

XAML Basics/Stack Layout Lab - MAUI

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Create a .NET MAUI Application

Open Visual Studio 2022 and select Create a new project as shown in Figure 1.

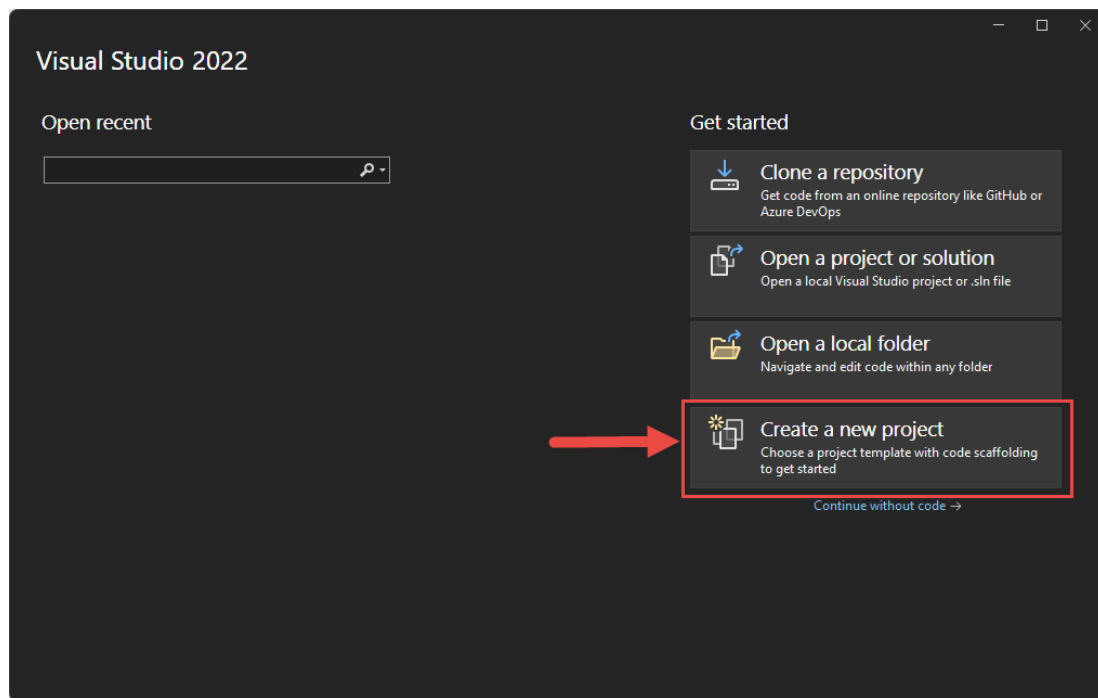


Figure 1: Select Create a new project from the Visual Studio screen.

On the Create a new project page, select a **.NET MAUI App** from the list of templates (Figure 2). Be sure to choose the **C#** language. Click the **Next** button.

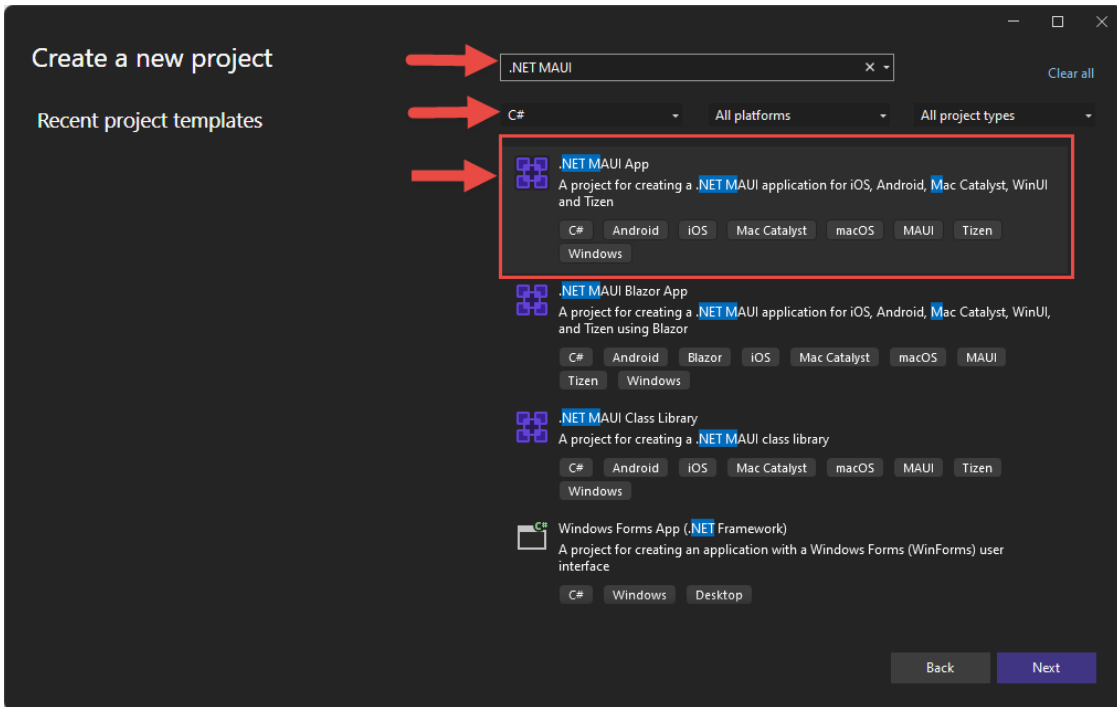


Figure 2: Select a .NET MAUI App from the list of templates.

On the Configure your new project screen, set the **Project name** of the new project to **XamlBasicsMAUI** (Figure 3). Place the project into a folder on your hard drive. Click the **Next** button.

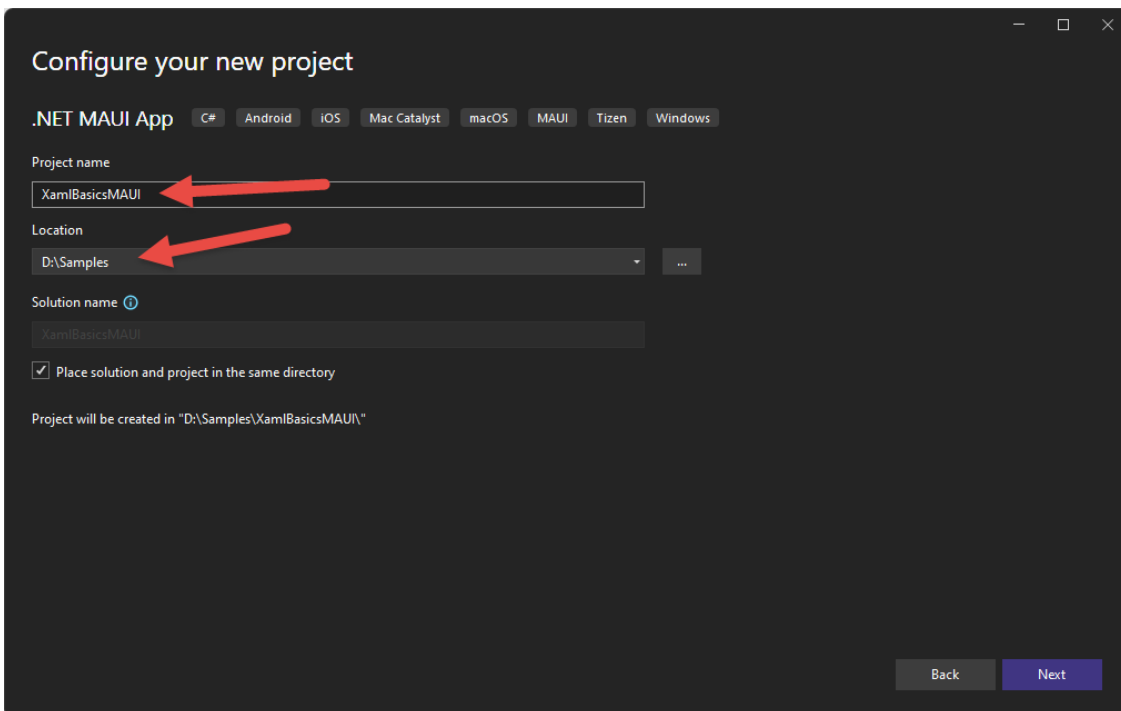


Figure 3: Set the project name and location.

On the Additional information screen, choose **.NET 7** or later, then click on the **Create** button.

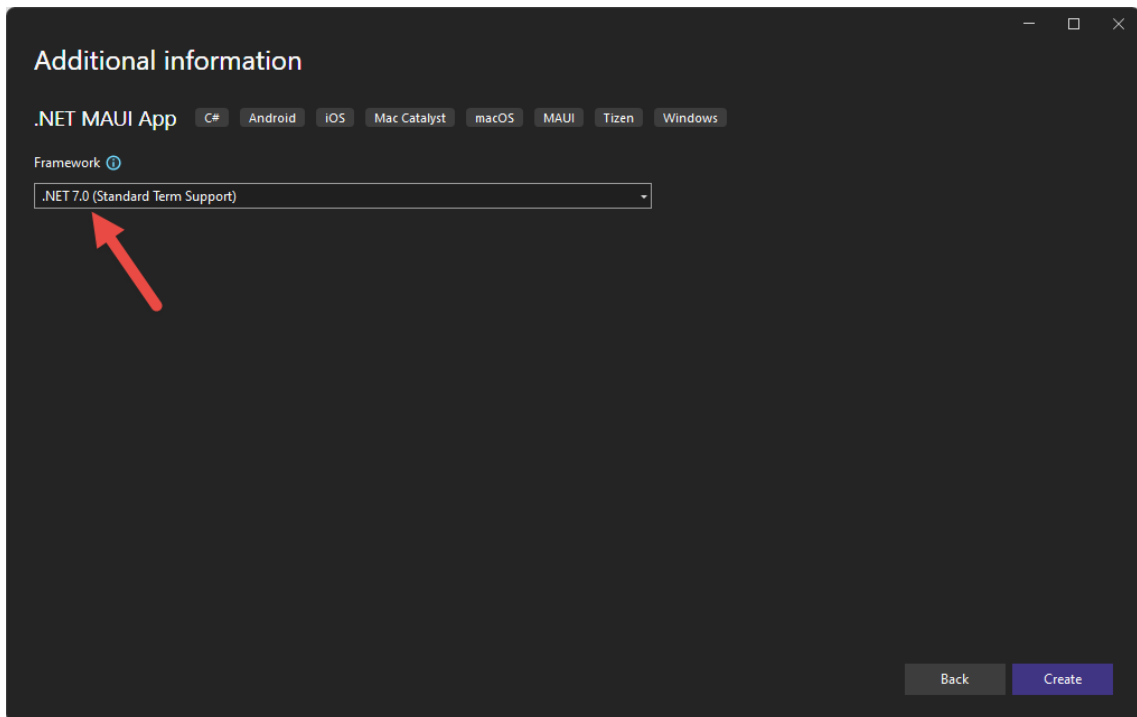


Figure 4: Choose .NET 7 or later when creating a .NET MAUI application.

Try It Out

Once the application has been created, click on the Windows Machine button (Figure 5) in the VS toolbar to run this application on Windows.

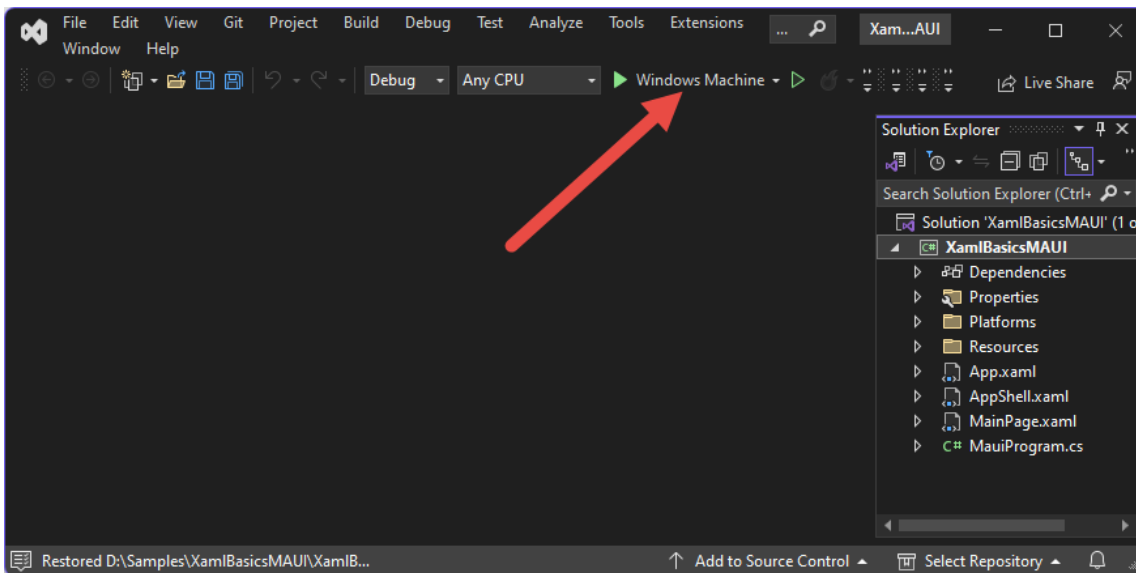


Figure 5: Run the .NET MAUI application on Windows.

The application should look like Figure 6.

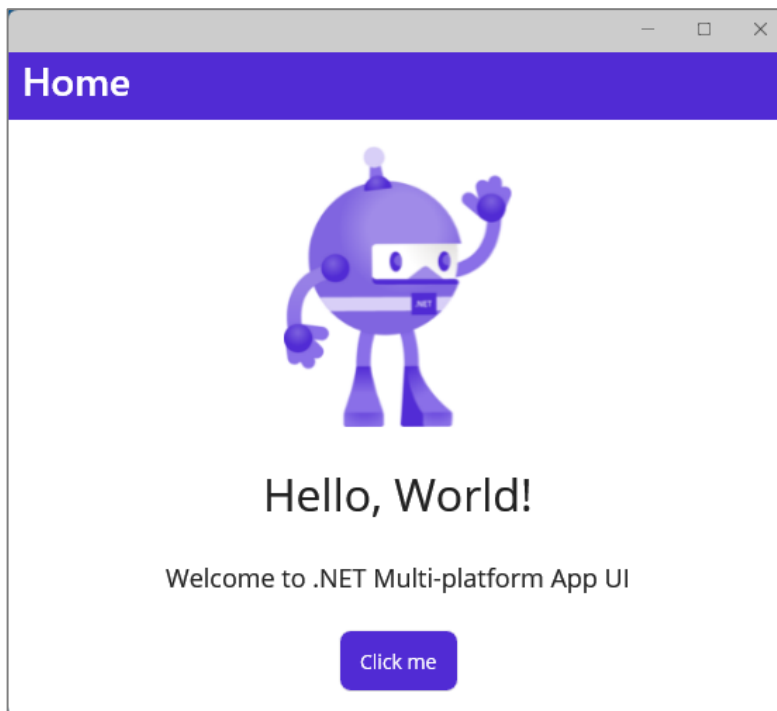


Figure 6: The .NET MAUI template application running on Windows.

Stop the application and now change to use the Android Emulators | Pixel 5 – API 33 as shown in Figure 7

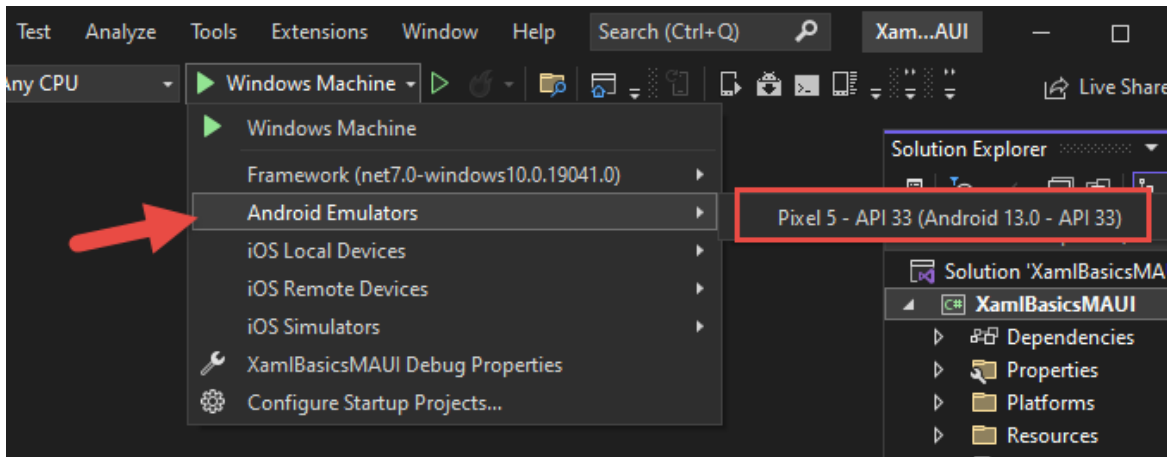


Figure 7: Choose the Android Emulators | Pixel 5 – API 33.

After selecting the Android emulator, it appears as the configuration item (Figure 8). Click on this button to run the Android emulator. It may take a few minutes to run this emulator, so be patient.

Even if the emulator comes up, it may still take a few minutes for VS to compile the application and deploy it to the emulator. Keep an eye on the status bar at the bottom of VS as it should show you the progress and provide you wish messages as to what is happening.

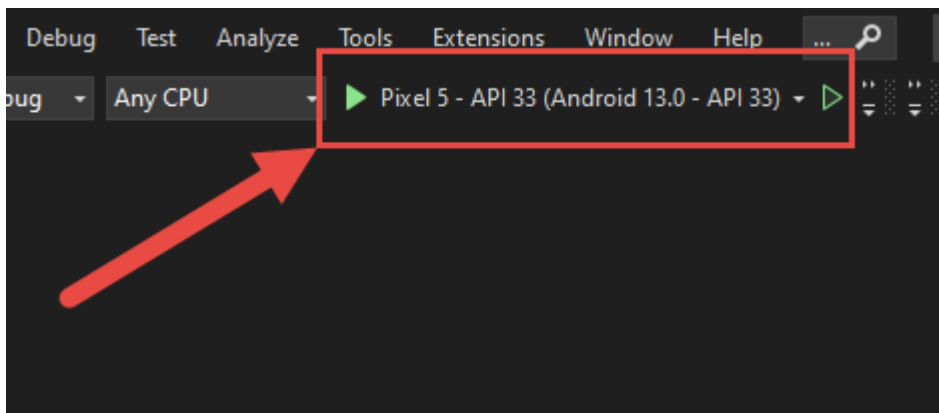


Figure 8: Run the application in the Android emulator.

Once it is completely deployed, you should see a screen that looks like Figure 9.

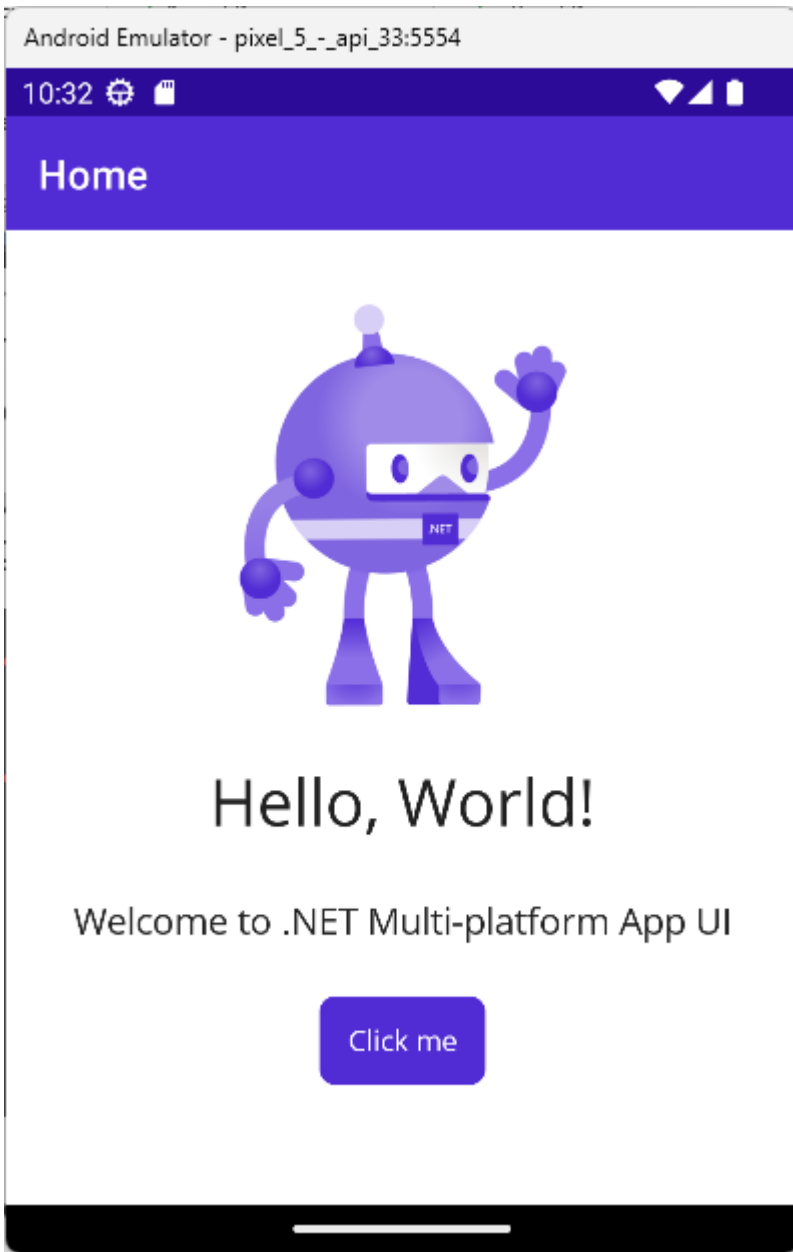
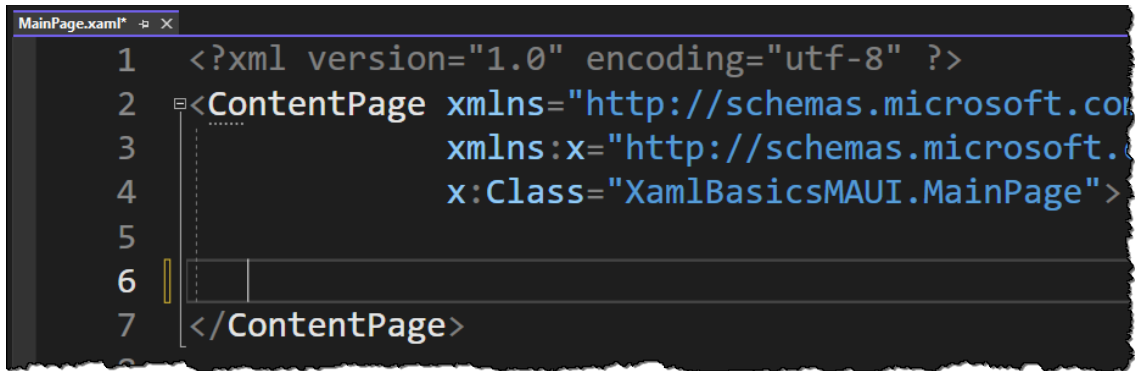


Figure 9: The .NET MAUI template application running on the Android emulator.

Demo 2: Content Page

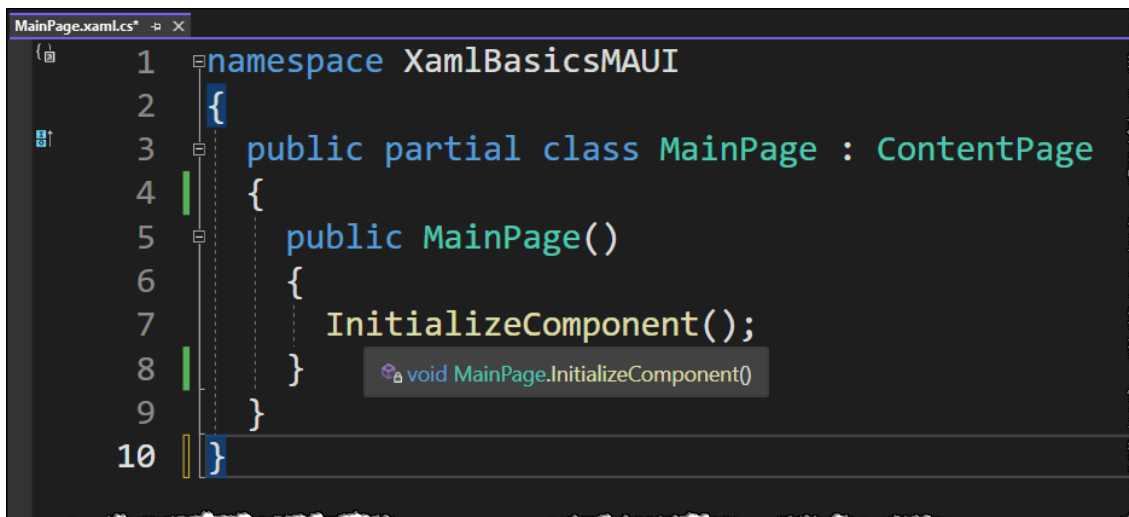
Open the **MainPage.xaml** file and remove the lines 6 through 39. This is the `<ScrollView>` element and everything down to (and including) the closing `</ScrollView>` element as shown in Figure 10.



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://schemas.microsoft.com
3             xmlns:x="http://schemas.microsoft.com
4             x:Class="XamlBasicsMAUI.MainPage">
5
6
7 </ContentPage>
```

Figure 10: Remove all the XAML from within the <ContentPage> element.

Open the **MainPage.xaml.cs** file and remove the code behind, except for the constructor as shown in Figure 11.



```
1 namespace XamlBasicsMAUI
2 {
3     public partial class MainPage : ContentPage
4     {
5         public MainPage()
6         {
7             InitializeComponent();
8         }
9     }
10 }
```

Figure 11: Remove everything but the constructor from the code-behind.

Set the Title Property

Open the **MainPage.xaml** file and click on line 2 so you are focused on the <ContentPage> element. Bring up the **Properties** window (Figure 12) by selecting **View | Properties Window** from the VS menu. The set of properties you can set for the <ContentPage> element are listed within this window.

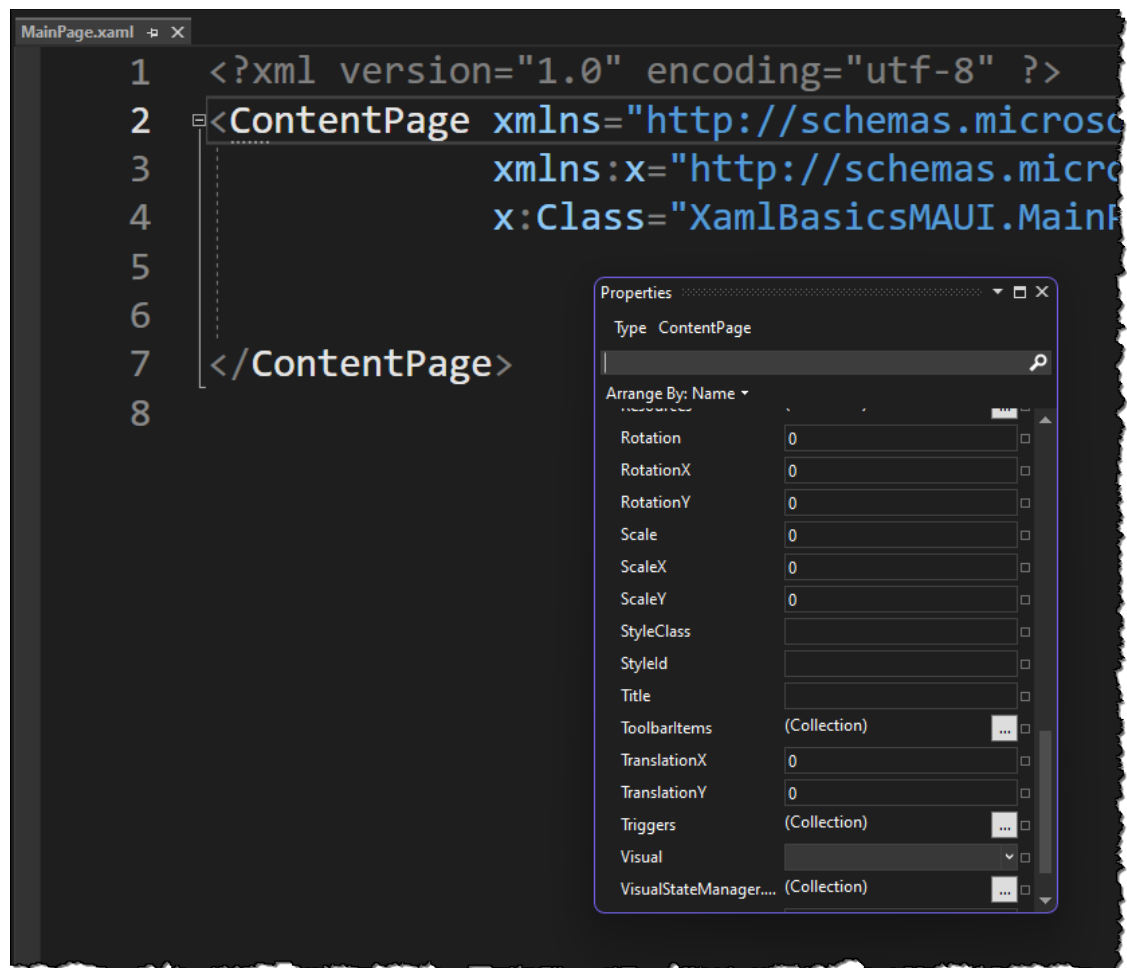


Figure 12: The Properties window allows you to set properties on whatever control has focus.

Set the **Title** property of the ContentPage to **XAML Basics** and press the **Enter** key. You should see the XAML has been written for you in the editor.

Close the **Properties** window and go to the editor and click just after the **Title="XAML Basics"** attribute and hit the spacebar. You should see an IntelliSense window open in your editor that has the same list of properties you saw in the Properties window.

Set the **Background** attribute to **Gray**.

Try It Out

Run the application either on Windows or the Android emulator to see the new **Title** attribute on the screen. Notice the Blue background on the title area and the Gray background (Figure 13).

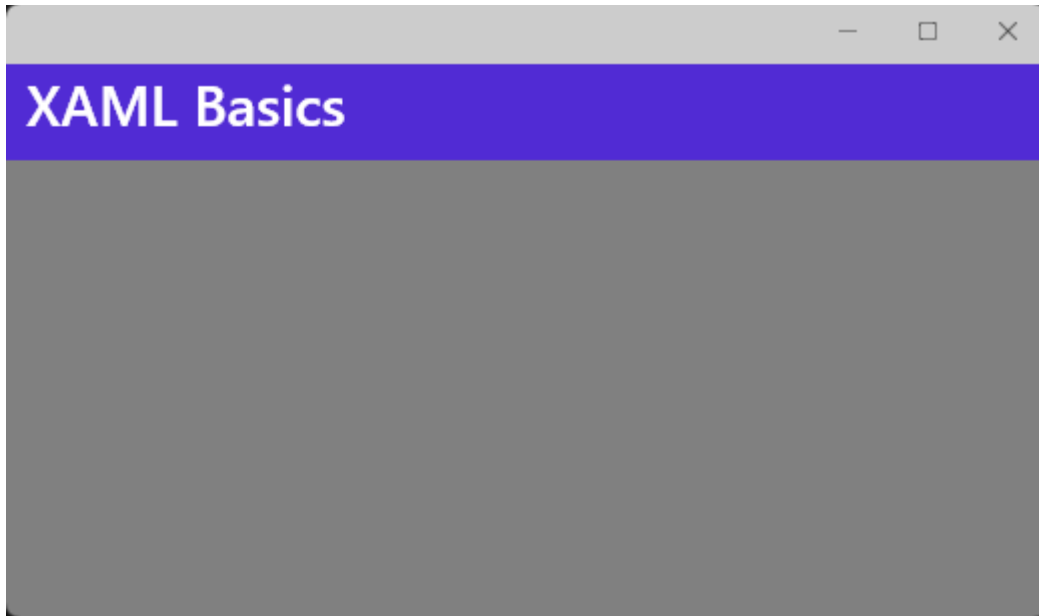


Figure 13: Setting properties on the <ContentPage> affects the appearance of the view.

Stop the application from running and completely remove the **Background** property.

Demo 3: Shell Properties

The blue background on the title area is controlled by the <Shell> class. Open the **AppShell.xaml** file and set the attributes on the <Shell> element shown in bold below.

```
<Shell x:Class="XamlBasicsMAUI.AppShell"
  xmlns="http://schemas.microsoft.com/XamlBasicsMAUI"
  xmlns:x="http://schemas.microsoft.com/XamlBasicsMAUI"
  xmlns:local="clr-namespace:XamlBasicsMAUI"
  Shell.TitleColor="Black"
  Shell.BackgroundColor="Gray"
  Title="My .NET MAUI Application"
  Shell.FlyoutBehavior="Disabled">
```

Try It Out

Run the application and view the background of the title area is gray and the title color is black (Figure 14). Also notice there is now a title on the Windows title bar

area. This only shows up on Windows and not on Android or iOS as they do not have a title bar area.

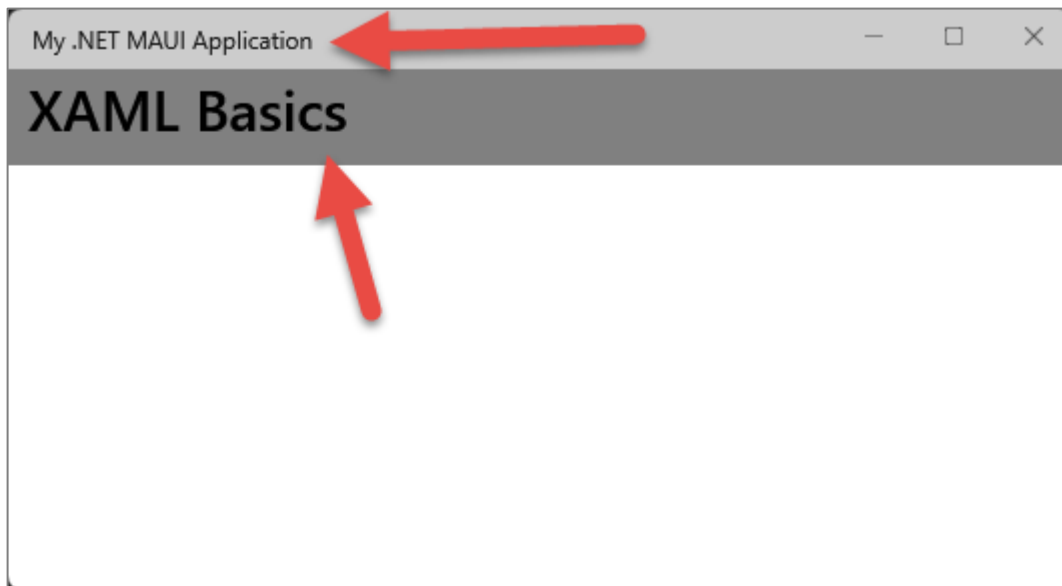


Figure 14: Changes to the application shell are controlled by the `<Shell>` element.

Remove the following two attributes from the `<Shell>`.

```
Shell.TitleColor="Black"  
Shell.BackgroundColor="Gray"
```

Lab 4: Create a Login Screen

Open the **MainPage.xaml** and create the screen shown in Figure 15 using just `<VerticalStackLayout>`, `<HorizontalStackLayout>`, `<Label>`, `<Entry>`, and `<Button>` elements.

Some things to keep in mind.

- Put some distance between these controls and the side of the `ContentPage`.
- Put some distance between each of the controls.
- Request a width of 200 on each entry control.
- Make sure the label text is aligned with the entry controls.
- Put the buttons under the entry controls.
- Make sure this screen looks good on both Windows and the Android Emulator.

Try It Out

Run the application on **Windows**.

Run the application on the **Android** Emulator.

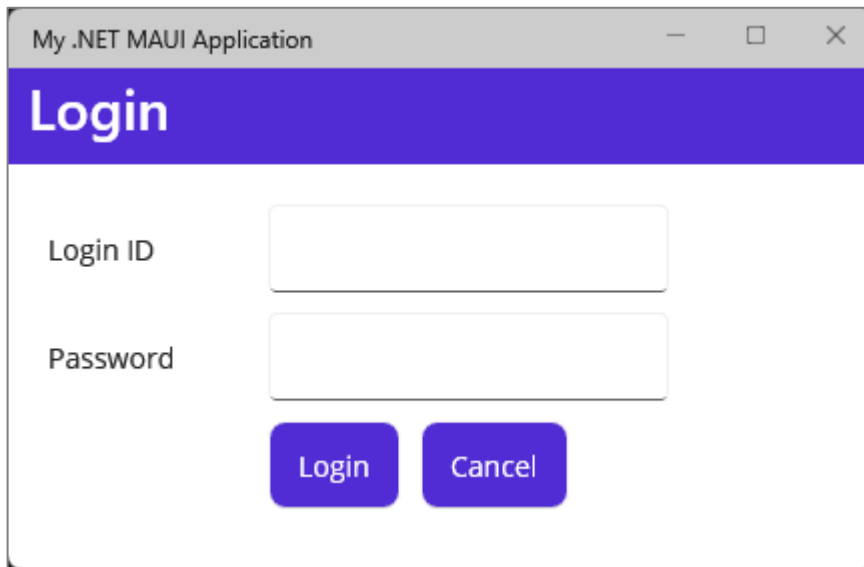


Figure 15: A Login screen