App Starters (v2.5.x)

Xamarin Forms Shopping Cart

Getting Started

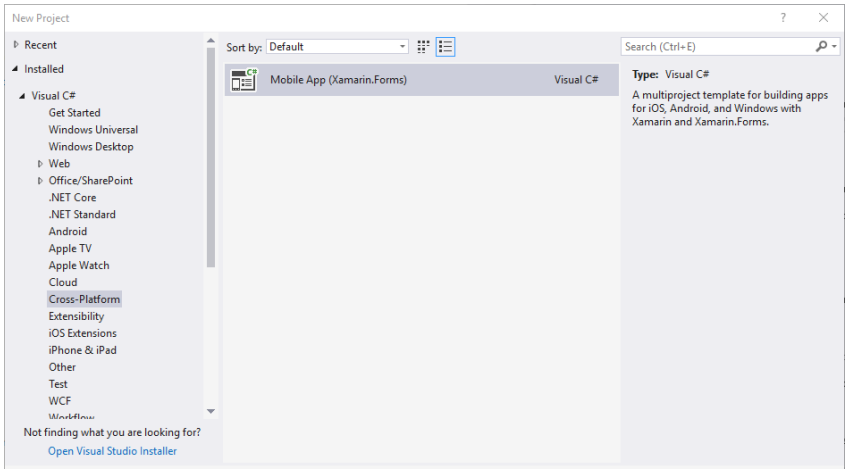
To start developing Xamarin application first you need to install Visual Studio or Xamarin Studio and make sure to include all necessary Xamarin components. For this example, we will use Visual Studio. On this link you can read step by step how to download and install Visual Studio for Xamarin applications.

In this section we have basic explanation how this demo project is developed. If you need more detailed explanation and more information about the objects and methods please check this link

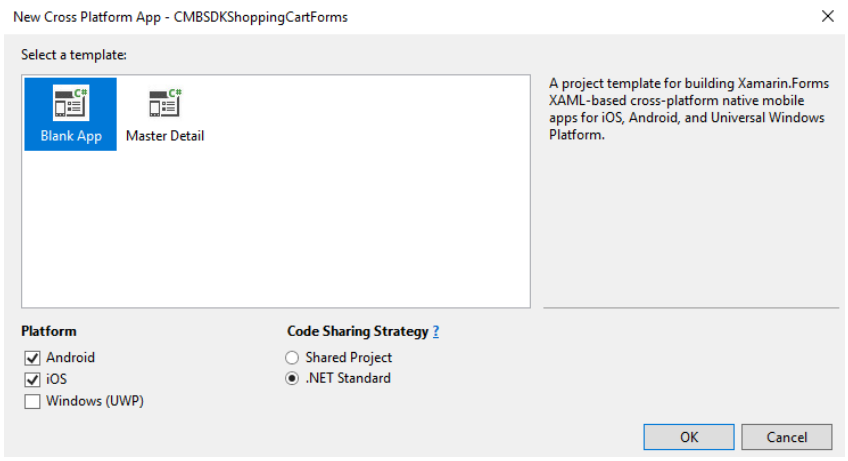
Once you finish with installation open Visual Studio and follow these steps:

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

2. Create **Mobile App (Xamarin.Forms).**



3. Select **Blank App** template for **Android** and **iOS** platform and **.NET Standard** code sharing strategy.



This solution contains three projects:

1.Portable Class Libraries (PCL) Project

A Portable Class Library (PCL) is a special type of project that can be used across disparate CLI platforms such as Xamarin.iOS and Xamarin.Android, as well as WPF, Universal Windows Platform, and Xbox. The library can only utilize a subset of the complete .NET framework, limited by the platforms being targeted.

2. Android Platform-Specific Application Project

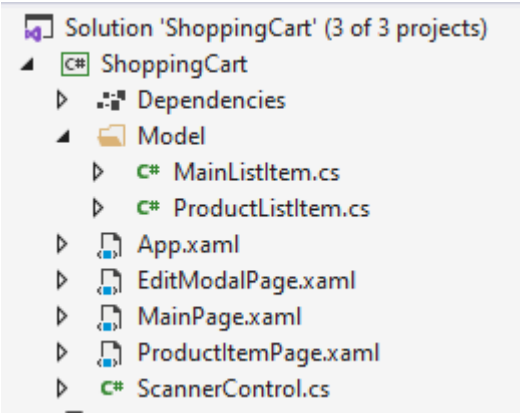
Android platform-specific project must reference the assemblies required to bind Xamarin.Android platform SDK, as well as the Core, shared code project.

3. iOS Platform-Specific Application Project

iOS platform-specific project must reference the assemblies required to bind Xamarin.iOS platform SDK, as well as the Core, shared code project.

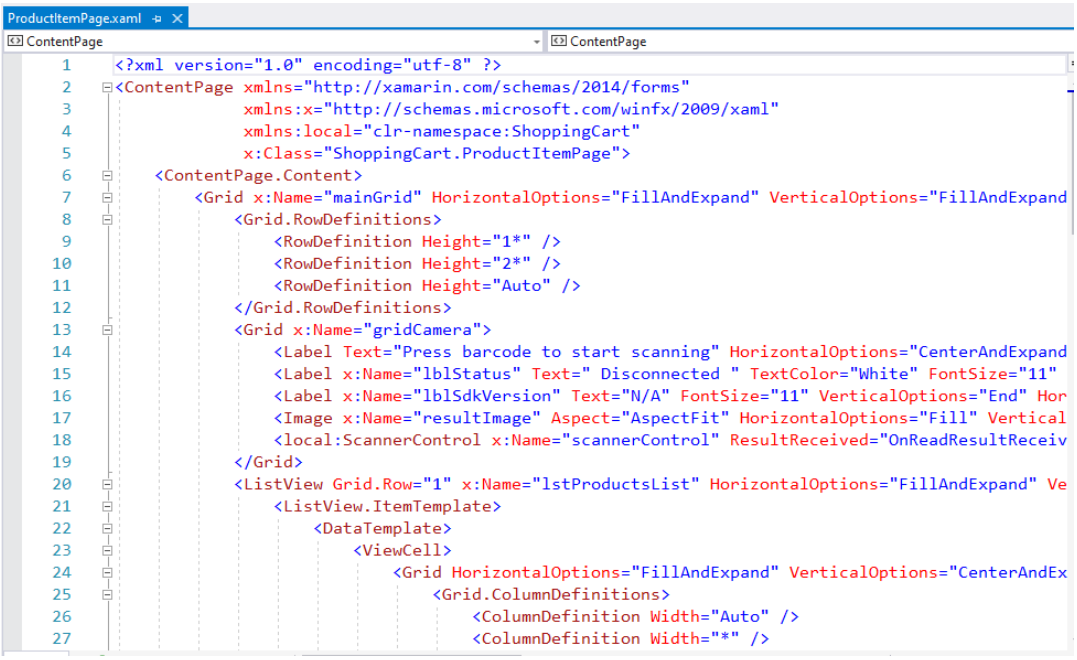
Portable Class Libraries (PCL) Project

In the portable project we have **MainPage** where all main lists are presented, **ProductItemPage** where are shown all products for a specific list, **EditModalPage** is a custom popup to edit list/product name, **MainListItem** and **ProductListItem** models that present list items for lists/products, and **ScannerControl** custom View control.



ScannerControl is a class that inherits from Xamarin.Forms.View control and we add some additional custom properties and event handlers. Later we will implement custom renderer for this class in platform-specific projects.

Here we are using **ScannerControl** in **ProductItemPage** to add new or edit an existing items.



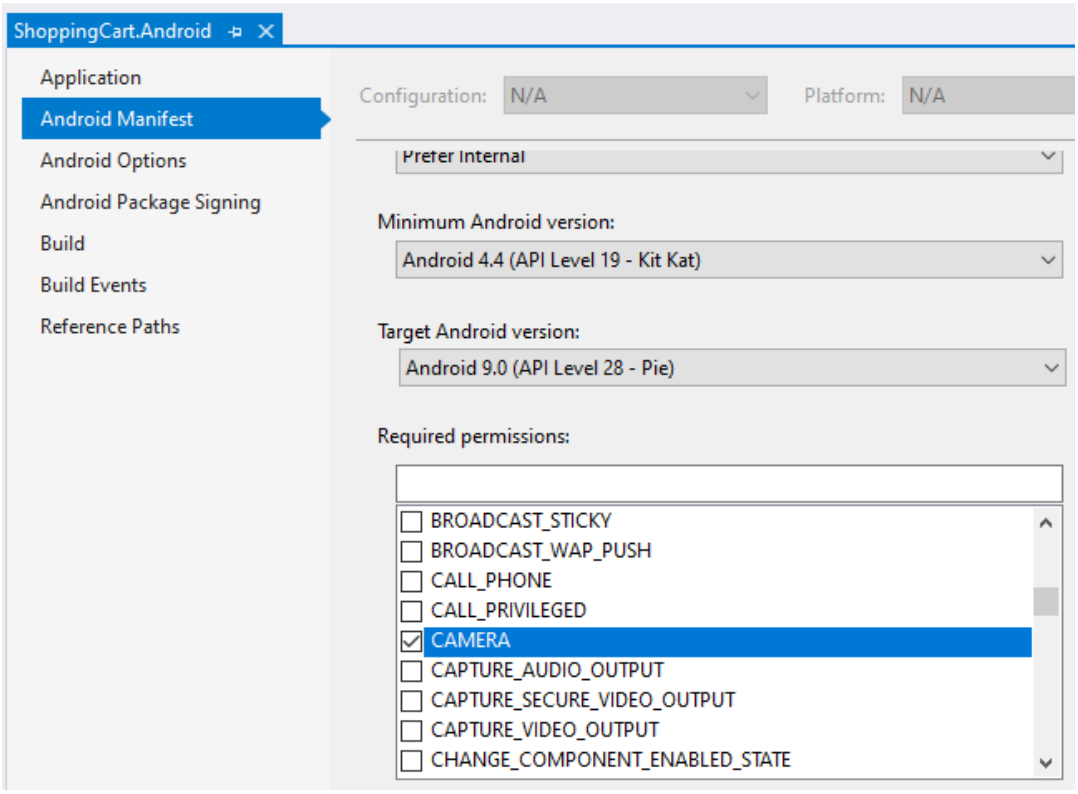
In code behind first we need to initialize, connect and configure **scannerControl** in order to start scanning process

With **scannerControl.StartScanning()** we start the scanning process

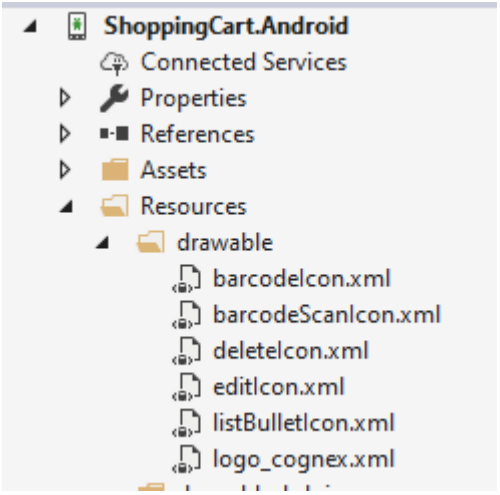
Android Platform-Specific Project

In this project we will set all settings that we need for android platform (min android version, target android version, app name, package name ..), require permissions that we need, add resources for android platform , create custom renderers for android platform , etc..

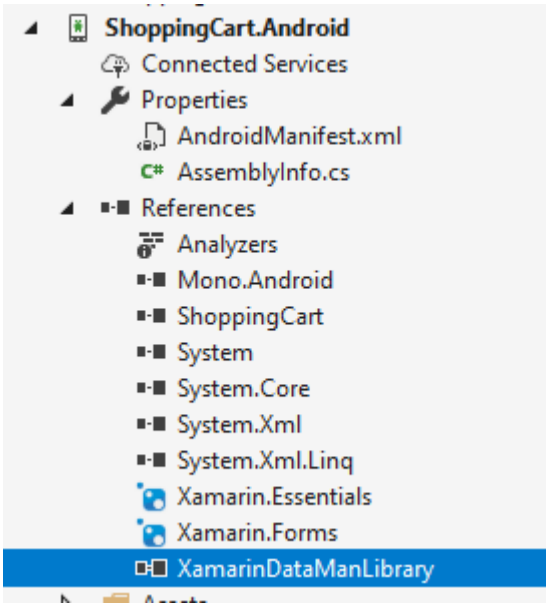
First we will check **Camera** as required permission for this application.



Next will add resources(that we use in portable project) in drawable folder.



Now we need to reference **XamarinDataManLibrary.dll** in order to use the cmbSDK.



*At the time when this document is written, there is a bug for Xamarin Forms with navigation bar icon on the android platform, that's why in this project we have a small modification on **Toolbar.axml** and there is a custom renderer for Navigation Page(**CustomNavigationRenderer**).*

Here we will create custom renderer for ScannerControl (PCL custom control) for Android platform. You don't need to edit this class. Use the same one in your project

```
[assembly: Xamarin.Forms.ExportRenderer(typeof(ShoppingCart.ScannerControl), typeof(ShoppingCart.Droid.ScannerControl))]
namespace ShoppingCart.Droid
{
    public class ScannerControl : ViewRenderer<ShoppingCart.ScannerControl, RelativeLayout>, IOnConnectionCompletedListener
        IReaderDeviceListener, IOnSymbologyListener
    {
        private RelativeLayout rlMainContainer;

        private ReaderDevice readerDevice;
        private bool availabilityListenerStarted = false;

        private static Bitmap svgBitmap;

        public ScannerControl(Context context) : base(context)
        {
        }

        protected override void OnElementChanged(ElementChangedEventArgs<ShoppingCart.ScannerControl> e)
        {
            base.OnElementChanged(e);

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

            rlMainContainer = new RelativeLayout(Context)
            {
                LayoutParameters = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MatchParent, RelativeLayout.
            };

...
```

iOS Platform-Specific Project

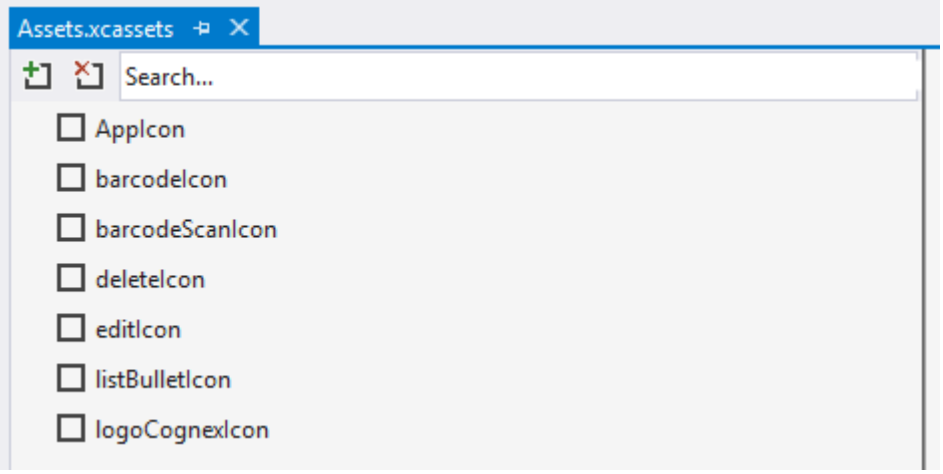
In this project, we will set all settings that we need for the ios platform (deployment target, app name, bundle identifier ..), require permissions that we need, add assets, create custom renderers for the ios platform, etc...

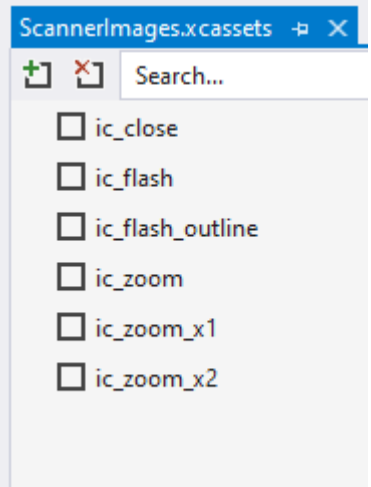
Open **Info.plist** file with some text editor and add the following lines:

```
<key>NSCameraUsageDescription</key>
<string>Camera used for scanning</string>
```

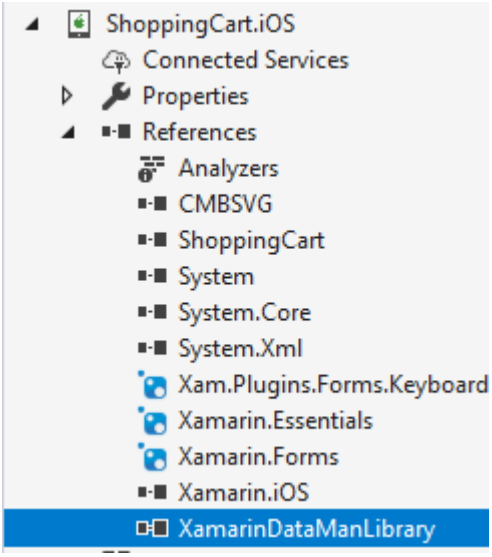
NSCameraUsageDescription key is for camera permission.

Next will add new icons in the Assets catalog and will create one more asset catalog named **ScannerImages** and add icons in that catalog





Now we need to reference **XamarinDataManLibrary.dll** in order to use the cmbSDK.



ScannerControl class is custom renderer for **ScannerControl(PCL custom control)** for iOS platform.

```
[assembly: Xamarin.Forms.ExportRenderer(typeof(ShoppingCart.ScannerControl), typeof(ShoppingCart.iOS.ScannerControl))]
namespace ShoppingCart.iOS
{
    public class ScannerControl : ViewRenderer<ShoppingCart.ScannerControl, UIView>, ICMBReaderDeviceDelegate
    {
        private UIView container;

        private CMBReaderDevice readerDevice;
        private CDMCameraMode cameraMode = CDMCameraMode.NoAimer;

        private UIAlertController connectingAlert;

        private NSObject didBecomeActiveObserver;

        protected override void OnElementChanged(ElementChangedEventArgs<ShoppingCart.ScannerControl> e)
        {
            base.OnElementChanged(e);

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

            container = new UIView();

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


...
```

Licensing the SDK

If you plan to use the **cmbSDK** to do mobile scanning with a smartphone or a tablet (without the MX mobile terminal), the SDK requires the installation of a license key. Without a license key, the SDK will still operate, although scanned results will be blurred (the SDK will randomly replace characters in the scan result with an asterisk character).

Contact your Cognex Sales Representative for information on how to obtain a license key including trial licenses which can be used for 30 days to evaluate the SDK.

After obtaining your license key there are two ways to add your license key to an application.

For **Android Platform** open your manifest file and add this meta tag inside the application tag

```
<meta-data android:name="MX_MOBILE_LICENSE" android:value="YOUR_MX_MOBILE_LICENSE" />
```

or you can register your SDK directly from code when you create camera scanner

```
// Create a scanner device
        private void CreateScannerDevice()
        {
            //*********************************************************************************************
            // Create a camera scanner
            //
            // NOTE: SDK requires a license key. Refer to
            //       the SDK's documentation on obtaining a license key as well as the methods for
            //       passing the key to the SDK (in this example, we're relying on an entry in
            //       plist.info and androidmanifest.xml--also sdk key can be passed
            //       as a parameter in this (GetPhoneCameraDevice) constructor).
            //*********************************************************************************************
            scannerControl.GetPhoneCameraDevice(ScannerCameraMode.NoAimer, ScannerPreviewOption.Defaults, false, "SDK_KEY")

            // Connect to device
            scannerControl.Connect();
        }
```

For **iOS Platform** open Info.plist file and add this key

```
<key>MX_MOBILE_LICENSE</key>
<string>Your license key</string>
```

or you can register your SDK directly from code when you create camera scanner

```
// Create a scanner device
        private void CreateScannerDevice()
        {
            //*********************************************************************************************
            // Create a camera scanner
            //
            // NOTE: SDK requires a license key. Refer to
            //       the SDK's documentation on obtaining a license key as well as the methods for
            //       passing the key to the SDK (in this example, we're relying on an entry in
            //       plist.info and androidmanifest.xml--also sdk key can be passed
            //       as a parameter in this (GetPhoneCameraDevice) constructor).
            //*********************************************************************************************
            scannerControl.GetPhoneCameraDevice(ScannerCameraMode.NoAimer, ScannerPreviewOption.Defaults, false, "SDK_KEY")

            // Connect to device
            scannerControl.Connect();
        }
```