Xamarin

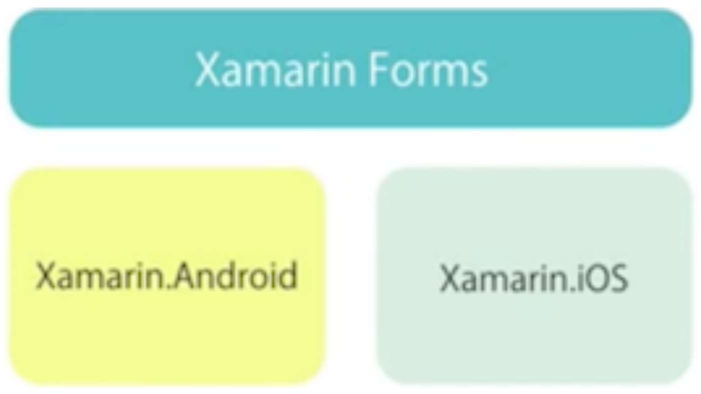
Similar product exist, like Xojo

Latest version

Xamarin Studio Community Version 6.2.1 (build 3)
Installation UUID: 0cff0983-4151-4e96-ba13-bde6ac1dfc41
Apple Developer Tools: Xcode 8.3 (12169) Build 8E162
Supported Android versions: 6.0

Introduction

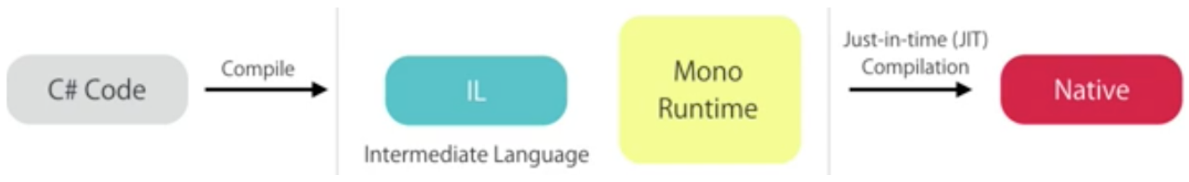
Xamarin uses C# language.
Xamarin products rely upon .NET Base Class Library, and also on the native APIs of iOS or Android.
The two libraries are called Xamarin.Android and Xamarin.iOS.
There is a third library which relies on top of the aforementioned two, Xamarin.Forms.



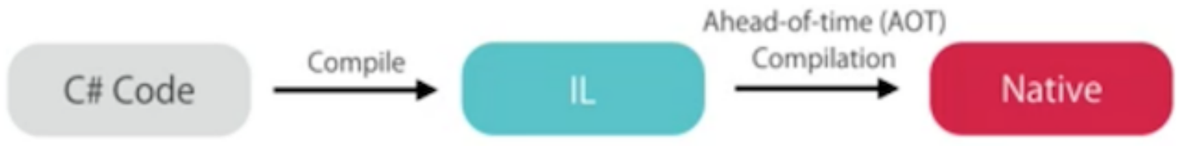
You need to know about Xamarin.Android or Xamarin.iOS only if you want to build platform specific user-interface elements.
Xamarin.iOS/.Android/.Forms are based on Mono, the open source version of .NET which runs on Unix, linux and iOS.
Mono for Android and MonoTouch are the two mono releases renamed Xamarin.Android and Xamarin.iOS

CLR (Common Language Runtime) is the .NET component which generates JIT (Just-In-Time) the native machine language from the CIL (Common Intermediate Language) obtained by the C# compiler.
In Mono, CLR is called Mono Runtime, and CIL is called IL.

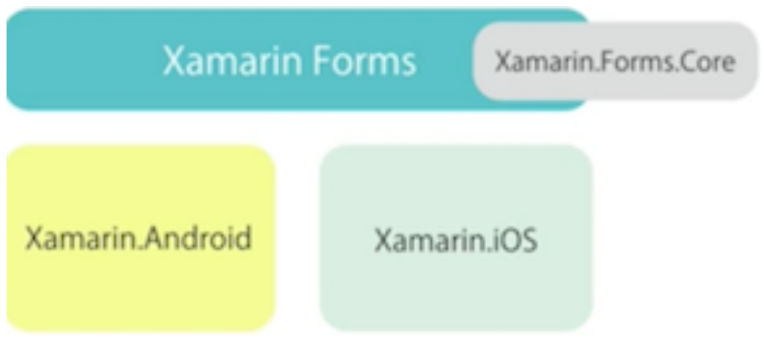
When you build a Xamarin.Android app, you get (and ship) a IL code with Mono Runtime



When you build a Xamarin.iOS app, you get (and ship) directly a native code that can run on iOS, because Apple doesn't allow JIT compilation



The last compiler requires a Mac.
In Xamarine.Forms there is a specific assembly called Xamarin.Forms.Core



which maps into platform specific assemblies like Xamarin.Forms.Platform.Android and Xamarin.Forms.Platform.iOS

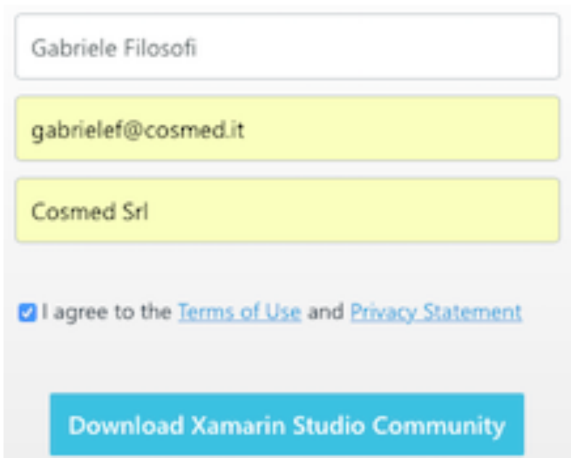
Two IDE exist, Visual Studio (running on a Windows PC) and Xamarin Studio (running on Mac)

- To build iOS apps you need a Mac
- To build Windows apps you need Visual Studio on a Windows PC (either real or VM)
- If you want to create iOS and Android apps, you can use Xamarin Studio on Mac

Installation of Xamarin Studio

go to https://www.xamarin.com/download
Two Xamarin Studio products exist:
- Community Edition: it is free
- Enterprise Edition: requires a licence

Set your account data and download Xamarin Studio Community

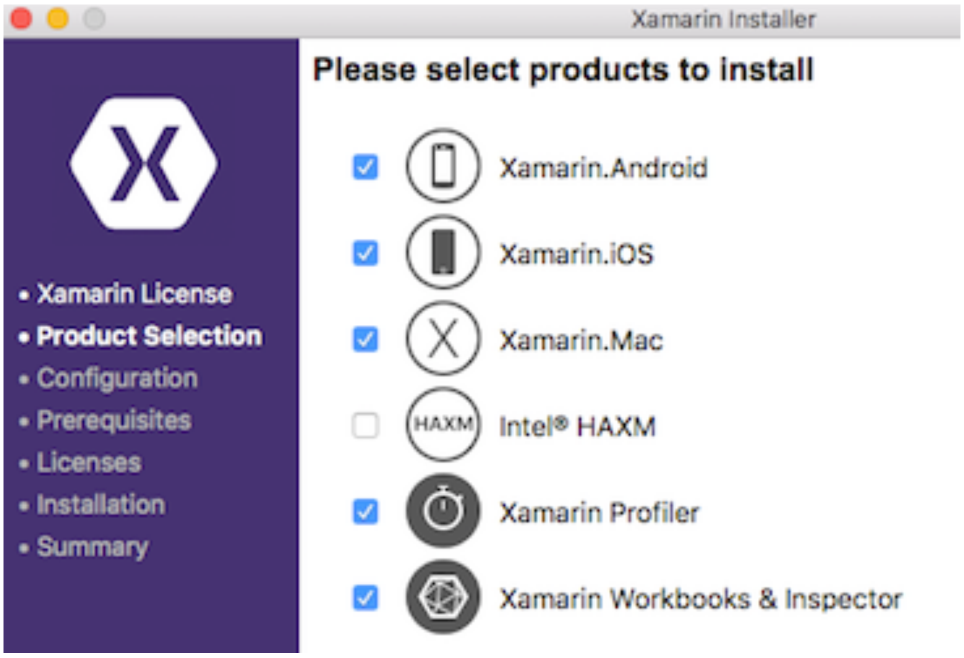

Gabriele Filosofi

gabrielef@cosmed.it

Cosmed Srl

☑ I agree to the Terms of Use and Privacy Statement

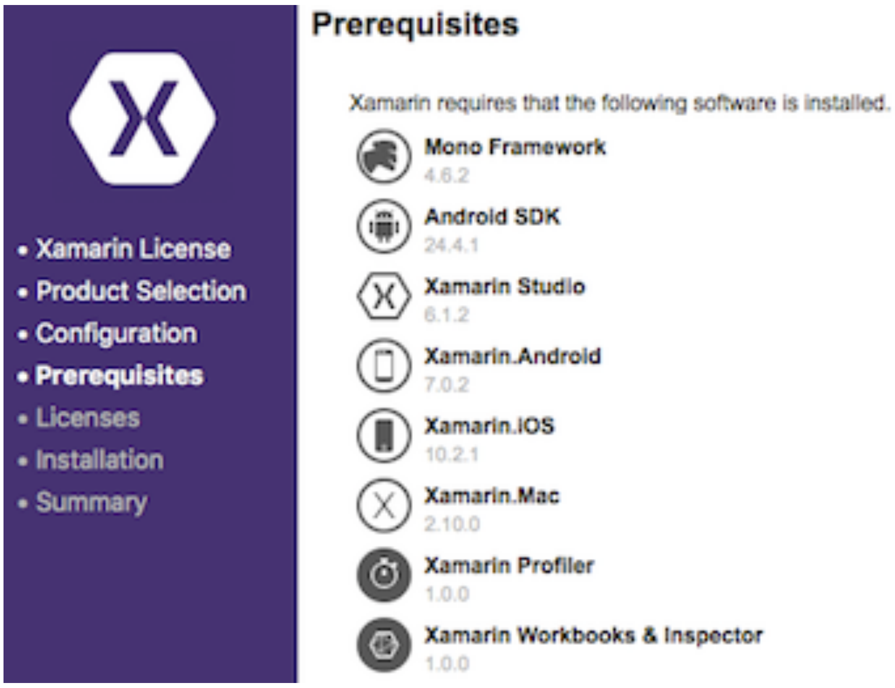
Select components to install



Please select products to install

- ☑ Xamarin.Android
- ☑ Xamarin.iOS
- ☑ Xamarin.Mac
- ☐ Intel® HAXM
- ☑ Xamarin Profiler
- ☑ Xamarin Workbooks & Inspector

Check the prerequisites

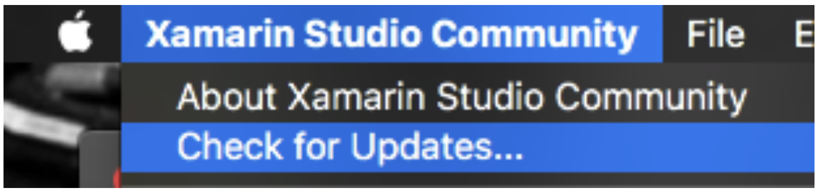

Prerequisites

Xamarin requires that the following software is installed.

Mono Framework	4.6.2
Android SDK	24.4.1
Xamarin Studio	6.1.2
Xamarin.Android	7.0.2
Xamarin.iOS	10.2.1
Xamarin.Mac	2.10.0
Xamarin Profiler	1.0.0
Xamarin Workbooks & Inspector	1.0.0

Create an account on Xamarine.com in order to start a Xamarin University Trial (30 days).
Then you have access to XAM101, XAM120, Lightning Lectures,
and Guest Lectures. Even after your 30-day trial has ended, you'll continue to have access to our Self-Guided Learning.

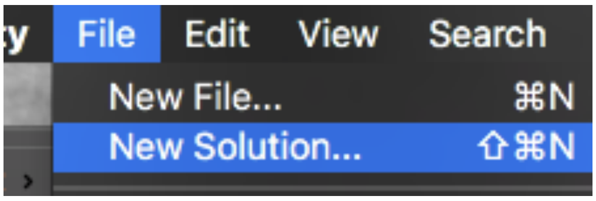
From time to time, check for package updates. They are frequently released.



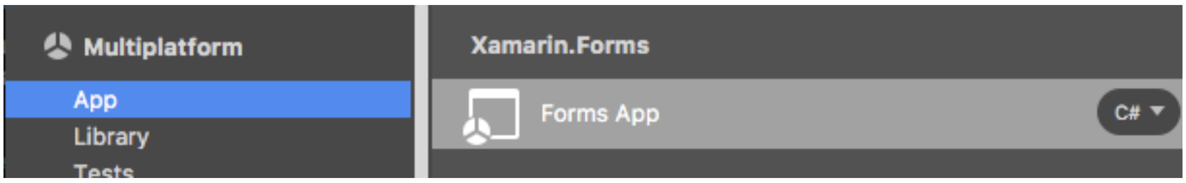
Xamarin Forms: Build Native Cross-platform Apps with C#

By Mosh Hamedani
https://www.udemy.com/cart/success/76091746/

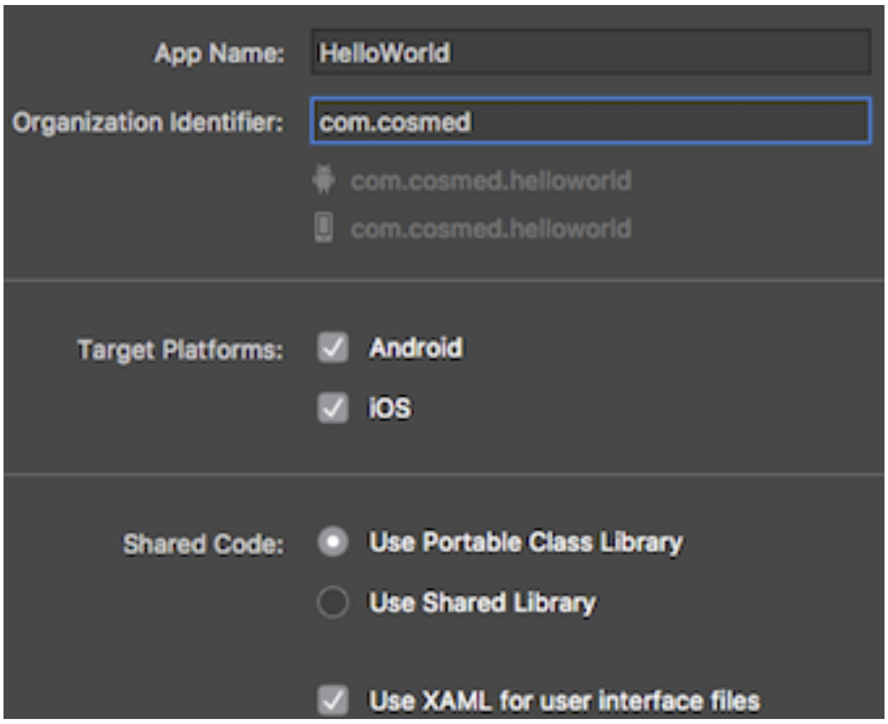
Let's create a new project



of type Xamarine.Form PCL (Portable Class Library) in C#



call it HelloWorld, then click Next



App Name: HelloWorld

Organization Identifier: com.cosmed

com.cosmed.helloworld

com.cosmed.helloworld

Target Platforms: [✓ Android] [✓ iOS]

Shared Code: [✓ Use Portable Class Library] Use Shared Library

[✓ Use XAML for user interface files]

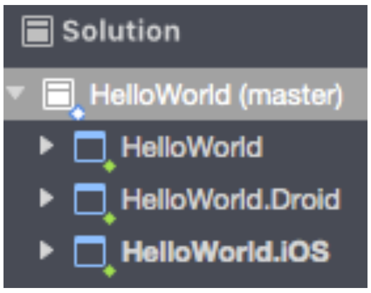
- The organisation ID is used to generate the app identifier com.companyname.productname (known as Bundle ID in iOS)
- A Portable Class Libraries (PCL) have access to a subset of .NET framework.
- Select Portable Class Library (PCL) if you plan to distribute the resulting assembly to other developers
- There are times that we need to write platform-specific code in the PCL. For example, on iOS, we often set 20 units padding on top of the page. We want the code for setting the padding to run only on an iOS device. To achieve this, we use the Device class in Xamarin Forms:

```
if (Device.OS == TargetPlatform.iOS) { ... }
else if (Device.OS == TargetPlatform.Android) { ... }
```

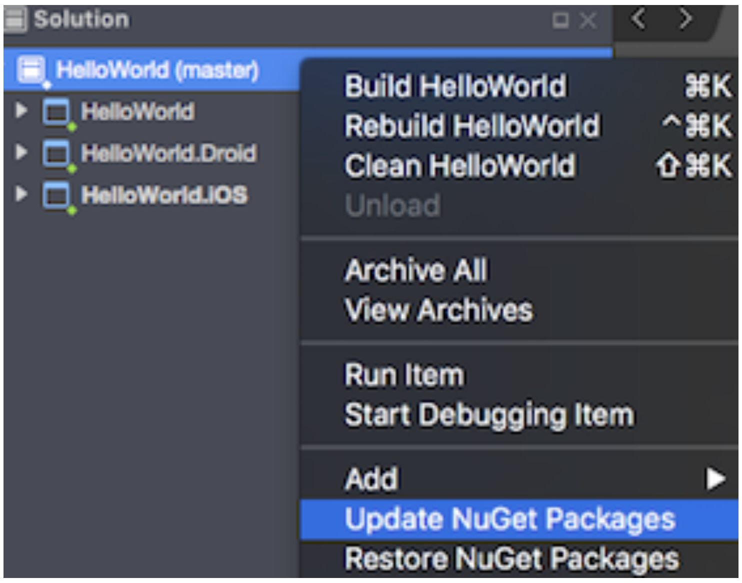
- Select Shared Library if it has to be shared only within your solution. It typically employs conditional compiler directive to specify code paths for different platforms:

```
#if __IOS__   ...
#elif __ANDROID__  ...
#endif
```

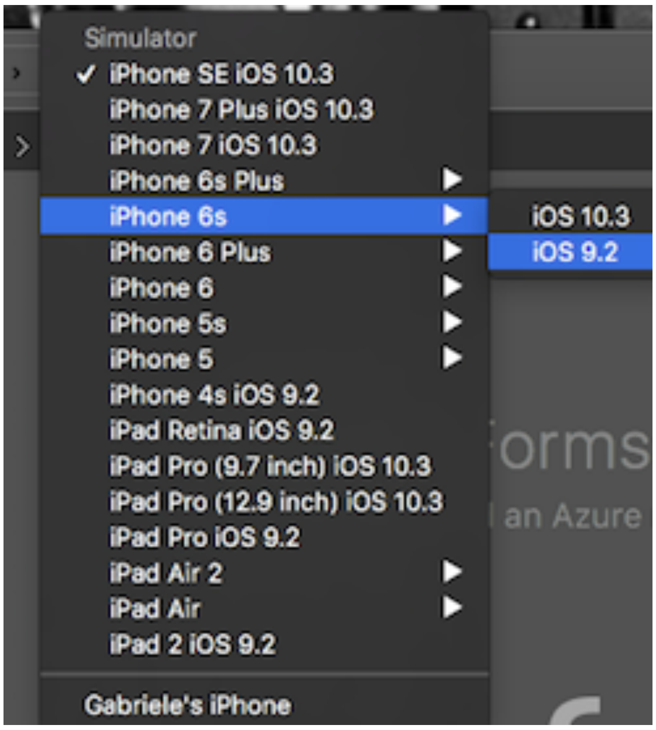
We get this perspective



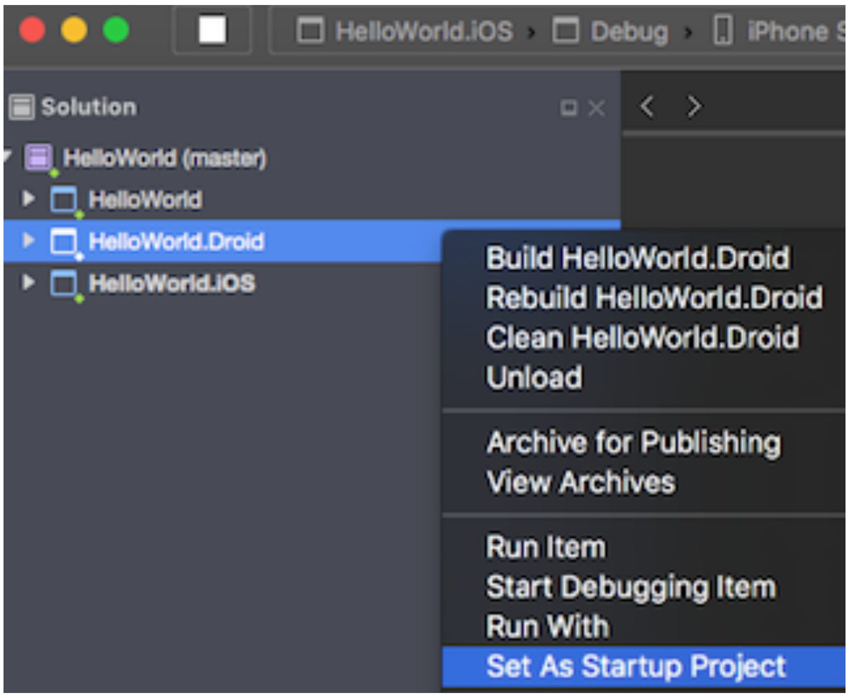
There is a Portable project and one project for each different platform you want to support.
The first thing to do is to update the packages (there are new versions time by time). Right click on the solution and click Update NuGet Packages (Xamarin.Forms is packaged as a NuGet)



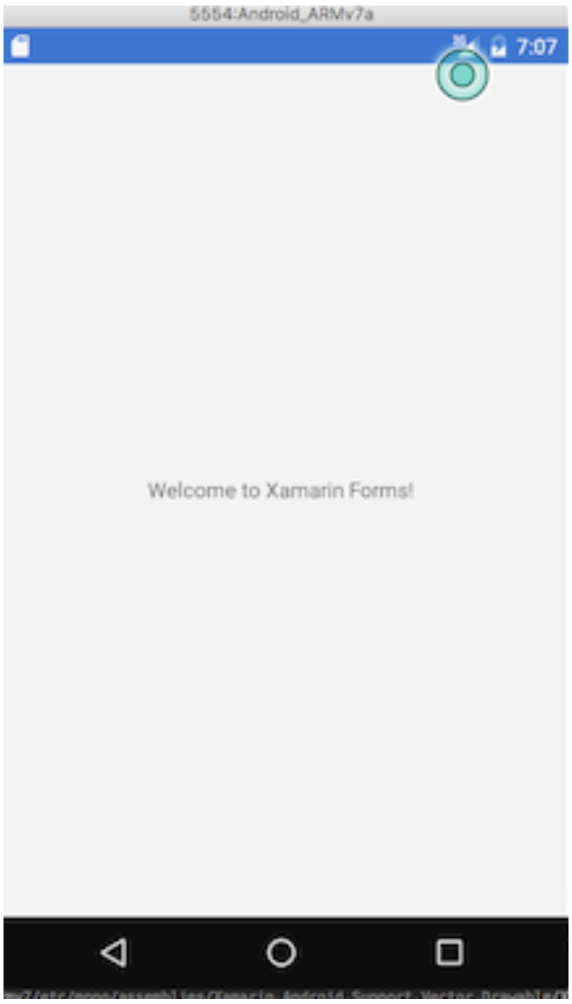
You can select the iOS device (Simulator or real device)



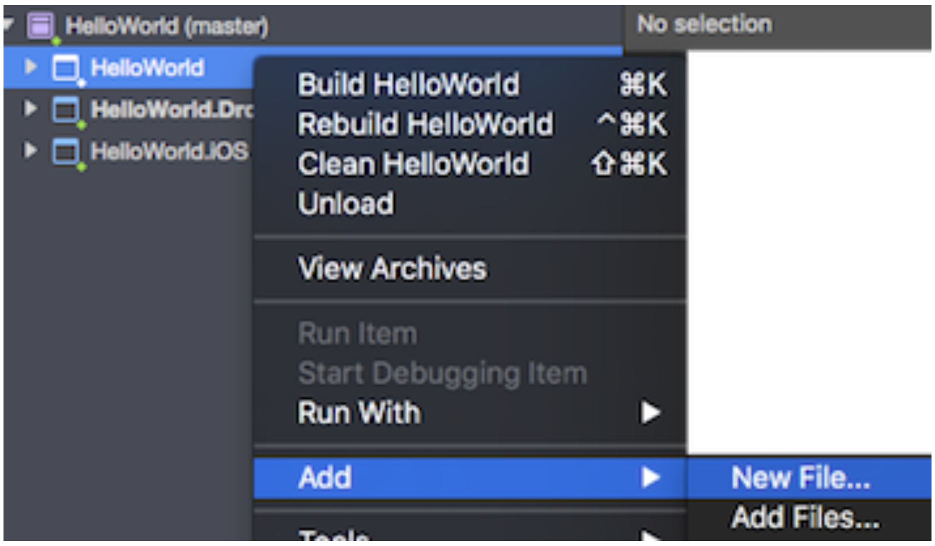
To run the app in iOS Simulator : Cmd + Option + Enter
Enable the Android app



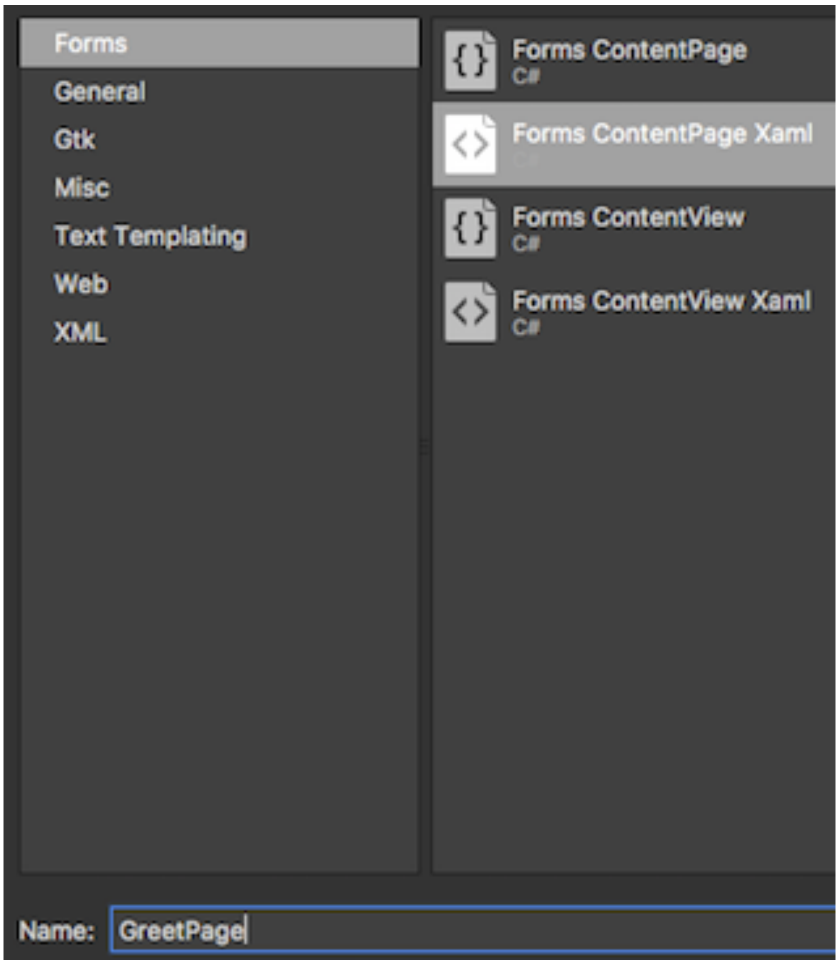
Build and then run the app in Android Emulator



Now we want add a button in the screen. Pressing the button we get an alert "Hello World".
Let's add a Content Page



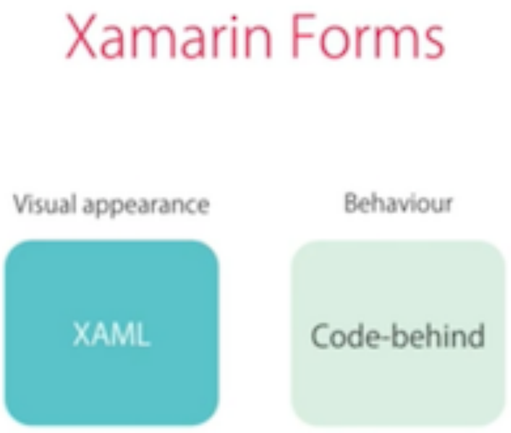
and rename it GreetPage



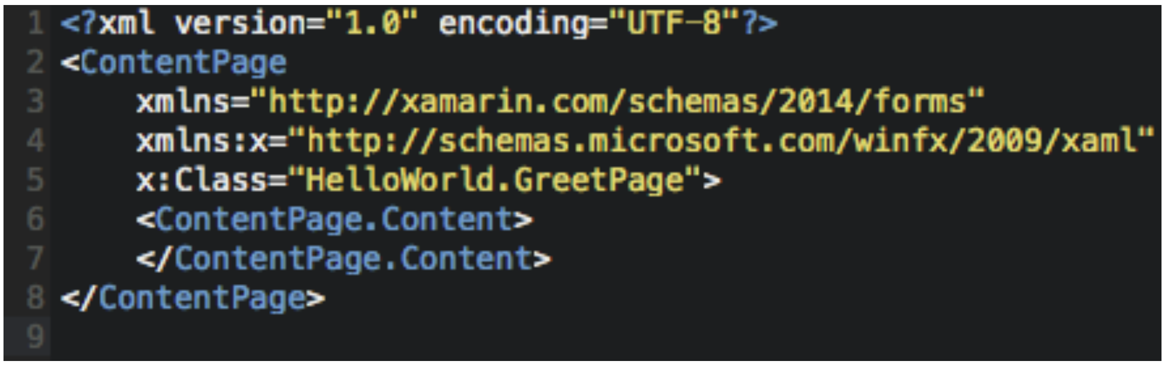
Name: GreetPage

Xamarin.Forms uses XAML (eXtensible Application Markup Language) to create the visual appearance of the UI. The Code-behind is the class which implements the behaviour. This architecture is similar to the View and

ViewController pair in iOS.



This is how the GreetPage.xaml looks like

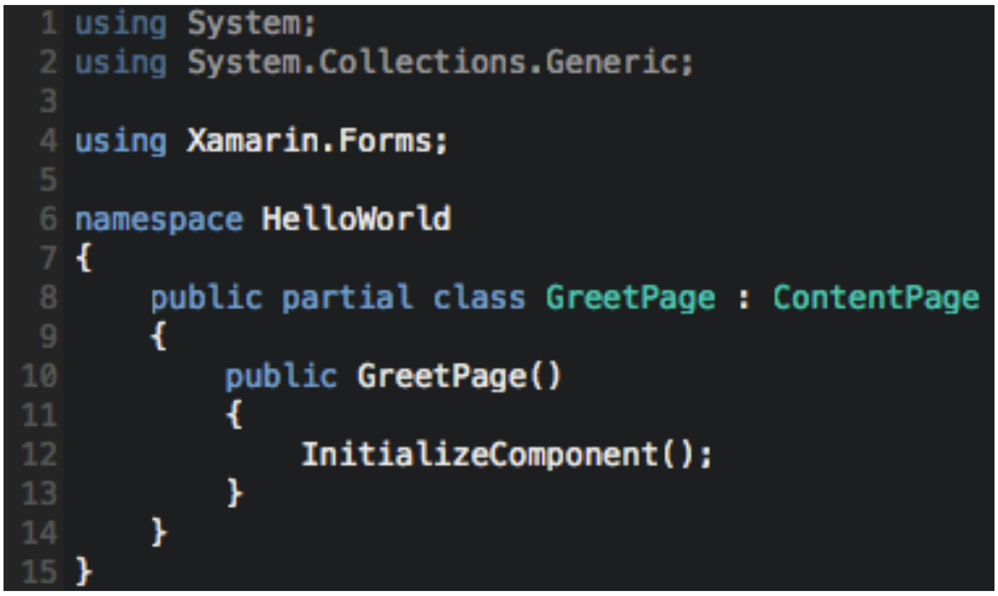


```xml
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="HelloWorld.GreetPage">
    <ContentPage.Content>
    </ContentPage.Content>
</ContentPage>
```

ContentPage is the type of page. It is used to present text and/or images contents to the user.
We have four different pages in Xamarin.Forms

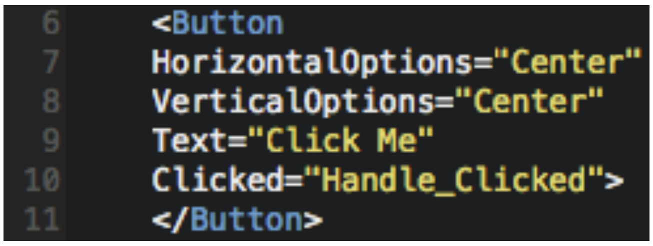
- Content
- MasterDetail
- Navigation
- Carousel

xmlns and xmlns:x are namespaces. x:Class is the Class attribute, which belongs to the xmlns:x namespace because has prefix x.
The value of the Class attribute is the link between the xcml file and the code-behind file.
In our case the code-behind file is the GreetPage.xaml.cs



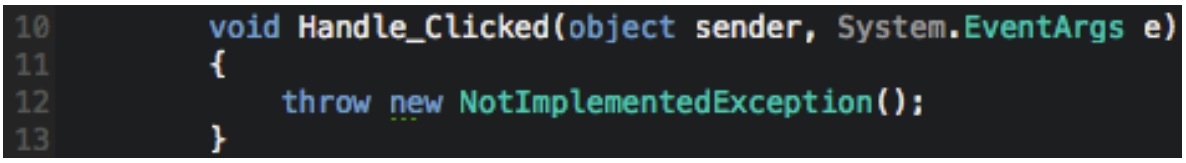
```
using System;
using System.Collections.Generic;

using Xamarin.Forms;

namespace HelloWorld
{
    public partial class GreetPage : ContentPage
    {
        public GreetPage()
        {
            InitializeComponent();
        }
    }
}
```

Now, replace line 6-7 of xaml source with a button



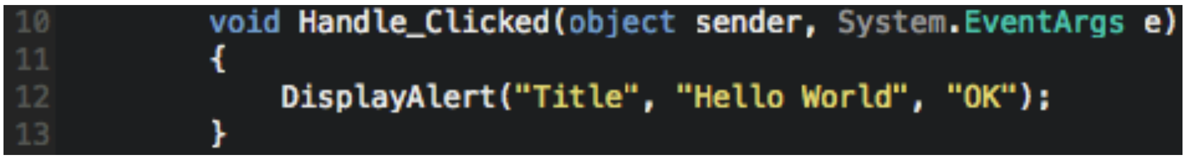
```
    <Button
    HorizontalOptions="Center"
    VerticalOptions="Center"
    Text="Click Me"
    Clicked="Handle_Clicked">
    </Button>
```

The autocompletion of the Clicked event attribute expands automatically into a method in the behind-code class



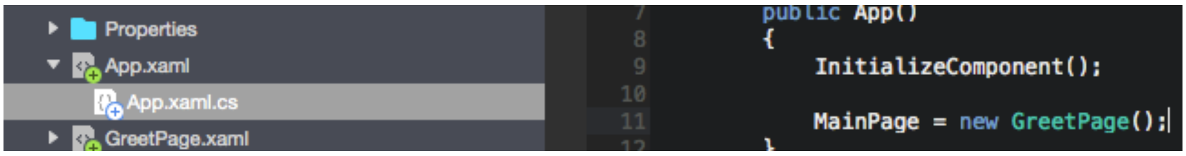
```
        void Handle_Clicked(object sender, System.EventArgs e)
        {
            throw new NotImplementedException();
        }
```

Let's replace the default behaviour with an alert



```
        void Handle_Clicked(object sender, System.EventArgs e)
        {
            DisplayAlert("Title", "Hello World", "OK");
        }
```

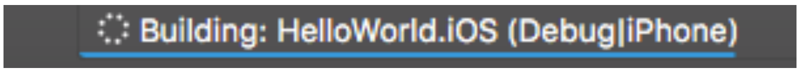
Now, set the MainPage property of the App.xaml.cs to GreedPage()



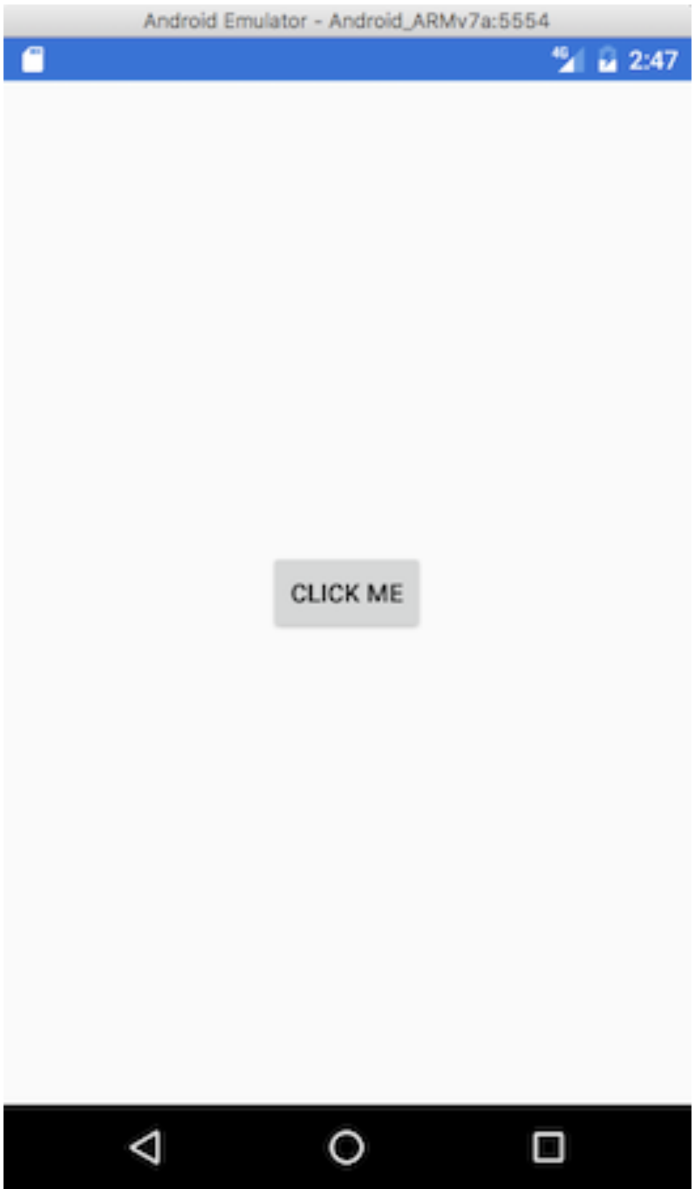
```
Properties
App.xaml
    App.xaml.cs
GreetPage.xaml
```

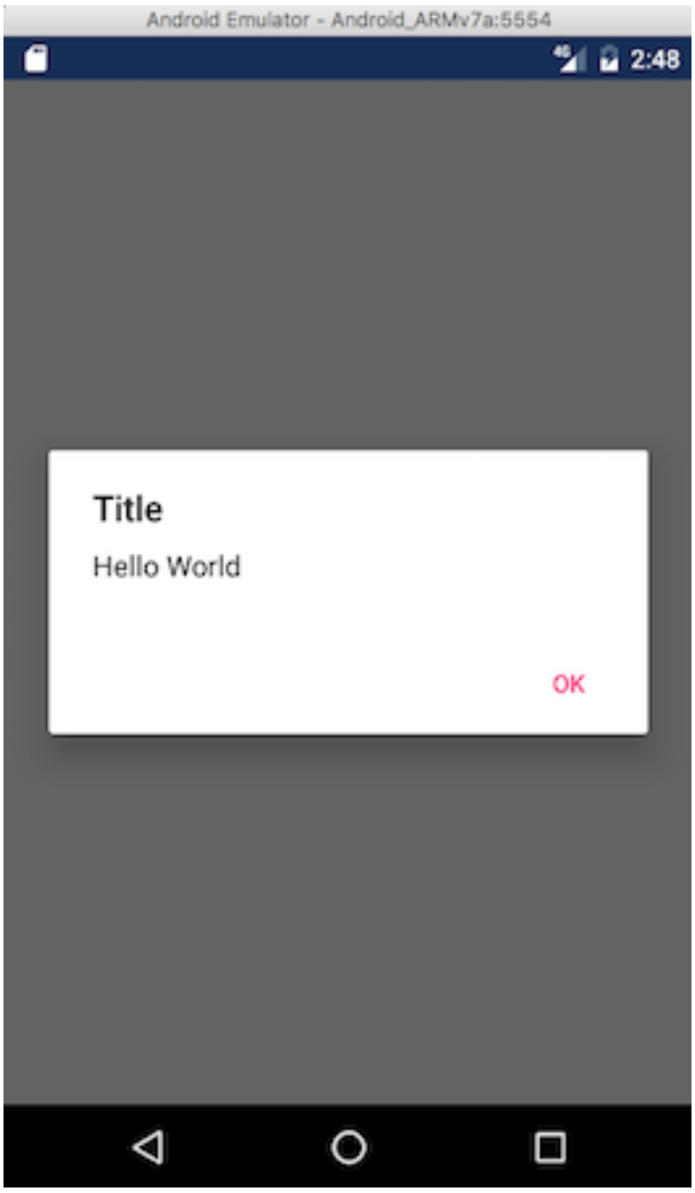
```
        public App()
        {
            InitializeComponent();

            MainPage = new GreetPage();
        }
```

It sets the start page of the application.
Now, type Cmd+Alt+Enter to run the app..



Building: HelloWorld.iOS (Debug|iPhone)


Android Emulator - Android_ARMv7a:5554
2:47
CLICK ME



Title

Hello World