Lab 7 – Navigation

Mr. Yousif Garabet Arshak
Computer Science Department
University of Zakho
yousif.arshak@uoz.edu.krd

Outlines

- **Hierarchical Navigation**
- **TabbedPage**
- **FlyoutPage**

Navigation Pgae

Pushing Pages to the Navigation Stack

To navigate to Page2Xaml, it is necessary to invoke the PushAsync method on the Navigation property of the current page, as demonstrated in the following code example:

```
async void OnNextPageButtonClicked (object sender, EventArgs e)
{
  await Navigation.PushAsync (new Page2Xaml ());
}
```

- When the PushAsync method is invoked, the following events occur:
  - The page calling PushAsync has its OnDisappearing override invoked.
  - The page being navigated to has its OnAppearing override invoked.
  - The PushAsync task completes.

Popping Pages from the Navigation Stack

The active page can be popped from the navigation stack by pressing the *Back* button on the device, regardless of whether this is a physical button on the device or an on-screen button.

```
async void OnPreviousPageButtonClicked (object sender, EventArgs e)
{
  await Navigation.PopAsync ();
}
```

- This causes the Page2Xaml instance to be removed from the navigation stack, with the new topmost page becoming the active page. When the PopAsync method is invoked, the following events occur:
  - The page calling PopAsync has its OnDisappearing override invoked.
  - The page being returned to has its OnAppearing override invoked.
  - The PopAsync task returns.

As well as PushAsync and PopAsync methods, the Navigation property of each page also provides a PopToRootAsync method,

```
async void OnRootPageButtonClicked (object sender, EventArgs e)
{
  await Navigation.PopToRootAsync ();
}
```

Animating Page Transitions

```
async void OnNextPageButtonClicked (object sender, EventArgs e)
{
  // Page appearance not animated
  await Navigation.PushAsync (new Page2Xaml (), false);
}

async void OnPreviousPageButtonClicked (object sender, EventArgs e)
{
  // Page appearance not animated
  await Navigation.PopAsync (false);
}

async void OnRootPageButtonClicked (object sender, EventArgs e)
{
  // Page appearance not animated
  await Navigation.PopToRootAsync (false);
}
```

TabbedPage

- Two approaches can be used to create a TabbedPage:
  - Populate the TabbedPage with a collection of child Page objects, such as a collection of ContentPage objects. For more information, see Populate a TabbedPage with a Page Collection.
  - Assign a collection to the ItemsSource property and assign a DataTemplate to the ItemTemplate property to return pages for objects in the collection. For more information, see Populate a TabbedPage with a template.

Note: It's recommended that a TabbedPage should be populated with NavigationPage and ContentPage instances only. This will help to ensure a consistent user experience across all platforms.

Populate a TabbedPage with a Page collection

```
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-
namespace:TabbedPageWithNavigationPage;assembly=TabbedPageWithNavigationPage"
        x:Class="TabbedPageWithNavigationPage.MainPage">
  <local:TodayPage />
  <NavigationPage Title="Schedule" IconImageSource="schedule.png">
    <x:Arguments>
      <local:SchedulePage />
    </x:Arguments>
  </NavigationPage>
</TabbedPage>
```

- The following XAML code example shows a FlyoutPage that sets the Flyout and Detail properties:

```
<FlyoutPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-namespace:FlyoutPageNavigation;assembly=FlyoutPageNavigation"
        x:Class="FlyoutPageNavigation.MainPage">
    <FlyoutPage.Flyout>
        <local:FlyoutMenuPage x:Name="flyoutPage" />
    </FlyoutPage.Flyout>
    <FlyoutPage.Detail>
        <NavigationPage>
            <x:Arguments>
                <local:ContactsPage />
            </x:Arguments>
        </NavigationPage>
    </FlyoutPage.Detail>
</FlyoutPage>
```

FlyoutPage

- A FlyoutPage contains Flyout and Detail properties that are both of type Page, which are used to get and set the flyout and detail pages respectively.

Note
A FlyoutPage is designed to be a root page, and using it as a child page in other page types could result in unexpected and inconsistent behavior. In addition, it's recommended that the flyout page of a FlyoutPage should always be a ContentPage instance, and that the detail page should only be populated with TabbedPage, NavigationPage, and ContentPage instances. This will help to ensure a consistent user experience across all platforms.

Any Questions?