Tab and Flyout Navigations

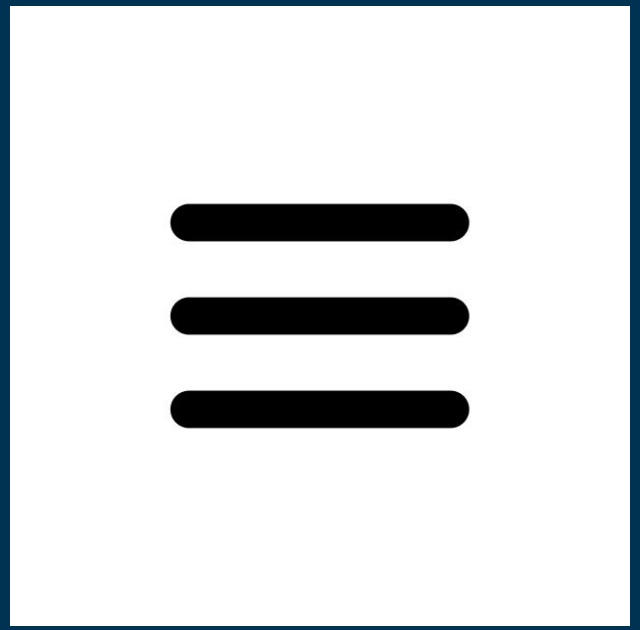
Introduction

- Part of planning your app's architecture is deciding how the user moves between pages:
  - Should users go forward and backward through a sequence of pages?
  - Does your app have one start page, or are there several top-level pages that are equally important?

- Your choice of navigation must work well with your app's content.
- It should also feel native to each platform you target.

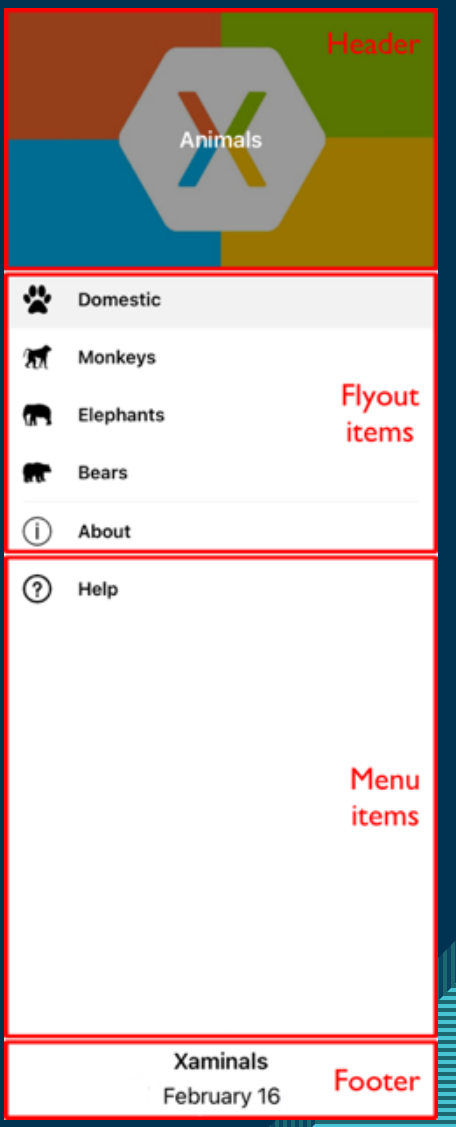
Flyout Navigation

- Flyout navigation is a type of navigation where a window of menu items slides (or flies out) from the side of the device's screen.
- It's invoked by tapping on what's called a hamburger menu, or an icon with three horizontal lines stacked on top of each other.



Flyout Navigation

- The flyout menu is composed of several parts:
  - Header
  - FlyoutItems
  - MenuItems
  - Footer



- Because the flyout menu isn't always visible, it can be used to switch context between conceptually different parts of your application.
- For example, one flyout item can lead to a data entry page, and another to an about page.

Flyout Navigation in a .NET MAUI App

- You use the `FlyoutItem` class to implement flyout navigation in .NET MAUI.
- Navigation with a flyout in .NET MAUI occurs when a `FlyoutItem` item is tapped.
- You specify what gets displayed when a `FlyoutItem` is tapped by setting its `ShellContent` property.
- That property points to a page in your application.
- The `FlyoutItem` needs to be hosted in a `Shell` page, which serves as your application's main page.
- And you can have as many `FlyoutItems` as you'd like.

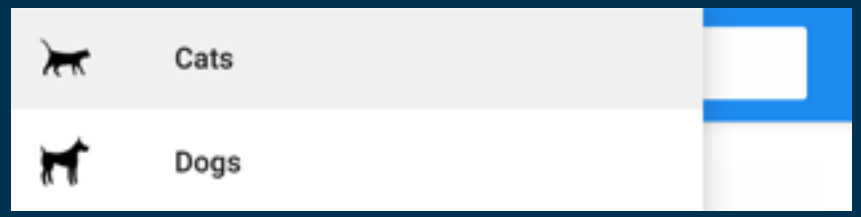
Flyout Navigation in a .NET MAUI App

- The following example creates a flyout menu containing two flyout items:

```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:controls="clr-namespace:Xaminals.Controls"
       xmlns:views="clr-namespace:Xaminals.Views"
       x:Class="Xaminals.AppShell">

    <FlyoutItem Title="Cats"
                Icon="cat.png">
       <Tab>
           <ShellContent ContentTemplate="{DataTemplate views:CatsPage}" />
       </Tab>
    </FlyoutItem>
    <FlyoutItem Title="Dogs"
                Icon="dog.png">
       <Tab>
           <ShellContent ContentTemplate="{DataTemplate views:DogsPage}" />
       </Tab>
    </FlyoutItem>
</Shell>
```

Simplify the XAML Code

- A `FlyoutItem` object represents each flyout item.
- Each `FlyoutItem` object should be a child of the subclassed `Shell` object that serves as your app's `MainPage`.
- The `Shell` object has implicit conversion operators that enable the Shell visual hierarchy to be simplified.
- This simplification is possible because a subclassed `Shell` object can only ever contain `FlyoutItem` objects or a `TabBar` object, which can only ever contain `Tab` objects, which can only ever contain `ShellContent` objects.
- These implicit conversion operators can be used to remove the `FlyoutItem` and `Tab` objects from the previous example.




```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:controls="clr-namespace:Xaminals.Controls"
       xmlns:views="clr-namespace:Xaminals.Views"
       x:Class="Xaminals.AppShell">
   <ShellContent Title="Cats"
                 Icon="cat.png"
                 ContentTemplate="{DataTemplate views:CatsPage}" />
   <ShellContent Title="Dogs"
                 Icon="dog.png"
                 ContentTemplate="{DataTemplate views:DogsPage}" />
</Shell>
```

Exercise

- In the sample scenario, you have a MAUI app that contains pages for displaying information about astronomical bodies, the phases of the moon, and sunrise/sunset times.
- The app also includes an About page.
- Currently, these pages are all stand-alone, but you want to provide a logical way for the user to move between them.
- In this exercise, you add flyout navigation to the app.

- This exercise uses the .NET 8.0 SDK.
  - Ensure that you have .NET 8.0 installed.

- Clone or download the exercise repo from Github:
  - https://github.com/MicrosoftDocs/mslearn-dotnetmaui-create-multi-page-apps

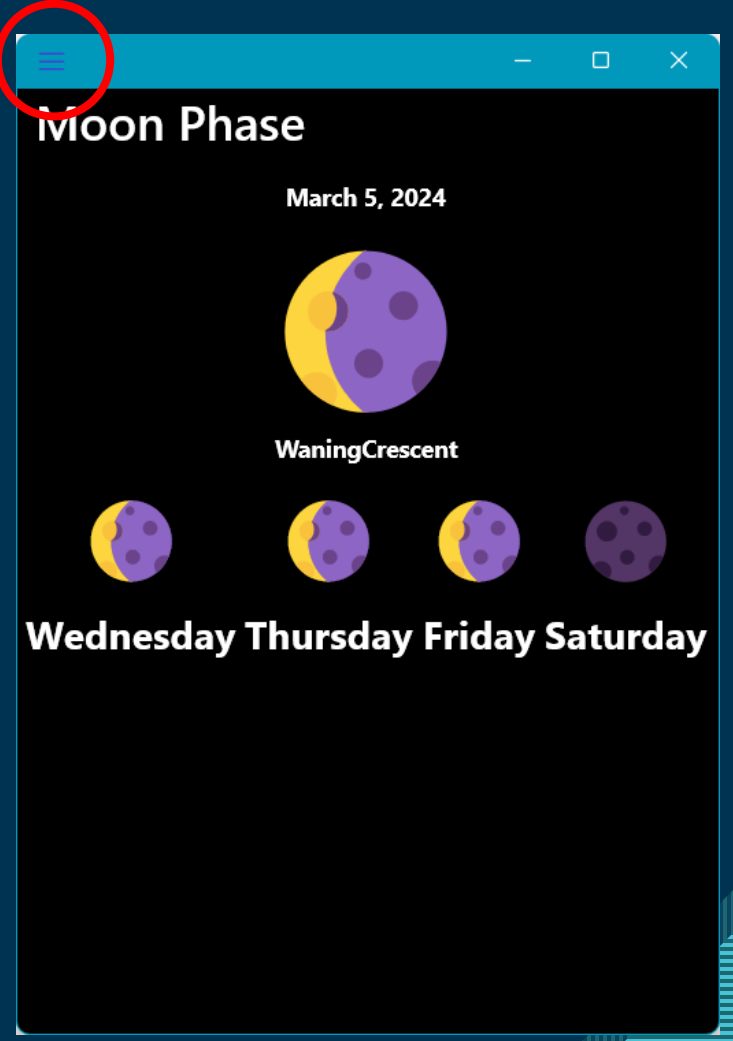
Exercise

- Go to the exercise1 folder in the cloned repo, and then move to the start folder.
- Use Visual Studio to open the Astronomy.sln solution.
- In the Solution Explorer window, in the Astronomy project, expand the Pages folder.
- This folder contains the following pages:
  - AboutPage: This page displays about information for the app.
  - MoonPhasePage: This page displays specific information about the phases of the Moon as seen from Earth.
  - SunrisePage: This page displays sunrise and sunset times for locations on Earth.
    - The data is provided by the Sunrise Sunset web service.
- Build and run the app.
- When the app starts, the MoonPhasePage displays, but currently there's no means provided to enable the user to navigate to the other pages.
- Close the app and return to Visual Studio.

Exercise: Add Flyout Navigation

- In the Solutions Explorer window, open up the AppShell.xaml page.
- In the XAML markup editor, surround the existing `<ShellContent>` item with a `<FlyoutItem>`. Set the `Title` property of the `<Flyout>` item to be Moon Phase. The markup should look like this:

```
<FlyoutItem Title="Moon Phase">
    <ShellContent
        ContentTemplate="{DataTemplate local:MoonPhasePage}"/>
</FlyoutItem>
```

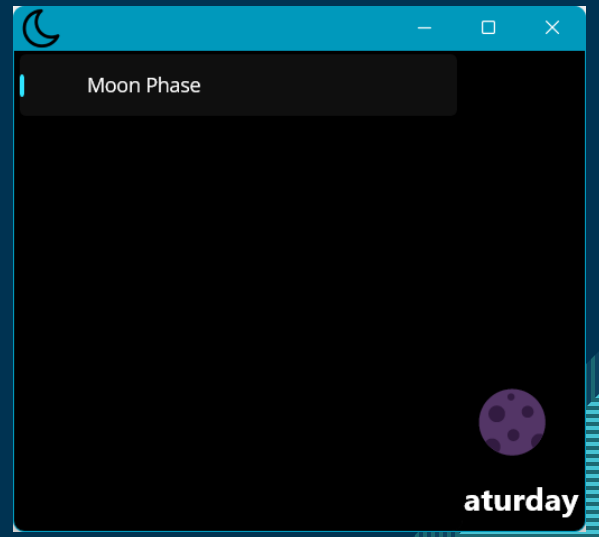


Exercise: Add Flyout Navigation

- Add a `FlyoutIcon` property to the `<Shell>` node to display an image.
- By default, it displays three horizontal bars, but we can change it to be whatever we like. The markup should look like this:

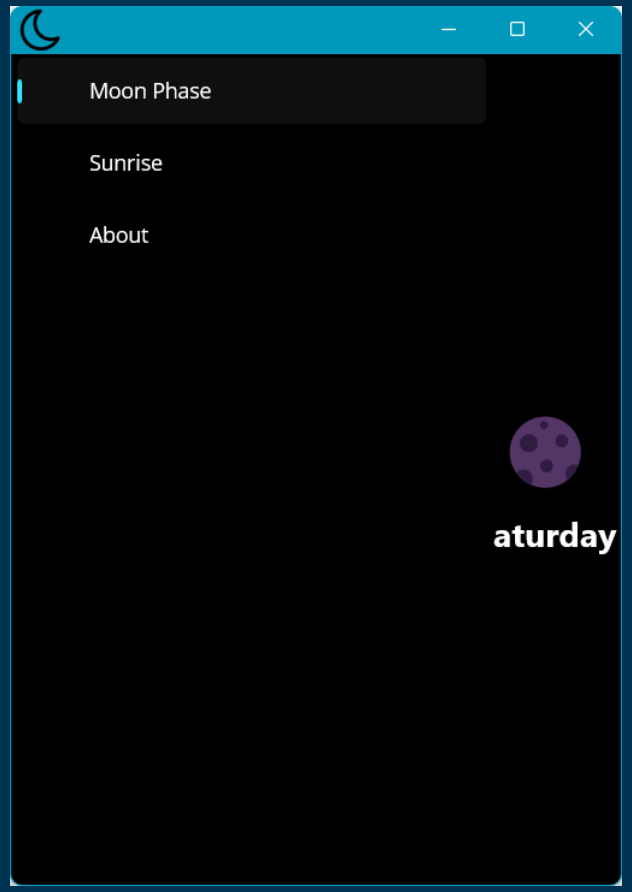
```
<Shell
    x:Class="Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:Astronomy.Pages"
    FlyoutIcon="moon.png">
```

- Run the application. You should now see a moon image in the upper left corner of the app.
- Tap on the icon and the flyout appears.



Exercise: Add Flyout Navigation

- Now add in more flyout options.
- Create a new `<FlyoutItem>` below the one you just made and set its `Title` to Sunrise.
- Its `ShellContent` should point to the SunrisePage page.
- Add another `<FlyoutItem>`, set its `Title` to About. This time set the `ShellContent` to AboutPage.

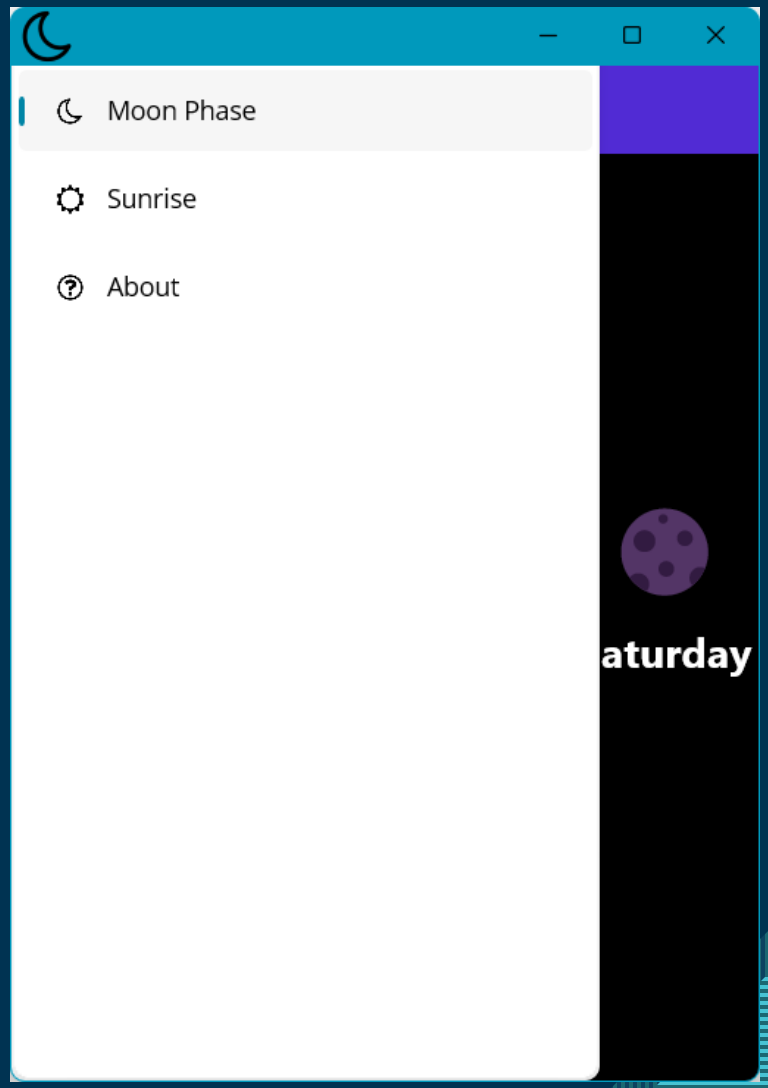



```
<FlyoutItem Title="Sunrise">
    <ShellContent
        ContentTemplate="{DataTemplate local:SunrisePage}"/>
</FlyoutItem>

<FlyoutItem Title="About">
    <ShellContent
        ContentTemplate="{DataTemplate local:AboutPage}"/>
</FlyoutItem>
```

Exercise: Adding Icons

- You might have noticed the flyout items look a bit empty.
- You can add icons to the flyout items by using the `Icon` property.
- Some images have already been added to the Resources\Images folder for you to use.
- Set the `Icon` property of the first `FlyoutItem` to moon.png.
- Repeat for the other 2 flyout items, using sun.png and question.png respectively.

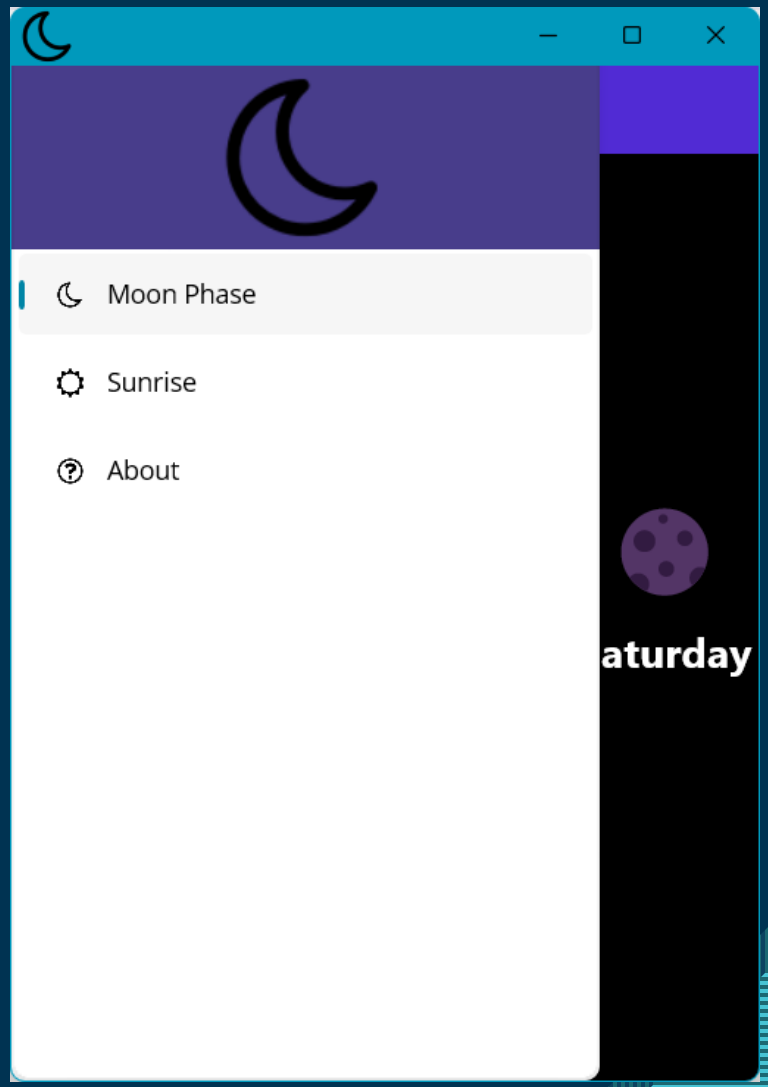
```
<FlyoutItem Title="Moon Phase" Icon="moon.png">
    <ShellContent
        ContentTemplate="{DataTemplate local:MoonPhasePage}" />
</FlyoutItem>

<FlyoutItem Title="Sunrise" Icon="sun.png">
    <ShellContent
        ContentTemplate="{DataTemplate local:SunrisePage}"/>
</FlyoutItem>

<FlyoutItem Title="About" Icon="question.png">
    <ShellContent
        ContentTemplate="{DataTemplate local:AboutPage}"/>
</FlyoutItem>
```



Exercise: Add a Flyout Header

- The flyout items are at the top of the flyout menu, making them difficult to distinguish.
- We can add some space to the top, and even an entire `View` by using the `<Shell.FlyoutHeader>`.
- Add a flyout header as a child of the `<Shell>` node.

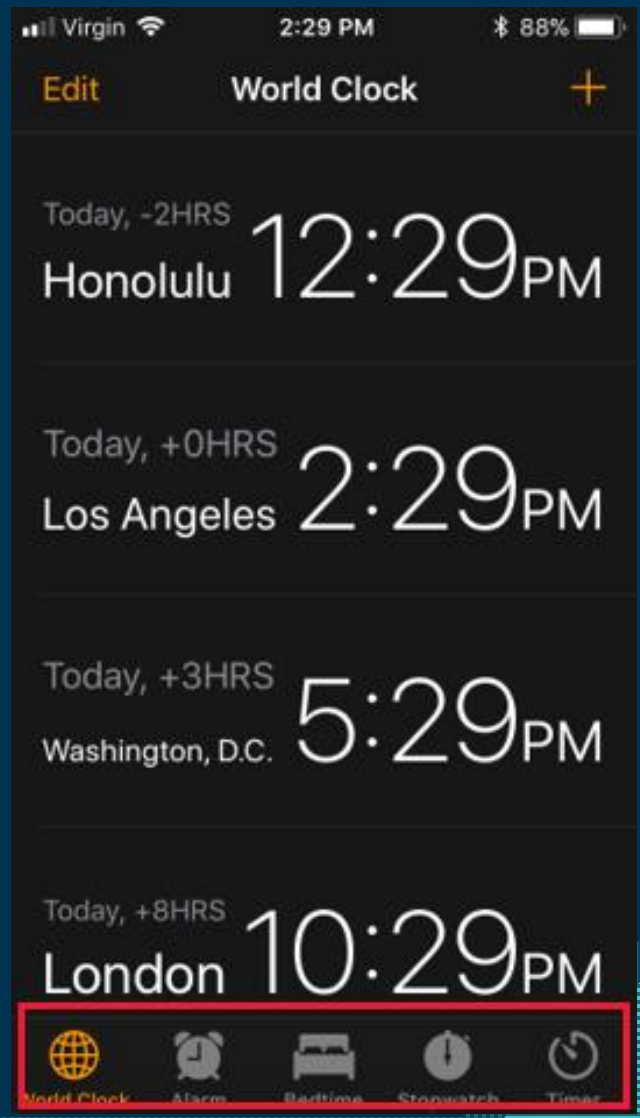
```
<Shell
    x:Class="Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:Astronomy.Pages"
    FlyoutIcon="moon.png">

    <Shell.FlyoutHeader>
        <Grid HeightRequest="100" BackgroundColor="DarkSlateBlue">
            <Image Source="moon.png" />
        </Grid>
    </Shell.FlyoutHeader>

    <FlyoutItem Title="Moon Phase" Icon="moon.png">
        <ShellContent
        ContentTemplate="{DataTemplate local:MoonPhasePage}" />
    </FlyoutItem>
```



Tab Navigation

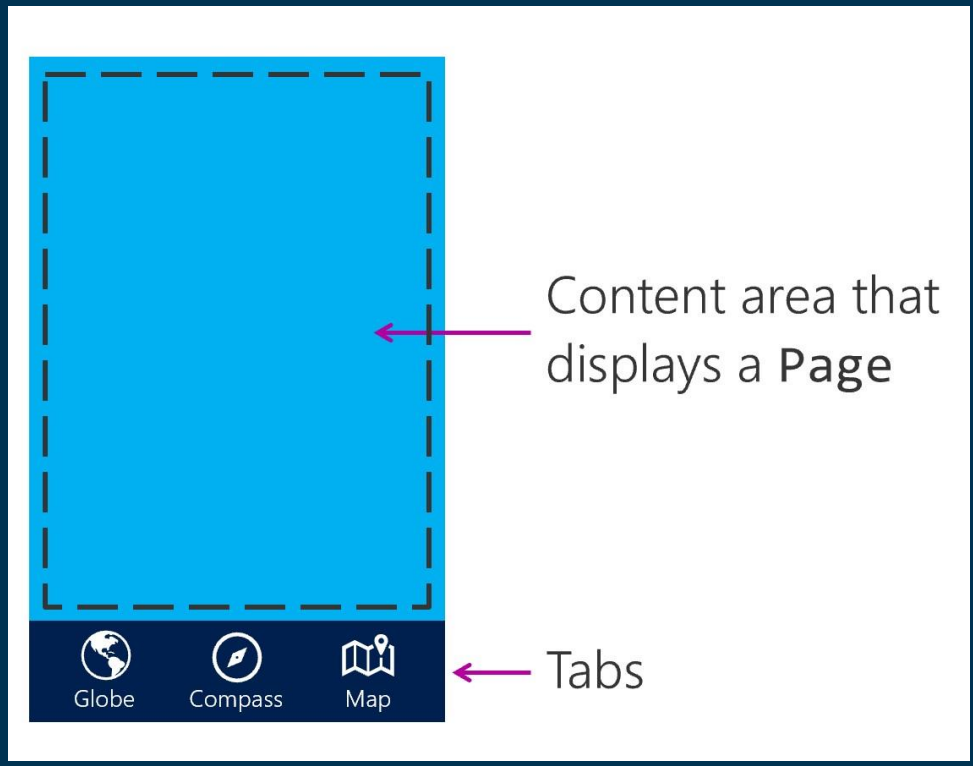
- Tab navigation is a navigation pattern where a tab strip (a row of touchable controls) is permanently displayed at the top or bottom of the screen.
- Tab navigation provides a mechanism for a user to select between pages in a multi-page app.
- As an example, this illustration shows the use of tab navigation in the iOS Clock app.
- The icons highlighted at the base of the page, enable you to switch between different views.

- Tab navigation is ideal when an application has several sections that a user is likely to use frequently.
- Clock applications are an excellent example.
- The clock, alarm, and stopwatch sections are likely to be frequently used.



Tab Navigation in a .NET MAUI App

- You use the `TabBar` object to implement tab navigation in a .NET MAUI shell app.
- The `TabBar` object displays a set of tabs and automatically switches the displayed content when the user selects a tab.
- To use tabs in a .NET MAUI Shell application, create an instance of the `TabBar` class as a child of the `Shell` class.
- Then add `Tab` objects to the `TabBar`.
- Within the `Tab` object, a `ShellContent` object should be set to a `ContentPage` object.



Create a TabbedPage

- You can create a `TabBar` instance as a child of the `Shell` class.
- Add `Tab` objects as children to the `TabBar` as needed.
- Within the `Tab` object, a `ShellContent` object should be set to a `ContentPage` object.

```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:views="clr-namespace:Xaminals.Views"
       x:Class="Xaminals.AppShell">
    <TabBar>
       <Tab Title="Moon Phase"
            Icon="moon.png">
           <ShellContent ContentTemplate="{DataTemplate local:MoonPhasePage}" />
       </Tab>
       <Tab Title="Sunrise"
            Icon="sun.png">
           <ShellContent ContentTemplate="{DataTemplate local:SunrisePage}" />
       </Tab>
    </TabBar>
</Shell>
```

Exercise

- In the astronomy app, you've been asked to combine tabs with the flyout to help with navigation between the different pages.
- The first thing you decide to do is remove all the pages from the flyout and add them to a `TabBar`, so you can see how the app feels.

Exercise: Adding a TabBar

- In the Solution Explorer window, open the AppShell.xaml page.
- In the XAML markup page, delete everything inside of the `<Shell>`.
- Create a `<TabBar>` and an empty `<Tab>`.

```
<Shell
    x:Class="Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:Astronomy.Pages"
    FlyoutIcon="moon.png">

    <TabBar>
        <Tab>

        </Tab>
    </TabBar>
</Shell>
```

Exercise: Adding a TabBar

- Next, add a `ShellContent` to the `Tab` and set its content to the `MoonPhasePage`.

```
<TabBar>
    <Tab>
        <ShellContent ContentTemplate="{DataTemplate local:MoonPhasePage}" />
    </Tab>
</TabBar>
```

- Now give the tab a title to be displayed and an icon using the `Title` and `Icon` properties.

```
<TabBar>
    <Tab Title="Moon Phase" Icon="moon.png">
        <ShellContent ContentTemplate="{DataTemplate local:MoonPhasePage}" />
    </Tab>
</TabBar>
```

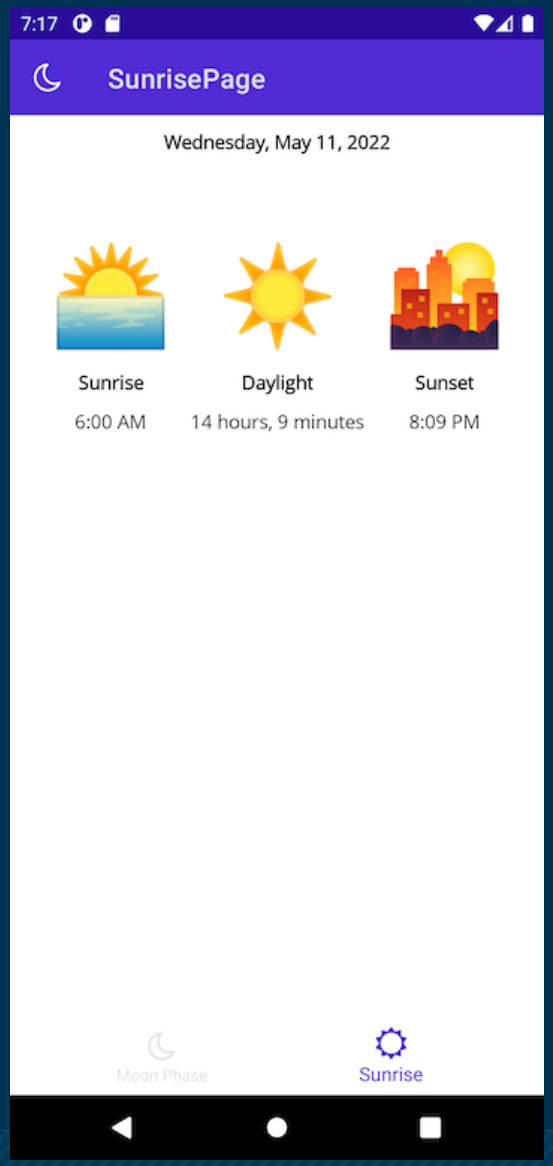
Exercise: Adding a TabBar

- Add in another `Tab` for the `SunrisePage`. Set its `Title` to sunrise and its `Icon` to sun.png.

```
<Shell
    x:Class="Astronomy.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:Astronomy.Pages"
    FlyoutIcon="moon.png">

    <TabBar>
        <Tab Title="Moon Phase" Icon="moon.png">
            <ShellContent ContentTemplate="{DataTemplate local:MoonPhasePage}" />
        </Tab>

        <Tab Title="Sunrise" Icon="sun.png">
            <ShellContent ContentTemplate="{DataTemplate local:SunrisePage}"/>
        </Tab>
    </TabBar>
</Shell>
```

Exercise: Adding a TabBar

- Run the app to see how it looks.



Exercise: Combine Tab Pages with a Flyout

- You decide it makes sense to have the moon phase and sunrise pages in the same tab page.
- And to keep the about page separate.
- So, you decide to add the flyout back in.
- The first flyout item displays the tab page, and the second the about page.

Exercise: Combine Tab Pages with a Flyout

- Delete the `TabBar` and all of the child items contained in it.
- In its place, add in a `<FlyoutItem>`.
- Set its `Title` property to Astronomy and `Icon` to moon.png.

```
<FlyoutItem Title="Astronomy" Icon="moon.png">

</FlyoutItem>
```

- Within the `<FlyoutItem>`, add a `<ShellContent>` that points to the MoonPhasePage.
- Set its `Title` property to Moon Phase and `Icon` property to moon.png.

```
<FlyoutItem Title="Astronomy" Icon="moon.png">
    <ShellContent Title="Moon Phase" Icon="moon.png"
        ContentTemplate="{DataTemplate local:MoonPhasePage}"/>
</FlyoutItem>
```

Exercise: Combine Tab Pages with a Flyout

- Within the same `<FlyoutItem>`, add another `<ShellContent>` to point to the SunrisePage.
- Set its `Title` property to Sunrise and `Icon` property to sun.png.

```
<FlyoutItem Title="Astronomy" Icon="moon.png">
    <ShellContent Title="Moon Phase" Icon="moon.png"
        ContentTemplate="{DataTemplate local:MoonPhasePage}"/>

    <ShellContent Title="Sunrise" Icon="sun.png"
        ContentTemplate="{DataTemplate local:SunrisePage}"/>
</FlyoutItem>
```

- Now, tapping on this flyout item displays a tab page with two tabs.

Exercise: Combine Tab Pages with a Flyout

- To create a new flyout item that points to the AboutPage, add a new `<FlyoutItem>`.
- Set its `Title` property to About and `Icon` property to question.png.
- Within that `<FlyoutItem>`, add a `<ShellContent>` that points to the AboutPage.

```
<FlyoutItem Title="About" Icon="question.png">
    <ShellContent
        ContentTemplate="{DataTemplate local:AboutPage}"/>
</FlyoutItem>
```

- Run the app again.
- You should see two items in the flyout.
- The first one opens up a tab page that contains the MoonPhasePage and SunrisePage.
- The second displays the AboutPage by itself.

Self Learn

- **Navigation Stack:**
  - For hierarchical data, stack navigation enables the user to drill down a series of pages, where the next page in the stack provides a more detailed view of a selected item on the previous page.

- Learn more:
  - https://learn.microsoft.com/en-us/training/modules/create-multi-page-apps/6-use-tabbed-pages-with-navigation-pages
- Exercise:
  - https://learn.microsoft.com/en-us/training/modules/create-multi-page-apps/7-exercise-use-tabbed-pages-with-navigation-pages

Thank You

Copyright

The materials provided in class and in SLATE are protected by copyright. They are intended for the personal, educational uses of students in this course and should not be shared externally or on websites such as Chegg, Course Hero or OneClass. Unauthorized distribution may result in copyright infringement and violation of Sheridan policies.

References

Material has been taken as is from:

- Microsoft Official Documentation:
  - https://learn.microsoft.com/en-us/training/modules/create-multi-page-apps/