Week3 – MasterDetail page demo

There is a variety of ways to navigate through Xamarin.Forms, on which more can be found in your eBook – **chapter 24: Page navigation**. You can find various demo's in this repository to demonstrate different page types and navigation.

In this document, we will explain how to use a **MasterDetailPage**. This page type allows you to create a master detail situation (for more on this, we refer to your theory notes). Before you go through this demo, we advice you to look at the NavigationPage demo.

Page type(s)

A masterdetail page requires 3 parts:

- A **master page** (ContentPage) which contains the master data
- A **detail page** (ContentPage) which relies on the master page for detail data
- A **masterdetail page** (MasterDetailPage) which does nothing else than connect the master and detail page together.

How to create a MasterDetail page

Do NOT create a MasterDetailPage by adding a new item of type MasterDetailPage! This will create 4 items (3 pages and a class), complicating everything more than it should.

1. **Turn a ContentPage into a MasterDetailPage**. This will do nothing else than connect the master and detail together.
2. **Define your Master page**, which is a ContentPage that contains the master data, **and** your **Detail page**, which is a ContentPage that contains the detail data.
3. **Refer from MasterDetailPage to Master page and Detail page** from steps 2 and 3.

1. Turn a ContentPage into a MasterDetailPage

- Either create new ContentPage (see demo NavigationPage), or change MainPage.xaml.cs
- In your xaml.cs code behind file, inherit from MasterDetailPage instead of ContentPage:

```
public partial class MainPage : MasterDetailPage
```

- In your xaml file: create this structure (for now, we leave Master and Detail empty):

```
<MasterDetailPage xmlns=http://xamarin.com/schemas/2014/forms
        … >
    <MasterDetailPage.Master>

    </MasterDetailPage.Master>
    <MasterDetailPage.Detail>
        <NavigationPage>
            <x:Arguments>

            </x:Arguments>
        </NavigationPage>
    </MasterDetailPage.Detail>
</MasterDetailPage>
```

2. Define your Master and Detail page

- Create a **ContentPage** for both.
- Give both of them a value for the **Title** property.
- Give them their desired content (most of the time, both will contain a listview).
- Example:

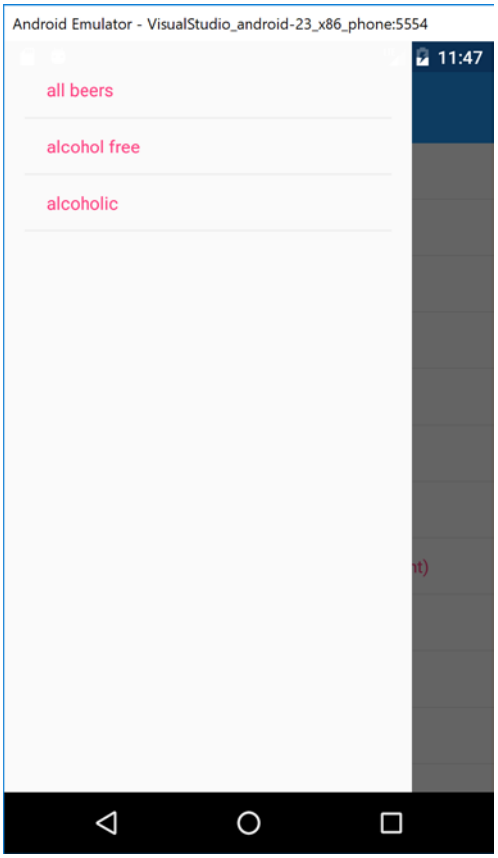

Figure 2: Master page - filters for detail page

Figure 1: Detail page - show beers filtered on what was chosen in the Master page

3. Refer from MasterDetailPage to Master & Detail page

- We will now go back to our MasterDetailPage (step 1) and fill the gaps of Master and detail.
- For the example: looking at the above screenshots, you can find **3** parts:
  - The **master page** (ContentPage) which is a sort of filter menu
  - The **detail page** (ContentPage) which relies on the master page
  - The **masterdetail page** (MasterDetailPage) which does nothing else than connect the master and detail page together. It looks like this:

```
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:view="clr-namespace:Demo_MasterDetail.View" … >
    <MasterDetailPage.Master>
        <view:MasterPage x:Name="masterPage" />
    </MasterDetailPage.Master>
    <MasterDetailPage.Detail>
        <NavigationPage>
            <x:Arguments>
                <view:DetailPage x:Name="detailPage" />
            </x:Arguments>
        </NavigationPage>
    </MasterDetailPage.Detail>
</MasterDetailPage>
```

- In this case, the master and detail page are literally called 'MasterPage.xaml' and 'DetailPage.xaml', but you can obviously choose whichever name you like.
- Both pages are inside a 'View' folder in the PCL project, as you can see in the referencing of the 'view' namespace.
- Instead of referencing to a different page, you can also set the content of the page(s) inside the MasterDetailPage.x tags, as you can see being done for the master page in the demo:

```
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
                  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                  xmlns:local="clr-namespace:Demo_MasterDetail"
                  xmlns:view="clr-namespace:Demo_MasterDetail.View"
                  x:Class="Demo_MasterDetail.MainPage">

    <MasterDetailPage.Master>
        <!--master page: list of filters (all, alcohol free, alcoholic)-->
        <ContentPage Title="Beer filters">
            <ListView Margin="16" x:Name="lvwFilters" ItemSelected="LvwFilters_ItemSelected" />
        </ContentPage>
    </MasterDetailPage.Master>

    <MasterDetailPage.Detail>
        <NavigationPage>
            <x:Arguments>
                <!--reference to the detail page-->
                <view:DetailPage_Overview x:Name="detailPage" />
            </x:Arguments>
        </NavigationPage>
    </MasterDetailPage.Detail>

</MasterDetailPage>
```

- In the code behind of the masterpage (in this case: mainpage.xaml.cs), you connect the master and detail page together:

```
private void LvwFilters_ItemSelected(object sender, …)
{
    string filter = lvwFilters.SelectedItem as string;

    if (filter == null)
        return; //nothing is selected; ignore

    detailPage.FilterBeerList(filter);  //filter details based on masterpage

    IsPresented = false;    //HIDE the menu
}
```

  - Notice the IsPresented property of a MasterDetailPage, that hides the menu again (otherwise it just stays hovered in front of your detailpage).