

C# Mobile

Les 5

Shell: Tab & Flyout navigation



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be



Doelstellingen

- De junior-collega kan *tab* navigatie implementeren met .NET MAUI Shell
- De junior-collega kan navigeren tussen pagina's binnen *tabbed pages*
- De junior-collega kan *flyout* navigatie implementeren met .NET MAUI Shell
- De junior-collega kan naar een detail pagina navigeren via een aparte route in Shell



Intro

Intro

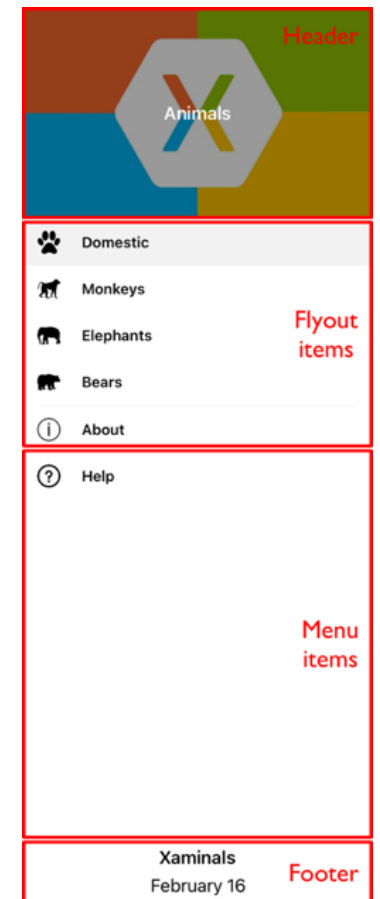
- Belangrijk onderdeel van elke applicatie: hoe zal de user navigeren?
 - Moeten de gebruikers door een hele reeks van pagina's kunnen heen-en-weer bewegen?
 - Is er één main-page of zijn er verschillende pages die 'even belangrijk' zijn?
- De keuze is afhankelijk van de inhoud van de app
- Ideaal: *native* gevoel van navigeren op het platform



Flyout navigation

Wat is flyout navigation?

- Flyout navigation geeft een menu weer in de applicatie dat snel opgeroepen kan worden via:
 - ‘Hamburger’ icoontje
 - Een veegbeweging van links naar rechts
- Het menu bestaat uit enkele onderdelen:
 - Header
 - FlyoutItems
 - MenuItem
 - Footer
- Het menu is niet altijd zichtbaar, maar meestal wel altijd beschikbaar
- Wordt gebruikt om een totaal andere pagina/inhoud te tonen die niet per se in de huidige context hoort



Flyout navigation in een .NET MAUI app

- Gebruik de `FlyoutItem` class om elementen in de flyout te definiëren
- Indien de user op een `FlyoutItem` drukt zal navigatie plaatsvinden
- De `ShellContent` property van het item bepaalt welke pagina ingeladen zal worden
- Een `FlyoutItem` moet in een `Shell` page staan
- Meerdere `FlyoutItems` zijn mogelijk

Flyout navigation in een .NET MAUI app

```
<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>
```

Opbouw:

<FlyoutItem>

<Tab>

<Shellcontent>

Een flyout maken

- In **AppShell.xaml**
- Eén of meerdere items kunnen worden toegevoegd door `FlyoutItem` objecten aan te maken
- De pagina waar dit item naar verwijst zal de zichtbare pagina worden van de app indien er op geklikt of gedrukt wordt
- Voorgaande voorbeeld kan vereenvoudigd worden:
 - Shell kan enkel `FlyoutItem(s)` of een `TabBar` object bevatten
 - Deze bevatten op hun beurt enkel `Tab` objecten
 - Een `Tab` kan enkel een `ShellContent` bevatten

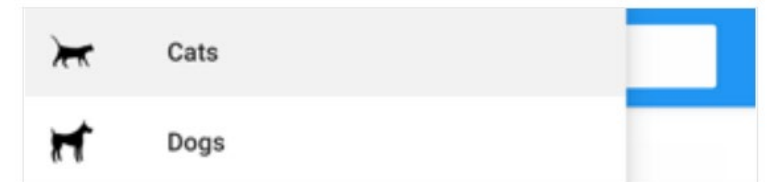
Een flyout maken

- We kunnen de `FlyoutItem` and `Tab` objects weglaten in de XAML:

```
<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>
```



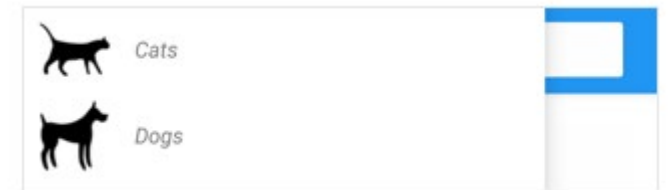
```
<ShellContent Title="Cats"
              Icon="cat.png"
              ContentTemplate="{DataTemplate views:CatsPage}" />
<ShellContent Title="Dogs"
              Icon="dog.png"
              ContentTemplate="{DataTemplate views:DogsPage}" />
```



Aanpassen layout

- Standaard FlyoutItem kan (beperkt) aangepast worden via attached property “Shell.ItemTemplate” met daarin een DataTemplate
 - Zie CollectionView en de [documentatie](#)
- Er zijn enkel bindings voor *FlyoutIcon* en *Title*

```
XAML Copy  
  
<Shell ...>  
  ...  
  <Shell.ItemTemplate>  
    <DataTemplate>  
      <Grid ColumnDefinitions="0.2*,0.8*">  
        <Image Source="{Binding FlyoutIcon}"  
          Margin="5"  
          HeightRequest="45" />  
        <Label Grid.Column="1"  
          Text="{Binding Title}"  
          FontAttributes="Italic"  
          VerticalTextAlignment="Center" />  
      </Grid>  
    </DataTemplate>  
  </Shell.ItemTemplate>  
</Shell>
```



Aanpassen layout

- Kan ook via Styles
 - Bv in Resources/Styles/Styles.xaml



The screenshot shows a code editor window titled 'XAML' with a 'Copy' button in the top right corner. The code defines three styles for a 'FlyoutItem' in a XAML file. The first style targets a 'Label' and sets its text color to black and height to 100. The second style targets an 'Image' and sets its aspect to fill. The third style targets a 'Layout' and sets its background color to teal, with a note that it applies to derived types.

```
<Style TargetType="Label"
      Class="FlyoutItemLabelStyle">
    <Setter Property="TextColor"
      Value="Black" />
    <Setter Property="HeightRequest"
      Value="100" />
</Style>

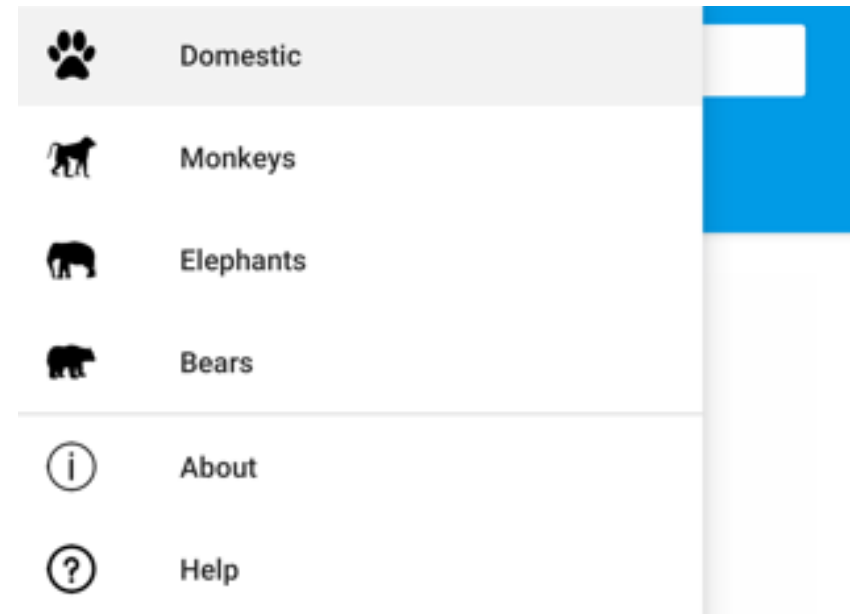
<Style TargetType="Image"
      Class="FlyoutItemImageStyle">
    <Setter Property="Aspect"
      Value="Fill" />
</Style>

<Style TargetType="Layout"
      Class="FlyoutItemLayoutStyle"
      ApplyToDerivedTypes="True">
    <Setter Property="BackgroundColor"
      Value="Teal" />
</Style>
```

- Of volledig zelfgemaakte items ([documentatie](#))

Flyout menu items

- Menu items zijn optioneel
- Worden aangemaakt met een `MenuItem` object
- Ze triggeren geen navigatie maar voeren een *command* uit
 - We komen hier nog op terug
 - Voorbeeld:



Flyout header en footer

- The flyout header wordt (optioneel) bovenaan het flyoutmenu getoond
- Voorbeeld:

```
<Shell ...>
  <Shell.FlyoutHeader>
    <Grid>
      <Image Source="header-image.png">
    </Grid>
  </Shell.FlyoutHeader>
</Shell>
```

Flyout header en footer

- The flyout footer wordt (optioneel) aan de onderkant van het flyoutmenu getoond
- Voorbeeld:

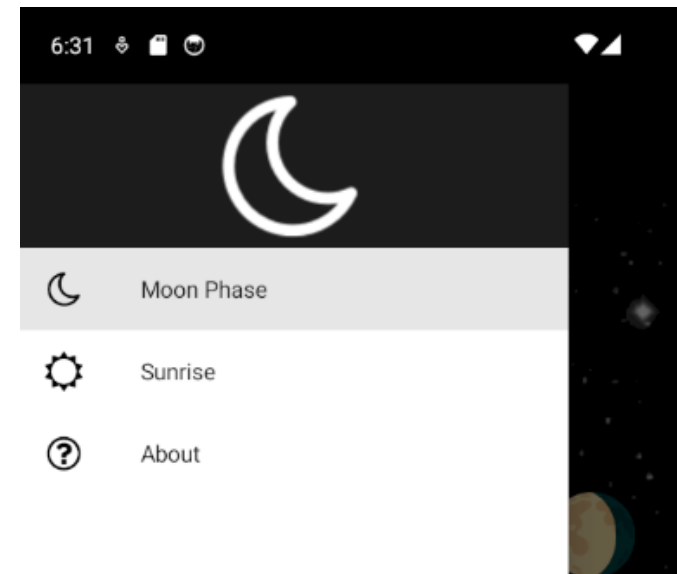
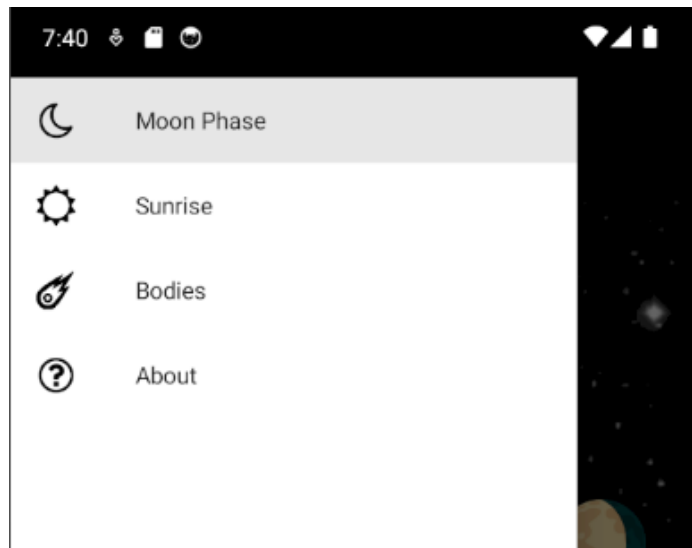
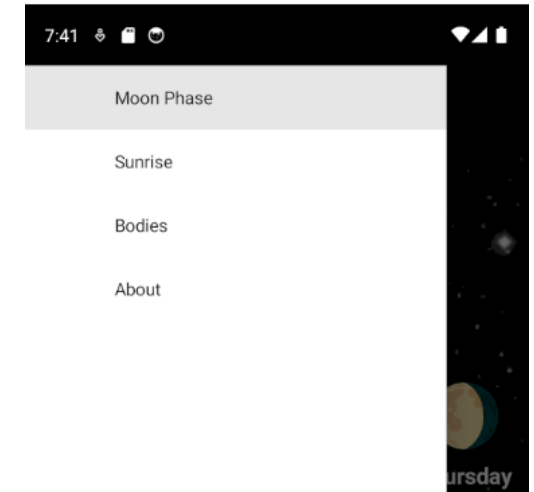
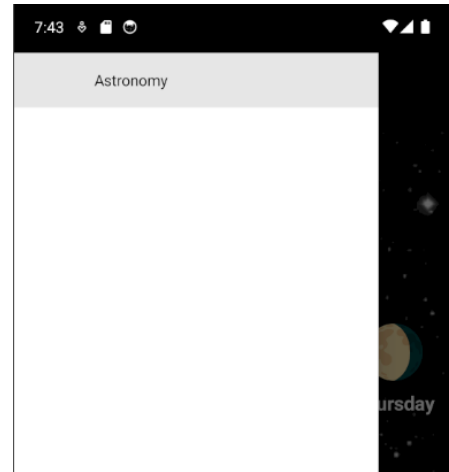
```
<Shell ...>
  <Shell.FlyoutFooter>
    <Grid>
      <Image Source="footer-image.png">
    </Grid>
  </Shell.FlyoutFooter>
</Shell>
```

Flyout styling

- Er zijn nog een aantal zaken die we kunnen instellen op het menu:
 - Breedte en hoogte
 - Icoontje
 - Achtergrond
 - Backdrop
 - ...

Zie [documentatie](#)

Oefening 1: Zie Lab

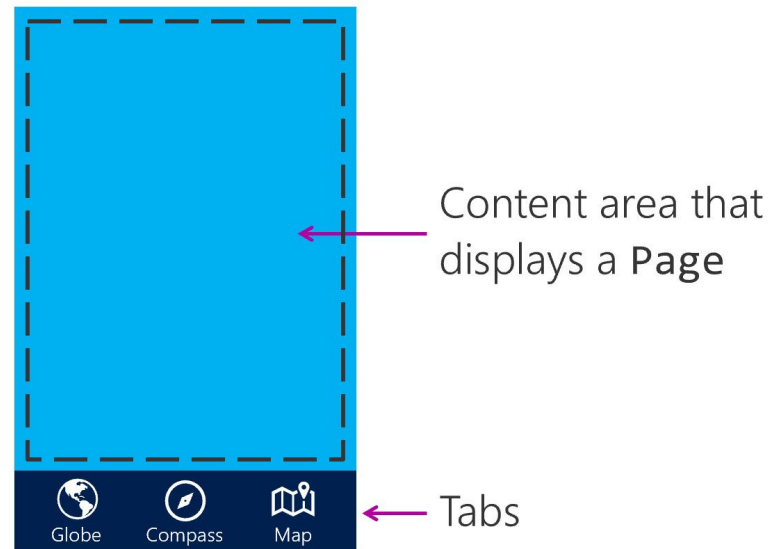




Tab navigation

Tab navigation

- Er wordt een tabbar getoond op het scherm die altijd in beeld blijft
- De tabbar staat onder- of bovenaan het scherm
- Via de tabs kan de gebruiker tussen verschillende pagina's in de app navigeren



Tab navigation: voorbeeld

- Om een tabbar te maken in een .NET MAUI Shell applicatie:
 1. Voeg een `TabBar` toe in `Shell.xaml`
 2. Voeg `Tab` objecten toe aan de `TabBar`
 - Bevat het icon en title van de tab
 3. Binnen het `Tab` object plaats je een `ShellContent` object dat verwijst naar een `ContentPage`

```
<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>
```

Combinatie Tabs en Flyout

- Een flyout item kan een pagina openen met een tab bar
 - Binnen een `<FlyoutItem>` van het flyoutmenu voeg je **meerdere** `<ShellContent>` items toe voor elke tab die je wil laten zien
 - Title en Icon op het `<ShellContent>` zetten om een titel en icon in de tab bar te tonen
- Voorbeeld:

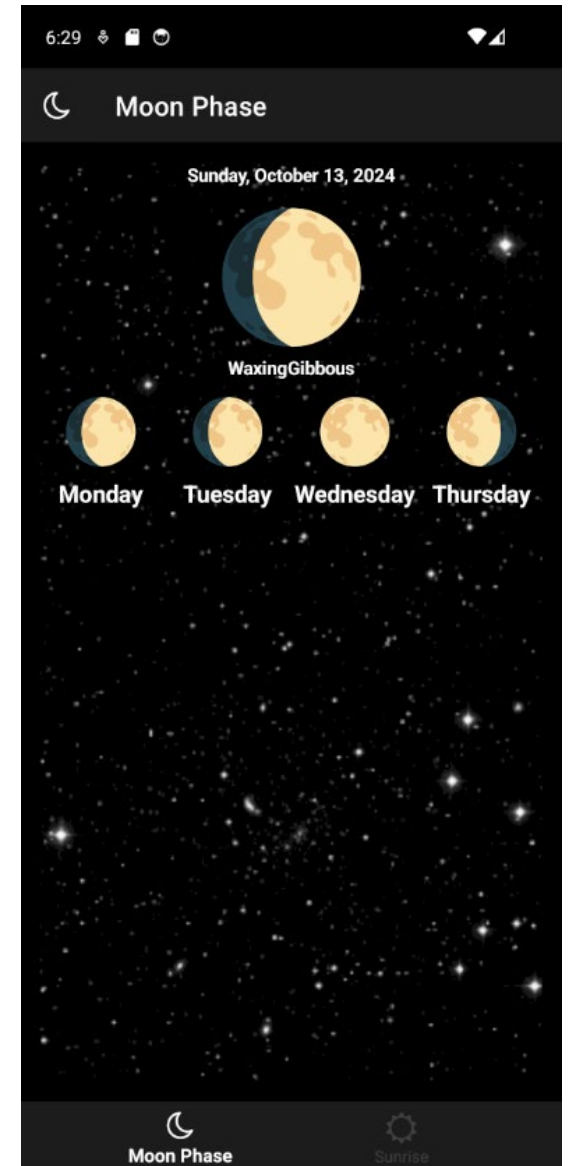
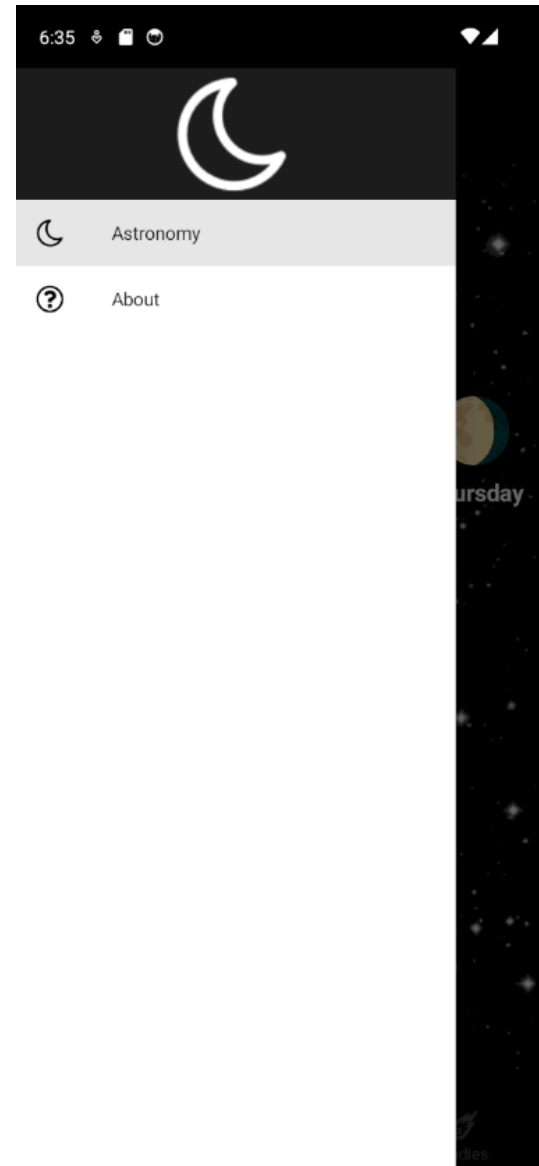
```
<FlyoutItem Title="Astronomy" Icon="moon.png">
  <ShellContent Title="Moon Phase" Icon="moon.png"
    ContentTemplate="{DataTemplate local:MoonPhasePage}"/>

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

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

Oefening 2: Zie Lab

- Combinatie Flyout en Tabs





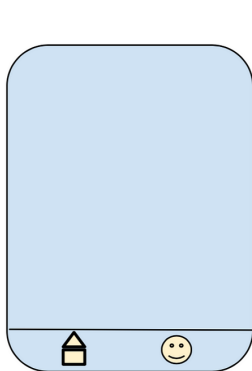
Stacked navigation

Combinatie tabbed pages met stacked navigation

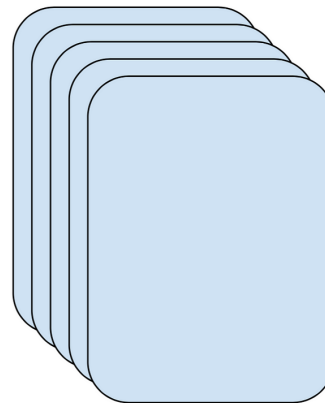
- Flyout navigation
- Tabbed navigation
- **Stack Navigation**
 - Dit is de navigatie die eerder behandeld is (RegisterRoute, GoToAsync)

Review tab navigation & stack navigation

- Flyouts en tab navigation veranderen de hele pagina
 - Handig om te switchen tussen grote onderdelen van de app die niet per se rechtstreeks iets met elkaar te maken hebben
 - Voorbeeld: In de SolarSystemApp
 - De Sun, Moon en About pages zijn 'even belangrijk', er is geen rechtstreekse relatie in de app tussen de verschillende pagina's



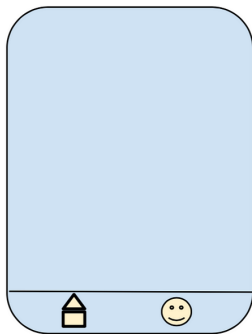
Tab-Navigation



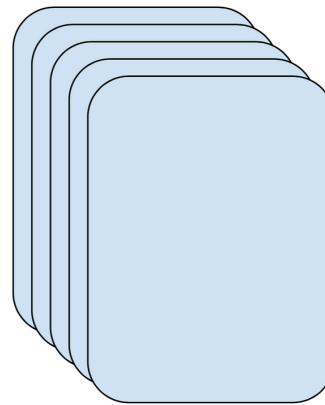
Stack-Navigation

Review tab navigation & stack navigation

- Voor hiërarchische data is stack navigation meestal geschikter
 - Via stack navigation kan de user dieper ingaan op een bepaald onderdeel van de app
 - Bijvoorbeeld: een detail pagina wordt geopend die meer informatie toont van een item op de hoofdpagina



Tab-Navigation



Stack-Navigation

Navigatie met stack navigation

- .NET MAUI Shell bevat een URI-based navigation (lijkt op URLs) waarmee routing tussen de verschillende pagina's mogelijk is

Format	Description
<code>route</code>	The route hierarchy will be searched for the specified route, upwards from the current position. The matching page will be pushed to the navigation stack.
<code>/route</code>	The route hierarchy will be searched from the specified route, downwards from the current position. The matching page will be pushed to the navigation stack.
<code>//route</code>	The route hierarchy will be searched for the specified route, upwards from the current position. The matching page will <u>replace</u> the navigation stack.
<code>///route</code>	The route hierarchy will be searched for the specified route, downwards from the current position. The matching page will <u>replace</u> the navigation stack.

- Er is ook de mogelijkheid om terug te navigeren

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/navigation>

Navigatie met stack navigation

- Shell biedt een aantal properties en functionaliteit:
 - `BackButtonBehavior`: bepaalt het gedrag van de back button
 - `CurrentItem`: een property die de huidige geselecteerde item bijhoudt
 - `CurrentPage`: bevat de pagina (page) die op dit moment getoond wordt
 - `Current`, van het type `Shell`, een type-casted alias for `Application.Current.MainPage`
 - ...
- Navigation wordt uitgevoerd door de `GoToAsync` method, van de `Shell` class:
 - Hiervoor kunnen we `Current` gebruiken

Routes

- Navigatie wordt uitgevoerd door een URI mee te geven
- Navigation URIs kunnen drie componenten hebben:
 - Een ***route***
 - Een geregistreerde route naar een onderdeel van de app
 - Deze kan relatief of absoluut zijn
 - Een ***page***
 - Pages die niet gedefinieerd zijn in Shell kunnen rechtstreeks meegegeven worden
 - Worden via een *push* op de navigation stack geplaatst
 - Een of meer ***query parameters***
 - Query parameters zijn parameters die doorgegeven kunnen worden aan de pagina waarnaar genavigeerd wordt

Routes registreren

- Routes kunnen ingesteld worden op een `FlyoutItem`, `TabBar`, `Tab` en `ShellContent` object met de `Route` property:

```
<Shell ...>
  <FlyoutItem ...
    Route = "astronomy">
      <ShellContent ...
        Route="moonphase" />
      <ShellContent ...
        Route="sunrise" />
    </FlyoutItem>
  <FlyoutItem>
    <ShellContent ...
      Route="about" />
    </FlyoutItem>
</Shell>
```

→ Voorbeeld: om naar de moonphases te navigeren kan je de absolute route URI gebruiken: `//astronomy/moonphase`

Registrerer detail routes

- In de `Shell` subclass constructor, en eigenlijk overal in de applicatie, kunnen extra routes geregistreerd worden die niet in `AppShell.xaml` staan

→ Dit kan met de `Routing.RegisterRoute` method:

```
Routing.RegisterRoute("astronomicalbodydetails", typeof(AstronomicalBodyPage));
```

→ Om te navigeren naar deze page:

```
await Shell.Current.GoToAsync("astronomicalbodydetails");`
```

Backwards navigation

- Backwards navigation kan gedaan worden door ".." als argument van de `GoToAsync` method te geven:

```
await Shell.Current.GoToAsync("..");
```

Passing data

- Simpele data kan als *string-based query parameters* doorgegeven worden via de URI

→ We voegen een ? toe na de route, gevolgd door een query parameter ID, =, en een waarde:

```
string celestialName = "moon";  
  
await Shell.Current.GoToAsync($"astronomicalbodydetails?bodyName={celestialName}");
```

→ De route is hier *astronomicalbodydetails*, de parameter is *bodyName* en de waarde wordt opgehaald uit `celestialName`.
(=="moon" in dit voorbeeld)

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/navigation#pass-data>

Passing data

- Objecten kunnen doorgegeven worden via een `IDictionary<string, object>` argument:
→ Hierbij is string de naam van het object (ID) en de waarde het object:

```
C# Copy  
  
async void OnCollectionViewSelectionChanged(object sender, SelectionChangedEventArgs e)  
{  
    Animal animal = e.CurrentSelection.FirstOrDefault() as Animal;  
    var navigationParameter = new Dictionary<string, object>  
    {  
        { "Bear", animal }  
    };  
    await Shell.Current.GoToAsync($"beardetails", navigationParameter);  
}
```

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/navigation#pass-data>

Retrieving data

- Navigation data kan opgehaald worden m.b.v. het [QueryPropertyAttribute] boven de klasse voor elke string-based query parameter and object-based navigation:

```
[QueryProperty(nameof(Bear), "Bear")]
public partial class BearDetailPage : ContentPage
{
    Animal bear;
    public Animal Bear
    {
        get => bear;
        set
        {
            bear = value;
            OnPropertyChanged();
        }
    }


    public BearDetailPage()
    {
        InitializeComponent();
        BindingContext = this;
    }
}
```

- Opgelet:** QueryProperty heeft 2 argumenten
- Het eerste argument bevat de property die de waarde zal opvangen
 - Het tweede argument is de gekozen ID van de parameter

Retrieving data (alternatief)

- Alternatief kunnen we de ontvangende klasse de IQueryable interface laten implementeren:

C#

 Copy

```
public class MonkeyDetailViewModel : IQueryable, INotifyPropertyChanged
{
    public Animal Monkey { get; private set; }

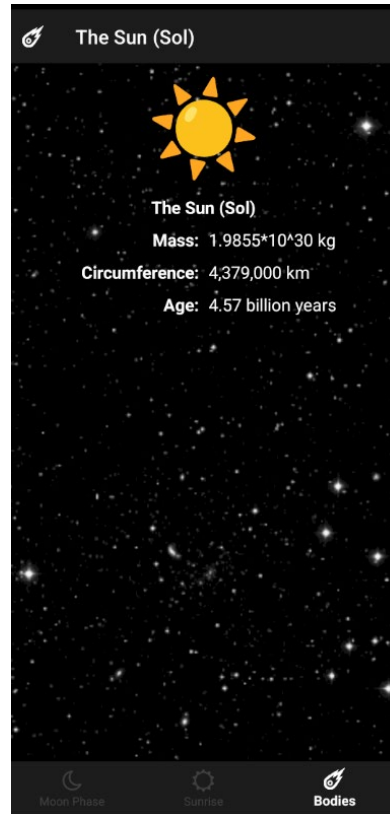
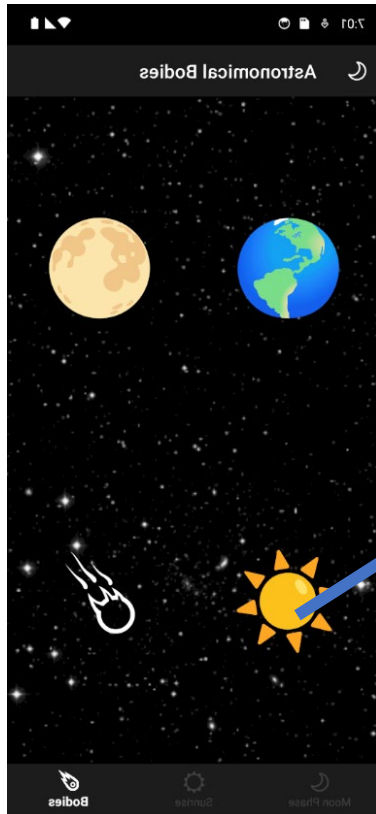
    public void ApplyQueryAttributes(IDictionary<string, object> query)
    {
        Monkey = query["Monkey"] as Animal;
        OnPropertyChanged("Monkey");
    }
    ...
}
```

Opgelet:

We moeten de ontvangen waarde nog omzetten naar het juiste type

Oefening 3

- Zie Lab deel 3



Extra: Navbar verbergen

- De navbar kan verborgen worden
- Op de ContentPage waar je deze niet meer wil zien:

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  
             x:Class="ShellExample.Views.WaterView"  
             Title="WaterView"  
             Shell.NavBarIsVisible="False">
```