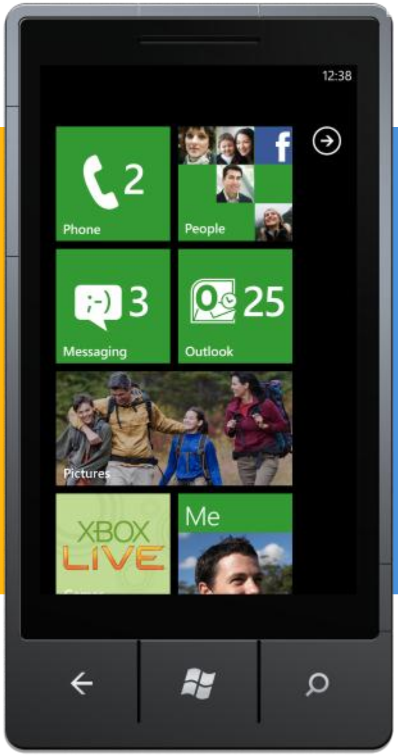
Windows® Phone

JAOO conference 2010

Exploring Windows Phone 7 Application Development



7

Jeff Wilcox

Senior Software Development Engineer
Microsoft Corporation
http://www.jeff.wilcox.name/

Agenda

- Introduction to Windows Phone 7 & Silverlight
- Controls
- Threading
- Navigation & Pages
- Phone controls
- Performance
- Limitations
- Application Lifetime, Launchers & Choosers, etc.
- Questions

IT’S A NEW PLATFORM!

- New OS
- Modern UX & modern UI framework
  - Multi-touch devices
  - Emphasis on design
- Excellent for developers
  - Integrated free toolset
    - One download… Visual Studio, Blend, Emulator, Controls
  - Quickly build professional apps
    - … and stop worrying about pointers and raw OS messaging

Hardware

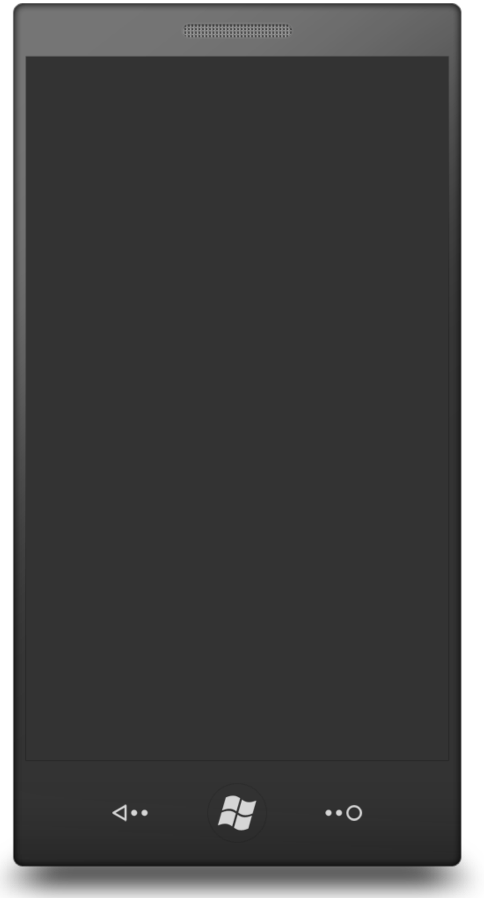
Display
480x800 QVGA
Other resolutions in the future

Capacitive touch
4+ contact points

Sensors
A-GPS, Accelerometer, Compass, Light

Camera
5+ megapixels

Hardware buttons
Start, Search, **Back**



Multimedia
Codec acceleration

Memory
256MB RAM or more
8GB Flash or more

GPU
DirectX 9 acceleration

CPU
ARMv7

Framework Choices

- Silverlight for Windows Phone
  - Write code in C#, VB
  - Powerful client platform
  - Leverage a decade of .NET libraries
- XNA
  - Powerful 2D and 3D game experiences
  - Raw game loop and libraries
- Native code is not a choice today

Windows® Phone

Silverlight

for Windows Phone

Silverlight for Windows Phone

- Silverlight "3++"
  - Optimized for the phone GPU & ARM, input
  - Phone OS and sensor integration
  - Relaxed sandbox security compared to the plugin
- Silverlight plugin apps (in IE) are not supported

Interested in advanced Silverlight?

- Tomorrow there is a session on Silverlight and view model by Nikhil Kotari at 10:15
- Purchase a WP7 book *or* a SL3 book

Silverlight

- Input
  - Keyboard, Mouse
  - Touch
  - Ink
- Modern App Framework
  - Data binding
  - Model – ViewModel (MVVM)
- Data
  - LINQ, LINQ to XML, XML
  - Isolated Storage
- Base Class Library (BCL)
  - Generics
  - Collections
  - Cryptography
  - Threading

- UI Core
  - Vector Shapes
  - Layout
  - Animation
  - Text
  - Images
- XAML
- Media Stack
  - VC1, WMA, MP3
  - H.264, AAC
- Communication & WCF
  - REST
  - JSON
  - RSS/ATOM
  - SOAP

XAML

- Declarative XML markup for UI trees
  - Resources
    - Brushes, gradients, helper code and converters
  - Vector graphics, layout containers, text
  - Controls
  - Set properties, bindings, hook up event handlers
- Enables the rich design surfaces in Visual Studio and Expression Blend 4

```
<Grid>
    <TextBox Text=“Hello World" TextChanged=“MyMessageChangedHandler"/>
</Grid>
```

Layout

- Flexible Layout System
- Core Panels
  - StackPanel: Vertical or horizontal stacks
  - VirtualizingStackPanel: Fast item lists when fixed height
  - Grid: Rows, columns, very flexible
- Other Panels include Canvas and WrapPanel

- TIP: Avoid using Canvas

Styling

- Phone styles are built in
  - Normal text, large text, etc.
  - Aware of phone themes (accent color, light & dark, …)
- Can also create your own styles
  - Per-app, per-page, or per-element

```
<Style TargetType="TextBlock"
       x:Key="SubHeadingText">
    <Setter Property="FontSize” Value="42" />
    <Setter Property="Foreground“ Value="Blue" />
</Style>
```

```
<TextBlock Text=“What’s up?“
           Style=“{StaticResource SubHeadingText}“/>
```

Touch Events

- Events exposed by UI elements
- Raw Touch Points
- Silverlight Toolkit GestureService
  - Tap, DoubleTap, Hold, Flick, Drag & Pinch events
- XNA offers a ‘TouchPanel’ for gestures

Data Binding

- Decouples apps and views when using MVVM
- Two types of binding
  - Property binding (i.e. Foreground color)
  - List binding (List of people)
- Set DataContext on an element tree
- Based on a change notification system

```
<TextBlock Text=“{Binding Name}”
           Foreground=“{Binding Name, Converter={StaticResource
MyPersonForegroundBrushSelector}}” />
```

Property Change Notifications

- Data objects implement INotifyPropertyChanged
  - TIP: Don't use Reflection here
  - TIP: Store a single event argument instance (link)
  - TIP: Constant strings offer a simple level of static checking
- Collections have INotifyCollectionChanged

```
// Bad for performance
NotifyPropertyChanged(()=> FullName);

// Good
NotifyPropertyChanged(“FullName”);

// Great
NotifyPropertyChanged(PropertyFullName);
```

Code and Design Reuse

- User interface
  - Pages and user controls
  - Custom controls
- Code
  - Class and control library projects
  - Across the .NET platform
    - Silverlight plugin (Mac, Windows)
    - Other platforms (Mono)
- TIP: Carefully evaluate UserControl use in your application, they can have performance problems from XAML parsing.

Standard Controls

Controls

- Built-in controls “just work”
  - Theme-aware
  - Manipulation events
  - Create event connections in XAML or in code
- Basic controls
  - Button ‘clicks’
  - ScrollViewer pans and flicks
  - TextBox handles the cursor and works with software & hardware keyboard

Standard Controls

- Border
- Button
- *Canvas*
- CheckBox
- *Grid*
- HyperlinkButton
- Image
- ListBox
- MediaElement
- PasswordBox
- ProgressBar
- RadioButton
- ScrollViewer
- Slider
- *StackPanel*
- TextBlock
- TextBox

SDK

- Map
- Panorama
- Pivot
- WebBrowser

Toolkit

- ContextMenu
- DatePicker
- *Gesture service*
- TimePicker
- ToggleSwitch
- *WrapPanel*

TextBox Input

- TextBox
  - Integrates with the SIP (software keyboard)
- Input Scopes
  - Optimized key layouts
    - Email, web address, numeric, etc.
  - Enables auto-correction
- PasswordBox
  - Standard phone experience, no special code

Windows® Phone

Lists

ListBox

- Chose over ItemsControl if you can
- Supports data and UI virtualization
  - Uses VirtualizingStackPanel by default
  - The native runtime optimizes performance for ListBox
  - Data virtualization is not standard, your collection must inherit from Ilist and provide a Count parameter (link)

ListBox Item Tips

- Virtualized items should have a fixed height to prevent stutter
- Keep items as simple as possible!
- Get creative with a Grid instead of 20 panels
- Avoid custom controls inside data templates
- There've been strategies blogged
  - wait to load images until after scrolling is complete
  - perform decoding on a background thread

Bad ListBox DataTemplate

```
<ListBox.ItemTemplate>
    <DataTemplate>
        <StackPanel>
            <StackPanel Orientation="Horizontal">
                <Image
                    Source="{Binding PicSource}"
                    Width="{Binding ApproxWidth}"/>
                <controls:MyPresenceControl
                    User="{Binding}"
                    OnlineText="{Binding ElementName=Resources, Path=UserOnline, Conve
rter={StaticResource CurrentCultureConverter}}"/>
                <TextBlock
                    Text="{Binding Name}"
                    Foreground="{Binding Name, Converter={StaticResource MyNameForegro
undConverter}, ConverterParameter='Red,Blue'}"/>
            </StackPanel>
        </StackPanel>
    </DataTemplate>
</ListBox.ItemTemplate>
```

Good ListBox Data Template

- Fixed Grid height (eliminates stutter) and exact image sizing
- Raw & direct binding to a model object
  - Expose visibility, brushes, image sources as properties
  - Avoid using converters, converter parameters
- No element name binding
- Avoids custom controls and user controls
- Model should avoid custom logic in getters

```
<ListBox.ItemTemplate>
    <DataTemplate>
        <Grid Height="180">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="80"/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>
            <Image Source="{Binding PicSource}" Width="110" Height="110"/>
            <Image Source="{Binding PresencePic}" Width="70" Height="30"
                    Grid.Column="1"
                    Visibility="{Binding PresenceVisible}"/>
            <!-- clipped for space -->
```

ListBox without Virtualization

- Slower startup time (full layout pass on all items)
- For grouping scenarios, accordions, etc.

```
<ListBox>
    <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
            <StackPanel/>
        </ItemsPanelTemplate>
    </ListBox.ItemsPanel>
</ListBox>
```

ListBox Performance

Demo

Threading

Threading

- Core Threads
  - Compositor Thread
  - User Interface (UI) Thread
- Background Threads

Compositor Thread

- Interacts with the GPU
- Enables Independent Animations
  - Opacity
  - Translation
  - Scaling
  - Perspective transform
- TIPS:
  - Elements need to be in a list (items are auto-cached) or have CacheMode="BitmapCache"
  - Avoid OpacityMask on items, it cannot be accelerated

User Interface Thread

- Primary thread for Silverlight
- Handles important events
    - Input and touch
    - OS messages
- Layout
    - System for arranging and sizing elements
    - Directly related to how complex pages are
- All UI properties must be set on this thread
- TIP: Keep this thread free and clear!

Dispatching to the UI Thread

- Must set UI properties on the UI thread
  - If you don't…
    - UnauthorizedAccessException: "Invalid cross-thread access"
- Use a Dispatcher or other cross-thread communication technique to make the call
  - `Dispatcher.BeginInvoke(()=>TextBox1.Text="Hi.");`
- Beware property change calls from model and data objects on your background threads

TIP: Be smart when dispatching

- A dispatch typically costs at least 16ms (1 tick)
- Group off-thread property change notifications
- Use a smart dispatcher

```
private void Dispatch(Action action)
{
    // No need for a dispatcher invoke, this will be quicker.
    if (Dispatcher.CheckAccess())
    {
        action();
    }
    else
    {
        Dispatcher.BeginInvoke(action);
    }
}
```

Background Threads

- BackgroundWorker
- .NET thread queue
- Important for performance
  - Help keep the UI thread available for input & events
  - Use for XML, JSON, serialization processing
  - Complex calculations, logic, server communication
- TIP: Use the background thread often to keep the UI thread as responsive as possible

BackgroundWorker

```
BackgroundWorker bw = new BackgroundWorker();
    bw.DoWork += BackgroundWork;
    bw.RunWorkerCompleted += (x, xe) => {
        // This is called on the UI thread
    };
bw.RunWorkerAsync();

private void BackgroundWork(object sender, DoWorkEventArgs e)
{
    // This is where your background magic happens.
}
```

Windows® Phone

Media

MediaElement

- Media element plays audio or video
- Proper visual element – place it inside your app, overlay with controls and visuals

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition Height="180"/>
    </Grid.RowDefinitions>
    <MediaElement Grid.RowSpan="2"
                  Source="/Intro.wmv"
                  AutoPlay="True"
                  Stretch="UniformToFill"/>
    <Grid Grid.Row="1" Background="Black" Opacity=".6">
        <!-- Put playback controls here -->
    </Grid>
</Grid>
```

MediaPlayerLauncher

- Opens the OS media player with content of your choice
- You select the playback controls (pause, etc.)

```
MediaPlayerLauncher mediaPlayerLauncher = new MediaPlayerLauncher();
mediaPlayerLauncher.Media = new Uri(“Intro.wmv”, UriKind.Relative);
mediaPlayerLauncher.Location = MediaLocationType.Data;

mediaPlayerLauncher.Controls =
    MediaPlaybackControls.Pause | MediaPlaybackControls.Stop;

mediaPlayerLauncher.Show();
```

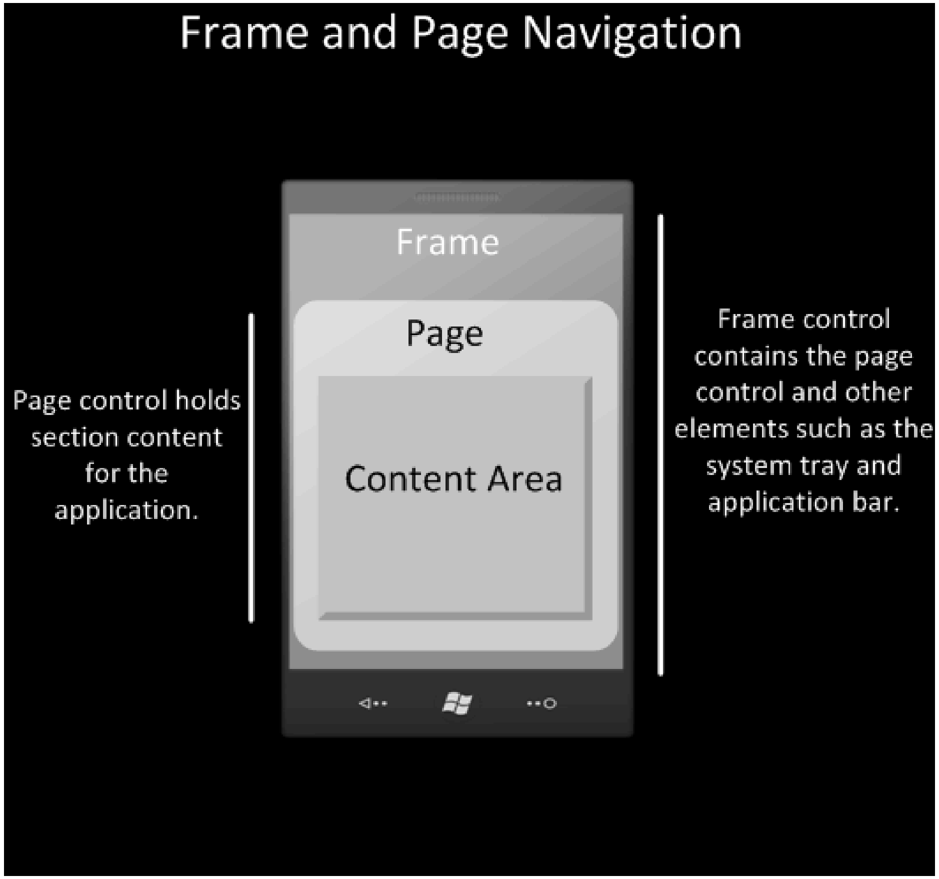
Media Tips

- Only one MediaElement can be used at a time
- XNA sound effect API can play multiple sounds
- No VideoBrush on the phone
- Emulation & Debugging…
  - Media playback in the emulator may not be supported (see documentation)
  - With an attached debugger, media may not playback on the device

Navigation & Pages

Frame and Page

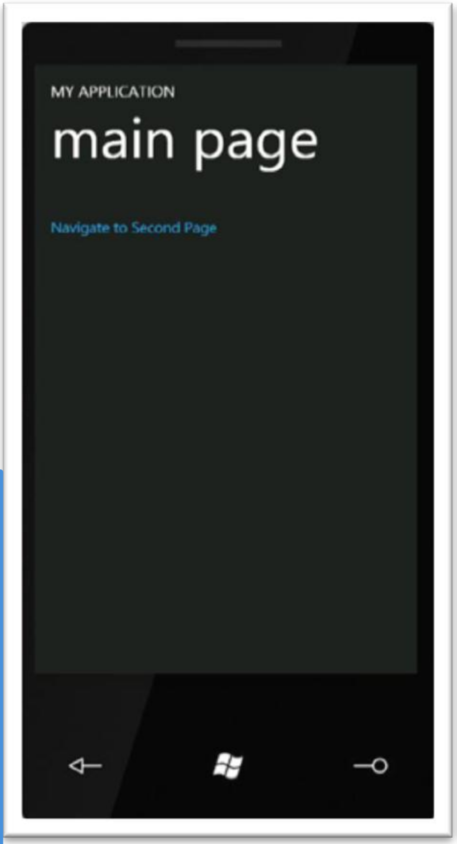
- All apps must have a root phone frame
- Can derive from it to customize
- Frame and pages handle back button navigation automatically
- Back stack is managed by the app platform

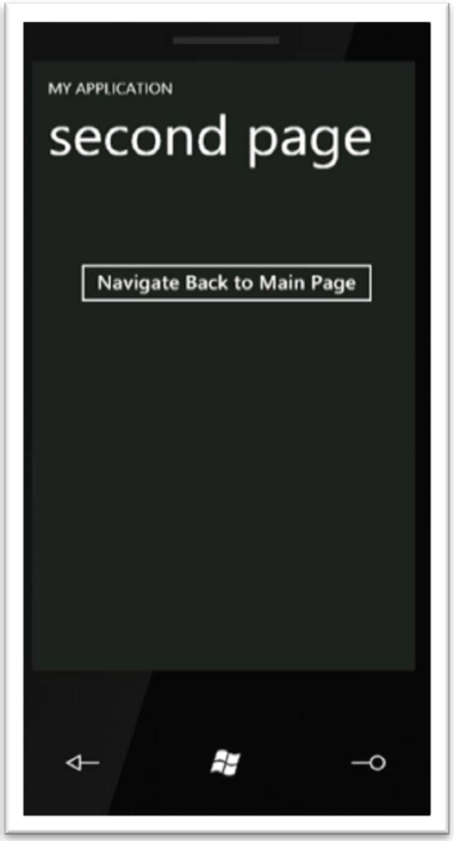

Frame and Page Navigation
Frame
Page
Content Area
Page control holds section content for the application.
Frame control contains the page control and other elements such as the system tray and application bar.

Page Navigation

- Page-based navigation model
  - Similar to web page model
  - Each page identified by a URI
  - Each page is essentially stateless

```
private void hyperlinkButton1_Click(
      object sender, RoutedEventArgs e)
{
  NavigationService.Navigate(
    new Uri("/SecondPage.xaml",
           UriKind.RelativeOrAbsolute)
  );
}
```



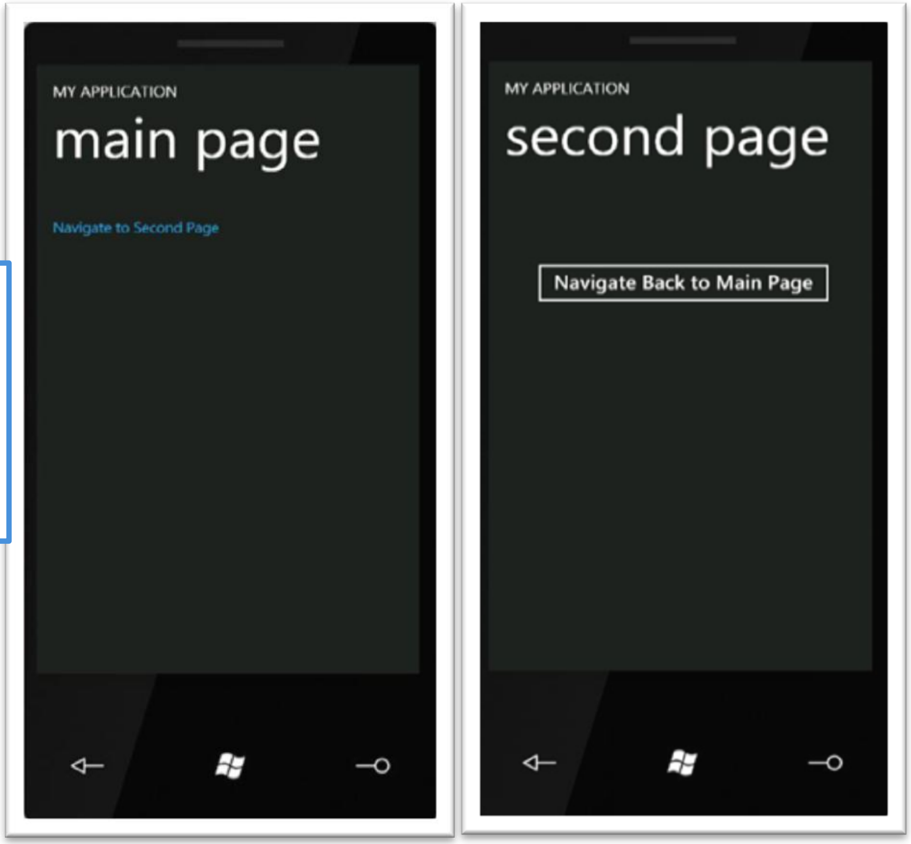


Navigating Back

- Application can provide controls to navigate back to preceding page

```
private void button1_Click(
  object sender, RoutedEventArgs e)
{
    NavigationService.GoBack();
}
```

- The hardware Back key will also navigate back to preceding page
  - No code required – built-in behavior
  - Can override as needed to maintain logical experience in controls & app



Passing Data Between Pages

- Can pass string data between pages using query strings

```
private void passParam_Click(object sender, RoutedEventArgs e)
{
  NavigationService.Navigate(new Uri("/SecondPage.xaml?msg=" +
                                        textBox1.Text, UriKind.Relative));
}
```

- On destination page

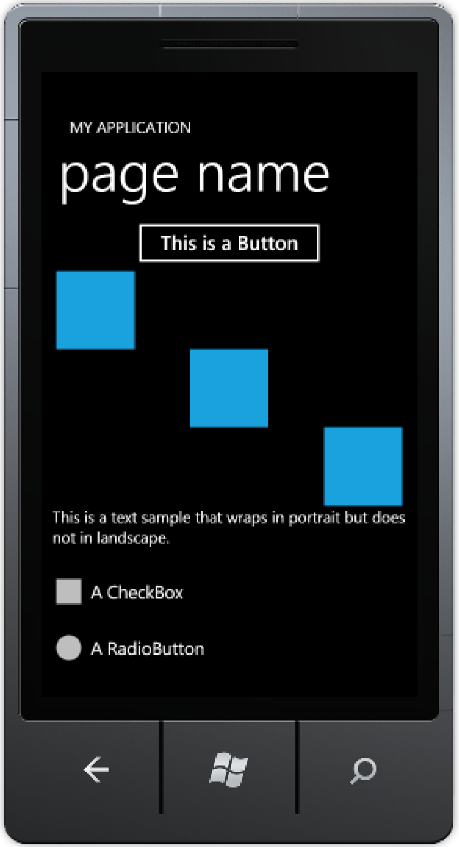
```
protected override void OnNavigatedTo(
    System.Windows.Navigation.NavigationEventArgs e)
{
  base.OnNavigatedTo(e);
  string msg = string.Empty;
  if (NavigationContext.QueryString.TryGetValue("msg", out msg))
         textBlock1.Text = msg;
}
```

Passing Objects Between Pages

- Often, you will need to pass a data object from one page to another
  - For example, the user selects an item in a list and navigates to a Details page
- One solution is to store your ViewModel (that is, data) in your App class
  - Global to whole application
- Pass the selected item index in query string

```
NavigationService.Navigate(
        new Uri("/SecondPage.xaml+?Param=value" ,
                    UriKind.RelativeOrAbsolute
```

Orientation-aware Forms



MY APPLICATION

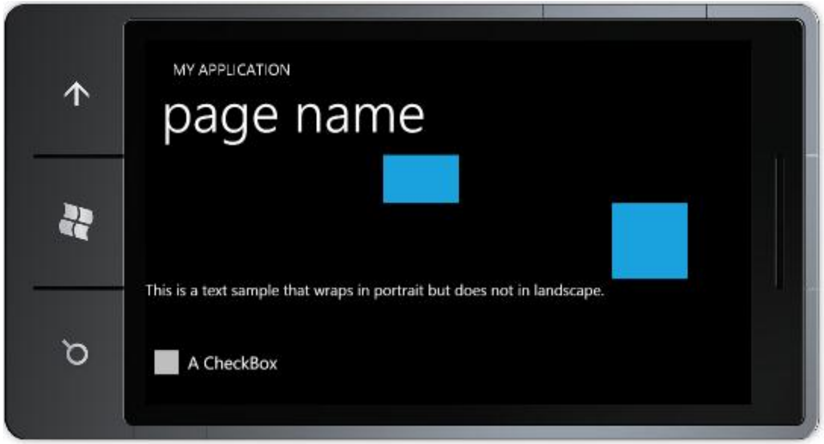
page name

This is a Button

This is a text sample that wraps in portrait but does not in landscape.

□ A CheckBox

○ A RadioButton

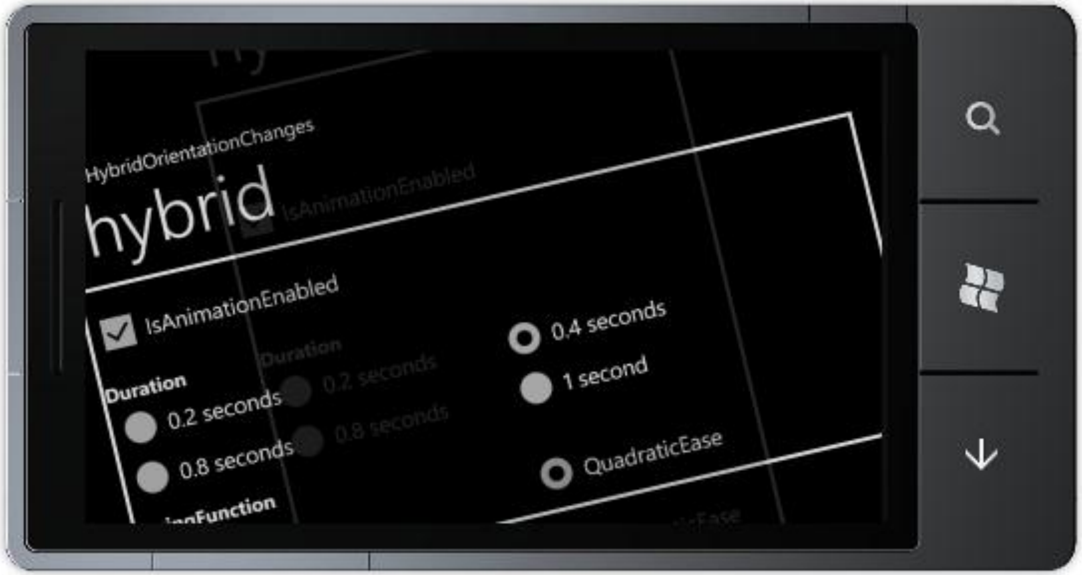


MY APPLICATION

page name

This is a text sample that wraps in portrait but does not in landscape.

□ A CheckBox



HybridOrientationChanges

hybrid

☑ IsAnimationEnabled

Duration

● 0.2 seconds

● 0.8 seconds

○ 0.4 seconds

● 1 second

ingFunction

○ QuadraticEase

Orientation-aware Pages

```
<phone:PhoneApplicationPage
    x:Class="OrientationSample.MainPage“
    …
    SupportedOrientations="PortraitOrLandscape" Orientation="Portrait"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768“>
```

- TIPS
  - Can use a ScrollViewer to scroll content when in landscape
  - Use Grid and StackPanel over Canvas, since they respect layout
  - Can provide MinWidth, MaxWidth, etc. property values
  - Avoid hard-coding widths
  - Provide a smooth transition animation automatically (link)

Windows® Phone

Phone Controls

Phone Controls

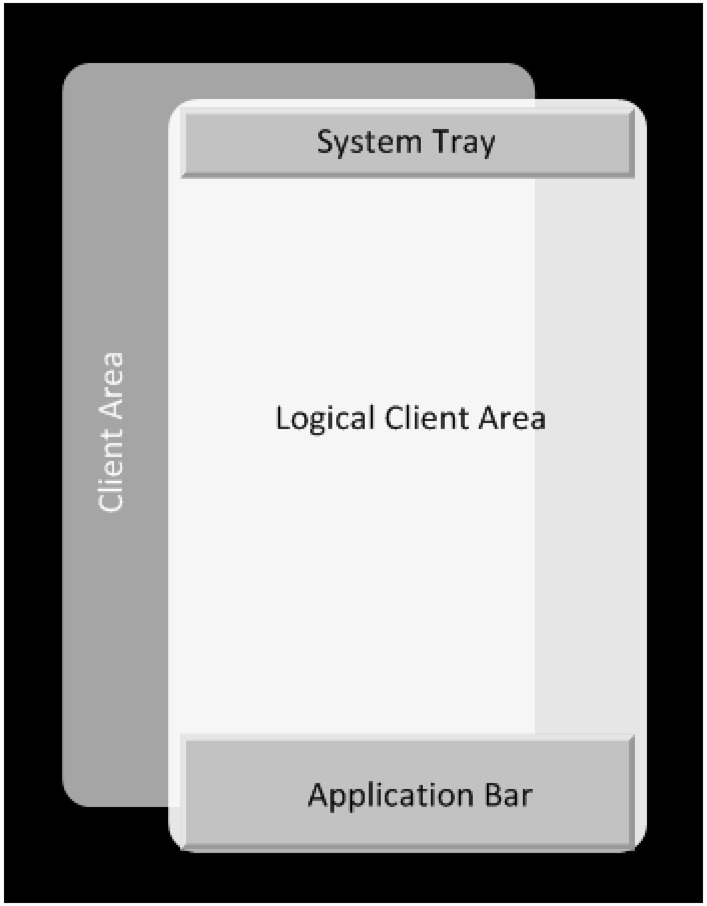
- Application Bar
- System Tray
- Web Browser Control
- Pivot
- Panorama
- *Map*

Application Chrome
System Tray and Application Bar

- System Tray
  - System owned indicator area that displays system-level status information
  - Apps can show/hide the Sys Tray

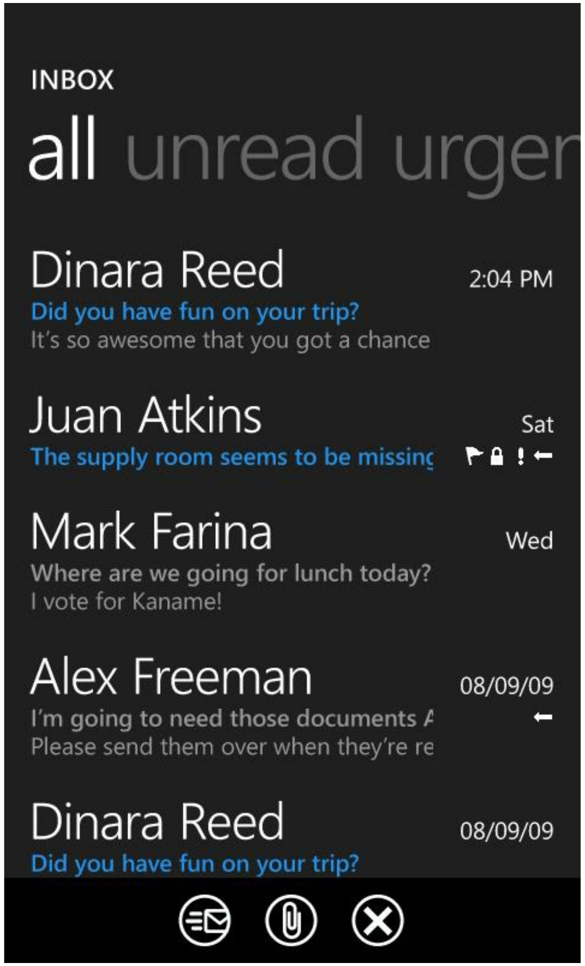
```
Microsoft.Phone.Shell.SystemTray.IsVisible = false;
```

- Application Bar
  - Area where applications can display buttons for the most common tasks
  - Can display pop-up menu for less common tasks



Application Bar

- Use the Application Bar instead of creating your own menu system
- Up to 4 buttons plus optional menu
- Unfortunately data binding does not work with app bar! ☹



INBOX

all unread urgen

Dinara Reed 2:04 PM
Did you have fun on your trip?
It's so awesome that you got a chance

Juan Atkins Sat
The supply room seems to be missing

Mark Farina Wed
Where are we going for lunch today?
I vote for Kaname!

Alex Freeman 08/09/09
I'm going to need those documents A
Please send them over when they're re

Dinara Reed 08/09/09
Did you have fun on your trip?

WebBrowser Control

- Embeds a browser window into your app
  - No back/forward/URL chrome
- Can navigate to a URI or you can send it HTML
- Alternatively can open pages in IE with a chooser.

```
webBrowser1.Navigate(new Uri("http://www.bing.com/"));

webBrowser1.NavigateToString(
     "<b>Hello JAOO Conference!</b>");
```

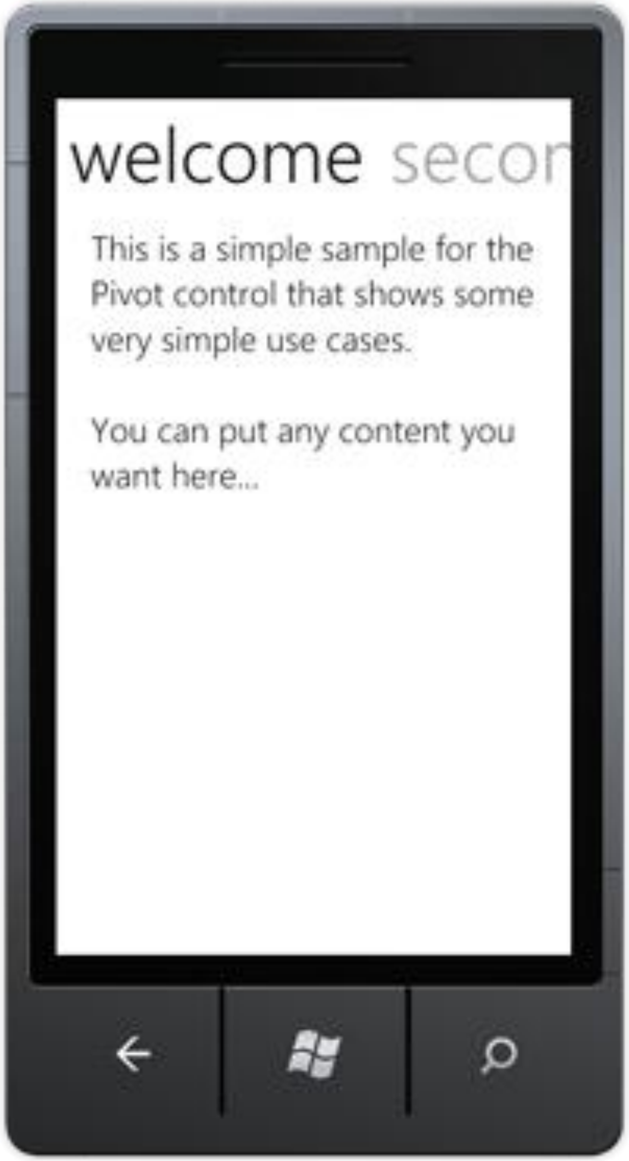
WebBrowser Control

- TIP: Minimize the number of browser controls to 1

- Best illusion of performance:
  - Use a single web browser control throughout the app
  - Translate the control off-screen *or* set Opacity to 0
  - When the control has loaded, navigate to the HTML string or page
  - Consider not showing the control until after the LoadCompleted event has fired

Pivot

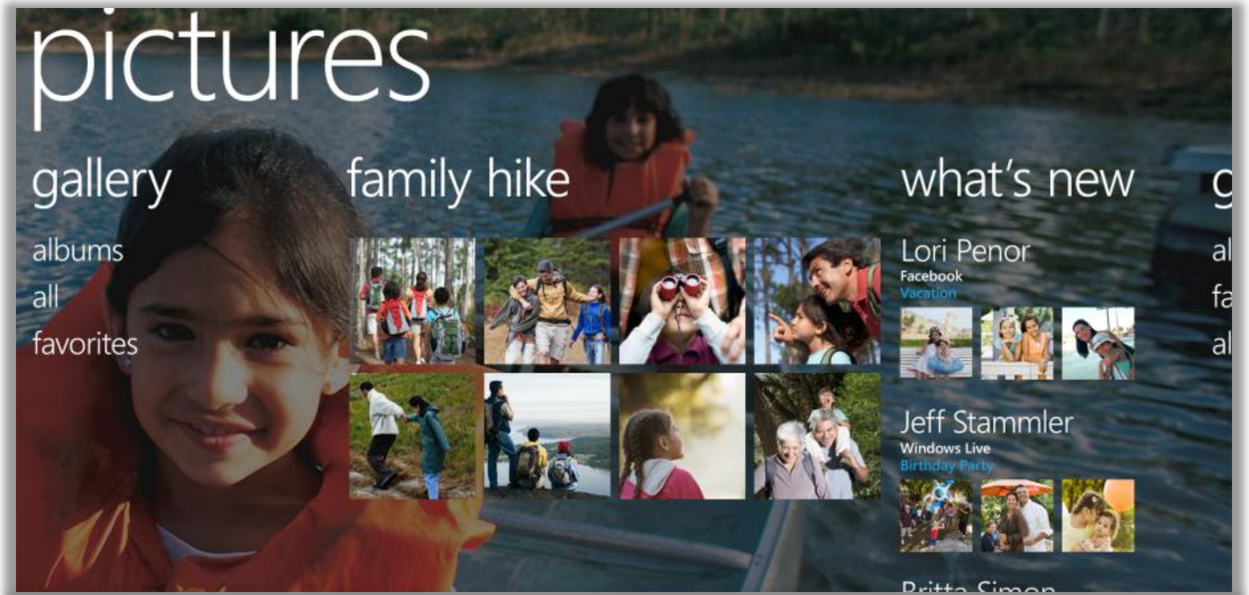
- Like a tab control
- Optimized for quick startup time
- Used for filtering data and views
- I recommend over Panorama for most apps



welcome secor

This is a simple sample for the Pivot control that shows some very simple use cases.

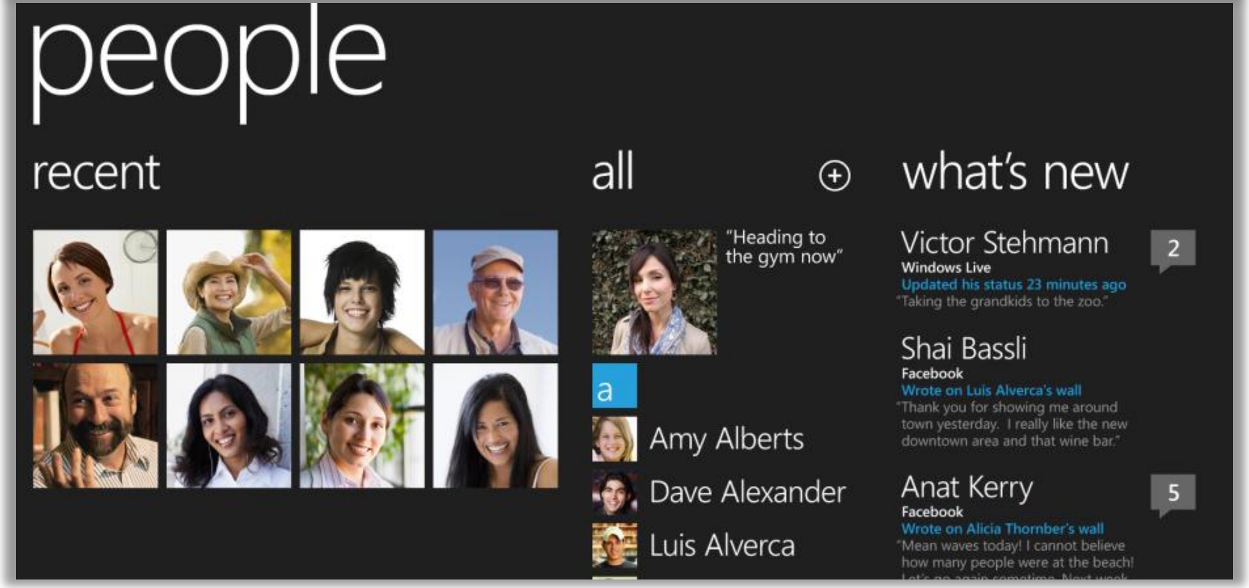
You can put any content you want here...



Lori Penor
Facebook
Vacation

Jeff Stammler
Windows Live
Birthday Party

Britta Simon



"Heading to the gym now"

Amy Alberts

Dave Alexander

Luis Alverca

Victor Stehmann
Windows Live
Updated his status 23 minutes ago
"Taking the grandkids to the zoo."
2

Shai Bassli
Facebook
Wrote on Luis Alverca's wall
"Thank you for showing me around town yesterday. I really like the new downtown area and that wine bar."

Anat Kerry
Facebook
Wrote on Alicia Thornber's wall
"Mean waves today! I cannot believe how many people were at the beach! Let's go again sometime. Next week
5

Panorama

- UX: “Rich magazine-like experience”
  - Parallax background image is optional
  - Allows the user to explore a larger canvas area than traditional phone pages
- Items with Orientation=“Horizontal” extend beyond a typical phone page’s size
- A full layout pass happens on all items at startup!

Simple Pivot App

Demo

Pivots Vs. Panoramas

Pivot

- Supports a larger number of "items" because the layout is simpler
- Supports programmatic selection for rich code-behind scenarios
- LoadingPivotItem and UnloadingPivotItem events make it easy to defer content creation
- More efficient use of screen space

Panorama

- Tends to create more visually stunning experiences
- DefaultItem property makes matching operating system restore behavior easy
- Background images of any size automatically wrap properly
- Horizontally-oriented PanoramaItems allow variably-sized content to flow smoothly

Performance

Performance

- Phone hardware is very different than your quad core desktop PC
- Minimize complex visuals
- Offload animations to the compositor thread
- Make list item templates as simple as can be
- Use BackgroundWorker & async coding throughout

Emulator

- Windows Phone Emulator is great for developing applications
- Contains an actual X86 build of the WP7 OS
- *Emulates* the OS but does not have hardware capability limiting
- May not see performance ‘issues’ until you use a real Windows Phone 7

Startup

- Splash Screen
  - A JPEG image shown at startup
  - Optional
  - Some apps use the Startup + Startup Overlay concept
- Minimize the startup DLL size
- After 10 seconds your app is closed if not responsive

Layout Performance

- Layout happens on the UI thread
- Very expensive but necessary computations

- Fast startup time, slow to display when toggled:
  - Visibility="Collapsed"
- Slower page startup time, ready to display anytime:
  - Opacity="0"

Vectors vs. Images

- Avoid complex vector paths that could be images
- JPEGs are decoded quicker than PNGs
- Mark images as **Content** instead of **Resource** when possible (places them outside the DLL and in the XAP instead)
- Image code paths are optimized to load Content and isolated storage streams over memory streams
- Use Resource for Panorama backgrounds to avoid a flicker or glitch

Loading Data

- Consider smart caching of web service data
  - A Dictionary cache in memory
  - FIFO Queue in Isolated Storage
  - Web service requests
- Parse data on the background thread
- Beware serialization/deserialization costs
- JSON is great

Progress Bar Control

- “Indeterminate” progress bars are used throughout the UX on the phone
- Control in the framework is sub-optimal as its animations are not offloaded to the compositor thread
- Please use my PerformanceProgressBar (link)
- Always set/bind IsIndeterminate to False when the effect is no longer needed to stop the storyboards

Memory

- 90MB peak memory use limit for marketplace apps on devices with 256 MB RAM
- Panorama and Pivot pages often have many views and can add up in size
- Compressed JPEG images may be small in file size, but are still uncompressed at runtime

```
var deviceMem=(long)DeviceExtendedProperties.GetValue("DeviceTotalMemory");
var app=(long)DeviceExtendedProperties.GetValue("ApplicationCurrentMemoryUsage");
var appPeak=(long)DeviceExtendedProperties.GetValue("ApplicationPeakMemoryUsage");
var managedMemory = GC.GetTotalMemory(false);
```

Frame Rate Counters

- Default projects enable the counters when a debugger is attached
- Find out more about the counters (link)
- Left to right:
  - Compositor Thread frames per second (fps)
  - UI Thread fps
  - Texture memory use
  - Surface counter
  - Intermediate overlay texture count
  - Screen Fill Rate (TIP: < 2.50)

Limitations

It's a new platform!

- This is a first version named 7
- We're committed to providing platform updates

What the OS doesn't do today

- Multitasking
- Copy & paste

Silverlight limitations on the phone

- Shaders and effects
- No ellipsis/text trimming built into the framework
- No 24-bit color in Silverlight
  - TIP:
    Avoid defining gradients that may exhibit banding
  - TIP:
    Use image dithering for gradients if required for your brand

App platform limitations today

- No Sockets
- No built-in relational database
- No platform invoke (p/invoke)
- No native code development option
- Cannot access live video feed from the camera

Application Lifetime

“A world of interruptions”

- Phone calls
- Texts
- Reminders
- Low battery!
- Changing apps
- Lock screen

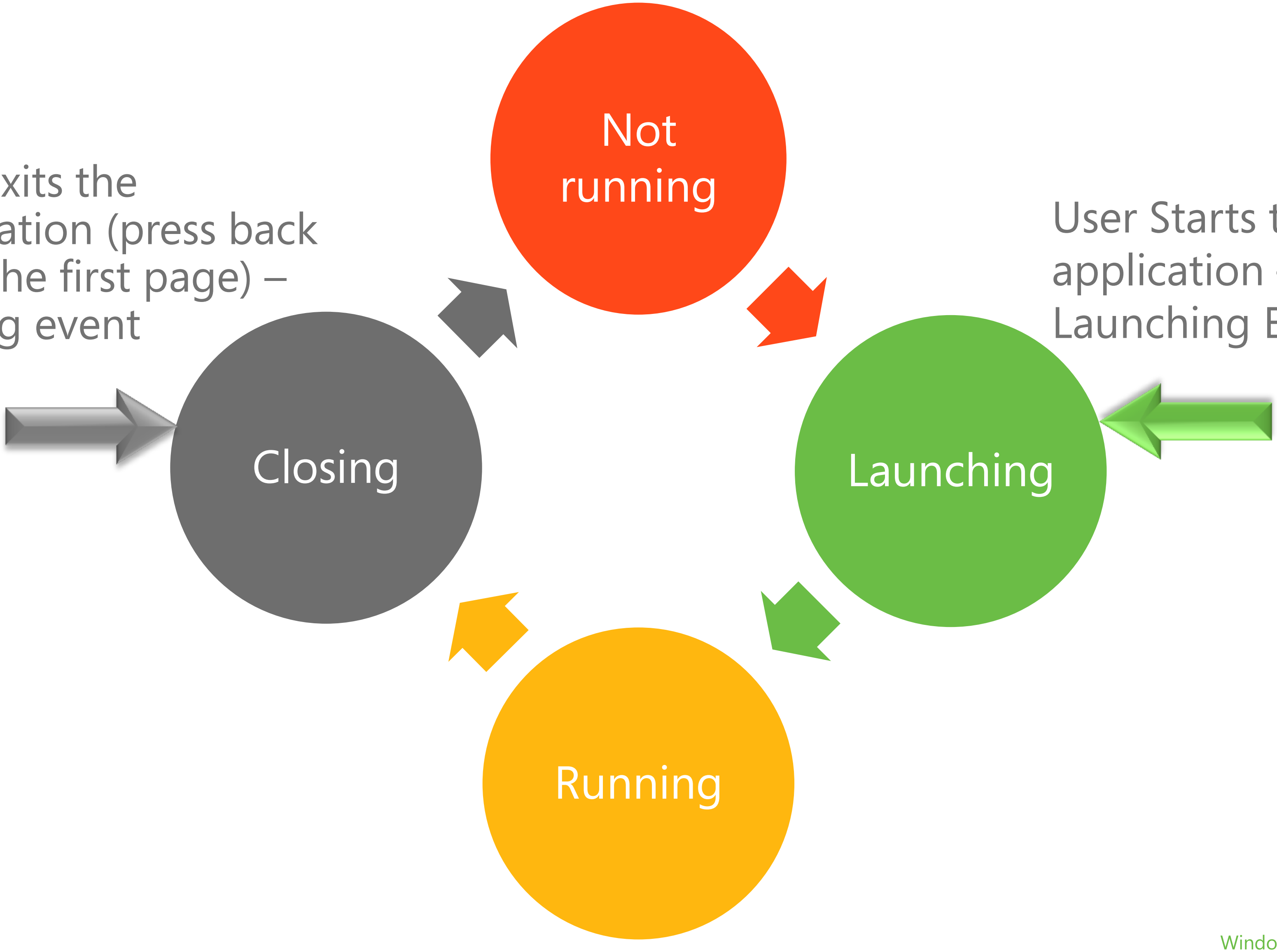
One app at a time

- WP7 maximizes app performance & resource use
  - Only one app in the foreground at a time
  - OS terminates your app when the user navigates away
  - OS may terminate your app if something else takes over the screen
- Your app is **Tombstoned** when user hits Start

  - Save state information
  - When the user comes back, new app instance is started and provided the state info

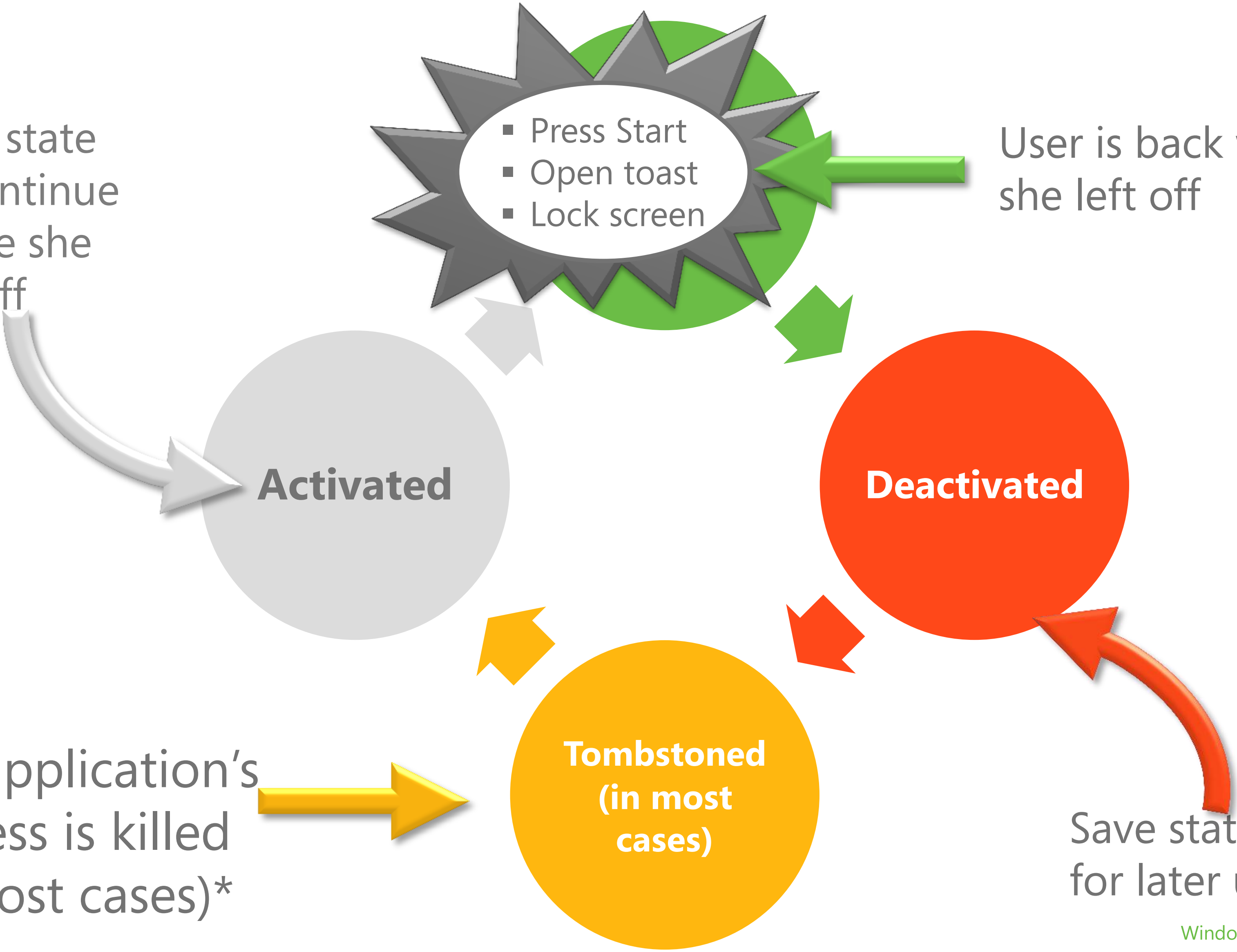
Basic Lifecycle

User exits the application (press back from the first page) – Closing event



User Starts the application – Launching Event

"Tombstone" Lifecycle



Load state to continue where she left off

User is back where she left off

The application's process is killed (in most cases)*

Save state for later use

Lifecycle Events

- Application_Launching
  - Not fired when the application is reactivated
- Application_Activated
  - When the application is activated (brought to foreground)
  - Not fired when started the first time
- Application_Deactivated
  - When the application is deactivated (sent to background)
- Application_Closing
  - When the user presses Back on the first Page
  - Not fired when the application is deactivated / tombstoned

Page State

- Restore the visual state of the page when the app is reactivated
  - User should see the page exactly as it was
  - Include things such as scroll position of a ScrollViewer control and the contents of TextBox controls
- Use the State property of the PhoneApplicationPage class to store transient page state
  - Save state in OnNavigatedFrom event handler
  - Restore state in OnNavigatedTo event handler

Launchers & Choosers

Launchers and Choosers

- The Windows Phone execution model isolates every application in its own sandbox
  - Apps cannot directly access information stores, such as contacts
  - Apps cannot directly invoke other applications, such as phone or messaging
- Launchers and Choosers allow applications indirect access to these phone features
- Launcher and Chooser APIs invoke distinct built-in applications that replace the currently running application
  - See Tombstoning

Launchers & Choosers

Launchers

- PhoneCallTask
- SearchTask
- SMSComposeTask
- WebBrowserTask
- EmailComposeTask
- MarketplaceDetailTask
- MarketplaceHubTask
- MarketplaceReviewTask
- MarketplaceSearchTask
- MediaPlayerLauncher

Choosers

- CameraCaptureTask
- EmailAddressChooserTask
- PhoneNumberChooserTask
- PhotoChooserTask
- SaveEmailAddressTask
- SavePhoneNumberTask

Programming Launchers

```
// Launches the Email application: displays a new email message
EmailComposeTask emailComposeTask = new EmailComposeTask();
emailComposeTask.To = "user@example.com";
emailComposeTask.Body = "Email message body";
emailComposeTask.Cc = "user2@example.com";
emailComposeTask.Subject = "Email subject";
emailComposeTask.Show();
```

```
// Launches the Windows Phone Marketplace client application
// which then shows the search results based on search terms
MarketplaceSearchTask marketplaceSearchTask = new
MarketplaceSearchTask();
marketplaceSearchTask.SearchTerms = "accelerometer xna";
marketplaceSearchTask.Show();
```

Programming Choosers

```
public partial class MainPage : PhoneApplicationPage {
    // Declare the PhotoChooserTask object with page scope.
    PhotoChooserTask photoChooserTask;

    public MainPage()     {
        InitializeComponent();
        // Initialize the PhotoChooserTask and assign the Completed handler
        photoChooserTask = new PhotoChooserTask();
        photoChooserTask.Completed += new
            EventHandler<PhotoResult>(photoChooserTask_Completed);
    }
    private void button1_Click(object sender, RoutedEventArgs e)     {
        photoChooserTask.Show();
    }
    void photoChooserTask_Completed(object sender, PhotoResult e)     {
        if (e.TaskResult == TaskResult.OK)
        {
            BitmapImage bmp = new BitmapImage();
            bmp.SetSource(e.ChosenPhoto);
        }
    }
}
```

Other Topics

Hardware APIs

- Accelerometer
  - High-frequency event
  - TIP: Events fire on a background thread, remember to dispatch to the UI
  - TIP: Consider writing a simple band pass filter
- Vibration
  - Just like Beep in classic apps

  -
    ```
    Microsoft.Devices.VibrateController.Default.Start(
      TimeSpan.FromSeconds(0.25));
    ```

Communication Services

- Push Notifications
- Location Service
  - 3 sources of information
    - WiFi
    - GPS
    - Cell Towers

Windows® Phone

QUESTIONS?

And some answers…

Thank You!

- Please evaluate (comments too!) 🟩 🟨 🟥
- Download the tools
  - http://developer.windowsphone.com/
  - http://silverlight.codeplex.com/
- MSFT loves feedback: what's good, bad, painful?
- Jeff Wilcox

  http://www.jeff.wilcox.name/

  jwilcox@microsoft.com

Windows® Phone

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.