Xamarin.Forms (v2.0.x)

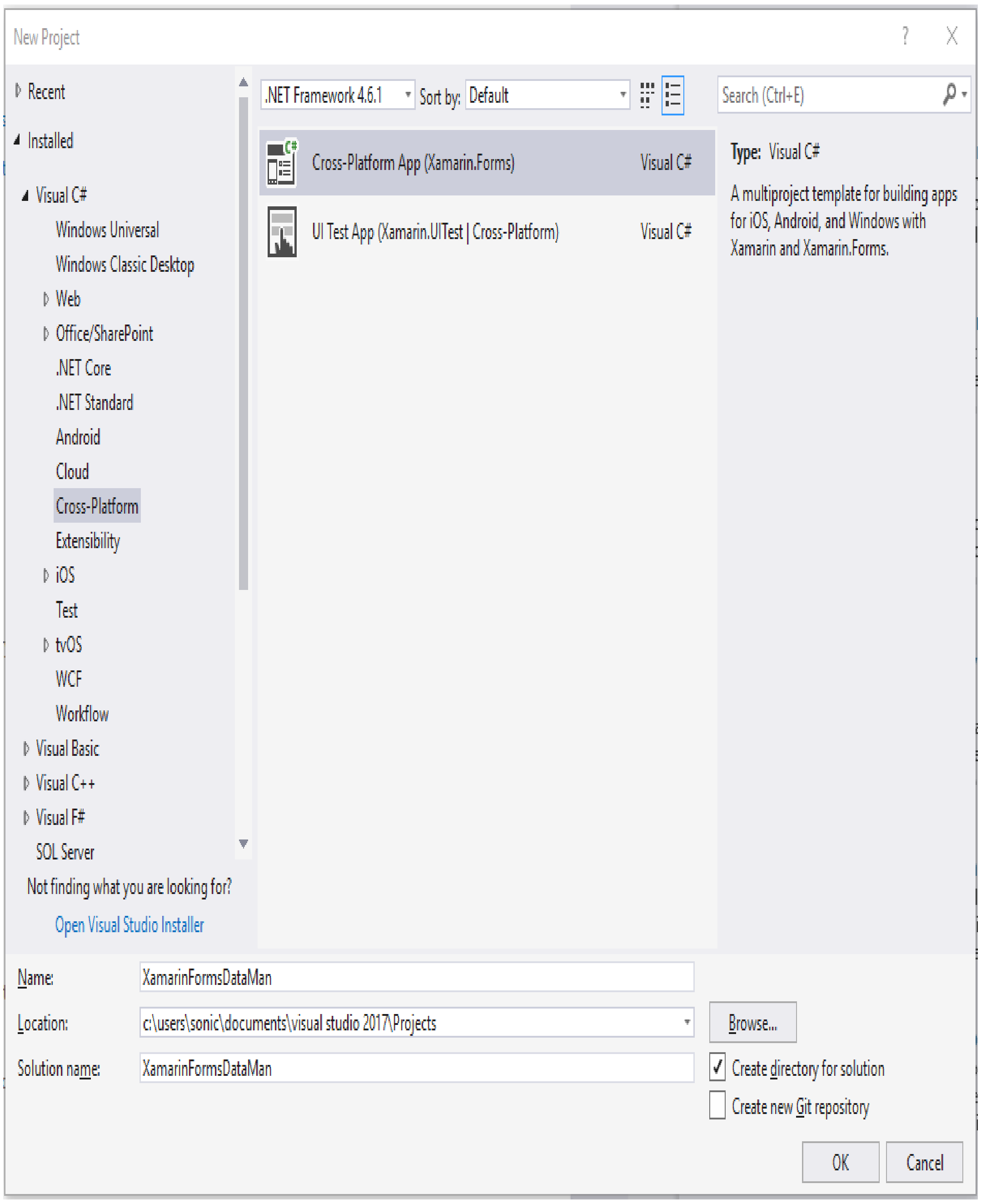
Getting Started

In the following sections we will explain how our sample app is developed step by step.

Open Visual Studio and follow these steps:

1. Go to **File -> New -> Project.**

2. Create **Cross-Platform App(Xamarin.Forms).**



New Project

Installed › Visual C# › Cross-Platform

.NET Framework 4.6.1 | Sort by: Default

Cross-Platform App (Xamarin.Forms)	Visual C#
UI Test App (Xamarin.UITest | Cross-Platform)	Visual C#

Type: Visual C#

A multiproject template for building apps for iOS, Android, and Windows with Xamarin and Xamarin.Forms.

Name: XamarinFormsDataMan

Location: c:\users\sonic\documents\visual studio 2017\Projects

Solution name: XamarinFormsDataMan

[✓] Create directory for solution

[ ] Create new Git repository

After loading, see Android Getting Started section to see how to setup references, and the **manifest.xml** for the **Android** platform, or iOS Getting Started and **Info.plist** file for the **iOS** platform.

Portable Project

In the portable project we create one page (**MainPage**) where we create a layout for a scanning page and one class(CameraPreview.cs) that will inherit from View control and have some properties and events.

CameraPreview control in **MainPage** is added from code behind on **OnAppearing** method so we can set some public properties and set **ResultReceived** and **ConnectionStateChanged** events.

```
protected override void OnAppearing()
        {
            base.OnAppearing();

           if (cameraPreview == null)
            {
                cameraPreview = new CameraPreview();

                //You can choose device from here, or you can select on every appearing on this page. Default is SelectFrom
                cameraPreview.SelectedDevice = ScanningDevice.MobileCamera;

                //Preview enable. Default is true
                //cameraPreview.ScanningPreviewEnable = false;

                //Event that will be triggered when result is received
                cameraPreview.ResultReceived += CameraPreview_ResultReceived;

                //Event that will be triggered when connection state will be changed
                cameraPreview.ConnectionStateChanged += CameraPreview_ConnectionStateChanged;

                //Add this control in this content
                gridCamera.Children.Insert(0, cameraPreview);

                /*If you use scanner in navigation page and you add another page in navigation stack(where you will use sca
                 * from this page like Navigation.PushAsync(new AnotherPage()) please use this code before you navigate */
                //if (cameraPreview != null)
                //{
                //    cameraPreview.ResultReceived -= cameraPreview_ResultReceived;
                //    cameraPreview.ConnectionStateChanged -= cameraPreview_ConnectionStateChanged;
                //    gridCamera.Children.Remove(cameraPreview);
                //    cameraPreview = null;
                //}
                //Navigation.PushAsync(new AnotherPage());
            }
        }
```

In **Android** and **iOS** platform specific projects we have custom renderers (ScannerView.cs) for this class, and with that we can use native elements in portable project.

Custom Renderer

ViewRenderer in Android platform specific project:

```
protected override void OnElementChanged(ElementChangedEventArgs<XamarinFormsDataMan.CameraPreview> e)
        {
            base.OnElementChanged(e);

            if (e.OldElement != null || Element == null)
            {
                return;
            }

            rlMainContainer = new RelativeLayout(Context);

            ...
            MainActivity.selfActivity.setActiveReader(Control, Element);
         }
```

In **MainActivity** in Android platform specific project we are handling with reader device object (almost the same like ScannerActivity for **Xamarin.Android**) and from custom renderer we are just call setActiveReader method and pass Control(native RelativeLayout element) and Element(CameraPreview object which is initialized from MainPage).

ViewRenderer in iOS platform specific project:

```
protected override void OnElementChanged(ElementChangedEventArgs<XamarinFormsDataMan.CameraPreview> e)
        {
            base.OnElementChanged(e);

            if (e.OldElement != null || Element == null)
            {
                return;
            }

            container = new UIView();
            ivPreview = new UIImageView();
            ivPreview.ContentMode = UIViewContentMode.ScaleToFill;
            ......

            AppDelegate.selfDelegate.setActiveReader(Control, Element);
        }
```

In **AppDelegate** in iOS platform specific project we are handling with reader device object (almost the same like View Controller for **Xamarin.iOS**) and from custom renderer we are just call setActiveReader method and pass Control(native UIImageView element) and Element(CameraPreview object which is initialized from MainPage).

Configuring ReaderDevice, Connecting to Device, Scanning Barcodes and **Disconnecting from Device** are the same and you can read about them in Xamarin.Android and Xamarin.iOS sections.

Licensing the SDK

Licensing the SDK must also be implemented separately in Android and in iOS projects.

For the Android oriented solution please check this link, and for the iOS solution you can refer to this resource.