  resolution settings, 670
CameraButtons class, 650
CameraCaptureTask, 428-430
CameraCaptureViewModel, 429
CanBeNull property, 849
CanCalibrate() method, 503
Cancel property, 79
CancelEventArgs class, 79
cancelling
  page navigation, 79
  progress bars, 152
CanExecute() method, 49
CanExecuteChanged() method, 49
CanPause property, 969
CanSeek property, 969
capabilities
  app required, analyzing, 28
  camera, 646
  defined, 26
  discovery process, 28
  displaying, 26
  exceptions, 27
  listing of, 26-27
  manifest file, 26-27
CaptureImageAsync method, 671
CaptureImageAvailable event, 649
CaptureSource class, 669
CaptureSourceView page, 674-676
CaptureSourceViewModel class
  commands, 671
  HandleCaptureImageCompleted method, 672
  PlayVideo method, 674
  Start method, 670
  TakePhoto method, 671
  ToggleCapturingVideo method, 673
  UpdateCommands method, 671
CaptureThumbnailAvailable event, 649
capturing audio/video, 669
  audio formats, 670
  CaptureSource class, 669
  helium voice app, 692-698
  images, 649
    formats, 670
    saving, 671
  microphone, 691-692
  playing, 674-676
  resolution settings, 670
  starting/stopping, 673-674
  user authorization, 669
  case sensitive parameter, 858
  cell tower triangulation, 531
Center property
  BingMapsTask, 392
  Map class, 561
certifications
Back button, 82
background audio user opt-in setting, 978
Extras applications, 621
photo share applications, 634
push notifications, disabling, 457
change notifications
data models, 851
properties, 44
alternative implementation, 46-48
traditional implementation, 44-46
channelName parameter, 458
ChannelOpenFailed errors, 461
channels, push notifications
cloud, creating, 480
creating, 458
errors, 460-461
finding, 458
shell, binding, 459
stock ticker app, 482-484
ChannelUriUpdated event, 460
chat client, 728
asynchronous testing, 736-737
coded UI tests, 734-736
internal main project members
access, 736
PhoneApplicationPage verification, 735
Send button, testing, 736
IChatService interface, 728
mock chat service, 728
UI elements, runtime manipulation,
737-738
unit tests, creating, 730-732
view elements, 732-734
viewmodel, 729-730
ChatClientUITests class, 735
ChatClientViewModel class
code listing, 729-730
test methods, 730-732
view elements, 732-734
ChatClientView.xaml, 733
check boxes, 133-140
data binding
SelectedItems property, 138-140
viewmodel collection, 134-137
example, 134
states, 133
Checked event, 129
CheckingAccount class, 897
child elements, positioning, 315
child element spacing, 315
layout modes, 315
list box example, 317-318
overview, 315
vertical/horizontal sample code, 315-317
ChooserBase class
Completed event, 387-388
overview, 386
choosers
AddressChooserTask, 425-427
API, 385-386
CameraCaptureTask, 428-430
ChooserBase class, 386
Completed event, 387-388
EmailAddressChooserTask, 394-396
listing of, 386
overview, 385
EmailAddressChooserTask, 394-396
overview, 385
PhoneNumberChooserTask, 415-417
PhotoChooserTask
Completed event, 432
cropping photos, 434-436
displaying photos, 437
Extras applications, launching, 628
image streams, 434
Photo Picker, launching, 432-433
PhotoChooserView page, 436
properties, 432
sample code, 434-435
Show method event, 433
SaveContactTask, 427-428
SaveEmailAddressTask, 397-400
BitmapImage, 622
BitmapImageConverter, 603
BookshopService, 97
BooleanToBrushConverter, 304
BooleanToVisibility, 84
Button, 125
ButtonBase, 309
CameraButtons, 650
CancelEventArgs, 79
CaptureSource, 669
CaptureSourceViewModel
commands, 671
HandleCaptureImageCompleted method, 672
PlayVideo method, 674
Start method, 670
TakePhoto method, 671
ToggleCapturingVideo method, 673
UpdateCommands method, 671
ChatClientUITests, 735
ChatClientViewModel
code listing, 729-730
test methods, 730-732
view elements, 732-734
CheckBox, 133-140
data binding to SelectedItems property, 138-140
data binding to viewmodel collections, 134-137
example, 134
states, 133
CheckingAccount, 897
ChooserBase, 386-388
City, 253-254
Clipboard, 182-183
CollectionAssert, 725-726
ColorToBrushConverter, 241
CommonResourceExample, 947
Compass
Calibrate event, 515
IsSupported property, 509
TimeBetweenUpdates property, 509
CompassViewModel, 510
calibration, 516
properties, 513
Start method, 509, 514
Contacts, 440
Accounts property, 442
instance, defining, 441
instantiating, 441
properties, 299
sample code, 442-445
SearchAsync method, 441
ContactsViewModel
results, displaying, 444-445
retrieving contacts, 442-444
ContentControl, 122-123
Control, 158
ControlExamplesViewModel, 149-152
controls hierarchy, 118
CredentialsProvider, 578
CultureName, 604
custom item, 816
CustomDatePickerView, 281-282
CustomGrouping, 292-294
DatabaseSchemaUpdater, 888
AddAssociation method, 891-893
AddColumn method, 888
AddIndex method, 890-891
AddTable method, 889-890
DatabaseSchemaVersion property, 894
DatabaseUtility, 855
DataboundPivotViewModel, 340
DataContextChangedListener, 782
code listing, 782-784
Subscribe method, 782
DataErrorNotifier, 767
AddValidationProperty method, 768
decoupling validation, 772
ViewModelBase class validation, 767
DataErrorValidator, 775
DataItem, 130-133
DataValidation, 769
DateGroupingKey, 931
DateTimePickerBase, 273
DebugStreamWriter, 885-887
DelegateCommand, 49
Dependency, 739
DeviceExtendedProperties, 39
DeviceProperties, 958
DeviceStatus
  ApplicationMemoryUsageLimit property, 40
  DeviceStatusView page, 39
  KeyboardDeployedChanged event, 41
  overview, 38
  PowerSourceChanged event, 41-42
  properties, 39
EbayClient, 804
EbaySearchStateViewModel, 827
EbaySearchView, 809-813
EbaySearchViewModel, 806-809
  commands, 806
  FetchMoreDataCommand, 816
  HandleEbayClientSearchItemsComplete method, 808-809
  Search method, 807-808
EnhancedAccelerometer, 501
  AverageAcceleration property, 502
  calibration methods, 503
  DetectShake method, 507-508
  LowPassFilterCoefficient property, 502
  LowPassFilteredAcceleration property, 502
  NoiseThreshold property, 503
  OptimallyFilteredAcceleration property, 502
  Reading property, 501
  sample view, 503-506
  shake detection, 507-508
  ShakeCount property, 508
  starting, 501
EnumUtility, 173, 702
Expression, 838
FadeOrientationChangesFrame, 110
FlatListViewModel, 288-289

FMRadio, 698
FMRadioViewModel
  Frequency property, 701
  PoweredOn property, 700-701
  Regions property, 702
  SignalStrength property, 705
  UpdateFrequencyRanges, 704-705
FrameNavigationService, 883-884
FrameworkElement, 118
  DataContext property, 782
  FlowDirection property, 612
  HorizontalAlignment, 123
  VerticalAlignment, 123
Game, 16
GamePageViewModel, 683
GameTimer, 681-682
  events, 681
  properties, 682
  Update event, 685
GeoCoordinateWatcher, 534
  DesiredAccuracy property, 534
  GeoPositionAccuracy enum, 534
  instantiating, 534
  MovementThreshold property, 534
  position change events, 535-537
  Start method, 537
  TryStart method, 537
  Wait method, 541
  WalkPath method, 539-540
GeoPositionSampler, 548-550
GeoPositionViewModel, 541
  DelegateCommands, 543
  Start method, 542
GestureEventArgs, 125, 365
GestureListener, 369
  double tap gestures, 370
  drag gestures, 371-372
  flick gestures, 373
  hold gestures, 370
  pinch gestures, 374-376
  tap gestures, 369-370
GestureService, 369

How can we make this index more useful? Email us at indexes@samspublishing.com
Gyroscope
   IsSupported property, 518
   Stop method, 520
   TimeBetweenUpdates property, 518
GyroscopeViewModel, 519
IChatService, 332
ImageCacheConverter, 96-97
ImageInfo, 662
ImageUploadService, 637
InkPresenterView, 191, 194
InkPresenterViewModel, 193-194
InputScopeValueConverter, 174-175
IsolatedStorageFile, 821-822
IsolatedStorageSettings, 824-826
IsolatedStorageUtility, 880-881
IsolatedStorageUtilityTests, 882
ItemsControl, 140
LicenseInformation, 741-743
ListBox, 132, 142-144
ListBoxItem, 142, 309
LocalizabilityViewModel
   CultureName property, 606
   SetCulture method, 605
   SetCurrentCultureValues method, 607
   SupportedCultures property, 605
LocalizedToggleSwitchConverter, 313
LockScreenManager, 66
   code listing, 67
   idle detection mode, 68
   ILockScreenManager interface, 66
   storing, 67
LockScreenViewModel, 68-69
MainWindowViewModel, 979-982
   PlayerState property, 980
   Position property, 982-983
   Refresh method, 979
   TrackArtist/TrackTitle properties, 979
   ViewState property, 981
ManipulationEventsView, 362-363
Map
   Center property, 561
   Mode property, 559
   SetView method, 561
   ZoomBarVisibility property, 563-564
   ZoomLevel property, 562
   MapItineraryItem, 590
   MapMode, 559
   MarketDetailTask, 743
   MarketplaceApp, 291
      categories array, 291
      grouping items, 292
      instantiating/populating, 291
      prices array, 291
      viewmodel, 292
   MediaElement, 195-196
   MediaLibrary, 621
   MediaLibraryImageViewModel
      LoadImage method, 667
      Title property, 668
   MediaPlayerLauncherViewModel, 409-411
   MediaStreamSource, 985
   MediaView, 197-199
      AppBar control, 201-202
      ScrollView, 200-201
   MenuItem, 310
   MessageService, 52-54
   MessagesViewModel, 334-336
   Microphone, 691-692
   MicrophoneViewModel
      code listing, 693
      commands, 693
      HandleBufferReady method, 694
      StartRecording method, 694
      StopRecording method, 694
      ToggleRecording method, 693
      UpdateCommands method, 694
   MockChatService, 728
   MockGeoCoordinateWatcher, 539
   MockLicensingService, 741
   MockMarketplaceDetailTask, 743
   Motion, 522
      CurrentValueChanged event, 524
      TimeBetweenUpdates property, 523
PushNotificationSubscriber, 458
PushNotifier
  raw notifications, 474-476
toast notifications, 465-466
Pushpin, 567
PushpinIconConverter, 571
PushpinViewModel
  addPushpinCommand, 572
code listing, 568
displaying, 570
properties, 569
RadiansToDegreesConverter, 525
RadioButton, 129-130
data binding to viewmodel collections,
  130-133
events, 129
GroupName property, 130
RangeBase, 144-146
ReadingSmoother, 511-513
ReminderView, 915-916
ReminderViewModel, 914-915
RepeatButton, 126-128
Request, 955
ResourceIntensiveTask, 922
ResultEventArgs, 805
Route, 583
RouteCalculationResult, 577
ScheduledAction
  BeginTime property, 908
  ExpirationTime property, 909
ScheduledActionService, 909
ScheduledTaskAgent, 919
ScrollBar, 154
ScrollViewerMonitor, 813
  AtEndCommand property, 813
code listing, 813-816
VerticalOffset property, 813
SearchItemsCompleteEventArgs, 806
SelectedItems, 138-140, 411-413
Selector, 141
SendEmailView, 786
SendEmailViewModel, 786, 789
SendMessageViewModel, 333-334
SensorBase, 495-497
SharedGraphicsDeviceManager, 679-681
ShellTileSchedule, 473-474
SilverlightSerializer, 834
SilverlightTest, 724-725, 737
Slider, 153-154
  Orientation property, 153
templates, 154
Value property, 153
SoundEffect, 195, 202-204
StateManager, 829
StockQuote, 489
StockQuoteService, 484-487
StringAssert, 726
StringToImageConverter, 664-665
SuggestionItem, 264
SystemTray, 243
TableAttribute, 848
TaskScheduler
  OnInvoke method, 942-943
  ProcessPeriodicTask method, 943-944
Table<TEntity>, 855
TemplatedItemsControl, 324
test, creating, 714-716
TextBlock, 168
TextBox, 168
ThumbnailsViewModel, 663-664
TimelineItem, 846-848
TodoDataContext, 928-929
TodoItem, 927-928
TodoItemView, 935
TodoItemViewModel
  AppBar button, 941
  LoadItem method, 939
  SaveItem method, 937-938
  updating, 941
  VisualState property, 940-942
TodoListViewModel, 932, 960
TodoService, 929-930
TodoTileDataCreator, 936
ToggleButton, 128-129
ToggleSwitch, 311-314
ClickMode property, 125
Clipboard class, 182-183
clipboard text, 182-183
cloud push notifications, 456, 480
code-beside files, 4, 10-11
Code Contracts tool, 50
code-first data model creation, 845-850
  benefits, 845
cached data, retrieving, 863-864
cached timelines, retrieving, 869-870
change notifications, 851
columns
  attributes, 848
  change conflicts, 850
data types, 849
entity members, synchronizing, 850
inheritance hierarchy, 850
null values, 849
OCC, 850
primary keys, 849
SQL expressions, 850
synchronizing automatically, 849
connection strings, 856-859
CRUD operations, 855
database file, 856
display controls, 871-872
encryption, 858
entities
  creating, 851-853
disconnected, attaching/detaching, 855
one-to-many relationships, 853
public fields, 855
enum value conversions, 859
indexes, 851
initialization, 856
isolated storage files, creating, 855
population, 861
results, handling, 870
retrieving users from database, 860
table names, 848

How can we make this index more useful? Email us at indexes@samspublishing.com
users
- credentials, 864-867
- details/status updates, 860
- web API queries, 862-863

coded UI tests
- chat client app, 734-736
  - asynchronous testing, 736-737
  - internal main project members access, 736
  - PhoneApplicationPage verification, 735
  - Send button, testing, 736
- elements, runtime manipulation, 737-738
  - overview, 711

CodePlex, 250
CollectionAssert class, 725-726
collections
- application bar, 229
- assertions, 725-726
- OData, populating, 802-803

colors
- AppBar, 234-235
- application bar, 241-242
- ARGB, extracting, 661
- borders, 358-359
- fonts, 163
- itineraries, 592
- Panorama control, 344
- resources, creating, 754
- styles, 230
- tile notifications, 473
- ToggleSwitch control, 314

ColorToBrushConverter class, 241
command-line options (Isolated Storage Explorer), 874

commands
- addPushpinCommand, 572
- AlarmSetCommand, 910
- CaptureSourceViewModel, 671
- DelegateCommands, 543
- EbaySearchViewModel, 806
- execution, evaluating, 788
- FetchMoreDataCommand, 806, 816
- ICommand interface, 49
- instantiating, 49
- itineraryToggleCommand, 586
- loadUserTimelineCommand, 868
- map visual state properties, 586
- MicrophoneViewModel class, 693
- PlayCommand, 199
- ReminderViewModel, 914
- routeSearchToggleCommand, 586
- SaveCommand, 629
- SearchCommand, 806
- SubmitCommand, 780
- support, 50
- ToggleCalibrationCommand, 517
- UploadCommand, 635-636
- ViewImageCommand, 664

common resources, accessing, 946-949
CommonResourceExample class, 947

communication
- cross-page, 80
- interpage, 74
- web pages, 216-217
  - loading web pages, 216-217
  - receiving messages from web pages, 219
  - script, enabling, 217
  - sending messages to web pages, 218-219
  - web page elements, 216

compass
- calibrating, 515-517
- interference, 509
- magnetic north arrow, 513-514
- noise, removing, 513
- orientation, 514-515
  - overview, 508
  - reading, 510
  - smoothing data, 511-513
  - starting, 509
  - support, 509
  - time between updates, 509
  - updates, receiving, 509
  - view page, 513
<table>
<thead>
<tr>
<th>Class</th>
<th>Properties/Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compass class</td>
<td>Calibrate event, 515</td>
</tr>
<tr>
<td></td>
<td>IsSupported property, 509</td>
</tr>
<tr>
<td></td>
<td>TimeBetweenUpdates property, 509</td>
</tr>
<tr>
<td>CompassView page</td>
<td>513</td>
</tr>
<tr>
<td>CompassViewModel class, 510</td>
<td>calibration, 516</td>
</tr>
<tr>
<td></td>
<td>properties, 513</td>
</tr>
<tr>
<td></td>
<td>Start method, 509, 514</td>
</tr>
<tr>
<td>Completed event, 387-388</td>
<td>AddressChooserTask, 426</td>
</tr>
<tr>
<td></td>
<td>CameraCaptureTask, 429</td>
</tr>
<tr>
<td></td>
<td>EmailAddressChooserTask, 394</td>
</tr>
<tr>
<td></td>
<td>GameInviteTask, 431</td>
</tr>
<tr>
<td></td>
<td>PhoneNumberChooserTask, 416</td>
</tr>
<tr>
<td></td>
<td>PhotoChooserTask, 432</td>
</tr>
<tr>
<td></td>
<td>SaveContactTask, 427</td>
</tr>
<tr>
<td></td>
<td>SaveEmailAddressTask, 398</td>
</tr>
<tr>
<td></td>
<td>SavePhoneNumberTask, 417</td>
</tr>
<tr>
<td></td>
<td>SaveRingtoneTask, 438</td>
</tr>
<tr>
<td>contacts</td>
<td>calling, 413-414</td>
</tr>
<tr>
<td></td>
<td>capability, 26</td>
</tr>
<tr>
<td></td>
<td>displaying, 298-304</td>
</tr>
<tr>
<td></td>
<td>email</td>
</tr>
<tr>
<td></td>
<td>saving, 397-400</td>
</tr>
<tr>
<td></td>
<td>selecting, 394-396</td>
</tr>
<tr>
<td></td>
<td>links, sharing, 422</td>
</tr>
<tr>
<td></td>
<td>phone numbers</td>
</tr>
<tr>
<td></td>
<td>saving, 417-419</td>
</tr>
<tr>
<td></td>
<td>selecting, 415-417</td>
</tr>
<tr>
<td></td>
<td>retrieving, 440</td>
</tr>
<tr>
<td></td>
<td>accounts available for searching, 442</td>
</tr>
<tr>
<td></td>
<td>sample code, 442-445</td>
</tr>
<tr>
<td></td>
<td>search results, displaying, 441</td>
</tr>
<tr>
<td></td>
<td>saving, 427-428</td>
</tr>
<tr>
<td></td>
<td>searching, 441</td>
</tr>
<tr>
<td></td>
<td>street addresses, selecting, 425-427</td>
</tr>
<tr>
<td>Contacts class, 440</td>
<td>Accounts property, 442</td>
</tr>
<tr>
<td></td>
<td>instantiating, 441</td>
</tr>
<tr>
<td></td>
<td>sample code, 442-445</td>
</tr>
<tr>
<td></td>
<td>SearchAsync method, 441</td>
</tr>
<tr>
<td>ContactsViewModel class</td>
<td>results, displaying, 444-445</td>
</tr>
<tr>
<td></td>
<td>retrieving contacts, 442-444</td>
</tr>
<tr>
<td>ContactsView.xaml, 444-445</td>
<td>Contains filter, 257</td>
</tr>
<tr>
<td></td>
<td>Contains method, 726</td>
</tr>
<tr>
<td></td>
<td>ContainsCaseSensitive filter, 257</td>
</tr>
<tr>
<td></td>
<td>ContainsOrdinal filter, 257</td>
</tr>
<tr>
<td></td>
<td>ContainsOrdinalCaseSensitive filter, 257</td>
</tr>
<tr>
<td></td>
<td>ContainsText() method, 182</td>
</tr>
<tr>
<td></td>
<td>content controls, 121-123</td>
</tr>
<tr>
<td>Content property</td>
<td>AlarmViewModel, 910</td>
</tr>
<tr>
<td></td>
<td>defining, 122-123</td>
</tr>
<tr>
<td></td>
<td>NavigationEventArgs class, 80</td>
</tr>
<tr>
<td></td>
<td>values, 122</td>
</tr>
<tr>
<td>ContentControl class, 122-123</td>
<td>ContentIdentifier property, 401</td>
</tr>
</tbody>
</table>
ContentManager, initializing, 682-683

ContentType property
MarketplaceDetailTask, 401
MarketplaceHubTask, 404
MarketplaceSearchTask, 405

context menus
adding, 267
button example, 267
list boxes, 271-273
menu items, binding, 270
overview, 267
page background scaling, 268-269
tilt effect, adding, 310
viewModel, 269-271

Control class, 158
ControlExamplesViewModel class, 149-152
controls. See specific controls
Controls property, 407

ConvertArgbToColor method, 661
ConvertColorToArgb method, 661
ConvertColorToGrayScale method, 661

converting
eum values, 174-175
radians to degrees, 525

CopyDatabaseToIsolatedStorage method, 882
CopyrightVisibility property, 560
CopyStreamBytes method, 225-226
CPU Usage graph (Performance Analysis tool), 36-37

CreateDelegate method, 838
CreateDirectory method, 822
CreateFile method, 822
CreateGetter method, 838
CreateNewPushPin method, 572
CreateOptions property, 622
CreateRandomApp method, 291
credentials (map routes), 578
CredentialsProvider class, 578
critical exceptions, 751
cropping photos, 434-436
cross-page communication, 80
cross-site restrictions, 221

cultures
availability, 604
codes
databases, 857
resx file names, 598
website, 857
date/time/currency formatting
properties, 607
names, 606
setting, 605
supported, listing of, 605

CumulativeManipulation property, 361
Current property, 61
CurrentRegion property, 699
CurrentSource property, 78
CurrentValue property, 496

CurrentValueChanged event
Accelerometer class, 499
Motion class, 524
SensorBase class, 496

CustomDatePickerView class, 281-282
CustomGrouping class, 292-294

customizing
application bar, 230, 241-242
apps
icons, 467
properties, 7, 13
titles, 8
AutoCompleteBox filters, 260-262
border colors, 358-359
DatePicker/TimePicker value formats, 277-278
full-screen picker pages, 278-282
constructor, 280
holding dates, 279
list of dates, 279
retrieving values, 281-282
languages at runtime, 603
cultures, 604-606
data binding path/source values, 611
data context changes, detecting, 782-785
retrieving over networks, 813-816
data binding
AutoCompleteBox control, 266, 259
check boxes
   SelectedItems property, 138-140
   viewmodel collections, 134-137
LoopingSelector control, 286
Path property, 611
property change notifications, 44
   alternative implementation, 46-48
   traditional implementation, 44-46
radio button controls, 749-752
SelectedItems property, 411-413
Source property, 611
ToggleSwitch, 312-313
data models
code-first creation, 845-850
benefits, 845
cached data, retrieving, 863-864
cached timelines, retrieving, 869-870
data source parameter, 857
database-first data models, 878-880
databases
backing up, 960-963
case sensitivity, 858
columns, adding, 888
concurrency, 899
   conflict detection, 900-901
   entities, designating, 899
culture codes, 857
database-first data models, 878-880
deploying to isolated storage, 880-882
foreign key constraints, 891-893
indexes, 890-891
LINQ to SQL, 841, 885-887
local. See local databases

How can we make this index more useful? Email us at indexes@samspublishing.com
locking modes, 857
max buffer size, 857
passwords, 857
restoring, 963-965
schema
displaying, 873
isolated storage file explorers, 873-876
SQL CE files, 876-878
updating, 887-893
versioning, 894
size, 857
SQL Server Compact Edition, 842
tables, 889-890
DatabaseSchemaUpdater class, 888
DatabaseSchemaVersion property, 894
methods
AddAssociation, 891-893
AddColumn, 888
AddIndex, 890-891
AddTable, 889-890
DatabaseSchemaVersion property, 894
DatabaseUtility class, 855
DataBoundPivotViewModel class, 340
DataBoundPivotView.xaml, 341
DataContext property, 43, 782
DataContextChanged Listener class, 782-784
DataContractJsonSerializer type, 476
DataErrorNotifier class, 767
AddValidationProperty method, 768
decoupling validation, 772
ViewModelBase class validation, 767
DataErrorValidator class, 775
DataItem class, 130-133
DataValidationException class, 769
Date property, 279
DateGroupingKey class, 931
DatePicker control, 273-275
alarms, 911
full-screen picker page, 278-282
constructor, 280
holding dates, 279
list of dates, 279
retrieving values, 281-282
header, 277
icons, 275
selection mode, 273
TimePicker, compared, 274
value format, 277-278
ValueChanged event, 275
dates
DatePicker, 273-275
alarms, 911
full-screen picker page, 278-282
header, 277
icons, 275
selection mode, 273
TimePicker, compared, 274
value format, 277-278
ValueChanged event, 275
LoopingSelector control, 283-286
data binding, 286
data source, 283
numeric values, presenting, 283-286
scheduled notifications, 908
DateTimePickerBase class, 273
DbType property, 849
d:DataContext markup extension, 94-96
Deactivated event, 61-62
debugging. See also troubleshooting
media database, unlocking, 641-643
Picture Hub reliant apps, 617
scheduled tasks, 932
DebugStreamWriter class, 885-887
decoding audio tracks, 985
decoupling validation, 772
Deep Zoom images
animations, enabling/disabling, 209
collection, accessing, 209
creating, 206-207
downloading, stopping, 210
instantiating, 208
logical coordinates, 209
disabling
   AppBar menu, 238
   idle detection mode, 68
   push notifications, 457
   scheduled tasks, 918

dismissing
   reminders, 917
   SIP, 170-171

Dispatcher property, 81
   display modes (ListPicker control), 252-255
   DisplayAllGroups property, 305
   DisplayGroupView method, 306

displaying
   3D XNA model, 683-684
   app details (Marketplace), 400-403
   application bar button in trial mode, 741-743
   background audio tracks on foreground app page, 981
   Bing Maps controls, 558-560
   camera photos, 657
   capabilities, 26
   contacts list, 298-304
   eBay logo, 813
   enum values, 176
   existing reminders, 915
   flat lists, 288-290
   FM radio, 706-708
   grouped lists, 290
   localized text/images, 607-610
   locations, 543-545
   maps
      itineraries, 590-593
      locations, 392
      pushpins, 573
   message boxes, 52-54
      class diagram, 52
      IMessageService interface, 52
      MessageService class, 52-54
   multiple application bars, 329-331
   OData search results, 810-812
   photos, 437
   PivotItems, 327
   product
      details, 90-93
      lists, 89-94
      ringtones, 437
   scheduled tasks, 933-935, 940-942
   schema, 873
      isolated storage file explorers, 873-876
      SQL CE files, 876-878
   shell tiles, 942
   test results, 715
   thumbnails, 655, 662-668
      creating, 663
      loading from media library, 666-668
      location, identifying, 662
      media library names, 662-664
      new images, detecting, 663
      page title, 668
      pages, populating, 665-666
      path to image conversions, 664-665
      presenting, 664
   tile notifications, 472
   toast notifications, 467
   tweets, 868-872
      cached timeline, retrieving, 869-870
      controls, 871-872
      results, handling, 870
      screen name verification, 869
      status updates, retrieving, 868
   validation errors
      details, 762-765
      visual states, 752-757

Dispose method, 497
Distance property, 375
DistanceRatio property, 376
DoesNotContain method, 726
DoesNotMatch method, 726
Dolhai network connections article, 794
double tap gestures
   border example, 378
   Deep Zoom images, 213
DoubleTap event
GestureListener class, 370
UIElement, 366
downloading
Expression Blend, 753
high-resolution images, 210
SDK, 2
Toolkit, 250
UTF, 712
DownloadProgress property, 196
DownloadProgressOffset property, 196
drag gestures, 371-372
border example, 378-379
Deep Zoom images, 212
Toolkit, 371-372
DragCompletedGestureEventArgs parameter, 372
DragDeltaGestureEventArgs parameter, 372
DragStartedGestureEventArgs parameter, 372
Draw method, 16
drawing surfaces, 188
sketch page sample app, 192-195
strokes, 188-189
user input, 190-191
DrawingAttributes property, 189
DrawOrder property, 681
driving directions
Geocode/Route services, 576-578
credentials, 578
results, 577
itineraries, displaying, 590-593
retrieving, 388-391
searching for routes, 584-585
user provided to and from addresses, 582-584
visual states, 586-590
Duration property, 111, 970
dynamic localizability, 600-601
dynamic mocking, 740
dynamic population (suggestion lists), 262-264

EasingFunction property, 111-112
East Asian language support, 164
eBay OData service
documentation website, 803
EbayClient class, 804
queries
collections, populating, 802
item objects, populating, 803
model class instance, creating, 802
results, retrieving, 803
eBay search app
automatic state preservation, 827-828
eBay logo, displaying, 813
EbayClient class, 804
network connection changes, monitoring, 807
OData wrappers, creating, 804-806
proxy results class, 805-806
records, retrieving, 807-808
results, displaying, 808-812
search queries, entering, 810
view page, 809-813
viewmodel, 806-809
EbayClient class, 804
EbaySearchStateViewModel, 827
EbaySearchView class, 809-813
EbaySearchViewModel class, 806-809, 816
commands, 806
HandleEbayClientSearchItemsComplete method, 808-809
Search method, 807-808
EbaySearchView.xaml, 810-812
EBNF (Extended Backus-Naur Form), 717
edge tracing Extras application
asynchronous processing, 623
creating, 619-628
emulator execution, detecting, 620
images
completed, 626
dummy, 627
1020  edge tracing Extras application

pixel arrays, creating, 624
processing, 622
reducing to line drawings, 623
retrieving from phone library, 621
saving, 629-631
launching from Application List, 628
MainPage XAML, 630
tokens, extracting, 620
white/black pixels, creating, 624-625

Edson, Dave, accelerometer article, 501
EffectEnabled property, 658
email
addresses
saving, 397-400
selecting, 394-396
composing, 396-397
sending, 786-789
EmailAddressChooserTask, 394-396
EmailComposeTask, 396-397
embedding
fonts, 165-167
UIElements, 181
dmulator
acceleration, simulating, 499-500
defined, 6
floating menu, 6
hardware keyboards, 169
location simulator, 538-539
photo share app, 635
Silverlight apps
properties, 7
running, 6-7
startup pages, editing, 7
titles, 8
virtual machine support, 7
EnabledGestures property, 688
enabling
frame rate counter, 31
geographic locations, 533
INotifyDataErrorInfo interface, 766
property setter validation, 749
push notifications, 457
resx files, 597
Rx, 546
Send button, 786
End property, 388
endonym, 604
EndsWith method, 726
EndTime property, 447
dergy
background transfer requests, 952
push notifications, 454, 461-463
timers, 706
EnhancedAccelerometer class, 501
AverageAcceleration property, 502
calibration methods, 503
LowPassFilterCoefficient property, 502
LowPassFilteredAcceleration property, 502
NoiseThreshold property, 503
OptimallyFilteredAcceleration property, 502
Reading property, 501
sample view, 503-506
shake detection, 507-508

DetectShake method, 507-508
ShakeCount property, 508
starting, 501
EnqueueCallback method, 737
dentities
creating, 851-853
disconnected, attaching/detaching, 855
multiplicity, 853
OCC, designating, 899
OData, extending, 816-817
single table inheritance, 895
den values
alarm recurrence, 911
ApplicationStateType, 827
AutoCompleteFilterMode, 257-259
converting, 174-175, 859
displaying, 176
GeoPositionAccuracy, 534
image sizing, 187
InputScopeNameValue, 171
LocalDatabaseMode, 858
PlayState, 975
radio regions, 702
retrieving, 173
TransferPreferences property, 952
TransferStatus property, 955
UserAction, 977
EnumUtility class, 173, 702
Equals filter, 258
EqualsCaseSensitive filter, 258
EqualsOrdinal filter, 258
EqualsOrdinalCaseSensitive filter, 258
Error property, 969
ErrorAdditionalData property, 462
ErrorOccurred event, 460
errors
  background audio playback, 969
  push notification channels, 460-461
validation
  asynchronous, 770-772
  creating, 779
  custom templates, 752-757
  DataValidationError class, 769
details, displaying, 762-765
list, retrieving, 785-786
synchronous, 769-770
ErrorsChanged event, 766
ErrorType property, 461
events
  AutoFocusCompleted, 649
  BufferReady, 692
  Calibrate, 515
  CaptureImageAvailable, 649
  CaptureThumbnailAvailable, 649
  ChannelUriUpdated, 460
click
  AppBar, 237
  buttons, 125
Completed
  AddressChooserTask, 426
  CameraCaptureTask, 429
  choosers, 387-388
  EmailAddressChooserTask, 394
  GameInviteTask, 431
  PhoneNumberChooserTask, 416
  PhotoChooserTask, 432
  SaveContactTask, 427
  SaveEmailAddressTask, 398
  SavePhoneNumberTask, 417
  SaveRingtoneTask, 438
CurrentValueChanged
  Accelerometer class, 499
  Motion class, 524
  SensorBase class, 496
DoubleTap
  GestureListener class, 370
  UIElement, 366
drag, 371-372
ErrorOccurred, 460
Flick, 373
FrameReported
  border color example, 358-359
  subscribing, 356
GameTimer class, 681
GestureBegin, 376
GestureCompleted, 376
GetErrors, 766
Hold
  GestureListener class, 370
  UIElement, 367
HttpNotificationChannel, 460
HttpNotificationReceived, 460, 477
Initialized, 647-649
keyboard, 41
life cycle
  deactivating, 61-62
  launching, 60-61
  subscribing, 61
tombstoning, 62
load, 328-329
LongListSelector, 306

How can we make this index more useful? Email us at indexes@samspublishing.com
manipulation, 359-360
  arguments, 360
  inertia, 361
  listing of, 359
  moving and scaling UIElement example, 362-363
  sequence, 360-362
MotionFinished, 210
mouse, 354
  border background color, changing, 354-355
  listing of, 354
  touch event promotion, 357-358
NetworkAddressChanged, 794-797
OrientationChanged, 102-104
password boxes, 178
pinch, 374-376, 379
Populating, 262-264
PositionChanged
  GeoCoordinateWatcher class, 535-536
  sampling, 546-550
power source, 41-42
PropertyChanged, 851
RadioButton class, 129
RangeBase class, 145
routed, 360
SearchCompleted
  Appointments class, 446
  Contacts class, 441
SelectionChanged, 266
Selector class, 141
Shake, 507
ShellToastNotificationReceived, 460, 465
ValueChanged, 275
ToggleButton class, 129
ToggleSwitch control, 312
touch. See gestures
TransferProgressChanged, 954
TransferStatusChanged, 954
transition navigation, 112
Update, 685
ValueChanged, 275
exceptions
  capabilities, 27
  critical, 751
Execute() method, 49
executing
  arbitrary JavaScript, 220
  commands, 788
  Silverlight apps, 9
  XNA apps, 13-16
execution model
  application state, 59
  consistent navigation, 58
  life cycle events
    deactivating, 61-62
    launching, 60-61
    subscribing, 61
    tombstoning, 62
  persistent state, 65
  responsiveness, 58
  running under lock screens, 65-66
  transient state
    requirements, 64
    restoring, 64-65
    saving, 63-64
execution order (UTF), 719
execution profiling, 34
EXIF data, reading, 617
expanding
  application bar, 229
  SIP 170
ExpansionVelocity property, 361
ExpectedException attribute, 723
expiration
alarms, 911
scheduled tasks, 927, 931
ExpirationTime property
AlarmViewModel, 910
ScheduledAction class, 909
Expression Blend, 753
error visual states, displaying, 752-757
styles, defining, 753
Expression class, 838
Expression property, 850
expressions
Lambda
properties, identifying, 47
unwinding, 837-838
regular, constructing, 816
tag, 717
assigning explicitly, 717
editor, hiding, 726-727
entering, 717
implicit, 717
setting programmatically, 718
symbols, 717
syntax, 717
Expresso, 816
Extended Backus-Naur Form (EBNF), 717
external navigation, 72-73
Extract method, 590
extracting tokens, 620
Extras applications, 617-618
adding to Extras menu, 619
certification requirements, 621
delayed-tracing, 619-628
asyncronous processing, 623
completed image, 626
dummy images, 627
emulator execution, detecting, 620
image pixel array, creating, 624
launching from Application List, 628
MainPage XAML, 630
processing images, 622

reducing images to line drawings, 623
retrieving images from phone library, 621
saving images, 629-631
tokens, extracting, 620
white/black pixels, creating, 624-625
Extras menu, 619
Extras.xml file, 619

Files

FadeOrientationChangesFrame class, 110
defades, 110
Fail method, 725
fast application switching, 57
fast-forwarding background audio, 969
FastForward method, 969
FCL (Framework Class Library), 117
control types, 118
Pivot/Panorama controls, 324
text elements, 158
top-level elements, 118
feed readers (IE), disabling, 798
FetchMoreDataCommand, 806, 816
file explorers, isolated storage, 873
build-in, 873-875
WP 7, 875-876
FileExists method, 822
files
alarm audio, 909
AppointmentsView.xaml, 449
audio, 970
background transfer requests
app terminations, handling, 956
BackgroundTransferRequest class, 952
backing up local databases, 963
copying files to temporary directory, 960
current state, identifying, 955
deleting, 952
existing, retrieving, 956
local database, backing up, 960-962
progress, monitoring, 954, 961-962
queue limits, 952
requirements, 952
restoring local databases, 963-965
re-subscribing, 956
results, displaying, 962
saving files to Backups directory, 956-957
status changes, monitoring, 954
submitting, 952
todo list viewmodel example, 960
upload, 953, 961
URL rerouting, 957-958, 961
Windows Live anonymous IDs, retrieving, 958-960
BingMapsView.xaml, 557
camera image filenames, 651
ChatClientView.xaml, 733
code-beside, 4, 10-11
ContactsView.xaml, 444-445
database, creating in isolated storage, 856
DataBoundPivotView.xaml, 341
EbaySearchView.xaml, 810-812
Extras.xml, 619
FMRadioView.xaml, 706-708
isolated storage, creating, 855-856
LocalizabilityResxView.xaml, 608
MainPage.xaml, 11-12
manifest, 26-27
ManipulationEventsView.xaml file, 362
names, 823
OrientationView.xaml, 108-109
PanoramaView.xaml, 348-349
PivotView.xaml, 336-338
ProductDetailsView.xaml, 93
ProductDetailsViewSampleData.xaml, 95
ProgressIndicator.xaml, 147
resx, 596
access modifiers, setting, 598
creating, 597-600
culture codes, 598
enabling, 597
generated designer class, 599
images, localizing, 602-603
names, 598
runtime language changes. See resx files, runtime language changes
ServiceReferences.ClientConfig, 575
SQL CE, 876-878
TodoListView.xaml, 933-935
WebBrowserWithScriptingView.xaml, 218-219
WrapPanelView.xaml, 315-317
XAP
Application Deployment tool, 25
contents, 24
overview, 24
size, 25
FileStream objects, 673-674
filters
AutoCompleteBox
alternate, 260
availability, 259
custom, 260-262
selecting, 260
text, 257-260
values, 257-259
Sobel, 624-625
FinalVelocities property, 362
Find method, 956
FindEdgesUsingSobel method, 623
FindEdgesUsingSobelCore method, 624-625
FixDelayMs property, 539
flash modes (camera), 655
flat lists, displaying, 288-290
FlatListViewModel class, 288-289
flick gestures, 373
border example, 380-382
Toolkit, 373
FlickGestureEventArgs parameter, 373
FlowDirection property, 612
FM radio
displaying, 706-708
FMRadio class, 698
frequencies, 698-705
onscreen menu, 705
power modes, 700-701
regions, 699, 702
signal strength, 699, 705
turning on/off, 698
view page, 708
FMRadio class, 698
FMRadioView page, 708
FMRadioViewModel
  Frequency property, 701
  PoweredOn property, 700-701
  Regions property, 702
  SignalStrength property, 705
  UpdateFrequencyRanges method, 704-705
FMRadioView.xaml, 706-708
focus (camera), 649, 655
Focus method, 649
Font Preview tool, 166
FontFamily property, 162
fonts
  colors, 163
  copyright, 18
  East Asian languages, 164
  embedding, 165-167
  licensing, 165
  official, 18
  properties, 162-163
  size, 163
  standard, 163-165
  streaming, 168
  styles, 163
  XNA, 17-20
FontSize property, 162
FontSource property, 163, 168
FontStretch property, 163
FontStyle property, 163
FontWeight property, 163
force, measuring. See accelerometer
forcing
  page orientation, 107
  quitting apps, 64
foreground control of background audio, 978-979
  application bar buttons, 981
  playback progress, 982-983
  slider control, 983
  track information, displaying, 981
  user opt-in requirement, 978
  viewmodel, 982
  visual state, 980
foreign key constraints, 891-893
formats
  audio, 670
  DatePicker/TimePicker values, 277-278
  images, 186
  text, 181
  video images, 670
forms, validating, 760, 775-779
  completing, 777-779
  evaluating, 777
  property list, creating, 775-776
forward in navigation, 112
Frame Rate graph (Performance Analysis tool), 35
frame rates property, 16
FrameNavigationService class, 883-884
FrameReported event
  border color example, 358-359
  subscribing, 356
frames
  rate counter, 31-33
  touch events, 356
Framework Class Library. See FCL
FrameworkElement class, 118
  DataContext property, 782
  FlowDirection property, 612
  HorizontalAlignment property, 123
  VerticalAlignment property, 123
frameworks
  Automation, 737-738
  IoC, 739
free space available, measuring, 822
FreeDrag gesture, 689-690
frequency (FM radio), 698-705
Frequency property
  FMRadio class, 698
  FMRadioViewModel class, 701
front facing camera capability, 27, 646
full-screen mode, 239-241
full-screen picker page, customizing, 278-282
  constructor, 280
  holding dates, 279
  list of dates, 279
  retrieving values, 281-282
FullModeHeader property, 255

G

  g's (gravitational units), 497
  Gallardo, John, 885
Game class, 16
Game Startup Types, 13
Game template, 13
Game Thumbnail, 13
GameInviteTask, 430-432
GamePageViewModel, 683
games
  executing, 13-16
  inviting players, 430-432
  loops, 16, 681-682
  XNA, 12
GameTimer class, 681-682
  events, 681
  properties, 682
  Update event, 685
garbage collection, 37
Geocode service
  address search, 577
  overview, 574
  route calculation example, 576-578
GeoCoordinateWatcher class, 534
  DesiredAccuracy property, 534
  GeoPositionAccuracy enum, 534
  instantiating, 534
  MovementThreshold property, 534
  position change events, 535-537
    PositionChanged, 535-536
    StatusChanged, 536-537
  Start method, 537
  TryStart method, 537
  Wait method, 541
  WalkPath method, 539-540
geographic locations
  accuracy, 534
  architecture, 532-533
  capability, 26
  enabling, 533
  GeoCoordinateWatcher class, 534
  locations
    provider monitoring, starting, 537
    sensing technologies, 530-531
  position changes, monitoring, 535-537
    detecting changes, 535-536
    status, 536-537
  position viewer sample
    displaying locations, 543-545
    emulator, running, 542
    GeoPositionViewModel class, 541
    location notifications, receiving, 542
    starting/stopping monitoring, 543
  pushpinning, 567
    appearance, 568
    creating, 572
    displaying, 573
    icon images, 571-572
    itinerary items, 592
    populating, 567-569
    presenting, 570
    template, 570
    user prompt for naming, 573
routes, calculating
  driving directions, 388-391
  Geocode/Route services, 576-578
  itineraries, displaying, 590-593
  searching for routes, 584-585
user provided to and from addresses, 582-584
visual states, 586-590
sampling PositionChangedEventArgs
GeoPositionSampler class, 548-550
Rx, 546-547
simulator, 538
street address resolution, 545
testing
code-driven, 539-541
location simulator, 538-539
tracking, 564-566
GeoPositionAccuracy enum, 534
GeoPositionSampler class, 548-550
GeoPositionView page, 543-545
GeoPositionViewModel class, 541
DelegateCommands, 543
Start method, 542
GestureBegin event, 376
GestureCompleted events, 376
GestureEventArgs class, 125, 365
GestureListener class, 369
double taps, 370
drags, 371-372
flicks, 373
holds, 370
pinches, 374-376
taps, 369-370
gestures
before/after events, 376
border example, 377
double tapping, 378
dragging, 378-379
flicking, 380-382
holding, 378
pinching, 379
tapping, 378
defined, 364
design
components, 382
guidelines, 383
sizing/spacing constraints, 383
double tap
Deep Zoom images, 213
Toolkit, 370
UIElement, 366
drag
Deep Zoom images, 212
 Toolkit, 371-372
flick, 373
FreeDrag, 689-690
hold, 368
border example, 378
GestureListener class, 370
 Toolkit, 370
UIElement, 366-367
hybrid apps, 686-687
pinch, 374-376
border example, 379
Deep Zoom images, 212
sequence, 375-376
 Toolkit, 374-376
Pivot control support, 325
tap
buttons, 125
GestureListener class, 369-370
 Toolkit, 369-370
UIElement, 365-366
touch points, 354
user interfaces, 382
XNA, processing, 688-690
GestureService class, 369
GestureView page sample code, 377
double tapping, 378
dragging, 378-379
flicking, 380-382
holding, 378
pinching, 379
tapping, 378
get accessors, 838-839
GetCachedTimeline method, 860, 863-864
GetCredentials method, 578
GetDirectoryNames method, 822

How can we make this index more useful? Email us at indexes@samspublishing.com
GetErrors method, 766
GetFileNames method, 822
GetFuncType method, 838
GetIsNetworkAvailable method, 793
GetItemsInView method, 306
GetPosition method, 365
GetPreviewBufferArgb32 method, 659
GetPreviewBufferY method, 660
GetPreviewBufferYCbCr method, 660
GetPropertyErrors method, 769-770
GetPropertyInfo method, 837
GetStockQuote method, 487-488
GetText() method, 182
GetTimeline method, 860, 869-870
GetTimelineCore method, 862-863
GetTodoItems methods, 930
GetUserStoreForApplication method, 821
GetVisualAncestors method, 331
globalization, 595
GoBack method, 78, 883
GoldWave, 203
graphics. See images
gravitational units (g’s), 497
Gravity property, 524
grayscale video effect
  ARGB color components, extracting, 661
  converting back to color, 661
  grayscale conversion, 661
  previewing, 659-660
  processing, 659
  result, 661
  turning on/off, 658
greater than symbols (>), 159
GroupBy property, 296
GroupByPrice method, 294
GroupName property, 130
groups
  input validation, 760-762
  lists, displaying, 290
  marketplace apps, 292-298
    group picker, 298
    item templates, 296
  radio buttons, 130
todo items, 931
  validation, 786-789
gyroscope
  angular rotations, 519-520
  cumulative values, 521
  disposing, 520
  drift, 521
  instance, creating, 519
  overview, 518
  support, 518
  time between updates, 518
  updates, receiving, 519
  view page, 520-521
Gyroscope class
  IsSupported property, 518
  Stop method, 520
  TimeBetweenUpdates property, 518
GyroscopeView page, 520-521
GyroscopeViewModel class, 519

H
HandleAccelerometerCurrentValueChanged method, 515
HandleBufferReady method, 694
HandleCalculationResult method, 583
HandleCaptureImageCompleted method, 672
HandleCaptureThumbnailAvailable method, 653
Handled property, 360, 365
HandleDownloadTransferStatusChanged method, 964
HandleEbayClientSearchItemsComplete method, 808-809
HandleEdgesFound method, 626
HandleErrorsFound method, 785-786
HandleGameTimerDraw method, 687-688
HandleGameTimerUpdate method, 686-687
HandleGetTimelineResult method, 870
HandleGyroscopeCurrentValueChanged method, 519
HandlePhotoCameraInitialized method,
HandlePhotoChooserTaskCompleted method, 628
HandleTapAtPageLevel method, 370
HandleUpdatePropertyChanged method, 757
HandleValidationComplete method, 773

handling events

App class, 9
AutoFocusCompleted, 649
BufferReady, 692
   MotionFinished, 210
   OrientationChanged, 102-104
   pivot load, 328-329
   Populating, 262-264
   power source, 41-42
   RadioButton class, 129
   RangeBase class, 145
   SelectionChanged, 266
   Selector class, 141
   Tap, 125
   ToggleButton class, 129
   ToggleSwitch control, 312
Calibrate, 515
CameraCaptureTask, 429
CaptureImageAvailable, 649
CaptureThumbnailAvailable, 649
ChannelUriUpdated, 460
Click, 125
Completed
   AddressChooserTask, 426
   EmailAddressChooserTask, 394
   GameInviteTask, 431
   PhoneNumberChooserTask, 416
   PhotoChooserTask, 432
   SaveContactTask, 427
   SaveEmailAddressTask, 398
   SavePhoneNumberTask, 417
   SaveRingtoneTask, 438
CurrentValueChanged
   Accelerometer class, 499
   Motion class, 524
   SensorBase class, 496
DoubleTap
   GestureListener class, 370
   UIElement, 366
drag, 371-372
ErrorOccurred, 460
Flick, 373
FrameReported
   border color example, 358-359
   subscribing, 356
GameTimer class, 685
GestureBegin, 376
GestureCompleted, 376
GetErrors, 766
Hold
   GestureListener class, 370
   UIElement, 367
HttpNotificationChannel, 460
HttpNotificationReceived, 460, 477
Initialized, 647-649
keyboard, 41
LongListSelector control, 306
manipulation, 359-360
   arguments, 360
   inertia, 361
   listing of, 359
   moving and scaling UIElement example,
      362-363
   sequence, 360-362
mouse, 354
   border background color, changing,
      354-355
   listing of, 354
   touch event promotion, 357-358
NetworkAddressChanged, 794-797
page navigation, 78
pinch, 374-376
PositionChanged
   GeoCoordinateWatcher class, 535-536
   sampling, 546-550
PropertyChanged, 851
DataErrorNotifier class, 772-775
memory leaks, 562

How can we make this index more useful? Email us at indexes@samspublishing.com
PropertyChanging, 851
routed, 360
SearchCompleted
  Appointments class, 446
  Contacts class, 441
Shake, 507
ShellToastNotificationReceived, 460, 465
StatusChanged, 536-537
Tap
  GestureListener class, 369-370
  UIElement, 366
 touch, 357-358
TransferProgressChanged, 954
TransferStatusChanged, 954
hardware
  Back buttons, 81-82
  keyboards, 169
HasErrors property, 766
headers
  DatePicker/TimePicker controls, 277
  pivot, 327
  ToggleSwitch control, 312
HeadingAccuracy property, 510
helium voice app, 692
  play command, disabling, 694
  playback, 695-697
  recording audio, 693-694
  saving recordings, 694
  stopping recording, 694
  view page, 697-698
hiding
  application bar, 229, 239-241
  Bing logo/copyright text, 560
  itineraries, 593
  scheduled tasks, 940-942
  tag expressions editor, 726-727
hierarchies
  control classes, 118
  inheritance, mapping, 895-899
    base class, 895-897
    database file, initializing, 898
  storage/retrieval of entities, 897-898
    subclass, 897
high-resolution images
  animations, enabling/disabling, 209
  collection, accessing, 209
  creating, 206-207
  double tapping, 213
  downloading, stopping, 210
  dragging, 212
  instantiating, 208
  logical coordinates, 209
  motion finished event, 210
  overview, 204-205
  pinching, 212
  sample code, 210-213
  sources, 208
  tiling, 206
  visibility, 209
  zooming/panning, 209
history stacks, 74-75
hold gesture, 368
  border example, 378
  GestureListener class, 370
  Toolkit, 370
  UIElement, 366-367
HorizontalAlignment property, 123
HorizontalAlignment property, 372
HorizontalVelocity property, 373
hosting web content within apps, 73
HTTP services, 792
HttpNotificationChannel events, 460
HttpNotificationReceived event, 460, 477
HttpWebResponse, 478
hybrid apps
  components, 678
  game loops, 681-682
  pop-up controls, 688
  portability, 677
  project templates, 678-679
  rendering modes, 679
Silverlight content, rendering in XNA, 684-688
drawing to screen, 687-688
gesture input, 686-687
page navigation, 686
results, 688
XNA
3D XNA model, 683-684
content manager, initializing, 682-683
gestures, processing, 688-690
immediate mode XNA rendering, allowing, 679-681
HyperlinkButton control, 73, 126
hyperlinks
AppBar, 246
buttons, 126
external page navigation, 73
rich text boxes, 180
sharing, 422

IAudioPlaybackManager interface, 969
IChatService, 332, 728
ICommand interface, 49
Icon property, 569
icons
apps, customizing, 467
buttons
AppBar, 233-236
application bar, 245
images, 231-232
retrieving at runtime, 232-233
text, 230-231
DatePicker/TimePicker controls, 275
pushpin images, 571-572
Id methods, 929
IDateTimePickerPage interface, 279
IDependencyRegistrar interface, 739
idle detection mode, disabling, 68
IEbayClient interface, 804
Ignore attribute, 721
IIsolatedStorageUtility interface, 972
ILicenseService interface, 741
ILockScreenManager interface, 66
ILoopingSelectorDataSource interface, 283
ImageCacheConverter class, 96-97
ImageInfo class, 662
Imagery service, 574
images. See also animations
bits per pixel (bpp), 616
Build Action set, 186
caching, 96-97
converting to byte arrays, 638-639
datePicker/TimePicker controls, 275
drawing surfaces, 188
sketch page sample app, 192-195
strokes, 188-189
user input, 190-191
edge tracing application
asynchronous processing, 623
completed image, 626
creating, 619-628
dummy images, 627
evaluator execution, detecting, 620
image pixel arrays, creating, 624
launching from Application List, 628
MainPage XAML, 630
processing images, 622
reducing images to line drawings, 623
retrieving images from phone library, 621
saving images, 629-631
tokens, extracting, 620
white/black pixels, creating, 624-625
formats, 186
Game Thumbnail, 13
high-resolution
animations, enabling/disabling, 209
collection, accessing, 209
creating, 206-207
double tapping, 213
downloading, stopping, 210

How can we make this index more useful? Email us at indexes@samspublishing.com
dragging, 212
instantiating, 208
logical coordinates, 209
motion finished event, 210
overview, 204-205
pinching, 212
sample code, 210-213
sources, 208
tiling, 206
visibility, 209
zooming/panning, 209
icon buttons, 231-232
identifying, 186-187
limiting, 187
loading, 37, 187
localized, displaying, 607-610
localizing, 602-603
memory, 187
Panorama control, 324, 338, 344
Pivot control, 338
product lists, displaying, 89
product thumbnails, displaying, 89
pushpin icons, 571-572
saving, 629-631
scale transforms, 30
sending, 635-637
shell tiles
creating, 936-937
displaying, 942
updating, 942-944
Silverlight
threads, 29-30
visibility, 31
sizing, 187-188
Source property, 186
still
capturing, 651
displaying, 657
stock ticker app, 489
streaming, 434
stretching, 187
tile background, 469-472
tile notifications, 474
trailing slashes, 186
URIs, 186
Images property, 664
ImageUploadService class, 637
ImageUrl property, 868
IMessageService, 270
IMessageService interface, 52-54
immediate mode XNA rendering, allowing, 679-681
improving sensor accuracy, 522
INavigationService interface
FrameNavigationService implementation, 883-884
IoC container, 884
methods, 883
Inconclusive method, 725
Indeterminate event, 129
indexes (databases), 851, 890-891
inheritance
data models, 850
hierarchies, mapping, 895-899
base class, 895-897
database file, initializing, 898
storage/retrieval of entities, 897-898
subclass, 897
single table, 895
Initialized event, 649
InitializationMs property, 539
InitializeContainer method, 884
Initialized event, 647
InitializeDatabase method, 855-856
InitializedPhoneApplication method, 110
initializing
PhotoCamera class, 654
SharedGraphicsDeviceManager class, 680
XNA content manager, 682-683
InkPresenter element, 188
sketch page sample app, 192-195
strokes, 188-189
user input, 190-191
forms, 775-779
property changes, 772-775
semantic, 748
style locations, defining, 753
synchronous, 769-770
syntactic, 748
text boxes as user types, 757-760
XNA gestures, processing, 688-690
InputScope property, 171-172
InputScopeNameValue enum values, 171
InputScopeValueConverter class, 174-175
installing SDK, 2
instantiating
AppBar, 236-237
Appointments class, 446
commands, 49
Contacts class, 441
GeoCoordinateWatcher class, 534
high-resolution images, 208
MarketplaceApp class, 291
OData collections, 802-803
integration testing, 711
interfaces
IAudioPlaybackManager, 969
IChatService, 728
ICommand, 49
IDateTimePickerPage, 279
IDependencyRegistrar, 739
IEbayClient, 804
IIsolatedStorageUtility, 972
ILicenseService, 741
ILockScreenManager, 66
ILoopingSelectorDataSource, 283
IMessageService, 52-54
INavigationService, 883-884
INotifyDataErrorInfo, 766
enabling, 766
members, 766
ValidationSummary control, adding, 785-786
INotifyPropertyChanged
alternative implementation, 46-48
PropertyChanged event, 851
traditional implementation, 44-46
INotifyPropertyChanging, 851
IObservable, 546-547
IObservable, 546-547
IRouteCalculator
CalculateAsync method, 576
route searches, 582-584
IStatePreservation, 829
ITwitterService
GetTimeline method, 869-870
methods, 860
IValidateData, 767
Rx, 546-547
intermediate surface counter frame rate counter field, 33
internal URIs, 71-72
InternalsVisibleTo method, 727, 736
interpage communication, 74
intersection (*) symbol, 718
Interval property, 126, 473
Inversion of Control. See IoC
inviting game players, 430-432
InvokeScript method, 216
IObservable interface, 546-547
IObservable interface, 546-547
IoC (Inversion of Control), 739
containers
creating, 739-740
scheduled tasks, 940
DI, 739-740
frameworks, 739
mocking, 740
service location, 739
IRouteCalculator interface
CalculateAsync method, 576
route searches, 582-584
IsAllDayEvent property, 447
IsBouncy property, 305
IsChecked property, 128
IsComplete method, 775
IsConnectionStringValue method, 859
IsDataValid property, 496
IsDbGenerated property, 850
IsDescriminator property, 850
IsFalse method, 725
IsIndeterminate property
PerformanceProgressBar, 307
ProgressIndicator class, 147
IsInertial property, 361-362
IsInstanceOfType method, 725
IsKeyboardDeployed property, 39
IsKeyboardPresent property, 39
IsMenuEnabled property, 238
IsNotInstanceOfType method, 725
IsNotNull method, 725
IsNotSubsetOf method, 726
IsNull method, 725
Isolated storage
API, 820
application settings, 826
apps, minimizing, 820
databases
deploying, 880-882
files, creating, 856
file explorers, 873
built-in Isolated Storage Explorer, 873-875
WP 7 isolated Storage Explorer, 875-876
free space
available, measuring, 822
remaining notification, 820
managed dictionary, 820
offline browsing, 221
reading content, 225-226
storing content, 223-225
overview, 819
persistent state, 824-826
reading/writing data, 822-824
serialization performance, 824
virtual file system, 821-822
Isolated Storage Explorer
built-in, 873-875
WP 7, 875-876
IsolatedStorageFile class, 821-822
IsolatedStorageSettings class, 824-826
IsolatedStorageUtility class, 880-881
IsolatedStorageUtilityTests class, 882
isostore prefix, 856
IsPrimaryKey property, 849
IsPrivate property, 447
IsReadOnly property, 179
IsScrolling property, 305
IsSubsetOf method, 726
IsSupported property
Compass class, 509
Gyroscope class, 518
IsSynchronizedWithCurrentItem property, 141
IStatePreservation interface, 829
IsTextCompletionEnabled property, 267
IsThreeState property, 128
IsTiltEnabled property, 309
IsTrial() method, 741
IsTrue method, 725
IsVersion property, 850, 899
IsVisible property, 147
Item class (custom), 816
ItemCountThreshold property, 252
ItemFilter property, 260-262
ItemHeight property, 315
items controls, 140
combo boxes, 144
list boxes, 142-144
properties, 140
selecting, 141
items layer (Panorama control), 346
Items property, 140
ItemsControl class, 140
ItemsSource property, 132
ItemsControl class, 140
ListPicker control, 254
ItemWidth property, 315
itineraries (Bing Maps)
  displaying, 590-593
  visual states, 586-590
itineraryToggleCommand, 586
ITwitterService interface
  GetTimeline method, 869-870
  methods, 860
IValidateData interface, 767

J–K–L

JavaScript, 220
JSON serialization, 476
Jung, Alexander's articles
  input validation definition, 748
  source property assignment, 749
KeyboardDeployedChanged event, 41
keyboards
  events, 41
  hardware, 169
  SIP layouts, 169
Lambda expressions
  properties, identifying, 47
  unwinding, 837-838
languages
  endonym, 604
  right to left support, 612
  runtime changes, 603
    cultures, 604-606
    data binding, 611
    date/time/currency formatting
      properties, 607
    localized text/images, displaying,
      607-610
    Visual Studio resource properties,
      selecting, 610
  support, 597
LargeChange property, 145
Launch WPConnect.bat, 642

launchers
  API, 385-386
  BingMapsDirectionsTask, 388-391
  BingMapsTask, 392
  ConnectionSettingsTask, 393
  EmailComposeTask, 396-397
  GameInviteTask, 430-432
  listing of, 386
  MarketplaceDetailTask, 400-403
  MarketplaceHubTask, 404
  MarketplaceReviewTask, 405
  MarketplaceSearchTask, 405-406
  MediaPlayerLauncher, 407-411
  content locations, 407
  example, 408
  playback controls, 407
  sample code, 409-411
  overview, 385
  PhoneCallTask, 413-414
  SavePhoneNumberTask
    Completed event, 417
    sample code, 418
  SearchTask, 420-421
  ShareLinkTask, 422
  ShareStatusTask, 423
  SmsComposeTask, 423-424
  testing, 743-744
  WebBrowserTask, 424-425
LaunchForTest method, 932
launching
  apps, 60-61
  compass, 509
  Extras applications from Application
    List, 628
  Performance Analysis tool, 34
  Photo Picker app, 432-433
  SIP 170
  video recording, 673-674
Launching event, 60-61
LaunchWebBrowserView page, 424
LaunchWebBrowserViewModel, 425
layout
  Panorama control
    background images, 338
  Bookshop Service sample app, 346-350
    creating, 343
  FCL placement, 324
    images, 324
    items, 346
    layers, 344-346
    memory, 322
    overview, 323-324, 343
    performance, 322
  Pivot control, compared, 322
    styles, 322
    template, 345
    things to avoid, 351
    title, 345
    touch support, 322
  Pivot control
    adding, 325-326
    background images, 338
    creating app with New Project dialog, 327
    FCL placement, 324
    gestures/navigational effects support, 325
    headers, 327
    limiting, 323
    load events, 328-329
    lockable, 352
    memory, 322
    messaging app, 331-339
    multiple application bars, hosting, 329-331
    overview, 323
    Panorama control, compared, 322
    performance, 322
    PivotItems, 325-328
    populating with data-bound collection, 340-343
    styles, 322
    things to avoid, 351
    touch support, 322
    SIP keyboard, 169
    touch
      components, 382
      guidelines, 383
      sizing/spacing constraints, 383
    less than symbols (<), 159
    LicenseInformation class, 741-743
  licensing
    fonts, 165
    mock service, 741
  life cycle (apps)
    deactivating, 61-62
    launching, 60-61
    subscribing, 61
    tombstoning, 62
  line breaks, 159-161
    LinearVelocity property, 361
    LineHeight property, 162
    LineStackingStrategy property, 162
  Link event, 306
  LINQ to SQL, 841
    concurrency, 899
      conflict detection, 900-901
      entities, designating, 899
    inheritance hierarchies, mapping, 899
      base class, 895-897
      database file, initializing, 898
      storage/retrieval of entities, 897-898
      subclass, 897
    new features, 845
    phones, 844
    platform differences, 845
    query logs, 885-887
    single table inheritance, 895
  list boxes, 132, 142-144
    context menus, hosting, 271-273
    selecting/deselecting items, 144
    WrapPanel, adding, 317-318

How can we make this index more useful? Email us at indexes@samspublishing.com
list selector control

Boolean conversion into brushes, 304
code listing, 301-303
contacts list example, 298-304
events, 306
flat lists, 288-290
grouped lists, 290
Marketplace app list, 290-298
categories array, 291
group picker, 298
grouping items, 292-296
instantiating/populating, 291
item templates, 296
MarketplaceApp class, 291
prices array, 291
view model, 292
methods, 306
properties
BufferSize, 305
DisplayAllGroups, 305
IsBouncy, 305
IsScrolling, 305
MaximumFlickVelocity, 305
SelectedItem, 305
ShowListFooter, 306
ShowListHeader, 306
view model, 299-303
visual structure, 287
ListBox class, 142-144
ItemsSource property, 132
selection properties, 144
ListBoxItem class, 142, 309
listings
AccelerometerView class, 504-505
AccelerometerViewModel class, 503-504
App class, 9
AppBarHyperlinkButton class, 246
ApplicationBarModeToBooleanConverter class, 238-239
ApplicationBarView.xaml AppBar, 236-237
AppointmentsViewModel class, 448-449
AppointmentsView.xaml, 449
BackupService class, 956-957
BankAccount class, 895-897
BankingDataContext, 897-898
BindableChangeNotifier class, 601
BindableResources class, 599
BingMapsDirectionsTaskViewModel, 389-390
BingMapsView page visual state
groups, 587
BitmapImageConverter class, 603
BookshopService class, 97
BooleanToBrushConverter class, 304
BooleanToVisibility class, 84
BuildAux method, 244
ChatClientViewModel class, 729-732
CheckingAccount class, 897
City class, 253-254
ColorToBrushConverter class, 241
CommonResourceExample, 947
compass calibration UI XAML, 516
CompassViewModel.Start method, 509-514
contacts LongListSelector, 301-303
ContactsViewModel class, 442-444
ContactsView.xaml, 444-445
ControlExamplesViewModel class, 149-152
CopyDatabaseToIsolatedStorage method, 882
CopyStreamBytes method, 225-226
cultureName class, 604
custom item class, 816
CustomDatePickerView class, 281-282
CustomGrouping class, 292-294
DataBoundPivotViewModel class, 340
DataBoundPivotView.xaml, 341
DataContextChangedListener class, 782-784
DataItem class, 131-132
DebugStreamWriter class, 885-887
DetectShake method, 507-508
DeviceProperties class, 958
EbayClient class, 804
EbaySearchView class, 810
local databases
  backing up, 960-963
  case sensitivity, 858
  code-first data model creation, 845-846, 850
    benefits, 845
    cached date, retrieving, 863-864
    cached timelines, retrieving, 869-870
    change notifications, 851
    column attributes, 848
    connection strings, 856-859
    creating entities, 851-853
    CRUD operations, 855
    database file, creating in isolated storage, 856
    disconnected entities, attaching/detaching, 855
    display controls, 871-872
    encryption, 858
    enum value conversions, 859
    indexes, 851
    initialization, 856
    isolated storage files, creating, 855
    one-to-many entity relationships, 853
    population, 861
    public fields, 855
    results, handling, 870
    retrieving users from database, 860
    table names, 848
    user credentials, 864-867
    user details/status updates, 860
    web API queries, 862-863
  columns, adding, 888
  concurrency, 899
    conflict detection, 900-901
    entities, designating, 899
    culture codes, 857
  database-first data models, 878-880
  deployment, 842-843, 880-882
  foreign key constraints, adding, 891-893
  indexes, adding, 890-891
  LINQ to SQL
    concurrency, 899-901
    inheritance hierarchies, mapping, 895-899
    new features, 845
    phones, 844
    platform differences, 845
    query logs, 885-887
    single table inheritance, 895
    locking modes, 857
    max buffer size, 857
    passwords, 857
    restoring, 963-965
    schema
      displaying, 873
      isolated storage file explorers, 873-876
      SQL CE files, 876-878
      versioning, 894
    schema updates, 887
      columns, adding, 888
      indexes, adding, 890-891
      one-to-many associations, adding, 891-893
      tables, adding, 889-890
      size, 857
      tables, adding, 889-890
  LocalDatabaseMode enum, 858
  localization, 596
    images, 602-603
    multiple language support, 597
    resx files, 596
      access modifiers, setting, 598
      creating, 597-600
      culture codes, 598
      enabling, 597
      generated designer class, 599
      names, 598

How can we make this index more useful? Email us at indexes@samspublishing.com
runtime language changes, 603
cultures, 604-606
data binding, 611
date/time/currency formatting properties, 607
right to left support, 612
text/images, displaying, 607-610
Visual Studio resource properties, selecting, 610
ToggleButton control, 313-314
UI, updating when culture changes, 600-601
LocalizabilityResxView.xaml, 608
LocalizabilityView page, 607-610
LocalizabilityViewModel
  CultureName property, 606
  SetCulture method, 605
  SetCurrentCultureValues method, 607
  SupportedCultures property, 605
LocalizedToggleSwitchConverter class, 313
Location property
  Appointments class, 447
  MediaPlayerLauncher, 407
  Pushpin class, 567
  PushpinViewModel, 569
locations
  accuracy, 534
  architecture, 532-533
  capability, 26
  enabling, 533
  GeoCoordinateWatcher class, 534
  position changes, monitoring, 535-537
detecting changes, 535-536
  status, 536-537
  position viewer sample
displaying locations, 543-545
  emulator, running, 542
  GeoPositionViewModel class, 541
  location notifications, receiving, 542
  starting/stopping monitoring, 543
  provider monitoring, starting, 537
pushpinning, 567
  appearance, 568
  creating, 572
  displaying, 573
  icon images, 571-572
  itinerary items, 592
  populating, 567-569
  presenting, 570
template, 570
  user prompt for naming, 573
routes, calculating
  driving directions, 388-391
  Geocode/Route services, 576-578
  itineraries, displaying, 590-593
  searching for routes, 584-585
  user provided to and from addresses, 582-584
  visual states, 586-590
sampling PositionChanged events
  GeoPositionSampler class, 548-550
  Rx, 546-547
sensing technologies, 530-531
simulator, 538-539
street address resolution, 545
testing
code-driven, 539-541
location simulator, 538-539
tracking, 564-566
lock screens
  LockScreenManager class, 66
code listing, 67
  idle detection mode, 68
ILockScreenManager interface, 66
storing, 67
LockScreenViewModel class, 68-69
running apps under, 65-66
LockablePivot control, 352
locking modes, 857
LockScreenManager class, 66
code listing, 67
  idle detection mode, 68
ILockScreenManager interface, 66
storing, 67
LockScreenViewModel class, 68-69
logging, 885-887
logical coordinates, 209
LogoVisibility property, 560
LongListSelector control, 287
  Boolean conversion into brushes, 304
code listing, 301-303
contacts list example, 298-304
events, 306
flat lists, 288-290
grouped lists, 290
Marketplace app list, 290-298
categories array, 291
group picker, 298
grouping items, 292-296
instantiating/populating, 291
item templates, 296
MarketplaceApp class, 291
prices array, 291
viewmodel, 292
methods, 306
properties
  BufferSize, 305
  DisplayAllGroups, 305
  IsBouncy, 305
  IsScrolling, 305
  MaximumFlickVelocity, 305
  SelectedItem, 305
  ShowListFooter, 306
  ShowListHeader, 306
viewmodel, 299-303
visual structure, 287
LoopingSelector control, 283-286
data binding, 286
data source, 283
FM radio frequency, 703
numeric values, presenting, 283-286
loops (game), 16, 681-682
LowPassFilterCoefficient property, 502
LowPassFilteredAcceleration property, 502
LowPassFilterStrategy, 513
M
MagneticHeading property, 510
MagnetometerReading property, 510
MainPage code-beside, 10-11
MainPageViewModel, 979-982
  PlayerState property, 980
  Position property, 982-983
  Refresh method, 979
  TrackArtist/TrackTitle properties, 979
  ViewState property, 981
MainPage.xaml file, 11-12
manifest file capabilities, 26-27
manipulation events, 359-360
  arguments, 360
  inertia, 361
  listing of, 359
  moving and scaling UIElement example, 362-363
  sequence, 360-362
ManipulationComplete event, 361
ManipulationCompletedEventArgs parameter, 362
ManipulationContainer argument, 360
ManipulationDelta event, 361
ManipulationDeltaEventArgs parameter, 361
ManipulationEventsView class, 362-363
ManipulationEventsView.xaml file, 362
ManipulationOrigin argument, 360
ManipulationStarted event, 360
Map class
  Center property, 561
  Mode property, 559
  SetView method, 561
  ZoomBarVisibility property, 563-564
  ZoomLevel property, 562
MapItineraryItem class, 590
MapMode class, 559

How can we make this index more useful? Email us at indexes@samspublishing.com
Bing Maps control
  accounts, creating, 554
  adding, 556-557
  API key registration, 554-555
  application bar items, 558
  Bing logo, hiding, 560
  BingMapsView.xaml, 557
  center location, setting, 561
  copyright text, hiding, 560
  location tracking, 564-566
  map modes, 558-560
  panning, 563
  pushpins, 567-573
  SOAP services. See SOAP
    (Simple Object Access Protocol)
  viewable area, setting, 561
  zooming, 562-564

Driving directions, 390
  itineraries, displaying, 590-593
  retrieving, 388-391
  searching for routes, 584-585
  user provided to and from addresses, 582-584
  visual states, 586-590

Geocode/Route services, 576-578
  credentials, 578
  results, 577
  inheritance hierarchies, 895-899
    base class, 895-897
    database file, initializing, 898
    storage/retrieval of entities, 897-898
    subclass, 897

URIs, 75-77

MarketDetailTask class, 743

Marketplace
  app list example, 290-298
    categories array, 291
    group picker, 298
    grouping items, 292-296
    instantiating/populating, 291
    item templates, 296
  MarketplaceApp class, 291
    prices array, 291
    viewmodel, 292
  apps
    details page, 400-403
    reviewing, 405
    certification requirements, 82
    launching, 404
    push notifications, disabling
      certification, 457
    searching, 405-406
    submissions, 27
  Test Kit, 28
  MarketplaceApp class, 291-292
    categories array, 291
    grouping items, 292
    instantiating/populating, 291
    prices array, 291
    viewmodel, 292
  MarketplaceDetailTask, 400-403
  MarketplaceHubTask, 404
  MarketplaceReviewTask, 405
  MarketplaceSearchTask, 405-406
  MarketplaceView page, 402
  MarketplaceViewModel
    DetailCommand, 402
    HubCommand, 404
    reviewCommand, 405
    searchCommand, 406
  Matches method, 726
  max buffer size parameter, 857
  max database size, 857
  Maximum property, 144
  MaximumFlickVelocity property, 305
  MaxUpdateCount property, 474

Measuring
  direction
    calibrating, 515-517
    interference, 509
    magnetic north arrow, 513-514
    noise, removing, 513
PropertyChanged event leaks, 562
requirements, 39
scheduled task agents, 925
Memory Usage MB graph (Performance
Analysis tool), 37
MenuItem class, 310
menus
AppBar
controls, 233
disabling, 238
visibility, 234
application bar, 228
limiting, 229
visibility, monitoring, 245
context
adding, 267
button example, 267
list boxes, 271-273
menu items, binding, 270
overview, 267
page background scaling, 268-269
tilt effect, adding, 310
viewmodel, 269-271
emulator, 6
Extras, 619
FM radio, 705
retrieving items, 232-233
Share, 631-633
text, 230-231
message boxes, displaying, 52-54
class diagram, 52
IMessageService interface, 52
MessageService class, 52-54
Message property, 868
MessageBox BadContent errors, 461
MessageService class, 52-54
MessagesViewModel class, 334-336
MessageViewModelTemplate, 342
Messaging app
messages, composing, 423-424
pivot items, 331-339
IChatService, implementing, 332
multiple AppBar messages, 336-338
PivotViewModel class, 331
sending messages, 333-334, 339
viewing received messages, 334-338
methods
Abort, 925
accelerometer calibration, 503
Add, 909
AddAssociation, 891-893
AddColumn, 888
AddIndex, 890-891
AddOrUpdateItem, 929
AddTable, 889-890
AddValidationProperty, 768
arguments, validating, 50-51
AskYesNoQuestion, 565
Assert class, 725
AssertLessThan, 51
AssertNotNull, 50
AssertNotNullOrWhiteSpace, 51
AssertNotNullAndOfType, 51
BackgroundAudioPlayer, 969
BackupDatabase, 960-963
BeginGetPropertyErrorsFromValidator, 772
BeginValidation, 769-772
BindToShellTile, 459
BindToShellToast, 459
Build, 243-245
BuildAux, 244
ButtonShouldBeEnabled, 736
CalculateAsync
IRouteCalculator, 576
RouteCalculator class, 578
Calibrate(), 503
CanCalibrate(), 503
CaptureImageAsync, 671
Clipboard class, 182
CollectionAssert class, 725-726
ConvertArgbToColor, 661
Navigate, 215, 883
NavigationService class
GoBack, 78
Navigate, 77
NotifyComplete, 925
OnBeginStreaming, 984-985
OnError, 977-978
OnErrorsChanged, 775
OnInvoke, 925, 942-943
OnNavigatedFrom, 74, 654
OnNavigatedTo, 75, 654, 686
OnOrientationChanged, 102
OnPlayStateChanged, 974-975
OnPropertyChanged, 46
OnUserAction, 975-977
OpenFile, 822
Page class navigation, 78
Pause, 969
PhotoCamera preview, 660
Play, 969
PlayTrack, 973
PlayVideo, 674
Populate, 663
PopulateComplete, 263
ProcessEffectFrames, 659-660
ProcessImage, 622
ProcessInput, 689-690
ProcessPeriodicTask, 943-944
ProcessReading, 511-513
ProcessTestImage, 627
RaisePropertyChanged, 601
Refresh, 979
RegisterForNotification, 245
RegisterState, 827, 831
RegisterStatefulProperty, 827, 836-837
Remove, 952
ReplaceFileContents, 947
RestoreDatabase, 963
Rewind, 969
RunWorkerCompleted, 152
SaveFile, 956-957
SaveHtmlToIsolatedStorage, 224-225
SaveImage, 629, 652
SaveItem, 937-938
SavePicture, 629
SaveState, 828
Search, 807-808
SearchAsync, 441, 447
SendEmail, 788
SetAlarm, 910-911
SetCulture, 605
SetCurrentCultureValues, 607
SetEffectState, 658-659
SetImageStream, 634
SetReminder, 914
SetView, 561
ShouldDisplayDefaultSize, 735
Show
AddressChooserTask, 426
CameraCaptureTask, 429
EmailAddressChooserTask, 394
GameInviteTask, 431
PhoneCallTask, 414
PhoneNumberChooserTask, 418
SaveEmailAddressTask, 398
SaveRingtoneTask, 439
ShareStatusTask, 423
SkipNext, 969
SkipPrevious, 969
Start
CaptureSourceViewModel, 670
CompassViewModel, 514
CompassViewModel method, 509
GeoCoordinateWatcher class, 537
GeoPositionViewModel class, 542
GyroscopeViewModel class, 519
SensorBase class, 496
StartRecording, 694
state, 827
Stop, 969
  Gyroscope class, 520
  PhotoCameraViewModel class, 653
  SensorBase class, 496
StopRecording, 694
streaming, 225-226
StringAssert class, 726
Subscribe
  DataContextChangedListener, 782
  PushNotificationSubscriber, 482-484
SuspendMousePromotionUntilTouchUp, 357
TakePhoto, 671
ToggleCapturingVideo, 673
ToggleRecording, 693
TrackArtists, 979
TryStart, 537
UnBindToShellTile, 459
UnBindToShellToast, 459
Update, 16
UpdateCommands
  CaptureSourceViewModel, 671
  MicrophoneViewModel, 694
UpdateFrequencyRanges, 704-705
UpdateSource, 760
UploadImage, 637
Validation class, 751
Wait, 541
WaitOne, 948
WalkPath, 539-540
microphone
  audio, recording, 691-692
capability, 27
  helium voice app, 692
    play command, disabling, 694
    playback, 695-697
    recording audio, 693-694
    saving recordings, 694
    stopping recording, 694
    view page, 697-698
Microphone class, 691-692
MicrophoneView page, 697-698
MicrophoneViewModel class
  code listing, 693
  commands, 693
  HandleBufferReady method, 694
  StartRecording method, 694
  StopRecording method, 694
  ToggleRecording method, 693
  UpdateCommands method, 694
Microsoft
  Intermediate Language (MSIL), 28
  Push Notification Service (MPNS), 463, 478-479. See also push notifications
  minimizing
    AppBar, 238-239
    application bar, 230
Minimum property, 144
MockChatService class, 728
MockGeoCoordinateWatcher class, 539
mocking
  dynamic, 740
  IoC, 740
  launchers/choosers, 743
  licensing service, 741
MockLicensingService class, 741
MockMarketplaceDetailTask class, 743
Mode property, 559, 857
Model-View-ViewModel pattern. See MVVM pattern
monitoring
  accelerometer readings, 498-499
  geographic location position changes, 535-537
Monster, Mark, shake detection article, 507
Motion class, 522
  CurrentValueChanged event, 524
  TimeBetweenUpdates property, 523
motion sensor, 522
  availability, 523
  calibrating, 526
  device position, 524
  modes, 522
  properties, reading, 524
How can we make this index more useful? Email us at indexes@samspublishing.com
motion sensor

- radians to degrees conversion, 525
- sample code, 523
- time between updates, 523
- updates, receiving, 524
- vector, displaying, 525
- view page, 526

MotionFinished event, 210
MotionView page, 526
mouse events, 354
  - border background color, changing, 354-355
  - listing of, 354
  - touch event promotion, 357-358
MouseEnter event, 354
MouseEventsView class, 355
MouseLeave event, 354
MouseLeftButtonDown event, 354
MouseLeftButtonUp event, 354
MouseMove event, 354
MouseWheel event, 354
MovementThreshold property, 534
moving borders, 362-363
MPNS (Microsoft Push Notification Service). See also push notifications
  - batching intervals, 478-479
  - response codes, 463
MSIL (Microsoft Intermediate Language), 28
multiline text blocks, 161-162
multiple language support, 597
MultiScaleImage element
  - collection, accessing, 209
  - creating, 206-207
  - double tapping, 213
  - downloading, stopping, 210
  - dragging, 212
  - instantiating, 208
  - logical coordinates, 209
  - motion finished event, 210
  - overview, 204-205
  - pinching, 212
  - sample code, 210-213
  - Source property, 208-209
  - tiling, 206
  - visibility, 209
  - zooming/panning, 209
MultiTouchGestureEventArgs parameter, 376
Mutex, 946-949
muting media playback, 197
MVVM pattern (Model-View-ViewModel), 42
  - benefits, 42
  - ContextMenu control, 269-271
  - elements, 42
  - implementing, 43
  - MVC, compared, 42
  - property change notifications, 44
    - alternative implementation, 46-48
    - traditional implementation, 44-46
  - viewmodel base class, 43

Name property
  - AlarmViewModel, 910
  - BackgroundServiceAgent, 924
  - TableAttribute class, 848
names
  - camera image files, 651
  - cultures, 606
  - database tables, 848
  - endonym, 604
  - files, 823
  - pushpins, 573
  - resx files, 598
  - thumbnail media library, 662-664
Navigate method, 77, 215, 883
NavigatingCancelEventArgs class, 79
navigation, 70
  - application bar support, 246-247
  - automatic state preservation, 830
  - backward, 78
  - cancelling, 79
  - classes, 70
  - consistent, 58
  - cross-page communication, 80
direct control, 77-78
display elements, 70
external
  Button control, 72
  HyperlinkButton control, 73
  URLS, 236
frame, nesting, 71
handling, 78
hardware Back button, 81-82
history, 74-75
hosting web content within apps, 73
hybrid app pages, 686
internal, 71-72
interpage communication, 74
pages, 112, 424-425
Panorama control
  background images, 338
  Bookshop Service sample app, 346-350
    creating, 343
    FCL placement, 324
    images, 324
    items, 346
    layers, 344-346
    memory, 322
    overview, 323-324, 343
    performance, 322
    Pivot control, compared, 322
    styles, 322
    template, 345
    things to avoid, 351
    title, 345
    touch support, 322
Pivot control
  adding, 325-326
  background images, 338
  creating app with New Project dialog, 327
  FCL placement, 324
  gestures/navigational effects
    support, 325
  headers, 327
  limiting, 323
  load events, 328-329
lockable, 352
memory, 322
messaging app, 331-339
multiple application bars, hosting,
  329-331
overview, 323
Panorama control, compared, 322
performance, 322
PivotItems, 325-328
populating with data bound collection,
  340-343
styles, 322
things to avoid, 351
touch support, 322
redirection, 80-81
service, abstracting, 883-885
splash screens, creating, 82-85
URI mapping, 75-77
web content, 215-216

NavigationCacheMode property, 75
NavigationContext property, 74
NavigationViewEventArgs class, 79-80
NavigationViewMode property, 79
NavigationViewService class, 77
  CurrentSource versus Source property, 78
  methods, 77-78
  Source property, 77
NavigationViewService property, 78
nesting text blocks, 159
NetworkAddressChanged event, 794-797
NetworkChange class, 793
NetworkConnectionMonitor class, 795-796
NetworkInterface class, 793
  GetIsNetworkAvailable method, 793
  NetworkAddressChanged event, 794-797
  NetworkInterfaceType property, 793
NetworkInterfaceType property, 793
networks
  capability, 27
  connections, 792
    availability, checking, 793
    background transfer requests, 952

How can we make this index more useful? Email us at indexes@samspublishing.com
changes, monitoring, 794-797
classes, 793
priorities, 792
data retrieval, 813-816
technologies, 791
New Project dialog
pivot app, creating, 327
XNA, 12
NoiseThreshold property, 503
non-public members, testing, 727
None filter, 258
normalized coordinates, 209
not (!) operator, 717
NotificationChannelErrorEventArgs class
  ErrorAdditionalData property, 462
  ErrorType property, 461
NotificationRateTooHigh errors, 461
notifications
  change, 851
  location, receiving, 542
  property change, 44
    alternative implementation, 46-48
    traditional implementation, 44-46
  push
    benefits, 454-455
    channel errors, 460-461
    cloud, 456, 480
    enabling/disabling, 457
  HttpNotificationChannel events, 460
  identifying, 478
  Microsoft.Phone reference, 457
  power consumption, 454, 461-463
  prioritizing, 478-479
  process, 455
  registered app limits, 460
  sending, 463
  stock ticker app, 480-493
  subscribing, 458-459
  types, 453
  URIs, 455
raw, 454, 474
  receiving, 477
  sending, 474-476
  stock ticker app, 490-492
scheduled, 906
  alarms. See alarms
classes, 907
  limits, 907
  registering, 907
  reminders. See reminders
time and date, 908
types, 907
  user actions, 906
  window, defining, 909
storage space, 820
tile, 453
  background images, 469-472
colors, 473
  displaying, 472
  fields, 469
  image size, 474
overview, 468
  sending, 470-472
  simplicity, 473
  stock ticker app, 489
  template, 469
  updating with shell tile schedule, 473-474
toast, 453, 946
  app icons, customizing, 467
displaying, 467
  overview, 463-464
  receiving, 464-465
  sending, 465-468
  stock ticker app, 489
touch, registering, 356
NotifyComplete method, 925
NotifyOnValidationError property, 752
NotifyPropertyChanged error property, 727
Nowak, Peter, 97
null values (databases), 849
NumericDataSource class, 283-286
obscurity characters, 178
OCC (optimistic concurrency control), 899
  conflict detection, 900-901
  entities, designating, 899
OData (Open Data Protocol), 792
  benefits, 797
  eBay search app
    eBay logo, displaying, 813
    EbayClient class, 804
    network connection changes, monitoring, 807
    OData wrappers, creating, 804-806
    records, retrieving, 807-808
    results, 808-812
    search queries, entering, 810
    view page, 809-813
    viewmodel, 806-809
  entity classes, extending, 816-817
  implementing, 797
  proxies, generating, 800
  queries, 802
    collections, populating, 802-803
    customizing, 802
    model class instance, creating, 802
    results, retrieving, 803
    service operation parameters, 802
    system, 801
    types, 800-802
 URI, 798-799
 website, 797
 wrappers, creating, 804-806
official fonts, 18
offline browsing, 221
  reading content, 225-226
  storing content, 223-225
OnBeginStreaming method, 984-985
OnError method, 977-978
OnErrorsChanged method, 775
OnInvoke method, 925, 942-943
OnNavigatedFrom method, 74, 78, 654
OnNavigatedTo method, 75, 78, 654, 686
OnNavigatingFrom method, 78
OnOrientationChanged method, 102
OnPlayStateChanged method, 974-975
onPropertyChanged method, 46
OnUserAction method, 975-977
opacity (AppBar), 239
Opacity property, 31, 230
Open Data Protocol. See OData
OpenFile method, 822
OptimallyFilteredAcceleration property, 502
optimistic concurrency control (OCC), 899-901
Organizer property, 447
orientation
  compass, 514-515
  page
    animations, 109-112
    determining, 104
    fades, 110
    forcing, 107
    layout visibility conversions, 104-106
    PhoneApplicationPage class, 101
    rotations, 110
    setting at runtime, 106-108
    switching, 102-104
    UI elements, 102, 108-109
    valid device orientations, 103
PanoramaItems, 346
Orientation property, 101, 104-106, 153
OrientationChanged event, 102-104
OrientationChangedEventArgs class, 103
OrientationView.xaml, 108-109
OriginalFileName property, 434
OriginalSource argument, 360
Overdue property, 931
Owner attribute, 722
Page class, 78
PageOrientationExtensions class, 104
PageOrientationToVisibilityConverter class, 105-106

Pages
  animated transitions, 112
  adding, 113-114
  navigation events, 112
  style, 115-116

Navigating, 70
  automatic state preservation, 830
  backward, 78, 112
  cancelling, 79
  classes, 70
  cross-page communication, 80
  direct control property, 77-78
  display elements, 70
  external, 72-73
  forward in, 112
  frame, nesting, 71
  handling, 78
  hardware Back button, 81-82
  history, 74-75
  hosting web content within apps, 73
  internal, 71-72
  interpage communication, 74

Panorama control. See Panorama control
Pivot control. See Pivot control

Redirection, 80-81
Splash screens, creating, 82-85
Web page behaviors, adding, 219-221

Panning (Bing Maps), 563

Panorama control
  background images, 338
  Bookshop Service sample app, 346-350
  creating, 343
  FCL placement, 324
  images, 324
  items, 346
  layers, 344
    background, 344-346
    items, 346
    template, 345
    title, 345
  memory, 322
  overview, 323-324, 343
  performance, 322
  Pivot control, compared, 322
  styles, 322
  things to avoid, 351
  touch support, 322
PanoramaItem control, 323, 346
PanoramaView class, 348
PanoramaViewModel class, 347
PanoramaView.xaml, 348-349

Parameters
  channelName, 458
  DragCompletedGestureEventArgs, 372
  DragDeltaGestureEventArgs, 372
  DragStartedGestureEventArgs, 372
  FlickGestureEventArgs, 373
  PinchGestureEventArgs, 376
  MultiTouchGestureEventArgs, 376
  PinchGestureEventArgs, 376
  PinchStartedGestureEventArgs, 375
  serviceName, 458
  service, 458
HandlePhotoCameraInitialized method, 649-650
SaveImage method, 652
SetEffectState method, 658-659
Stop method, 653
VisualState property, 651, 657

PhotoChooserTask
Completed event, 432
Extras applications, launching, 628
image streams, 434
Photo Picker, launching, 432-433
PhotoChooserView page, 436
photos
cropping, 434-436
displaying, 437
properties, 432
sample code, 434-435
Show method, 433

PhotoChooserView page, 436
PhotoChooserViewModel class, 434-435
PhotoResult class, 434
photos
bits per pixel (bpp), 616
camera
capturing, 649-651
displaying, 657
filenames, 651
saving to media library, 652
Extras applications, 617-618
adding to Extras menu, 619
certification requirements, 621
edge tracing, 619-628
Photo Picker app
cropping, 434-436
displaying, 437
image streams, 434
launching, 432-433
picture viewer. See picture viewer
Pictures Hub, 617
saving, 671
Share menu, 631
apps, adding, 632-633
title, 632
taking, 428-430
thumbnails
creating, 663
displaying, 655, 662-668
loading from media library, 666-668
location, identifying, 662
media library names, 662, 664
new images, detecting, 663
page title, 668
pages, populating, 665-666
path to image conversions, 664-665
presenting, 664
saving, 653
upload share app
certification requirement, 634
converting images to byte arrays, 638-639
creating, 633-641
displaying images, 634, 639-641
dummy images, 633
emulator, 633-635
retrieving images, 633-634
sending images, 635-637
service references, adding, 637
PickerPageUri property, 278
picture viewer
extensibility, 615
Extras applications, 617-618
adding to Extras menu, 619
certification requirements, 621
edge tracing, 619-628
launching, 615
Share apps
certification requirement, 634
converting images to byte arrays, 638-639
creating, 633-641
displaying images, 634, 639-641
dummy images, 633
emulator, 633-635
retrieving images, 633-634
sending images, 635-637
service references, adding, 637
Share menu, 631
apps, adding, 632-633
title, 632
Pictures Hub, 617
pinch gestures, 374-376
border example, 379
Deep Zoom images, 212
sequence, 375-376
Toolkit, 374-376
PinchGestureEventArgs parameter, 376
PinchStartedGestureEventArgs parameter, 375
Pitch property, 524, 697
Pivot control
adding, 325-326
background images, 338
creating app with New Project dialog, 327
FCL placement, 324
gestures/navigational effects support, 325
headers, 327
limiting, 323
load events, 328-329
lockable, 352
memory, 322
messaging app, 331-339
IChatService, implementing, 332
multiple AppBars, 336-338
PivotViewModel class, 331
sending messages, 333-334, 339
viewing received messages, 334-338
multiple application bars, hosting, 329-331
overview, 323
Panorama control, compared, 322
performance, 322
PivotItems, 323-325
active, setting, 328
displaying, 327
populating with data bound collection,
340-343
DataBoundPivotViewModel class, 340
DataBoundPivotView.xaml, 341
MessageViewModelTemplate, 342
SendMessageViewModelTemplate, 342
TypeTemplateSelector class, 341
styles, 322
things to avoid, 351
touch support, 322
PivotItems, 323-325
active, setting, 328
displaying, 327
PivotViewModel class, 331
PivotView.xaml, 336-338
PixelHeight property, 432
PixelWidth property, 432
Plain Old CLR Object (POCO), 844
Play method, 969
playback
audio recordings, 695-697
background audio, 968-970, 982-983
controls, 196
local media files, 972-973
media player controls, 407
muting/unmuting, 197
video recordings, 674-676
PlayCommand, 199
PlayerControls property, 970
PlayerState property, 969, 980-981
playing
media files, 407-411
sound effects, 195
video, 195, 200
PlayState enum values, 975
PlayTrack method, 973
PlayVideo method, 674
POCO (Plain Old CLR Object), 844
pop-up controls, 688
Populate method, 663

How can we make this index more useful? Email us at indexes@samspublishing.com
background transfer requests, 954
bars, 146
cancelling, 152
described, 149-153
indeterminate, 307-308
PerformanceProgressBar, compared, 307
splash loading screens, 83
updating, 149
indicators, 147-149
adding, 148
background transfer requests, 961-962
initializing, 148-149
properties, 147
XML file, 147
ProgressBar class, 146
ProgressIndicator class, 147-149
projects
creating
Silverlight, 4
XNA, 12
Scheduled Task Agent template, 923-924
unit test, creating, 712-714
promoting touch events, 357-358
properties
Accounts, 447
AlarmViewModel, 910
Album, 970
AllowDownloading, 210
ApplicationSettings, 824
Appointments class, 447
apps, customizing
Silverlight, 7
XNA, 13
Artist, 970
Association, 853
AtEndCommand, 813
Attitude, 524
AttitudeReading, 524
AudioTrack, 970
AutoPlay, 196
AutoSync, 849
AvailableFreeSpace, 822
AverageAcceleration, 502
BackgroundAudioPlayer class, 969
BackgroundServiceAgent, 924
BarOpacity, 239
BeginTime, 908-910
BingMapsDirectionsTask, 388
BufferingProgress, 196, 969
Busy, 868
BuyOptionVisible, 742
CanBeNull, 849
Cancel, 79
CanPause, 969
CanSeek, 969
Center
BingMapsTask, 392
Map class, 561
change notifications, 44
alternative implementation, 46-48
traditional implementation, 44-46
ClickMode, 125
compass reading, 510
ConnectionSettingsType, 393
Contact class, 299
Content
AlarmViewModel, 910
defining, 122-123
NavigationEventArgs class, 80
values, 122
ContentType
MarketplaceDetailTask, 401
MarketplaceHubTask, 404
MarketplaceSearchTask, 405
Controls, 407
CopyrightVisibility, 560
CreateOptions, 622
CultureName, 606
Current, 61
CurrentRegion, 699
CurrentSource, 78
CurrentValue, 496
DatabaseSchemaVersion, 894
DataContext, 43, 782

How can we make this index more useful? Email us at indexes@samspublishing.com
Date, 279
DbType, 849
Default, 692
Delay, 126
dependency, 813
DesiredAccuracy, 534
DesiredFormat, 670
DeviceAcceleration, 524
DeviceRotationRate, 524
DeviceStatus class, 39
Dispatcher, 81
DownloadProgress, 196
DownloadProgressOffset, 196
DragDeltaGestureEventArgs parameter, 372
DrawingAttributes, 189
DrawOrder, 681
Duration, 111, 970
EasingFunction, 111-112
EffectEnabled, 658
EnabledGestures, 688
End, 388
Error, 969
ErrorAdditionalData, 462
ErrorType, 461
ExpirationTime
   AlarmViewModel, 910
   ScheduledAction class, 909
Expression, 850
FixDelayMs, 539
FlickGestureEventArgs parameter, 373
FlowDirection, 612
FMRadio class, 698
fonts, 162-163
FontSource, 168
Frequency
   FMRadio class, 698
   FMRadioViewModel class, 701
FullModeHeader, 255
GameTimer class, 682
Gravity, 524
GroupBy, 296
GroupName, 130
HasErrors, 766
HeadingAccuracy, 510
HorizontalAlignment, 123
Icon, 569
identifying, 47
ILockScreenState interface, 66
Images, 664
ImageUrl, 868
InitializationMs, 539
Inlines, 159
InputScope, 171-172
Interval, 126, 473
IsChecked, 128
IsDataValid, 496
IsDbGenerated, 850
IsDiscriminator, 850
IsIndeterminate, 307
IsMenuEnabled, 238
IsPrimaryKey, 849
IsReadOnly, 179
IsSupported
   Compass class, 509
   Gyroscope class, 518
IsTextCompletionEnabled, 267
IsTiltEnabled, 309
IsThreeState, 128
IsVersion, 850, 899
ItemCountThreshold, 252
ItemFilter, 260-262
ItemHeight, 315
ItemsControl class, 140
ItemsSource, 132, 254
LineWidth, 162
LineHeight, 162
LineStackingStrategy, 162
ListPickerMode, 253
Location
   MediaPlayerLauncher, 407
   Pushpin, 567
   PushpinViewModel, 569
LogoVisibility, 560
LongListSelector control, 305-306
LowPassFilterCoefficient, 502
LowPassFilteredAcceleration, 502
MagneticHeading, 510
MagnetometerReading, 510
ManipulationCompletedEventArgs parameter, 362
ManipulationDeltaEventArgs parameter, 361
MarketplaceDetailTask, 401
MarketplaceSearchTask, 405
MaxUpdateCount, 474
Media, 407
MediaElement audio output, 196
Message, 868
MockGeoCoordinateWatcher class, 539
Mode, 559, 857
MovementThreshold, 534
Name
  AlarmViewModel, 910
  BackgroundServiceAgent, 924
  TableAttribute class, 848
NavigatingCancelEventArgs class, 79
NavigationCacheMode, 75
NavigationContext, 74
NavigationMode, 79
NavigationService, 78
NetworkInterfaceType, 793
NoiseThreshold, 503
NotifyOnValidationError, 752
OptimallyFilteredAcceleration, 502
Orientation, 104-106
Overdue, 931
PasswordChar, 178
Path, 611
PhoneApplicationPage class, 101
PhoneNumber, 414
PhotoChooserTask, 432
PhotoResult class, 434
PickerPageUri, 278
PinchGestureEventArgs parameter, 376
PinchStartedGestureEventArgs parameter, 375
Pitch, 524, 697
PlayerControls, 970
PlayerState, 969, 980
Position, 969
  MainPageViewModel, 982-983
  MediaElement, 196
PositionChangedDelayMs, 539
PoweredOn, 700-701
PowerMode, 698
Products, 89
ProgressIndicator class, 147
PushpinViewModel, 569
Quaternion, 524
QueryString, 74
RangeBase class, 144-146
Reading, 501
Recurrence, 474
RecurrenceIntervals, 911
Region, 702
Regions, 702
RemoteImageUri, 473-474
RepeatButton control, 126
Roll, 524
RotationMatrix, 524
RotationRate, 519
SaveCommand, 399
ScreenName, 868
ScrollBar class, 154
SearchCount, 827
SearchQuery, 420
SearchTerms, 405
SelectedIndex, 144
SelectedItems, 144
SensorBase class, 496

How can we make this index more useful? Email us at indexes@samspublishing.com
ShakeCount, 508
ShellTileSchedule class, 473
ShowMessageCommand, 270
SignalStrength
  FMRadio class, 699
  FMRadioViewModel class, 705
Sound
  Alarm class, 908
  AlarmViewModel, 910
Source, 77-78, 970
  BackgroundServiceAgent, 924
data bindings, 611
  images, 186
  MediaElement, 196
  MultiScaleImage element, 208
  WebBrowser control, 215
Specifier, 924
Start, 388
StartTime, 474
stateful, identifying, 830
Status, 423
Stretch, 187
Strokes, 188
SubImages, 209
SupportedCultures, 605
SupportedFormats, 670
SupportedOrientations, 106-108
SupressTilt, 309
SwitchForeground, 314
Tag, 970
TagExpression, 718
TargetElapsedTime, 16
text
  MainPage.xaml, 12
  PushpinViewModel, 569
TextBlock control, 162-163
TextBox control, 162-163, 168
TimeBetweenUpdates
  Compass class, 509
  Gyroscope class, 518
  Motion class, 523
  SensorBase class, 496
TimelineItems, 868
TimeSpan, 816
Timestamp
  compass, 510
  motion sensors, 524
Title, 668, 970
TodoItem class, 927
Todos, 928
ToggleOn, 312
TouchPoint class, 356
Track, 969
TrackCommand, 565
TrackTitle, 979
TransferPreferences, 952
TransferStatus, 955
Trial, 741
TrueHeading, 510
TwitterTimelineViewModel, 868
type, 924
UI element visibility, 31
UpdateCheck, 850
UpdateInterval, 682
UpdateSourceTrigger, 760
UploadLocation, 953
Uri, 79, 80
UriMapper, 76
UriMapping class, 76
UseSprings, 209
UTF
  AssemblyCleanup, 720
  AssemblyInitialize, 719
  Asynchronous, 723
  Bug, 723
  ClassCleanup, 720
  ClassInitialize, 720
  Description, 722
  ExpectedException, 723
  Ignore, 721
  Owner, 722
  Priority, 724
  TestClass, 718
  TestCleanup, 721
TestInitialize, 721
TestMethod, 719
TestProperty, 721
timeout, 722
ValidatesOnExceptions, 752
validation
activities, 781
all at once, 775-779
asynchronously, 770-772
changes, 772-775
completing, 781
ersors, creating, 779
registration, 768-769
synchronously, 769-770
whitespace example, 780
ValueMemberPath, 265
ValueStringFormat, 277
VerticalAlignment, 123
VerticalOffset, 813
ViewportOrigin, 209
ViewportWidth, 209
ViewState, 981
Visual Studio resource, selecting, 610
VisualState, 587
PhotoCameraViewModel, 651, 657
TodoItemViewModel, 940-942
Volume, 196, 969
WayPoints, 539
WorkerReportsUpdates, 152
Yaw, 524
ZoomAboutLogicalPoint, 209
ZoomBar, 563-564
ZoomLevel, 562
property setter validation, 749
binding errors, 751-752
critical exceptions, 751
data binding steps, 749-751
enabling, 749
limitations, 765
source property assignment, 749
Validation class, 751

PropertyChanged event, 851
DataErrorNotifier class, 772-775
INotifyPropertyChanged interface, 44
memory leaks, 562
PropertyChangeNotifier class, 48
PropertyChanging event, 851
PropertyUtility class
CreateGetter method, 838
GetPropertyInfo method, 837
public fields, 855
push notifications
benefits, 454-455
capability, 27
channel errors, 460-461
cloud, 456, 480
enabling/disabling, 457
HttpNotificationChannel events, 460
identifying, 478
Microsoft.Phone reference, 457
power consumption, 454, 461-463
prioritizing, 478-479
process, 455
raw, 454, 474
receiving, 477
sending, 474-476
stock ticker app, 490-492
registered app limits, 460
sending, 463
stock ticker app, 480
channel subscription, 482-484
cloud service subscription, 484-487
images, 489
input controls, 481
server side implementation, 481
stock quotes, receiving, 487-488
storing information, 489
unregistering, 492-493
Yahoo! stock data format, 488
subscribing, 458-459
tile, 453
background images, 469-472
colors, 473

How can we make this index more useful? Email us at indexes@samspublishing.com
displaying, 472
fields, 469
image size, 474
overview, 468
sending, 470-472
simplicity, 473
stock ticker app, 489
template, 469
updating with shell tile schedule, 473-474
toast, 453
app icons, customizing, 467
displaying, 467
overview, 463-464
receiving, 464-465
sending, 465-468
stock ticker app, 489
URIs, 455
PushNotificationSubscriber class, 458
PushNotificationViewModel, 477
PushNotifier class
raw notifications, sending, 474-476
toast notifications, sending, 465-466
Pushpin class, 567
PushpinIconConverter class, 571
pushpins (Bing Maps), 567
appearance, 568
creating, 572
displaying, 573
icon images, 571-572
itinerary items, 592
populating
data-bound collections, 568-569
map layers, 567-568
presenting, 570
template, 570
user prompt for naming, 573
PushpinViewModel class
addPushpinCommand, 572
code listing, 568
displaying, 570
properties, 569

Q
Quaternion property, 524
queries
interpage communication, 74
LINQ to SQL, 885-887
OData, 802
collections, populating, 802-803
customizing, 802
model class instance, creating, 802
results, retrieving, 803
service operation parameters, 802
system, 801
types, 800-802
web APIs, 862-863
QueryString property, 74

R
radians to degrees conversion, 525
RadiansToDegreesConverter class, 525
radio buttons, 129-130
data binding to viewmodel collections, 130-133
events, 129
groups, 130
RadioButton class, 129-130
data binding to viewmodel collections, 130-133
events, 129
GroupName property, 130
RaisePropertyChanged method, 601
Randall, Ian, 739
range controls, 144-146
event, 145
progress bars, 146
cancelling, 152
example, 149-153
updating, 149
progress indicators, 147-149
adding, 148
initializing, 148-149
properties, 147
XAML file, 147
properties, 144-146
scrollbars, 154
sliders, 153-154
orientation, 153
Pivot/Panorama controls, 154
templates, 154
values, 153
RangeBase class, 144-146
raw notifications, 454, 474
  receiving, 477
  sending, 474-476
  stock ticker app, 490-492
Reactive Extensions. See Rx
reading
  isolated storage data, 822-824
  web content offline, 225-226
Reading property, 501
ReadingSmoother class, 511-513
ReadStreamBytes method, 225-226
  receiving
    location notifications, 542
    raw notifications, 477
    stock quotes, 487-488
    toast notifications, 464-465
  updates
    accelerometer, 499
    compass, 509
    gyroscope, 519
    motion sensor, 524
    sensors, 496
recording audio/video
  audio formats, 670
CaptureSource class, 669
helium voice app, 692-698
images, 649
  formats, 670
  saving, 671
microphone, 691-692
playing, 674-676
resolution settings, 670
starting/stopping, 673-674
user authorization, 669
Recurrence property, 474
RecurrenceIntervals property, 911
redirecting pages, 80-81
Refresh method, 979
Region property, 702
Regions property, 702
RegisterForNotification method, 245
registering
  alarms, 909
  audio streaming agents, 984
  background agents, 920
  Bing Maps API key, 554-555
  properties, 768-769
  reminders, 912-913
  scheduled notifications, 907
  scheduled tasks, 918, 926
  touch notifications, 356
RegisterState method, 827, 831
RegisterStatefulProperty method, 827, 836-837
regular expressions, 816
relative URIs, 186-187
releasing sensor resources, 497
Relyea, Dave, 154
reminders
  alarms, compared, 908
  creating, 914
  existing, displaying, 915
  properties, customizing, 916
  registering, 912-913
  set reminder view, 917
  snooze/dismiss actions, 917
  viewmodel, 914
ReminderView class, 915-916
ReminderViewModel, 914-915
RemoteImageUri property, 473-474
Remove method, 952
rendering
  hybrid apps, 679-681
  Silverlight content in XNA, 684-688
    drawing to screen, 687-688

How can we make this index more useful? Email us at indexes@samspublishing.com
gesture input, 686-687
page navigation, 686
results, 688
RepeatButton class, 126-128
example, 127-128
properties, 126
ReplaceFileContents method, 947
Representational State Transfer (REST), 791
Request class, 955
requirements
background
audio user opt-in, 978
transfer requests, 952
capabilities, 28
certification
Back button, 82
background audio user opt-in setting, 978
Extras applications, 621
photo share applications, 634
push notifications, disabling, 457
Extras applications, 621
memory, 39
performance, 33-34
photo share applications, 634
transient state, 64
resolution (camera), 649, 670
resource intensive tasks, 922
ResourceIntensiveTask class, 922
resources
color, 754
common, accessing, 946-949
properties, 610
REST (Representational State Transfer), 791
RestoreDatabase method, 963
restoring
local databases, 963-965
transient state, 64-65
viewmodel state, 829
ResultEventArgs class, 805
resx files, 596
access modifiers, setting, 598
creating, 597-600
culture codes, 598
enabling, 597
generated designer class, 599
images, localizing, 602-603
names, 598
runtime language changes, 603
cultures, 604-606
data binding, 611
date/time/currency formatting properties, 607
localized text/images, displaying, 607-610
right to left support, 612
Visual Studio resource properties, selecting, 610
retrieving
application bar buttons/menu items, 232-233
appointments, 446-450
accounts available for searching, 447
querying, 446
sample code, 447-450
search results, displaying, 446
background transfer requests, 956
cached data from local databases, 863-864
cached timelines, 869-870
contacts, 440
accounts available for searching, 442
sample code, 442-445
search results, displaying, 441
data over networks, 813-816
driving directions, 388-391
enum values, 173
images, 621
OData query results, 803
product IDs, 402
todo items, 930
URIs, 77
users from databases, 860
SavePhoneNumberTask, 417-419
Completed event, 417
sample code, 418-419
Show method, 418
SavePhoneNumberView page, 419
SavePicture method, 629
SaveRingtoneTask, 437-440
SaveState method, 828
saving
audio recordings, 694
contacts, 427-428
data to isolated storage, 822-824
e-mail addresses, 397-400
images, 629-631
persistent state, 65
phone numbers, 417-419
photos, 652, 671
ringtones, 437-440
scheduled tasks, 937
state, 830
thumbnails, 653
transient state, 63-64
Scale property, 361
scale transforms, 30
scheduled notifications, 906
alarms
AppBar button, 912
begin times, 911
creating, 910-911
expiration times, 911
properties, 910
recurrence, 911
registering, 909-911
reminders, compared, 908
set alarm view, 912
sounds, 908-909
time, setting, 909
classes, 907
limits, 907
registering, 907
reminders
alarms, compared, 908
creating, 914
existing, displaying, 915
properties, customizing, 916
registering, 912-913
set reminder view, 917
snooze/dismiss actions, 917
viewmodel, 914
time and date, 908
types, 907
user actions, 906
window, defining, 909
Scheduled Task Agent project template,
923-924
scheduled tasks, 906, 918
agents, 921
audio player, 919
background tasks, compared, 921
class, creating, 924
common resources, accessing, 946-949
completion, 925
feedback to users, providing, 946
memory limitations, 925
properties, 924
registration, 920
scheduled task, 919
shell tiles, updating, 942-944
types, 919, 925
unavailable resources, 925
API limitations, 945
AppBar button, 935
changes, updating, 941
CRUD operations, 935
debugging, 932
deleting, 939
disabling, 918
displaying, 933-935
editing, 938-941
expiration, 927, 931
hiding/displaying, 940-942
IoC container, 940
navigating back to view page, 937-938
periodic, 921-922
creating, 926
processing, 943-944
Shell tiles
- creating, 936-937
- displaying, 942
- updating, 942-944

Todo list app
- adding todo items, 929
- AppBar button, 935
- changes, updating, 941
- creating todo items, 927
- CRUD operations, 929-930, 935
- deleting todo items, 939
- displaying, 933-935
- editing, 938-941
- grouping todo items, 931
- hiding/displaying elements, 940-942
- IoC container, 940
- navigating back to TodoItemList page, 937-938
- populating, 931
- retrieving todo items, 928-930
- saving todo items, 937
- shell tiles, 936-937, 942-944
- testing, 932
- todo items, 928-929
- TodoListViewModel constructor, 932

ScheduledAction class
- BeginTime property, 908
- ExpirationTime property, 909

ScheduledActionService class, 909

ScheduledTaskAgent class, 919

schema
- displaying, 873
- isolated storage file explorers, 873
- built-in Isolated Storage Explorer, 873-875
- WP7 Isolated Storage Explorer, 875-876

SQL CE files, 876-878
- versioning, 894

schema, updating, 887
- columns, adding, 888
- indexes, adding, 890-891
- one-to-many associations, adding, 891-893
- tables, adding, 889-890

Schulte, René, 660

scope (input), 171-172
- default view, 176
- enum values, retrieving, 173
- example, 173-176
- InputScopeNameValue enum values, 171
- intellisense support, 172
- ListPicker control view, 176
- selected items example view, 176
- text entry example view, 176
- values, converting, 174-175
- word prediction, 172

ScreenName property, 868

screens
- fill/frame rate counter, 33
- orientation
  - animations, 109-112
  - determining, 104
  - fades, 110
  - forcing, 107
  - layout visibility conversions, 104-106
  - PhoneApplicationPage class, 101
  - rotations, 110
  - setting at runtime, 106-108
  - switching, 102-104
  - UI elements, 102, 108-109
  - valid device orientations, 103
- splash, creating, 82-85

ScrollBar class, 154
scrollbars, 154

ScrollingCompleted event, 306
ScrollingStarted event, 306
ScrollTo method, 306
ScrollViewer, 200-201

How can we make this index more useful? Email us at indexes@samspublishing.com
ScrollViewerMonitor class, 813
   AtEndCommand property, 813
   code listing, 813-816
   VerticalOffset property, 813
SDK (Software Development Kit)
   downloading, 2
   frame rate counter, 31
      enabling, 31
      field descriptions, 32-33
   installing, 2
   Marketplace Test Kit, 28
   Performance Analysis tool
      CPU Usage graph, 36-37
      detailed profiling information, viewing, 37
      Frame Rate graph, 35
      garbage collection, 37
      image loads, 37
      launching, 34
      Memory Usage MB graph, 37
      overview, 34
      results, viewing, 35
      running, 34
      stopping, 35
      storyboards, 37
   tools, 2
Search method, 807-808
Search service, 574
SearchAsync method, 441, 447
SearchCommand, 806
SearchCompleted event
   Appointments class, 446
   Contacts class, 441
SearchCount property, 827
searching
   appointments, 446
   contacts, 441-442
   map routes, 584-585
   Marketplace apps, 405-406
   web, 420-421
SearchItemsCompleteEventArgs class, 806
SearchQuery property, 420
SearchTask, 420-421
SearchTerms property, 405
SearchView page, 421
SearchViewModel, 421
security
   capabilities
      app requirements, analyzing, 28
      defined, 26
      discovery process, 28
      displaying, 26
      listing of, 26-27
      manifest file, 26
      marketplace submissions, 27
   cloud authentication, 480
   credentials, 578
   databases, 857
   user authorization, 669
SelectedIndex property, 141, 144
SelectedItem property, 141, 144
   ListBox control, 254
   LongListSelector control, 305
SelectedItems class, 138-140, 411-413
SelectedItems property
   binding, 411-413
      check boxes, data binding, 138-140
   SelectedRadioButton property, 132
SelectionChanged event, 141
   AutoCompleteBox, 266
   LongListSelector control, 306
SelectionMode property, 142
Selector class, 141
semantic validation, 748
Send button, 786
SendEmail method, 788
SendEmailView, 786
SendEmailViewModel, 786, 789
sending
   email, 786-789
   images, 635-637
   messages to web pages, 218-219
notifications
  push, 463
  raw, 474-476
  tile, 470-472
  toast, 465-468
SendMessageViewModel class, 333-334
SendMessageViewModelTemplate, 342
SensorBase class, 495
  CurrentValue property, 496
  IsDataValid property, 496
  properties, 496
  Start/Stop methods, 496
  TimeBetweenUpdates property, 496
sensors
  accelerometer
    axis values, 498
    calibrating, 503
    compass orientation, 514
    g's, 497
    overview, 497
    readings, monitoring, 498-499
    sample view, 503-506
    shake detection, 507-508
    simulating, 499-500
    smoothing readings, 500-503
    stationary readings, 497
    updates, receiving, 499
  accuracy, improving, 522
  capability, 27
  compass
    calibrating, 515-517
    interference, 509
    magnetic north arrow, 513-514
    noise, removing, 513
    orientation, 514-515
    overview, 508
    reading, 510
    smoothing data, 511-513
    starting, 509
    support, 509
  time between updates, 509
  updates, receiving, 509
  view page, 513
  data acquisition, starting/stopping, 496
  data validity, checking, 496
  device support, 523
  gyroscope
    angular rotations, 519-520
    cumulative values, 521
    disposing, 520
    drift, 521
    instance, creating, 519
    overview, 518
    time between updates, 518
    updates, receiving, 519
    view page, 520-521
  intervals between readings, 496
  motion, 522
    availability, 523
    calibrating, 526
    device position, 524
    modes, 522
    radians to degrees conversion, 525
    reading properties, 524
    sample code, 523
    time between updates, 523
    updates, receiving, 524
    vector, displaying, 525
    view page, 526
  overview, 495-497
  resources, releasing, 497
  support, 518
  updates, receiving, 496
  values, holding, 496
serialization
  binary, 834-837
  performance, 824
Serializer website, 834
  servers, uploading to files, 961
  service operation parameters, 802
  serviceName parameter, 458

How can we make this index more useful? Email us at indexes@samspublishing.com
ServiceReferences.ClientConfig file, 575

services
  cloud
    authentication, 480
    notification channels, creating, 480
    push notifications, 456, 480
  dialog, 52-54
    class diagram, 52
    IMessageService interface, 52
    MessageService class, 52-54

Geocode
  address search, 577
  overview, 574
  route calculation example, 576-578

geographic location, 26

HTTP, 792

Imagery, 574

IMessageService, 270

mock licensing, 741

MPNS
  batching intervals, 478-479
  response codes, 463

navigation, abstracting, 883-885

network
  data retrieval, 813-816
  technologies, 791

OData, 792
  benefits, 797
  entity classes, extending, 816-817
  implementing, 797
  proxies, generating, 800
  queries, 800-803
  URI, 798-799
  website, 797
  wrappers, creating, 804-806

REST, 791

Route, 574-578

Search, 574

SOAP, 791
  client configuration file, 575
  coordinating calls with Rx, 579-582

displaying routes/itineraries, 590-593

listing of, 574

reference, adding, 575

route calculation, 576-578

route searches, 584-585

user provided to and from addresses, 582-584

visual states, 586-590

Twitter, 860

WCF
  BookshopService, 97-99
  creating, 636
  URL routing, 957-958

set accessors, 838-839

set alarm view, 912

set reminder view, 917

SetAlarm method, 910-911

SetCulture method, 605

SetCurrentCultureValues method, 607

SetEffectState method, 658-659

SetImageStream method, 634

SetReminder method, 914

SetText() method, 182

SetView method, 561

shake detection, 507-508

Shake event, 507

ShakeCount property, 508

shapes (buttons), 123

Share menu, 631-633

SharedGraphicsDeviceManager class, 679-681

ShareLinkTask, 422

ShareLinkTaskView page, 422

ShareStatusTask, 423

sharing hyperlinks, 422

shell tiles
  creating, 936-937
  displaying, 942
  updating, 942-944

ShellTileSchedule class, 473-474

ShellToastNotificationReceived event, 460, 465

ShouldDisplayDefaultSize method, 735
overview, 34
results, viewing, 35
running, 34
stopping, 35
storyboards, 37
rendering, 679
Serializer, 834
Toolkit, 249-250
UTF (Unit Testing Framework). See UTF
WCF service, 636
XAP files
   Application Deployment tool, 25
   contents, 24
   overview, 24
   size, 25
XNA, compared, 2-4
SilverlightSerializer class, 834
SilverlightTest class, 724-725, 737
Simple Object Access Protocol. See SOAP
single table inheritance, 895
SIP (Software Input Panel), 168
dimensions, 169
dismissing, 170-171
expanding, 170
input scope, 171-172
   default view, 176
   enum values, retrieving, 173
   example, 173-176
   InputScopeNameWithValue enum values, 171
   intellisense support, 172
   ListPicker control view, 176
   selected items view, 176
   text entry example view, 176
   values, converting, 174-175
   word prediction, 172
keyboard layouts, 169
launching, 170
overview, 168
size
   application bar, 230
   borders
      pinch gestures, 379
      scaling, 362-363
      tap gestures, 378
   buttons, 123
   fonts, 163
   images, 187-188
   local databases, 857
   SIP 169
   system tray, 230
tile images, 474
UI elements based on page orientation, 102
XAP files, 25
Size property, 356
sketch page app, 192-195
SkipNext method, 969
SkipPrevious method, 969
Slider class, 153-154
   Orientation property, 153
templates, 154
Value property, 153
sliders, 153-154
   background audio player, 983
   Bing Maps zooming, 562
   orientation, 153
   Pivot/Panorama controls, 154
templates, 154
values, 153
SmallChange property, 145
smoothing
   compass data, 511-513
   accelerometer readings, 500-503
SMS messages, composing, 423-424
SmsComposeTask, 423-424
snooze reminders, 917
SOAP (Simple Object Access Protocol), 791
   client configuration file, 575
   coordinating calls with Rx, 579-582
itineraries, displaying, 590-593
listing of, 574
reference, adding, 575
route calculation
  completed route, displaying, 590
  Geocode/Route services, 576-578
  searching for routes, 584-585
  user provided to and from addresses, 582-584
  visual states, 586-590
Sobel filter, 624-625
social networks
  links, sharing, 422
  status updates, posting, 423
Software Development Kit. See SDK
Software Input Panel. See SIP
sorting Marketplace apps, 298
Sound property
  Alarm class, 908
  AlarmViewModel, 910
SoundEffect class, 195, 202-204
sounds
  alarms, 908-909
  background
    agent, creating, 970-972
    audio streaming agents, 983-986
    classes, 968
    controlling from foreground apps, 978-983
    foreground app actions, handling, 975-977
    local media file playback, 972
    play state, toggling, 973
    player, 968-970
    track information, 970
    track numbers, managing, 973
    track state changes, responding, 974-975
capturing, 669
  CaptureSource class, 669
  formats, 670
  playing, 674-676
  starting/stopping, 673-674
  user authorization, 669
effects, 202-204
FM radio
  displaying, 706-708
  FMRadio class, 698
  frequencies, 698, 701-705
  onscreen menu, 705
  power modes, 700-701
  regions, 699, 702
  signal strength, 699, 705
  turning on/off, 698
  view page, 708
media viewer page sample code, 196-202
  AppBar control, 201-202
  binding properties to viewmodel, 199
  MediaView class, 197-199
  muting/unmuting playback, 197
  position control, 199
  scroll viewer, 200-201
  video, playing, 200
MediaElement, 195
output, 196
playing
  agents, 919
  controls, 196
  media player, 407-411
recording with microphone
  helium voice app, 692-698
  Microphone class, 691-692
ringtones, 437-440
sources, 196
streaming, 196, 983-986
  app assembly audio files, playing, 985
  creating, 984
  decoding tracks, 985
  registering, 984
  track information, receiving, 984
  Web files, 986
volume, 196

How can we make this index more useful? Email us at indexes@samspublishing.com
Source property, 77-78, 970
BackgroundServiceAgent, 924
data bindings, 611
images, 186
MediaElement, 196
MultiScaleImage element, 208
WebBrowser control, 215
Specifier property, 924
splash screens, creating, 82-85
Sprite Fonts, adding, 17
SQL CE (SQL Server Compact Edition) files, 842, 876-878
SQL Server Management Studio, 878
SqlMetal, 878
standard fonts, 163-165
Start method
CaptureSourceViewModel, 670
CompassViewModel, 509, 514
GeoCoordinateWatcher class, 537
GeoPositionViewModel class, 542
GyroscopeViewModel class, 519
SensorBase class, 496
Start property, 388
starting. See launching
StartRecording method, 694
StartsWith filter, 258
StartsWith method, 726
StartsWithCaseSensitive filter, 258
StartsWithOrdinal filter, 258
StartsWithOrdinalCaseSensitive filter, 259
StartTime property
Appointments class, 447
ShellTileSchedule class, 474
startup pages, customizing, 7
stateful properties, identifying, 830, 836-837
StateManager class, 829
states, 59
automatic preservation, 826
binary serialization, 834-837
dictionaries, 831-834
eBay search app example, 827-828
Lambda expressions, unwinding, 837-838
loading state, 830
methods, 827
property accessor delegates, creating, 838-839
restoring state, 834
saving state, 830
stateful properties, identifying, 830, 836-837
type required, 827
viewmodels, customizing, 828-829
background audio, 969, 973,980
foreground app actions, handling, 975-977
track state changes, responding, 974-975
background transfer requests, 955
camera, 657
check boxes, 133
compass calibration, 516
dictionary, 59
fast application switching, 57
ListPicker control, 253
loading, 830
map routes/itineraries, 586-590
persistent,
apps, 59, 65
isolated storage, 824-826
saving, 830
toggle buttons, 128, 312
transient, 59, 63-65
viewmodels
restoring, 829
saving, 828
visibility, 123
visual, 752-757
Status property
Appointments class, 447
ShareStatusTask, 423
status updates, posting, 423
StatusChanged event, 536-537
still images. See photos
stock ticker app, 480
  channel subscription, 482-484
  cloud service subscription, 484-487
  images, 489
  input controls, 481
  server side implementation, 481
  stock quotes, receiving, 487-488
  storing information, 489
  unregistering, 492-493
  Yahoo! stock data format, 488
StockQuote class, 489
StockQuoteService class, 484-487
Stop method, 969
  Gyroscope class, 520
  PhotoCameraViewModel class, 653
  SensorBase class, 496
stopping
  background audio, 969
  video recording, 673-674
StopRecording method, 694
storage
  isolated
    API, 820
    application settings, 826
    apps, minimizing, 820
    databases, 856, 880-882
    file explorers, 873-876
    free space, 820-822
    managed dictionary, 820
    overview, 819
    persistent state, 824-826
    reading/writing data, 822-824
    serialization performance, 824
    virtual file system, 821-822
LockScreenManager class, 67
offline browsing, 221
  reading content, 225-226
  storing content, 223-225
  persistent state, 59
  transient state, 59
storyboards (Performance Analysis tool), 37
streaming
  audio, 196, 983-986
  app assembly audio files, playing, 985
  calling over built-in streaming, 984
  creating, 984
  decoding tracks, 985
  registering, 984
  track information, receiving, 984
  Web files, 986
  fonts, 168
  images, 434
  media content, 196
  Rx data subscription, 547
street addresses
  resolution, 545
  selecting, 425-427
Stretch property, 187
StretchingBottom event, 306
StretchingCompleted event, 306
StretchingTop event, 306
StringAssert class, 726
strings
  assertions, 726
  asynchronous validation example
    completing, 781
    errors, creating, 779
    validation activities, 781
  connection, 856-859
  query, 74
StringToImageConverter class, 664-665
strokes, 188
  appearance, 189
  collections, 189
  creating, 189
  user input, 190-191
Strokes property, 188
styles
  colors, 230
  defining, 753

How can we make this index more useful? Email us at indexes@samspublishing.com
fonts, 163
MainPage.xaml, 12
Pivot/Panorama controls, 322
predefined listing of, 12
pushpin itinerary items, 592
suggestion lists, 264-266
transitionPageStyle, 115
transitions, 115-116
SubImages property, 209
Subject property, 447
SubmitCommand, 780
submitting apps (Marketplace), 27
Subscribe method
DataContextChangedListener class, 782
PushNotificationSubscriber, 482-484
subscribing
Completed event, 387-388, 394
FrameReported event, 356
life cycle events, 61
push notifications, 458-459
Rx data stream, 547
stock ticker
cloud service, 484-487
notification channel, 482-484
suggestion list (AutoCompleteBox)
bug, 256
dynamic population, 262-264
filtering
alternate, 260
availability, 259
custom, 260-262
selecting, 260
two-way data binding, 259
values, 257-259
styles, 264-266
suggestions, displaying, 256
SuggestionItem class, 264
SupportedCultures property, 605
SupportedFormats property, 670
SupportedOrientations property, 101, 106-108
SuppressTilt property, 309
surface counter frame rate counter field, 33
SuspendMousePromotionUntilTouchUp
method, 357
SwitchForeground property, 314
switching page orientation, 102-104
UI elements, sizing, 102
valid device orientations, 103
synchronizing database columns automatically, 849
synchronous validation, 769-770
syntactic validation, 748
system
queries, 801
tray
appearance, 243
hiding, 239-241
size, 230
SystemTray class, 243

T

Table<TEntity> class, 855
TableAttribute class, 848
tables
CRUD operations, 855
databases
adding, 889-890
names, 848
tabs
adding, 325-326
background images, 338
creating app with New Project dialog, 327
FCL placement, 324
gestures/navigational effects support, 325
headers, 327
limiting, 323
load events, 328-329
lockable, 352
memory, 322
messaging app, 331-339
   IChatService, implementing, 332
   multiple AppBars, 336-338
   PivotViewModel class, 331
   sending messages, 333-334, 339
   viewing received messages, 334-338
   multiple application bars, hosting, 329-331
overview, 323
Panorama control, compared, 322
performance, 322
PivotItems, 323-325
   active, setting, 328
   displaying, 327
populating with data bound collection, 340-343
   DataBoundPivotViewModel class, 340
   DataBoundPivotView.xaml, 341
   MessageViewModelTemplate, 342
   SendMessageViewModelTemplate, 342
   TypeTemplateSelector class, 341
styles, 322
things to avoid, 351
touch support, 322
tag expressions, 717
   assigning explicitly, 717
   editor, 715, 726-727
   entering, 717
   implicit, 717
   setting programmatically, 718
   symbols, 717
   syntax, 717
Tag property, 970
TagExpression property, 718
TakePhoto method, 671
tap gestures, 366
   border example, 378
   buttons, 125
   GestureListener class, 369-370
   Toolkit, 369-370
   UIElement, 365-366
TargetElapsedTime property, 16
tasks
   background, 906, 921
   periodic, 921-922
      creating, 926
      processing, 943-944
   resource intensive, 922
   scheduled, 906, 918
      agents, 919-925
      API limitations, 945
      AppBar button, 935
      changes, updating, 941
      common resources, accessing, 946-949
      CRUD operations, 935
      debugging, 932
      deleting, 939
      disabling, 918
      displaying, 933-935, 940-942
      editing, 938-941
      expiration, 927, 931
      feedback to users, providing, 946
      hiding, 940-942
      IoC container, 940
      navigating back to view page, 937-938
      periodic. See periodic tasks
      project template, 923-924
      registering, 918, 926
      resource intensive, 922
      restrictions, 923
      saving, 937
      shell tiles, 936-937, 942-944
      todo list app. See todo list app
TaskScheduler class
   OnInvoke method, 942-943
   ProcessPeriodicTask method, 943-944
TemplatedItemsControl class, 324
templates
   AutoCompleteBox, 256
   Game, 13
   hybrid project, 678-679
   ListPicker, 255
Marketplace app list items, 296
MessageViewModelTemplate, 342
Panorama control, 345
pushpins, 570
Scheduled Task Agent, 923-924
SendMessageViewModelTemplate, 342
sliders, 154
tile notifications, 469
validation errors, 752-757
test classes, creating, 714-716
test harness, 715
TestClass attribute, 718
TestCleanup attribute, 721
testing
asynchronous, 736-737
automated, 710
coded UI tests, 711
integration tests, 711
unit tests, 710
coded UI
chat client app, 734-736
elements, runtime manipulation, 737-738
overview, 711
foreground control page of background audio, 978
geographic location features
code-driven, 539-541
location simulator, 538-539
integration, 711
IoC, 739
container, creating, 739-740
Dependency Injection, 739
DI, 739-740
frameworks, 739
mocking, 740
service location, 739
Mutex ownership, 948
partitioning, 711
scheduled tasks, 932
trial versions
application bar button, displaying, 741-743
benefits, 741
mock licensing service, 741
unit
assemblies, 727
assertions, 725-726
asynchronous, 736-737
attributes, 718-724
benefits, 709
chat client, 730-732
disadvantage, 709
execution order, 719
expressions editor, hiding, 726-727
inheritance, 724-725
launchers/choosers, 743-744
non-public members, 727
overview, 710
projects, creating, 712-714
results, displaying, 715
running, 715
tag expressions, 717-718
test classes, creating, 714-716
test harness, 715
UTF, 711-712
TestInitialize attribute, 721
TestMethod attribute, 719
TestProperty attribute, 721
TexFat file system, 823
text
blocks
creating, 159
font properties, 162-163
inline objects, 159
line breaks, 159-161
multiline, 161-162
nesting, 159
overview, 159
strings with different formatting, 160-161
boxes, 168
font properties, 162-163
input scope, 171-172
properties, 168
TimeBetweenUpdates property
Compass class, 509
Gyroscope class, 518
Motion class, 523
SensorBase class, 496
TimelineItem class, 846-848
TimelineItems property, 868
Timeout attribute, 722
TimePicker control, 273-275
alarms, 911
DatePicker, compared, 274
full-screen picker page, 278-282
header, 277
icons, 275
selection mode, 273
value format, 277-278
ValueChanged event, 275
timers, 706
times
alarms
begin, 911
expiration, 911
setting, 909
LoopingSelector control, 283-286
data binding, 286
data source, 283
numeric values, presenting, 283-286
scheduled notifications, 908
TimePicker, 273-275
DatePicker, compared, 274
full-screen picker page, 278-282
header, 277
icons, 275
selection mode, 273
value format, 277-278
ValueChanged event, 275
TimeSpan property, 816
Timestamp property
compass, 510
motion sensors, 524
title layer (Panorama control), 345
Title property, 668, 970
titles
apps, 8
Share menu, 632
thumbnail pages, 668
toast notifications, 453, 946
app icons, customizing, 467
displaying, 467
overview, 463-464
receiving, 464-465
sending, 465-468
stock ticker app, 489
todo list app
hiding/displaying elements, 940-942
testing, 932
todo items, 928
adding, 929
AppBar button, 935
changes, updating, 941
creating, 927
CRUD operations, 929-930, 935
deleting, 939
displaying, 933-935
editing, 938-941
grouping, 931
ids, 929
IoC container, 940
loading, 931
navigating back to TodoItemList page, 937-938
retrieving, 928-930
saving, 937
shell tiles, 936-937, 942-944
TodoListViewModel, 932, 960
TodoDataContext class, 928-929
TodoItem class, 927-928
TodoItems property, 928
TodoItemView class, 935
TodoItemViewModel
AppBar button, 941
LoadItem method, 939
SaveItem method, 937-938
updating, 941
VisualState property, 940-942
TodoListViewModel class, 932, 960
TodoListView.xaml, 933-935
TodoService class, 929-930
    AddOrUpdate method, 929
    GetTodoItems method, 930
    Id method, 929
TodoTileDataCreator class, 936
toggle buttons, 69, 128-129
    adding, 312
    binding to viewmodel property, 312-313
    checked/unchecked state, 128
    colors, 314
    events, 129, 312
    header, 312
    indeterminate state, setting, 128
    localizing, 313-314
    overview, 311
    state, 312
ToggleButton class, 128-129
ToggleCalibrationCommand, 517
ToggleCapturingVideo method, 673
ToggleOn property, 312
ToggleRecording method, 693
ToggleSwitchViewModel class, 312-313
toggling AppBar buttons, 236
tokens, extracting, 620
tombstoning, 62
tool tips, 140
ToolTip class, 140
TotalAngleDelta property, 376
TotalManipulation property, 362
touch
    buttons, 125
design
    components, 382
garbage collection, 37
    guidelines, 383
    inertia, 361
    listing of, 359
    promotion, 357-358
    sizing/spacing constraints, 383
    stopping, 35
    storyboards, 37
    swing, 361
events
    Deep Zoom images, 211
    Double Tap, 212-213
    drawing surfaces, 190-191
    Mouse Events, 354
    mouse events, 354
    Mouse Scroll, 196
    Pivot/Panorama controls, 322
    promotion, 357-358
frames, 356
garbage collection, 37
image loads, 37
garbage collection, 37
launching, 34
Memory Usage MB graph, 37
overview, 34
results, viewing, 35
running, 34
stopping, 35
storyboards, 37
SDK, 2
Silverlight Toolkit, 113-114
SqlMetal, 878
WPConnect, 641-643
ToolTip class, 140
TouchDelta property, 376
TotalManipulation property, 362
touch
    buttons, 125
design
    components, 382
garbage collection, 37
    guidelines, 383
    inertia, 361
    listing of, 359
    promotion, 357-358
    sizing/spacing constraints, 383
    stopping, 35
    storyboards, 37
    swing, 361
events
    Deep Zoom images, 211
    Double Tap, 212-213
    drawing surfaces, 190-191
    Mouse Events, 354
    mouse events, 354
    Mouse Scroll, 196
    Pivot/Panorama controls, 322
    promotion, 357-358
frames, 356
tool tips, 140
the toolkit, 249-250
tools
    Application Deployment, 25
    Code Contracts, 50
    Expresso, 816
    Font Preview, 166
    frame rate counter, 31-33
    GoldWave, 203
    Marketplace Test Kit, 28
    Performance Analysis
        CPU Usage graph, 36-37
detailed profiling information, viewing, 37
    Frame Rate graph, 35
How can we make this index more useful? Email us at indexes@samspublishing.com
notifications, registering, 356
points, 354
Touch class, 356-359
TouchFrameEventArgs class, 357
TouchPoint class, 356
user interfaces, 382
Touch class, 356
  border color example, 358-359
  subscribing, 356
TouchDevice property, 356
TouchFrameEventArgs class, 357
TouchPoint class, 356
TouchPointView class, 358
Track button (AppBar), 566
Track property, 969
TrackArtist method, 979
TrackCommand property, 565
tracking locations, 564-566
  AppBar Track button, 566
  starting, 565
  user confirmation prompt, 565
tracks
  audio player
    numbers, managing, 973
    play state, toggling, 973
    state changes, responding, 974-975
  background audio
    displaying on foreground app page, 981
    foreground control page, displaying, 979
    information, 970
    playback methods, 969
TrackTitle property, 979
trailing slashes (images), 186
transfer requests (background)
  app terminations, handling, 956-957
  BackgroundTransferRequest class, 952
  backing up local databases, 963
  copying files to temporary directory, 960
  current state, identifying, 955
  deleting, 952
  existing, retrieving, 956
  local databases, backing up, 960-962
progress, monitoring, 954, 961-962
queue limits, 952
requirements, 952
restoring local databases, 963-965
re-subscribing, 956
results, displaying, 962
status changes, monitoring, 954
submitting, 952
todo list viewmodel example, 960
upload, 953, 961
URL rerouting, 957-958, 961
Windows Live anonymous IDs, retrieving, 958-960
TransferPreferences property, 952
TransferProgressChanged event, 954
TransferStatus property, 955
TransferStatusChanged event, 954
transient state
  requirements, 64
  restoring, 64-65
  saving, 63-64
  storing, 59
TransitionPageStyle, 115
transitions (animated), 112
  adding, 113-114
  navigation events, 112
  style, 115-116
Translate property, 361
Trial property, 741
trial versions (apps)
  application bar button, displaying, 741-743
  benefits, 741
  mock licensing service, 741
troubleshooting. See also debugging
  AutoCompleteBox control suggestion list
  bug, 256
  progress bar performance, 146
TrueHeading property, 510
TryStart method, 537
tweets, displaying, 868-872
  cached timeline, retrieving, 869-870
  controls, 871-872
results, handling, 870
screen name verification, 869
status updates, retrieving, 868
Twitter timeline viewer, 846
Association attribute, 853
cached data, retrieving, 863-864
change notifications, 851
column properties, 848
  AutoSync, 849
  CanBeNull, 849
  DbType, 849
  Expression, 850
  IsDbGenerated, 850
  IsDescriminator, 850
  IsPrimaryKey, 849
  IsVersion, 850
  UpdateCheck, 850
connection strings, 856-859
data model, 846
encryption, 858
enum value conversions, 859
initialization, 856
isolated storage files, creating, 855-856
index, adding, 891
ITwitterService interface, 860
LINQ to SQL logging output, 887
navigating to TimelineView page, 885
populating, 861
retrieving users from database, 860
sign-in view, 864-867
tables
  adding, 889
  names, 848
TimelineItems, 846-848, 892
tweets, viewing, 868-872
  cached timeline, retrieving, 869-870
  controls, 871-872
  results, handling, 870
  screen name verification, 869
  status updates, retrieving, 868
Twitter web API queries, 862-863
  TwitterDatabaseUtility class, 855
  TwitterDataContext class, 854
  TwitterUser class, 851-853, 892
  user details/status updates, 860
TwitterDatabaseUtility class, 855
TwitterDataContext class, 854
TwitterFriend class, 889
TwitterService class
  GetCachedTimeline method, 863-864
  GetTimelineCore method, 862-863
  instantiating, 860
TwitterSignInViewModel, 865-866
TwitterTimelineViewModel, 868-872
TwitterUser class, 851-853
Type property, 924
  controls, 118
  OData queries, 800-802
  push notifications, 453
  scheduled task agents, determining, 925
TypeTemplateSelector class, 341

U

UI elements. See user interfaces
UIElementRenderer class, 685
UnBindToShellTile method, 459
UnBindToShellToast method, 459
Unchecked event, 129
Uniform Resource Identifiers. See URIs
Uniform Resource Locators. See URLs
union (+) symbol, 718
unit testing
  assemblies, 727
  assertions, 725-726
  asynchronous, 736-737
  attributes
    AssemblyCleanup, 720
    AssemblyInitialize, 719
    Assembly, 723
    Bug, 723

How can we make this index more useful? Email us at indexes@samspublishing.com
ClassCleanup, 720
ClassInitialize, 720
Description, 722
ExpectedException, 723
Ignore, 721
Owner, 722
Priority, 724
TestClass, 718
TestCleanup, 721
TestInitialize, 721
TestMethod, 719
TestProperty, 721
Timeout, 722
benefits, 709
chat client, 730-732
classes, 714-716
disadvantage, 709
downloading, 712
extension order, 719
epressions editor, 726-727
foreground control page of background audio, 978
harness, 715
inheritance, 724-725
launchers/choosers, 743-744
non-public members, 727
overview, 710
projects, creating, 712-714
results, displaying, 715
running, 715
tag expressions, 717-718
Unit Testing Framework. See UTF
UnitTestSettings class, 718
Universal Volume Control (UVC), 975
Unlink event, 306
UploadedPayloadItem event, 329
unlocking media databases, 641-643
unmuting media playback, 197
unregistering push notifications, 492-493
Update event, 685
Update method, 16
UpdateCheck property, 850
UpdateCommands method
CaptureSourceViewModel, 671
MicrophoneViewModel, 694
UpdateFrequencyRanges method, 704-705
UpdateInterval property, 682
updates
accelerometer, 499
background audio foreground control page, 979
compass, 509
game loops, 682
gyroscope, 519
motion sensor, 524
progress bars, 149, 153
scheduled task changes, 941
schema, 887
columns, adding, 888
indexes, adding, 890-891
one-to-many associations, adding, 891-893
tables, adding, 889-890
shell tiles, 942-944
sensor, 496
status, 423
tile notifications with shell tile schedule, 473-474
user interfaces, 600-601
UpdateSource method, 760
UpdateSourceTrigger property, 760
UpdateSourceTriggerExtender class, 757-759
UploadCommand, 635-636
UploadImage method, 637
uploading
background transfer requests, 953
files to servers, 961
photos
certification requirement, 634
converting images to byte arrays, 638-639
creating, 633-641
displaying images, 634, 639-641
dummy images, 633
emulator, 633-635
retrieving images, 633-634
sending images, 635-637
service references, adding, 637
UploadLocation property, 953
Uri property, 79-80
UriMapper property, 76
UriMapping class, 76
URIs (Uniform Resource Identifiers)
images, 186-187
internal navigation, 71-72
mapping, 75-77
OData, 798-799
push notifications, 455
retrieving, 77
URLs (Uniform Resource Locators)
images, 187
rerouting, 957-958, 961
user input
drawing surfaces, 190-191
keyboards
events, 41
hardware, 169
SIP layouts, 169
manipulation events, 359-363
arguments, 360
inertia, 361
listing of, 359
moving and scaling UIElement example, 362-363
sequence, 360-362
mouse events, 354
border background color, changing, 354-355
listing of, 354
promotion, 357-358
notifications, registering, 356
points, 354
property setter validation, 749
binding errors, 751-752
critical exceptions, 751
data binding steps, 749-751
enabling, 749
limitations, 765
source property assignment, 749
Validation class, 751
scope, 171-172
default view, 176
enum values, retrieving, 173
element, 173-176
InputScopeNameValuePair enum values, 171
intellisense support, 172
ListPicker control view, 176
selected items example view, 176
text entry example view, 176
values, converting, 174-175
word prediction, 172
sensors
accelerometer, 497-506
accuracy, improving, 522
compass, 508-517
data acquisition, starting/stopping, 496
data validity, checking, 496
device support, 523
gyroscope, 518-521
intervals between readings, 496
motion, 522-526
overview, 495-497
resources, releasing, 497
shake detection, 507-508
updates, receiving, 496
values, holding, 496
touch
design, 382-383
frames, 356
gestures. See gestures
mouse events, 354-358
notifications, registering, 356
points, 354
user interfaces, 382
user interfaces, 382
validation
asynchronous. See asynchronous validation

How can we make this index more useful? Email us at indexes@samspublishing.com
data context changes, detecting, 782-785
DataValidationError class, 769
decoupling, 772
defined, 748
decoupling, 772
errors, displaying, 752-757, 762-765
forms, 775-779
groups, 760-762, 786-789
property changes, 772-775
semantic, 748
style locations, defining, 753
synchronous, 769-770
syntactic, 748
text boxes as user types, 757-760
XNA gestures, processing, 688-690

user interfaces
  animating upon orientation changes, 108-109
  child, positioning
    child element spacing, 315
    layout modes, 315
    list box example, 317-318
    overview, 315
    vertical/horizontal sample code, 315-317
coded UI testing, 711, 734-736
development languages supported, 3
embedding elements, 181
gestures. See gestures
moving and scaling, 362-363
mouse events, 354
runtime language changes, 603
cultures, 604-606
data binding, 611
date/time/currency formatting properties, 607
localized text/images, displaying, 607-610
right to left support, 612
Visual Studio resource properties, selecting, 610
runtime manipulation, 737-738
Silverlight versus XNA, 2-4
sizing based on page orientation, 102
threads, 29, 33
touch friendly, 382-383
  components, 382
guidelines, 383
  sizing/spacing constraints, 383
updating, 600-601
visibility, 31
UserAction enum values, 977
UserExtendedProperties class, 958, 960
UserPrompted property, 66
UseSprings property, 209
using statements, placement, 10
UTF (Unit Testing Framework)
  assemblies, 727
  assertions, 725-726
  asynchronous testing, 736-737
attributes
  AssemblyCleanup, 720
  AssemblyInitialize, 719
  Asynchronous, 723
  Bug, 723
  ClassCleanup, 720
  ClassInitialize, 720
  Description, 722
  ExpectedException, 723
  Ignore, 721
  Owner, 722
  Priority, 724
  TestClass, 718
  TestCleanup, 721
  TestInitialize, 721
  TestMethod, 719
  TestProperty, 721
  Timeout, 722
benefits, 709
chat client, 730-732
classes, 714-716
disadvantage, 709
downloading, 709
execution order, 719
V

ValidatesOnExceptions property, 752
validation
  arguments, 50-51
  asynchronous, 770-772
    all properties at once, 775-779
    completing, 781
    DataErrorNotifier class, 767
    errors, creating, 779
    properties, registering, 768-769
    sending email example, 786-789
    validation activities, 781
  whitespace string properties
    example, 780
decoupling, 772
errors
  creating, 779
  DataValidationError class, 769
  details, displaying, 762-765
  displaying, 752-757
  list, retrieving, 785-786
forms, 760, 775-779
  completing, 777-779
  evaluating, 777
  property list, creating, 775-776
  group, 786-789
  input, 748
validation activities, 781
validation
  arguments, 50-51
  asynchronous, 770-772
    all properties at once, 775-779
    completing, 781
    DataErrorNotifier class, 767
    errors, creating, 779
    properties, registering, 768-769
    sending email example, 786-789
    validation activities, 781
  whitespace string properties
    example, 780
decoupling, 772
errors
  creating, 779
  DataValidationError class, 769
  details, displaying, 762-765
  displaying, 752-757
  list, retrieving, 785-786
forms, 760, 775-779
  completing, 777-779
  evaluating, 777
  property list, creating, 775-776
  group, 786-789
  input, 748

How can we make this index more useful? Email us at indexes@samspublishing.com
filter mode, 257-259
InputScopeNameValue enum, 171
sensors, 496
ValueStringFormat property, 277
Velocities property, 361
VerticalAlignment property, 123
VerticalChange property, 372
VerticalOffset property, 813
VerticalVelocity property, 373
video
capturing, 669
audio formats, 670
CaptureSource class, 669
image collections, 672
image formats, 670
playing, 674-676
resolution settings, 670
saving, 671
starting/stopping, 673-674
user authorization, 669
grayscale effect
ARGB color component, extracting, 661
converting back to color, 661
grayscale conversion, 661
previewing, 659-660
processing, 659
result, 661
turning on/off, 658
media viewer page sample code, 196-202
AppBar control, 201-202
binding properties to viewmodel, 199
MediaView class, 197-199
muting/unmuting playback, 197
position control, 199
scroll viewer, 200-201
video, playing, 200
MediaPlayer, 195
playback controls, 196
playing, 195, 200, 407-411
sources, 196
streaming, 196
view-first MVVM creation, 43
ViewImageCommand, 664
viewing. See displaying
ViewModelBase class, 43
AddValidationProperty method, 768
BeginValidation method, 770-772
decoupling, 772
GetPropertyErrors method, 769-770
PropertyChanged event, 772-775
RegisterStatefulProperty method, 836-837
SaveState method, 828
state methods, 827
validation, 767
viewmodel-first MVVM creation, 43
viewmodels
AccelerometerViewModel, 503-504
AddressChooserTaskViewModel, 426
AlarmViewModel, 910-911
AppBarViewModel, 234-236
AppointmentsViewModel, 448-449
AutoCompleteBox control, 256
alternate filter mode, 260
filter modes, 257-260
two-way data binding, 259
BingMapsDirectionsTaskViewModel, 389-390
BingMapsViewModel, 564-565
CameraCaptureViewModel, 429
CaptureSourceViewModel
commands, 671
HandleCaptureImageCompleted method, 672
PlayVideo method, 674
Start method, 670
TakePhoto method, 671
ToggleCapturingVideo method, 673
UpdateCommands method, 671
ChatClientViewModel
code listing, 729-730
test methods, 730-732
view elements, 732-734
check boxes, binding, 134-137
CompassViewModel, 510
    calibration, 516
    properties, 513
    Start method, 509, 514
ContactsViewModel
    results, displaying, 444-445
    retrieving contacts, 442-444
CustomDatePickerView
    constructor, 280
    holding dates, 279
    list of dates, 279
    retrieving values, 281-282
EbaySearchViewModel, 806-809
    commands, 806
    HandleEbayClientSearchItemsComplete method, 808-809
    Search method, 807-808
    SearchCount property, 827
EmailViewModel, 395
FlatListViewModel, 288-289
FMRadioViewModel
    Frequency property, 701
    PoweredOn property, 700-701
    Region property, 702
    Regions property, 702
    SignalStrength property, 705
    UpdateFrequencyRanges, 704-705
GamePageViewModel, 683
GeoPositionViewModel, 541
    DelegateCommands, 543
    Start method, 542
GyroscopeViewModel, 519
InkPresenterView, 194
InkPresenterViewModel, 193-194
input scope, 173-176
    default view, 176
    enum values, retrieving, 173
    ListPicker control view, 176
    selected items view, 176
text entry view, 176
     values, converting, 174-175
LaunchWebBrowserViewModel, 425
LocalizabilityViewModel
    CultureName property, 606
    SetCulture method, 605
    SetCurrentCultureValues method, 607
    SupportedCultures property, 605
LockScreenViewModel class, 68-69
LongListSelector, 299-303
MainPageViewModel, 979-982
    PlayerState property, 980
    Position property, 982-983
    Refresh method, 979
    TrackArtist/TrackTitle properties, 979
    ViewState property, 981
MarketplaceApp, 292
MarketplaceViewModel
    DetailCommand, 402
    hubCommand, 404
    reviewCommand, 405
    searchCommand, 406
MediaPlayerImageViewModel
    LoadImage method, 667
    Title property, 668
MediaPlayerLauncherViewModel, 409-411
MessagesViewModel, 334-336
MicrophoneViewModel
    code listing, 693
    commands, 693
    HandleBufferReady method, 694
    StartRecording method, 694
    StopRecording method, 694
    ToggleRecording method, 693
    UpdateCommands method, 694
PerformanceProgressBarViewModel, 307-308
PhoneCallViewModel
    phone calls, placing, 414
    phone numbers, selecting, 416
How can we make this index more useful? Email us at indexes@samspublishing.com
PhotoCameraViewModel
  code listing, 648-649
  ConvertArgbToColor method, 661
  ConvertColorToArgb method, 661
  ConvertColorToGrayScale method, 661
  EffectEnabled property, 658
  HandleCaptureThumbnailAvailable method, 653
  SaveImage method, 652
  SetEffectState method, 658-659
  Stop method, 653
  VisualState property, 651, 657

PivotViewModel, 331
  populating pivots with data bound collections, 340-343
    DataboundPivotViewModel class, 340
    DataBoundPivotView.xaml, 341
    MessageViewModelTemplate, 342
    SendMessageViewModelTemplate, 342
    TypeTemplateSelector class, 341
  ProductDetailsViewModel class, 91-93
  ProductsViewModel class, 86-87
  progress bar example, 149-152
  PushNotificationViewModel, 477
  PushpinViewModel
    addPushpinCommand, 572
    code listing, 568
    displaying, 570
    properties, 569
    radio button controls, binding, 130-133
  ReminderViewModel, 914
    commands, 914
    LoadReminder method, 915
    SetReminder method, 914
  SaveEmailAddressViewModel, 399
  SearchViewModel, 421
  SendMessageViewModel, 333-334
  state
    loading, 830
    restoring, 829
    saving, 828-830

ThumbnailsViewModel
  Images property, 664
  Populate method, 663
  ViewImageCommand, 664

TodoItemViewModel
  AppBar button, 941
  LoadItem method, 939
  SaveItem method, 937-938
  updating, 941
  VisualState property, 940-942

TodoListViewModel, 932, 960
  ToggleSwitchViewModel, 312-313
  TwitterSignInViewModel, 865-866
  TwitterTimelineViewModel, 868-872

ViewModelBase
  RegisterStatefulProperty method, 836-837
  SaveState method, 828
  state methods, 827
  validation, 767

WrapPanelViewModel, 318

ViewportOrigin property, 209

ViewportSize property, 154

ViewportWidth property, 209

virtual machines, 7

visibility
  AppBar buttons/menu items, 234
  application bar buttons/menu items, 245
  background audio application bar buttons, 981
  Bing logo/copyright text, 560
  buttons, 123
  full-screen mode, 239-241
  full-screen picker pages, customizing, 278-282
  itineraries, 593
  map routes/itineraries, 586-590
  PerformanceProgressBar, 307-308
  Silverlight graphics/animations, 31

Visibility property, 31
visual states
  background audio foreground app page, 980
  validation errors, 752-757
Visual Studio
  hybrid project templates, 678-679
  resource properties, selecting, 610
  SQL CE files, opening, 876
VisualState property, 587
  PhotoCameraViewModel, 651, 657
  TodoItemViewModel, 940-942
VisualStateUtility class, 588
volume
  audio, 196
  background audio playback, 969
Volume property, 196, 969
W
Wait method, 541
WaitOne method, 948
WalkPath method, 539-540
waypoints, 581
WayPoints property, 539
WCF service
  BookshopService, 97-99
  BookshopService class, 97
  ProductManager class, 98-99
  creating, 636
  URL routing, 957-958
web, 215
  capability, 27
  content, hosting, 73
  cross-site restrictions, 221
  navigation, 215-216
  offline browsing, 221-226
  pages
    behaviors, adding, 219-221
    communication, 216-219
    navigating to, 424-425
  services, 636
  sources, 215
  searching, 420-421
WebBrowserIsolatedStorageView class, 223
WebBrowserTask, 424-425
WebBrowserWithScriptingView.xaml file, 218-219
webcam API, 669
  audio formats, 670
  CaptureSource class, 669
  CaptureSourceViewModel Start method, 670
  images
    capturing, 671
    collections, 672
    formats, 670
    playing videos, 674-676
    recording, starting/stopping, 673-674
    resolution settings, 670
    user authorization, 669
Webpage.html, 216-217
websites
  accelerometer blog article, 501
  alien spacecraft model, 684
  AutoCompleteBox custom template, 256
  Bing Maps, 554
  Calcium project, 233
  CalciumSDK repository, 55
  camera color conversion blog article, 660
  Code Contracts tool, 50
  CodePlex, 250
  color styles, 230
  culture codes, 857
d:DataContext markup extension, 94
  Deep Zoom Composer, 206
dependency property article, 813
eBay OData service documentation, 803
Expression Blend, 753
Expresso, 816
Geocode service, 574

How can we make this index more useful? Email us at indexes@samsprofessional.com
GoldWave, 203
Imagery, 574
input validation definition, 748
IoC container, 739
LINQ to SQL
logging output, 885
platform differences, 845
MediaStreamSource class, 985
MPNS response codes, 463
network connections article, 794
Nowak, Peter, 97
OData, 797-798
predefined styles, 12
RichTextBox control MSDN article, 182
Route service, 574
SDK download, 2
Search service, 574
Serializer, 834
shake detection article, 507
Slider control, 154
source property assignment, 749
Toolkit download, 250
UTF download, 712
WP7 Isolated Storage Explorer, 875
Yahoo! stock data format, 488
Wi-Fi
settings dialog, 393
triangulation, 531
Wilcox, Jeff, 712
Windows
Live anonymous IDs, retrieving, 958-960
Mobile 6.5 compatibility, 1
Phone 3D Graphics Application
template, 678
words
completing, 256
predicting, 172
WorkerReportsUpdates property, 152
WP 7 Isolated Storage Explorer, 875-876
WPConnect tool, 641-643
WrapPanel control
child element spacing, 315
layout modes, 315
list box example, 317-318
overview, 315
vertical/horizontal sample code, 315-317
WrapPanelViewModel class, 318
WrapPanelView.xaml, 315-317
X–Y–Z
XAML design view (Silverlight apps), 6
XAP files
Application Deployment tool, 25
contents, 24
overview, 24
size, 25
XBOX Live capability, 26
XNA
3D model, displaying, 683-684
combining with Silverlight. See hybrid apps
ccontent manager, initializing, 682-683
development languages supported, 3
evaluation, 13-16
fonts, 17-20
official, 18
Sprite Fonts, adding, 17
game loops, 16, 681-682
Game template, 13
game Thumbnail, 13
gestures, processing, 688-690
Microphone class, 691-692
new projects, creating, 12
overview, 2
properties, 13
rendering, 679
Silverlight, compared, 2-4
SoundEffect class, 202-204
startup types, 13
Windows Phone Rich Graphics Application
template, 678
Yahoo! stock data format, 488
Yaw property, 524
ZoomAboutLogicalPoint property, 209
ZoomBarVisibility property, 563-564
zooming (Bing Maps), 563
  buttons, 563-564
  slider, 562
ZoomLevel property, 562
Zune software, 641-643

How can we make this index more useful? Email us at indexes@samspublishing.com