Xamarin.Forms (v2.1.x)

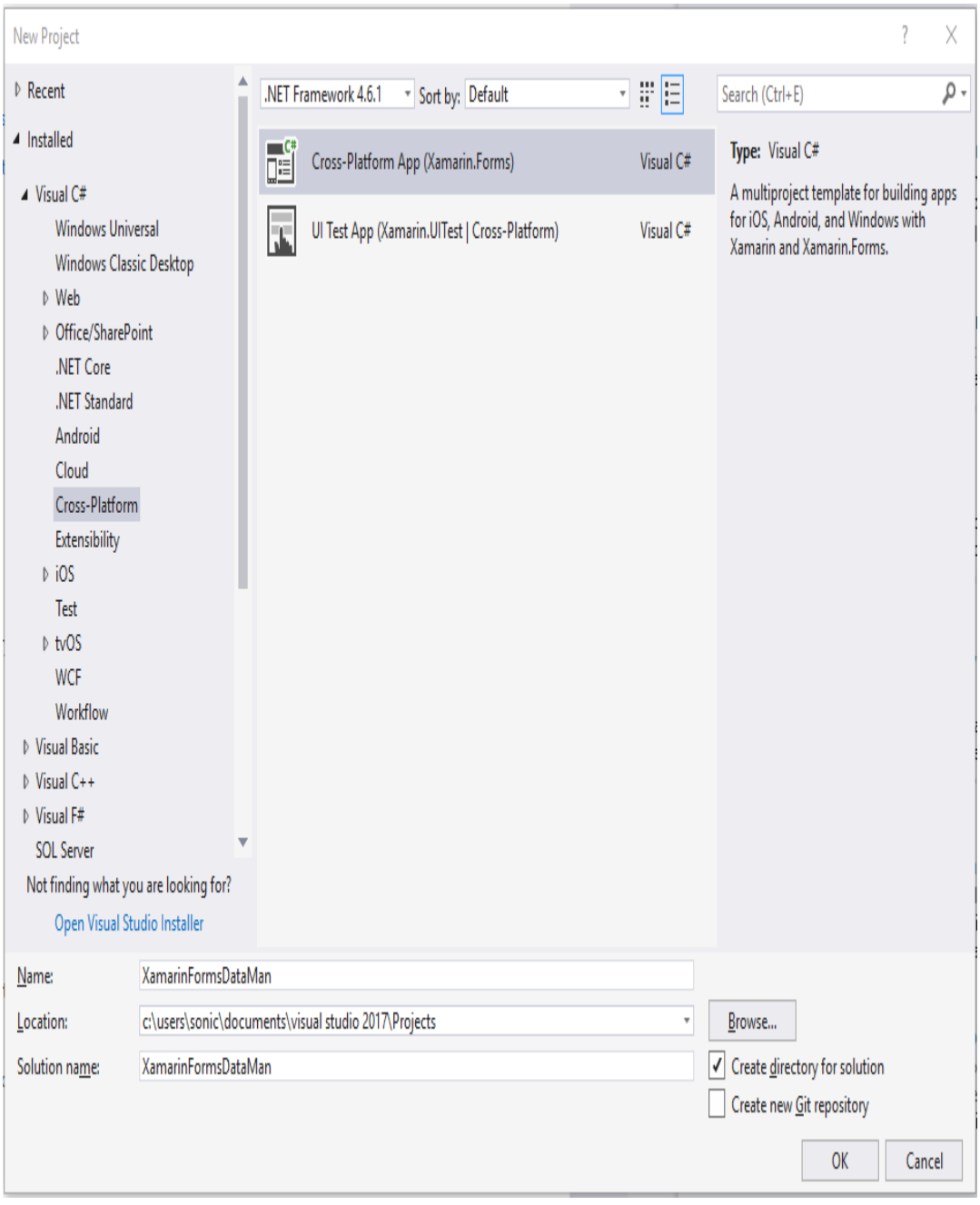
Getting Started

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).**



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 is added in **MainPage**:

```
...

<Grid x:Name="gridCamera">
         <local:CameraPreview x:Name="cameraPreview" SelectedDevice="MobileCamera" ResultReceived="CameraPreview_ResultRece
         <Label x:Name="lblStatus" Text=" Disconnected " TextColor="White" FontSize="11" VerticalOptions="Start" Horizontal
</Grid>

...
```

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<CameraPreview> e)
        {
            base.OnElementChanged(e);

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

            rlMainContainer = new RelativeLayout(Context);
            rlMainContainer.SetMinimumHeight(50);
            rlMainContainer.SetMinimumWidth(100);
            rlMainContainer.LayoutParameters = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MatchParent, Rel

            ivPreview = new ImageView(Context);
            ivPreview.SetMinimumHeight(50);
            ivPreview.SetMinimumWidth(100);
            ivPreview.LayoutParameters = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MatchParent, RelativeL
            ivPreview.SetScaleType(ImageView.ScaleType.FitCenter);

            rlMainContainer.AddView(ivPreview);

            if (Control == null)
                SetNativeControl(rlMainContainer);

            MainActivity.instance.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;
            ivSVG = new UIImageView();
            ivSVG.ContentMode = UIViewContentMode.ScaleToFill;

            container.AddSubview(ivPreview);
            container.AddSubview(ivSVG);

            ivPreview.Frame = new CoreGraphics.CGRect(0, 0, container.Frame.Size.Width, container.Frame.Size.Height);
            ivPreview.AutoresizingMask = UIViewAutoresizing.FlexibleHeight | UIViewAutoresizing.FlexibleWidth;

            ivSVG.Frame = new CoreGraphics.CGRect(0, 0, container.Frame.Size.Width, container.Frame.Size.Height);
            ivSVG.AutoresizingMask = UIViewAutoresizing.FlexibleHeight | UIViewAutoresizing.FlexibleWidth;

            if (Control == null)
                SetNativeControl(container);

            AppDelegate.instance.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.