DZone Refcardz

CONTENTS INCLUDE:

▪ Platform & Tools
▪ Silverlight Project Creation
▪ Application Navigation
▪ Data Handling
▪ Platform Integration
▪ Design Considerations and more...

Developing a Silverlight Application for

Windows® Phone 7

By Colin Melia

PLATFORM & TOOLS

Windows® Phone 7 is a new platform bringing together developers from the .NET, Silverlight, and XNA eco-systems. This document combines reference information and tips for developing and publishing a Silverlight application for the platform using Visual Studio.

Hardware Specification

All initially launched Windows Phone 7 devices come with: the same operating system; the same minimum set of built-in applications and platform capabilities; and the ability to run Silverlight and XNA applications. Additionally, all devices come with the following hardware capabilities:

- 480x800 display
- DirectX 9 hardware acceleration
- Capacitive 4-point multi-touch
- Back, Start & Search hardware buttons (as well as a power, camera and volume button)
- A-GPS, accelerometer, compass*, light*, proximity*
- 5 MP+ digital camera with flash
- 256MB+ RAM
- 8GB+ flash storage

* not accesible from .NET API in initial release

Hardware manufacturers may differentiate their devices by items such as: having higher RAM, flash storage or camera mega-pixels; having different screen sizes (but same resolution); adding a slide-out keyboard (horizontal or vertical); adding a single additional theme accent color; bundling additional built-in applications—but not removing/replacing built-in applications/UI.

> **Hot Tip**
> Flash storage space should be considered as opaque file system storage; it is not removable portable storage for transferring files. It can be extended in some devices with an additional card, but that requires a 'factory reset' of the device to wipie existing data.

Developer Tools

The developer tools for Windows Phone 7 are available for free. Go to http://go.microsoft.com/fwlink/?LinkID=189554 for the RTW ('Released to the Web') tools or go to http://create.msdn.com to find the latest set.

The developer tools include the following items in one package:

1. Visual Studio 2010 Express for Windows Phone*
2. XNA Game Studio 4.0*
3. Microsoft Expression Blend for Windows Phone*
4. Windows Phone Emulator Resources
5. Silverlight 4 Tools For Visual Studio

* Items 1 & 2 will install as standalone applications if Visual Studio 2010 Professional or higher is not installed, otherwise they will install additional project templates in Visual Studio. The same goes for item 3 with regard to Expression Blend 4.

To install the RTW version of the tools, one of the following is required:

- Windows® Vista® (x86 and x64) with Service Pack 2–all editions except Starter Edition
- Windows® 7 (x86 and x64)–all editions except Starter Edition

Be sure to check out the release notes for full requirements.
http://go.microsoft.com/fwlink/?LinkID=190374

> **Hot Tip**
> Before installing the RTW version of the tools, be sure to uninstall any pre-release version of the toolset in one step by uninstalling the item named, "Microsoft Windows Phone Developer Tools..." under Control Panel.

Silverlight Application Platform

The Windows Phone 7 platform can run applications based on Silverlight and XNA.

Silverlight is often the preferred choice for applications built with controls and events, similar to WinForms or WebForms. XNA is typically used for higher performance UI hardware rendering in 2D and 3D.

> **Hot Tip**
> Silverlight applications can use non-rendering classes provided by the XNA framework (e.g. for playing overlapping sounds or accessing microphone audio) and vice-versa. For choosing a development stack and calling across frameworks, see http://msdn.microsoft.com/en-us/library/ff402528(v=VS.92).aspx.

Silverlight applications run directly on the Windows Phone platform in a mode similar to the out-of-browser capability of Silverlight 3+ on the desktop. They do not run in a browser. In fact, Silverlight content on Web pages will not appear in the phone's Web browser, but there is a WebBrowser control to host HTML content (from the Web, injected, or from local storage) inside a Silverlight application.

The RTW Silverlight platform is based largely on Silverlight 3.

Windows® Phone
7
↓ Get the free tools
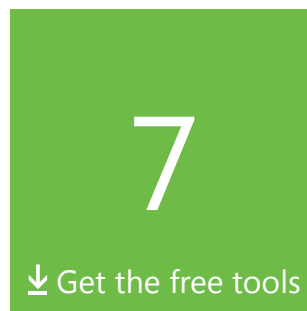
A notable Silverlight 3 feature that is missing is TCP socket network access. A notable Silverlight 4 feature that is included is multi-touch manipulation events and offline DRM.

For general information on Silverlight support on the phone, see the MSDN library section:
http://msdn.microsoft.com/en-us/library/ff426934(v=VS.95).aspx

> **Hot Tip**
>
> To see the extent to which a specific Silverlight class is supported on Windows Phone, check out any of the class pages within the namespace under .NET Class library for Silverlight (http://msdn.microsoft.com/en-us/library/cc838194(v=vs.95).aspx) and look for the 📞 symbol next to class members.

The phone platform includes a set of classes specific to Windows Phone: http://msdn.microsoft.com/en-us/library/ff626516(v=VS.92).aspx.

SILVERLIGHT PROJECT CREATION

Project Templates

Start building a Silverlight application for Windows Phone by using the File -> New Project feature in Visual Studio. The RTW tools include support for projects based on the C# language. Support for VB.NET will be officially introduced later on.

Project Template Type	Likely Used When…
Windows Phone Application	Starting with a basic one-page/screen application.
Windows Phone Databound Application	Starting with a one-page/screen list-based application and including MVVM (Model-View-ViewModel) design.
Windows Phone Class Library	Building a component or control to reuse in other projects.
Windows Phone Panorama Application*	Building an application that is based on an MVM design and that starts with a rich and wide content-based starting screen like the People, Music + Videos, Pictures, Office, Xbox Live, and Marketplace."
Windows Phone Pivot Application*	Building an application that is based on an MVM design and that starts with a screen showing set of views (or pivots) over data like the Calendar & email application on the phone.

* These project templates use the Panorama and Pivot controls on a single page and are often used to create 'Hub' applications; these controls can also be added to any page in an application.

> **Hot Tip**
>
> A 'Hub' is an application that typically starts with a Panorama or Pivot control and that generally brings together content and data under a single area of concern (e.g., People).

Design Surface + XAML

After choosing to create a new project based on the Windows Phone Application template, an interface is shown that includes windows for the Toolbox, Solution Explorer, and Properties, along with the editor window for the initial (and perhaps only) application page named MainPage.xaml.

The editor window is split between a preview design surface and Extensible Application Markup Language (XAML) editor. Drag and drop controls from the toolbox to either the design surface or the XAML text. By selecting items on the design surface or XAML text (which synchronize their selections), properties can be changed in the Properties window.

XAML is based on XML and is parsed at runtime to instantiate objects, creating a way to perform declarative programming.

The XAML used in Silverlight represents a tree of objects largely used to form the tree of visual controls in the running application. It can also be used to instantiate reusable resource objects and class instance objects.
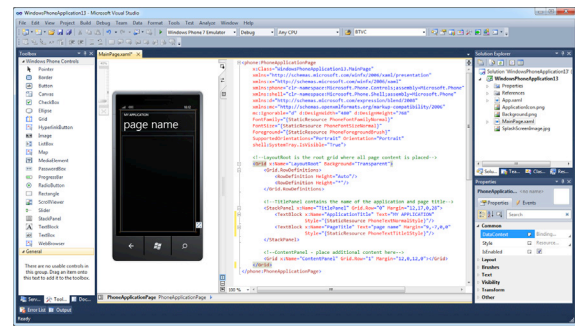

```xml
<phone:PhoneApplicationPage
    x:Class="WindowsPhoneApplication13.MainPage"
① xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
⑦ xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
② <Grid x:Name="LayoutRoot" Background="Transparent"> ③
        <Grid.RowDefinitions>
⑤          <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
                                                ⑥
        <!--TitlePanel contains the name of the application and page title-->
④      <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
                Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
                Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>  ⑨

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>  ⑧
</phone:PhoneApplicationPage>
```

Key elements of the XAML syntax are:

1. **Default namespace:** XML feature that defines the set of elements (and their attributes) that can be used in the document. At runtime, these elements are matched to classes in a .NET assembly.

2. **Element:** using an XML element means an object with a type named the same as the element will be created at runtime.

3. **Attribute:** at runtime, the value of an attribute will be set as the value of a property on the object created for the corresponding element. The XAML parser is also able to convert text to non-text types including a numerical type, an enumerated type, multiple initial values on a type (e.g. the different sides of a Margin), or a whole object initialized by the attribute values (e.g. a SolidColorBrush with a specific color).

4. **Nesting:** putting one element inside another means the created objects form a tree whereby the nested object is assigned to a default property (e.g., a single Child or Content property of the containing object), or it becomes one of several objects added to a default collection property (e.g., a Children collection property).

5. **Property Element:** when a property of an object is too complex to declare or parse from an XML text attribute, the Property Element syntax is used to declare the object fully.

6. **Attached Properties:** this allows a containing object to declare properties against each child object without the class of each child object needing to have members (e.g. a set of Button objects could be placed in a radial layout control which visually lays out the buttons in a circle; that layout control needs each Button to have an angle property value).

7. **Extensibility:** to introduce elements that become types from other .NET assemblies (e.g., other controls or classes), use this kind of XML namespace declaration.

8. **x:Name:** similar to ASP.NET ID attribute; to access the instantiated objects from procedure code, give them a name.

9. **Evaluated:** The {} syntax is used to evaluate value at runtime, e.g., accessing resources or binding to data.

You can (and may need to in some cases) use procedural code in the MainPage.xaml.cs file to build UI and populate it with data.

However, XAML provides a more efficient way to declare visual interfaces rather than writing procedural code that can be substantially longer. Procedural code is still used for data models and accessing data. XAML is also compatible with the Expression Blend designer tool allowing for projects to be edited in both tools with tool UI appropriate to each user.

> **Hot Tip**
> Adhering to the MVVM design pattern can maximize XAML and minimize UI code, increasing designer-developer workflow productivity and creating more maintainable code layers (from separation of concerns).

Controls

The following controls are provided on the phone:

Category	Control Class
Do Action	Button, HyperlinkButton
Data Display	Map, ProgressBar, TextBlock, WebBrowser
Data Input	CheckBox, PasswordBox, RadioButton, Slider, TextBox
Media	MediaElement
Drawing	Ellipse, Rectangle
Layout	Border, Canvas, Grid, ListBox, ScrollViewer, StackPanel, Panorama, Pivot

Many controls have a property named Content. Such controls (e.g., Border or even Button) act as visual/object container for rendering a single item of content which can be either:

- visual (derived from UIElement), e.g., another Control
- an object containing data (in which case the data is visually represented as text by calling ToString() or with other visual elements declared in a DataTemplate)

> **Hot Tip**
> Microsoft is also releasing other controls out-of-band from (and likely more frequently than) developer tools updates, in the Silverlight for Windows Phone Toolkit on Codeplex: http://silverlight.codeplex.com/

Layout

The desired UI layout is achieved by setting Margin and Padding on controls, setting their alignment in the space provided to them, and placing them in other controls (e.g., Grid, Canvas, or StackPanel) that arrange one or more controls.

> **Hot Tip**
> Try customizing controls by providing XAML for 'Template' properties. For custom child layout, derive a class from the Panel class and override the MeasureOverride() and ArrangeOverride() methods.

Layout of controls on the design surface is not limited to the physical dimensions of the device. A control can extend passed the dimensions of the design surface if the user is able to reach that UI through some means of interaction. The Panorama and Pivot controls are 'über-layout' controls that do exactly this.

Emulator

The toolset includes a powerful emulator running the Windows Phone OS 7.0 compiled for Intel processors. It emulates a device's processor, RAM, display, GPU, networking, media, keyboard, orientation changes, as well as application deployment and debugging. The PC mouse pointer is used to emulate a single screen touch. If the PC supports multi-touch, then this is passed through to the emulator.

> **Hot Tip**
> Press the Pause key on the PC keyboard to toggle keyboard input between the onscreen keyboard and the PC's keyboard (emulating a physical keyboard on a device).

The emulator starts automatically when a debug session is started in Visual Studio (if the Windows Phone 7 Emulator is selected).

> **Hot Tip**
> The emulator can take advantage of hardware virtualization if turned on in the PC BIOS as well as the PC's GPU - http://msdn.microsoft.com/en-us/library/ff402567(v=VS.92).aspx.

The emulator does not emulate A-GPS, light, proximity, and accelerometer sensors. It also does not include all user Settings pages.

> **Hot Tip**
> Use mock classes or the Reactive extensions for .NET (http://msdn.microsoft.com/en-us/library/ff431792(v=VS.92).aspx) to simulate streams of data from hardware that is not emulated.

> **Hot Tip**
> To take 480x800 snapshots of the emulator for use in marketplace application submissions, on Windows 7, use the Snipping Tool, select New -> Windows Snip and click on the emulator screen.

Debugging

Visual Studio allows debugging targeting to the Emulator or a connected device for debugging.

Regular debugging activities are available including using breakpoints, variable inspection, variable watch, call stack, and output window.

Launching a debug instance (e.g., by pressing F5) builds the project (as necessary), deploys the application to the target device (removing any previous version,) and then launches the application.

To debug with a real device, all of these points must be met:

1. The Zune (4.7+) software (free to download from http://zune.net) must be running on the development PC.
2. The device must be connected to the PC using the provided USB cable.
3. The device must be 'development unlocked' (not the same as 'network unlocked'); this is achieved by using the Windows Phone Developer Registration Tool (included with the tools) and having a developer account at http://create.msdn.com with a spare slot (out of 3 total for paying accounts; 1 for student accounts).
4. The device must be on and not on the lock screen.

When using the emulator, leave it running between debug sessions to avoid the start-up time and to keep the storage emulation intact and be able to launch applications deployed/debugged since it was started.

> **Hot Tip**
> The Zune software locks the media library on the phone. To debug an application that accesses the media library instead, use WPConnect.exe in the Windows Phone Developer Tools 2010 Update.

APPLICATION NAVIGATION

Frame & Page

Silverlight applications on the phone operate within a logical 'last-on-first-off' (LIFO) stack of running applications. Within an application, there is a logical LIFO stack of pages.

Other pages within an application (added with the 'Add New Item' menu item in Visual Studio) are launched by developed code, usually in response to user actions such as clicking a button.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/page2.xaml?param=" + value,
UriKind.Relative));
}
```

This example shows passing a parameter to the next page. During page navigation, the current page's OnNavigatedFrom() override is called and the destination page's OnNavigatedTo() override is called.

```
protected override void OnNavigatedTo(System.Windows.Navigation.
NavigationEventArgs e)
{
    var value = NavigationContext.QueryString["param"];
}
```

In this case, the query string passed earlier is being extracted. The user launches instances of an application using the Search or Start screen 'on top', which sit logically higher on the sack.

> **Hot Tip**
> If the user launches another instance of the same application, any previously launched instance (and any tombstone information – see later) is effectively removed from the stack.

Back Button

The phone's Back button removes the current page and navigates to the previous one on the stack, calling its OnNavigatedTo() override. It is possible to cancel this default behavior. In the example, the modal dialog box is cancelled and prompts the user to save unsaved data.

> **Hot Tip**
> Check marketplace submission requirements before overriding the Back button. To avoid complex parameter passing and to keep data 'alive' between pages, store data against the application or RootFrame DataContext, and for collections, pass an index parameter.

```
protected override void OnBackKeyPress(System.ComponentModel.
CancelEventArgs e)
{
    if (someallowedcondition)
        e.Cancel = true;
    base.OnBackKeyPress(e);
}
```

DATA HANDLING

Internet Data Access

There are three main methods provided for obtaining data from the Internet. All follow an asynchronous pattern.

Method	Usage
Client Proxy	Use Add Service Reference to create a strongly-typed proxy class on a Web Service or WCF service supporting wsBasicBinding.
WebClient class	Use to get (or send) text data (including arbitrary XML) or a file.
HttpWebRequest & HttpWebResponse	Use for fine-grained control of sending an HTTP request and receiving the response.

The typical pattern involves reacting to a user event (or timed event from System.Windows.Threading.DispatcherTimer) and setting up the data request by creating a class instance, setting up an event handler (or delegate), and initiating the asynchronous activity.

> **Hot Tip**
> Some asynchronous activities provide results on a different thread. If the application needs to directly modify the UI on receipt of data, use Dispatcher.BeginInvoke() to do it on the UI thread.

LINQ to XML

On the phone, LINQ allows set-based queries on in-memory objects and data transformation. This can include parsing text downloaded with WebClient and transforming it to objects.

LINQ to XML (by adding the System.Xml.Linq reference) allows parsing and transforming of XML data.

```
public class MyPerson
{
    public String Name { get; set; }
    public int Age { get; set; }
}
```

Code to parse XML downloaded using WebClient, into the class.

```
// Do something with e.Result string
XDocument doc = XDocument.Load(new StringReader(e.Result));

var results = from p in doc.Element("People")
              .Elements("Person")
              select new MyPerson()
              {
                  Name = p.Element("Name").Value,
                  Age = int.Parse(p.Attribute("Age").Value)
              };
```

Data Binding

Data can be set on UI controls using procedural code, but this can get quite complex and is not designer-tool friendly. Setting (in code or XAML) the DataContext of any UI object (inherited from FrameworkElement) to a 'data' object (e.g., of type MyPerson) 'binds' that data object to the UI object and allows the UI object (or descendants) to display properties of the data object using a simple XAML syntax.

```
public MainPage()
{
    InitializeComponent();

    ContentPanel.DataContext = new MyPerson();
}

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <TextBlock Text="{Binding Name}"/>
    <TextBlock Text="{Binding Age}"/>
</Grid>
```

Setting the DataContext of descendant UI objects to any object properties of the data object allows a hierarchy of binding.

> **Hot Tip**
> Use binding converters (classes based on IValueConverter) along with a resource declaration to provide for bindings between data object properties and UI object properties of different types or values (e.g., set a color based on a value). Use d:DataContext in XAML to set up design-time data in Visual Studio to preview sample data in the UI.

Isolated Storage

Isolated storage classes provide access to an area of storage on the device that is for the exclusive use of the calling application. This means applications cannot access data stored by other application (with only a few very special exceptions), and there is no file explorer for the end user.

Basic settings storage can be achieved like so:

```
// Store
IsolatedStorageSettings.ApplicationSettings["mysetting"] = person;

// Retrieve - Use Try/Catch or check first with ApplicationSettings.
Contains()
MyPerson person = (MyPerson)IsolatedStorageSettings.ApplicationSettings["m
ysetting"];
```

Alternatively, get a store object that allows file and directory operations. The CreateFile() and OpenFile() methods return back objects inherited from System.IO for use with Text, Binary, and XML Writers/Readers.

```
IsolatedStorageFile store = IsolatedStorageFile.
GetUserStoreForApplication();
```

> **Hot Tip**
> Isolated storage is best used for local settings, caching data, offline sync queues and data entry/activity in progress in some cases.

PLATFORM INTEGRATION

Application Lifecycle

While the OS maintains a history of navigation between applications and pages, applications (as a rule) terminate when the application goes into the background. Some examples of this are when the user launches other applications including the Start screen, a launcher/choose task is launched (see "Launchers & Choosers", or the screen locks. This provides the best user experience. Applications restart them at the last shown page when the user returns to the application using the Back button. The application developer must do the work to retain any UI or in-memory state by storing it either in isolated storage or with the 'bag' of data that the OS provides and retains in the stack history, known as the 'tombstone'. App.xaml.cs includes the stub for four application lifecycle events.

- The Launching event fires every time the application launches. Initiate loading any settings or cached data from Isolated Storage (and check the Internet for any new data if appropriate).
- The Deactivated event fires when the application is 'tombstoned', with ~10 seconds to save data into the tombstone or Isolated Storage before the application terminates.
- The Activated event fires when the application is resumed via the Back button. Load back data from the tombstone or Isolated Storage (and check the Internet for any new data if appropriate).
- The Closing event fires when the application exits from the initial back via the Back button.

Tombstone data can be stored under the global PhoneApplicationService.Current.State dictionary (during Deactivate/Activates events) or in the current page's State dictionary (during OnNavigatedFrom()/OnNavigatedTo()).

Note that the tombstone will be removed if the same application is launched again 'on top'—the user is abandoning any 'session' data stored in the tombstone—or if the OS needs to free up resources. In either case, the last two events will never be called on the previously running instance.

> **Hot Tip**
> To prevent the lock screen from engaging due to user inactivity, set PhoneApplicationService.Current. UserIdleDetectionMode = IdleDetectionMode.Disabled. To allow an application to run under the lock screen set PhoneApplicationService.Current. ApplicationIdleDetectionMode = IdleDetectionMode. Disabled (which has an associated marketplace submission requirement). Be careful not to drain the battery.

Launchers & Choosers

The Microsoft.Phone.Tasks namespace includes classes used to configure and prompt the user to perform tasks either with no resulting data returned (a Launcher) or with returned data (a Chooser), e.g.

```
private EmailAddressChooserTask task = null;
public MainPage()
{
    InitializeComponent();

    // These two things must be in the constructor
    task = new EmailAddressChooserTask();
    task.Completed += new EventHandler<EmailResult>(task_Completed);
}

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

private void button1_Click(object sender, RoutedEventArgs e)
{
    // Causes application to terminate while task is performed
    task.Show();
}
```

Application Bar

Each page can have an Application bar with up to four icon menu items and optionally additional pull-up text menu items (accessed by clicking '…'). See the sample code at the bottom of MainPage. xaml in a new project. Add a Click attribute to add a handler.

With Opacity of 1 the page size is reduced by the size of the bar. Opacity < 1 shows the bar over the page contents.

> **Hot Tip**
> Expression Blend makes it easy to add icons to App Bar Buttons from the built-in resources.

Accelerometer

Add a reference to Microsoft.Devices.Sensors and access information from an Accelerometer object.

```
private void StartAccelerometer()
{
    accel = new Accelerometer();
    accel.ReadingChanged += new EventHandler<AccelerometerReadingEventArgs>
(accel_ReadingChanged);
    accel.Start();
}
void accel_ReadingChanged(object sender, AccelerometerReadingEventArgs e)
{
    // Access e.X, e.Y, e.Z & e.Timetamp
}
```

The values of X (right), Y (top), Z (face), range from 1 when the stated side points to the ground to -1 for the opposite side.

Location Services

The location system talks to the Microsoft location services to help establish a location. Usage is similar to the accelerometer using an add reference to Microsoft.Devices and the Microsoft.Devices.Sensors.GeoCoordinateWatcher class, which has a StatusChanged event and PositionChanged event (returning GPS data).

Use the optional constructor parameter to request high-accuracy data using the GPS receiver (taking longer to get a position and using more battery power). Otherwise, just available Mobile phone tower and Wi-Fi information is used.

Touch

> **Hot Tip**
> Use the GestureListener in the Silverlight for Windows Phone toolkit to easily add a handler for Tap, Double Tap, Drag, Flick, Hold, and Pinch/Zoom gestures on part of the UI.

```
<Grid x:Name="LayoutRoot"  Background="#00FFFFFF">
    <tk:GestureService.GestureListener>
        <tk:GestureListener  Hold="GestureListener_Hold"
DoubleTap="GestureListener_DoubleTap" Flick="GestureListener_Flick"/>
    </tk:GestureService.GestureListener>
</Grid>
```

Push Notification Services

Microsoft's Push Notification Services (PNS) provide a way for others (including the application developer) to send notifications to the application. Create a Microsoft.Phone.Notification. HttpNotificationChannel (or reopen an existing one) for the phone to receive notifications on behalf of the application. The channel is globally unique to the application instance combined with a channel name. The ChannelUri provided in the channel's ChannelUriUpdated event should be sent to any party that needs to send notifications. The channel can receive Raw notifications (and optionally be configured to receive Toast and Tile notifications), which are sent to the PNS on the ChannelUri as simple xml+data packets.

Notification Type	Detail
Raw	Up to 4096 bytes in a format agreed between parties. Presented to application by OS as event, if running, otherwise discarded.
Toast	Two strings. Presented to application by OS as event, if running. Otherwise, shown by OS as 'toast' overlay at top of screen (first string is Bold) and clicking it launches the application (with no knowledge of Toast).
Tile	A string, number, and image URL that will update the application tile if it was pinned to the Start screen by the user. The image URL must be either a relative URL for an image in the application deployment or be an internet URL under a domain permitted when Tile notifications were requested.

If an error event occurs on the channel object, re-create the channel and inform any parties of the new URL.

DESIGN CONSIDERATIONS

Orientation

Set SupportedOrientations on a page as desired. Physical rotation of the devices causes the UI to be rotated if the new orientation is supported. To support both orientations, either build UI without absolute positioning (e.g., using ScrollViewer, text wrapping, scaling elements, and/or * width/height for Grid columns and rows) or support the OnOrientationChanged override to adjust UI layout.

Built-in Styling Resources

Resources like styles, brushes, widths, colors, fonts, and font sizes can be applied to UI elements in XAML using the syntax like FontFamily="{StaticResource ABC}" where ABC is a resource defined in XAML or is one of the built-in styles available on the phone. The built-in resources are listed when picking the Apply Resource option on select properties in the Property window. Some resources dynamically change according to the theme chosen on the phone.

Hot Tip Use dynamic built-in theme resources to make sure UI colors work correctly for any theme. Note that a device manufacturer can add an 11th unknown color that the user can choose.

Submission Requirements

Be sure to check on the latest submission requirements at http://go.microsoft.com/fwlink/?LinkID=183220 before submitting an application to marketplace.

Check out the Design Resources for Windows Phone 7 at http://msdn.microsoft.com/en-us/library/ff637515(VS.92).aspx and short design tutorials at http://www.microsoft.com/design/toolbox/school/tutorials.aspx.

ABOUT THE AUTHOR

Known for his broad and deep hands-on expertise with Microsoft technologies along with his creative problem-solving and solution-building mindset, Colin is the Principal Architect for Ace of Clouds, a speaker, trainer, author, user group leader, academic advisor, CTO, and company director. He has expertise creating rich UI with WPF/Silverlight, cloud development with Azure, mobile development on Windows Phone, and business intelligence with SQL Server. He has in-depth knowledge of core technologies such as .NET, OData, WCF, WF, LINQ, and WIF. He has been a hands-on Architect for over 17 years having developed award-winning simulation technology with rich UI, cloud-based learning portals and workflow-driven BI systems. He also created the first streaming video community site with Windows Media. Colin has worked in the finance, telecoms, e-learning, Internet communications, learning, and gaming industries, with his latest business solutions currently in use by thousands of users worldwide in corporations like GE, HP, O2, Cisco, IBM, Microsoft, and Reuters.
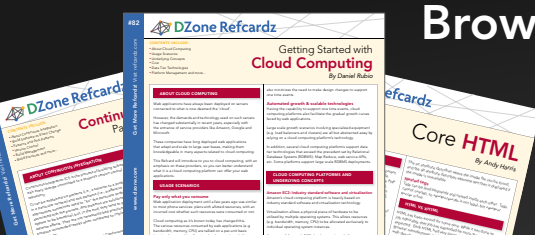
RECOMMENDED BOOK

Programming Windows Phone 7 is a free e-book from Charles Petzold and the Windows Phone 7 team. This book is divided into 3 parts: basic concepts of Windows Phone 7, Silverlight, and XNA 2D.

READ NOW
http://www.charlespetzold.com/phone/