# Module

Page Navigation

# Table of contents

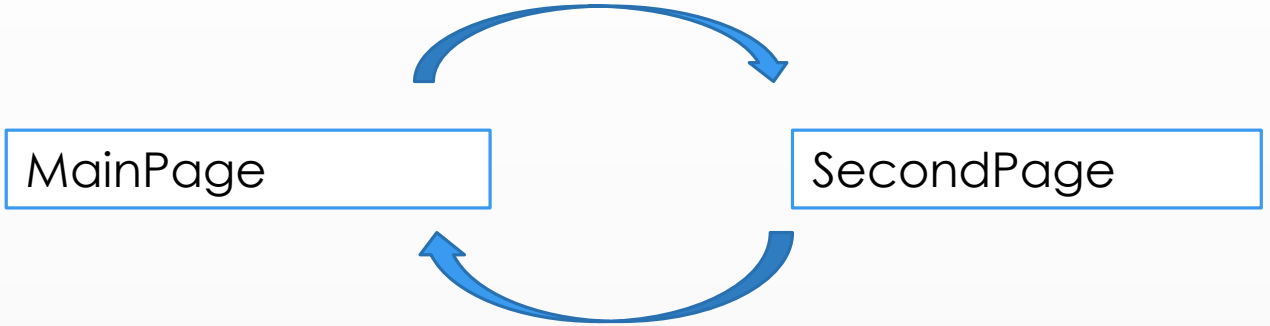

- Introduction
- Methods called through Navigation
- Forward Navigation
- Backward Navigation
- Navigation Bar Back key
- NavigationHelper
- Page Cache Mode

# Introduction

- One window in Windows Phone
- The window contains a single frame (100% of area)
- The frame contains a page (typically sized at 100% of the area)
  - Is created on app launching
  - Every time the app navigates from a page to another one, the frame will render the new page

# Introduction

- Schema




- Navigation history as the user navigates through pages
  - *Frame.BackStack property* returns an *IList<PageStackEntry>*

# Introduction

- In App.xaml.cs

```
protected override void OnLaunched (LaunchActivatedEventArgs e)
{
        Frame rootFrame = Window.Current.Content as Frame;
        …
        // Do not repeat app initialization when the Window already has
        // content, just ensure that the window is active
        if (rootFrame == null)
        {
            // Create a Frame to act as the navigation context and navigate  to the first page
            rootFrame = new Frame();

            ...

            // Place the frame in the current Window
            Window.Current.Content = rootFrame;
        }

        if (rootFrame.Content == null)
        {
            // Navigate to the first page
            rootFrame.Navigate(typeof(MainPage), e.Arguments);
        }
        // Ensure the current window is active
        Window.Current.Activate();
    }
```

# Introduction

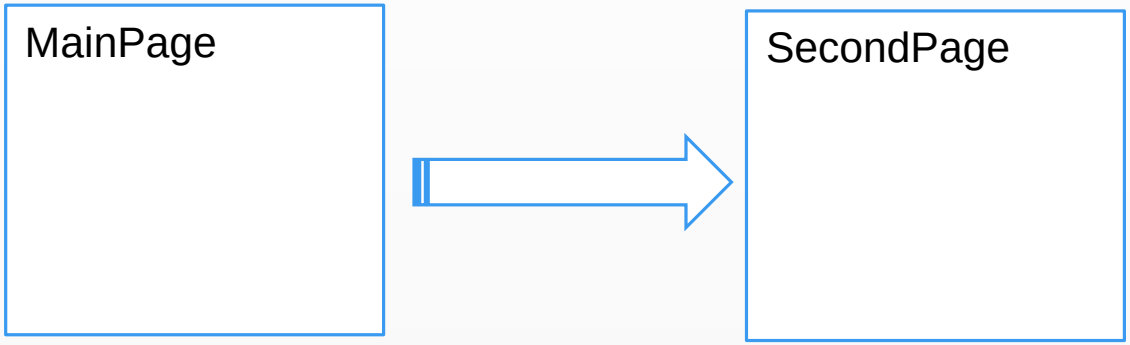
- Navigation Patterns
  - "To achieve a usable interaction model, you should spend the most time up front with the design of your info and how it's navigated. The navigation model determines what's on each screen and how you get from one screen to another.
  - See :
    - https://msdn.microsoft.com/fr-fr/library/windows/apps/dn596118.aspx

# Methods called through Navigation

- *OnNavigatedTo*
  - Called when the user navigates to this page
  - Loads data, populates fields, wires-up event handlers, …
- *OnNavigatedFrom*
  - Called when this page is no longer active in the frame, typically when a user navigates away

# Forward Navigation

- Navigation from one page to another page
- Through button, menu item, …



# Forward Navigation

- Sample (without MVVM Pattern)
  - In MainPage.xaml

```
<Page  x:Class="NavigationPages.MainPage" ….>

   <Grid>
      <TextBlock Text="Page 1" Style="{StaticResource HeaderTextBlockStyle}"
                 Width="200" Margin="10,1,0,0"
                 HorizontalAlignment="Left" VerticalAlignment="Top" />
      <Button x:Name="NavigateBtn" Content="Vers Page 2" HorizontalAlignment="Left"
                 Height="120" Margin="75,275,0,0"
                 VerticalAlignment="Top" Width="240"
                 FontSize="36" Click="NavigateBtn_Click"/>
   </Grid>
</Page>
```

# Forward Navigation

- Sample(…)
  - In the code-behind (MainPage.xaml.cs)

```
namespace NavigationPages
{
    4 references
    public sealed partial class MainPage : Page
    {
        0 references
        public MainPage()...

        1 reference
        private void NavigateBtn_Click(object sender, RoutedEventArgs e)
        {
            //Navigate to the Next Page
            Frame.Navigate(typeof(Page2), "Je passe en page 2");
        }

        /// <summary> ...
        1 reference
        protected override void OnNavigatedTo(NavigationEventArgs e)...
    }
}
```

Sent value to page 2

# Forward Navigation

- Sample(…)
  - In Page2.xaml

```
<Page   x:Class="NavigationPages.Page2" …>
<Grid>
      <TextBlock Text="Page 2" Style="{StaticResource HeaderTextBlockStyle}"
                 Width="200" Margin="10,0,0,0"
                 HorizontalAlignment="Left" VerticalAlignment="Top" />
      <TextBlock x:Name="paramTextBlock" Text="" FontSize="36" Height="100"
                 Margin="10,219,25,321" TextWrapping="Wrap"/>
   </Grid>
</Page>
```

# Forward Navigation

- Sample(…)
  - In Page2.xaml.cs

```
namespace NavigationPages
{
    /// <summary> ...
    4 references
    public sealed partial class Page2 : Page
    {
        0 references
        public Page2()...

        /// <summary>
        /// Invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached
        /// This parameter is typically used to configure the page.</param>
        1 reference
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            paramTextBlock.Text = e.Parameter.ToString();
        }
    }
}
```

# Forward Navigation

- With MVVM Pattern
  - E.g, in *MainPage*.xaml

```
< Button  Command="{ Binding  GoToSecondPageCommand }"  … >
```

# Forward Navigation

- In *MainPageViewModel.cs*

In the MVVMLight toolkit, implementation of the ICommand interface.
It is used by the Execute method that the ICommand interface requires.
It's a lambda expression in our case.

```
private ICommand  _goToSecondPageCommand;
public ICommand  GoToSecondPageCommand
{
        get
        {
            if (_goToSecondPageCommand == null)
              _goToSecondPageCommand = new RelayCommand( () => GoToSecondPage() );
            return _goToSecondPageCommand;
        }
}
private void GoToSecondPage()
{
        (Application.Current.RootVisual as PhoneApplicationFrame).Navigate(typeOf(Page2), "
…");
 }
```

# Backward Navigation

- An app can execute logic to navigate back to the preceding page
- E.g, without MVVM pattern

```
<Grid>
      <TextBlock Text="Page 2" Style="{StaticResource HeaderTextBlockStyle}" Width="200"
                 Margin="10,0,0,0" HorizontalAlignment="Left" VerticalAlignment="Top" />
      <TextBlock x:Name="paramTextBlock" Text="" FontSize="36" Height="100"
                                          Margin="10,219,25,321" TextWrapping="Wrap"/>
      <Button x:Name="GotoBack" Content="GotoBack" HorizontalAlignment="Left"
                 Margin="251,380,0,0" VerticalAlignment="Top" Click="GotoBack_Click" />
 </Grid>
```

# Backward Navigation

- E.g. (…)

```
private void GotoBack_Click(object sender, RoutedEventArgs e)
{
        if (this.Frame.CanGoBack) this.Frame.GoBack();
}
```

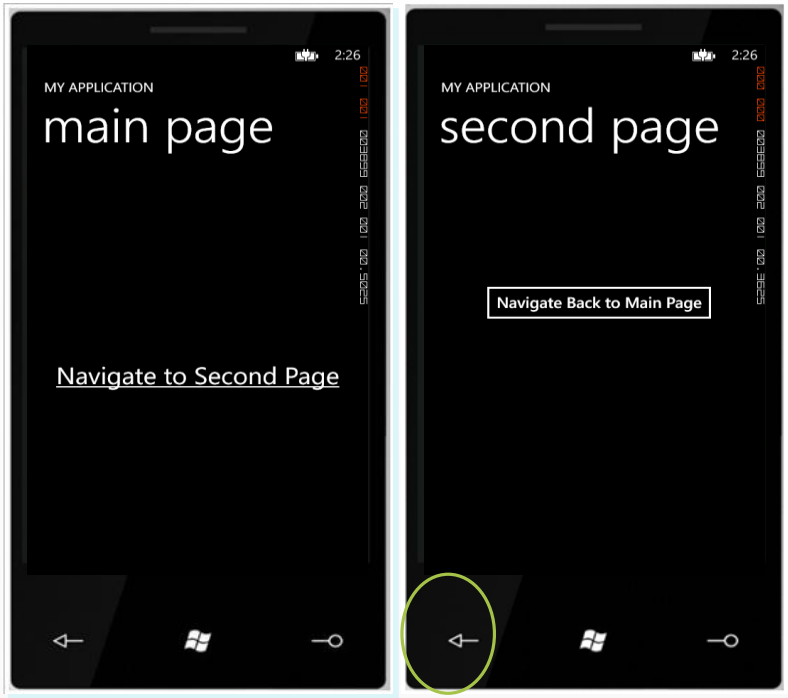
- **But …….**

# Navigation Bar Back Key

- Windows Phone devices have a Back button
  - To navigate backwards through the pages of the app
  - When the user has reached the first page of the app and presses the back button again, the OS
    - Will suspend the app
    - Navigates the user to the experience before the app was launched
      - Experience may be another app or it may be the Start screen

# Navigation Bar Back Key

- Standard Windows Phone app use the Back key
  - to navigate back
  - or to close transient UI




- By default, Back key causes a navigation back to the previous app (not to the previous page!) except for some type of pages

# Navigation Bar Back Key

- In a Windows Phone Store app, the programmer must include code to override this to cause a backward navigation within the app
- With correct Back key handling, pressing the Back key on the launch page suspends the current app and navigates back to previous app

# Navigation Bar Back Key

- Good
  - Handle the Back button if you want to navigate inside the app
    - Do handle the Back button to dismiss transient UI such as message boxes, menus, etc.
    - Do handle the Back button to navigate to the previous page on the navigation history for the current launch point

# Navigation Bar Back Key

- How to handle the Bar Back key?
- NavigationHelper.cs
  - Helps navigation between pages
    - Provides commands used to navigate back and forward as well as registers for standard mouse and keyboard shortcuts used to go back and forward
  - In addition, it integrates SuspensionManager to handle process lifetime management and state management when navigating between pages

# NavigationHelper Class

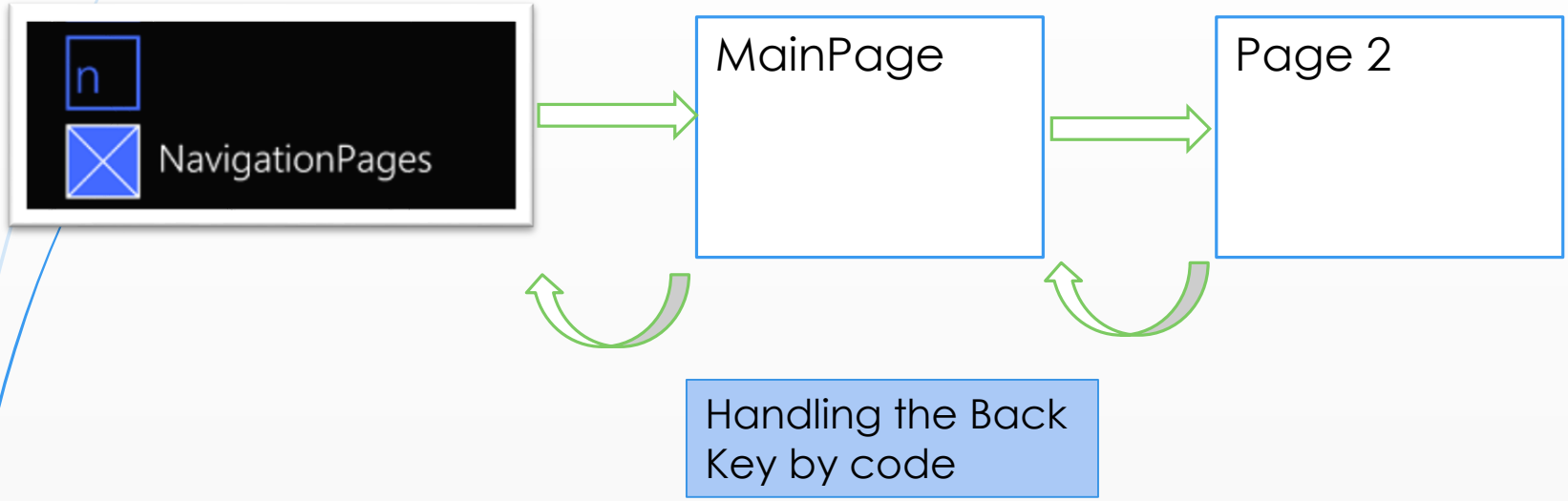
- Templates
  - Blank App
    - Does not include Back key handling
  - Hub App, Pivot App, Grid App, Split App
    - Does include Back key handling
    - *NavigationHelper* class automatically added to the solution
  - *NavigationHelper* class automatically added to an existing project if the developper add a *BasicPage* or any other Page item template other than *BlankPage*

# NavigationHelper Class

- But ….in every page, the developer must declare a property of NavigationHelper type
  - It must be instantiated in the constructor along with the registration of the events it raises
  - Call helper methods in *OnNavigatedTo/OnNavigatedFrom*
  - The code is repeated too much time!
    - We can centralize this code in a super class

# Navigation Bar Back Key

- How to do in another app?



# Navigation Bar Back Key

- E.g., without MVVM pattern,
  - in the Page2.xaml.cs,

```
public Page2()...

/// <summary> ...
1 reference
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    paramTextBlock.Text = e.Parameter.ToString();
    Windows.Phone.UI.Input.HardwareButtons.BackPressed +=  HardwareButtons_BackPressed;
}

1 reference
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    Windows.Phone.UI.Input.HardwareButtons.BackPressed -= HardwareButtons_BackPressed;
}

2 references
private async void HardwareButtons_BackPressed(object sender, Windows.Phone.UI.Input.BackPressedEventArgs e)
{
    e.Handled = true; // We've handled this button press
    // Standard page backward navigation
    if (this.Frame.CanGoBack)
        this.Frame.GoBack();
}
```

# Navigation Bar Back Key

- E.g. (…)
  - in the App.xaml.cs,

```
public sealed partial class App : Application
{
    private TransitionCollection transitions;

    /// <summary> ...
    1 reference
    public App()
    {
        this.InitializeComponent();
        this.Suspending += this.OnSuspending;
        HardwareButtons.BackPressed += HardwareButtons_BackPressed;
    }

    1 reference
    private async void HardwareButtons_BackPressed(object sender, BackPressedEventArgs e)
    {
        Frame frame = Window.Current.Content as Frame;
        if (frame == null) return;
        if (frame.CanGoBack)
        {
            frame.GoBack();
            e.Handled = true;
        }

        //throw new NotImplementedException();
    }
```

# Page Cache Mode

- When the user navigates to a Page type at the first time, a new instance is created
- When the user navigates farther in the app, the property *NavigationCacheMode* of the page determines if the state of the page will be saved, hidden or destroyed
  - Values :
    - *NavigationCacheMode.Disabled*
    - *NavigationCacheMode.Enabled*
    - *NavigationCacheMode.Required*

# Page Cache Mode

- *NavigationCacheMode.Disabled*
  - New instance of the page whether the user navigates forward or backward to the page
- *NavigationCacheMode.Enabled*
  - The page is cached
  - But the cached instance is discarded when the cache size for the frame is exceeded (determined by *Frame.CacheSize* property; by default on Windows Phone = 10)
- *NavigationCacheMode.Required*
  - The page is cached
  - The cached instance is reused for every visit regardless of the cache size for the frame