# Camera and Image Example

The camera is not simulated on the iOS simulator, so we’ll just be creating an Android project. We’ll use two plugins from NUGET to help access the camera.

## Plugins

Install the plugins from NUGET into the Android and Shared projects only: Xam.Plugin.Media

Android Manifest:

Under properties folder in solution explorer for the Android project, open Android Manifest xml file and add the following between the <application> tags, you will need to change the :

<provider android:name="android.support.v4.content.FileProvider"

android:authorities="${applicationId}.fileprovider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths"></meta-data>

</provider>

Add a new folder called xml to the Resources folder of the Android project then add a new xml file called file\_paths.xml. Inside the file add:

<?xml version="1.0" encoding="utf-8"?>

<paths xmlns:android="http://schemas.android.com/apk/res/android">

<external-files-path name="my\_images" path="Pictures" />

<external-files-path name="my\_movies" path="Movies" />

</paths>

## Assembly Info

Open the Assembly Info C# file under properties in the solution explorer for Android project and add:

[assembly: UsesFeature("android.hardware.camera", Required = false)]

[assembly: UsesFeature("android.hardware.camera.autofocus", Required = false)]

# Interface to Take Pictures

In App.xaml.cs constructor create a simple interface with an Image and a Button and code to take a photo:

StackLayout layout = new StackLayout();

Label lblPath = new Label { Text = "Path" };

Image photoImage = new Image();

Button btnCam = new Button { Text = "Take Photo" };

btnCam.Clicked += async (sender, e) =>

{

string filename = "img";

var photo = await Plugin.Media.CrossMedia.Current.TakePhotoAsync(new Plugin.Media.Abstractions.StoreCameraMediaOptions() { Name = filename });

if (photo != null)

{

lblPath.Text = photo.Path;

photoImage.Source = ImageSource.FromStream(() => { return photo.GetStream(); });

}

};

layout.Children.Add(lblPath);

layout.Children.Add(photoImage);

layout.Children.Add(btnCam);

MainPage = new ContentPage { Content = layout};

Run the app and try it out. The first time trying to take a photo might be weird as it will ask for permission and the actual camera app might need to be initialized, so take a picture and back out and then try again – the second time should work properly.

## List View to Show Photos Taken

Not going to go into great depth here, but we can show all of the photos we’ve taken by scanning the directory they are saved to. Note that we’ll have to give them unique names or they will be overwritten.

// Replace tmp.jpg

string filename = DateTime.Now.Millisecond.ToString() + DateTime.Now.Day.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Year.ToString();

Make a public list view that we can access from the Android project.

public ListView lFiles = new ListView { HeightRequest = 400, ItemTemplate = new DataTemplate(typeof(PhotoCell))};

Define a Photo Cell that extends View Cell

public class PhotoCell : ViewCell

{

public PhotoCell()

{

Image img = new Image();

// since the bound object is a string, “.” has it bind to the string's value

img.SetBinding(Image.SourceProperty,".");

Label lblPath = new Label { };

lblPath.SetBinding(Label.TextProperty, ".");

StackLayout layout = new StackLayout

{

Orientation = StackOrientation.Horizontal,

Children = {lblPath, img}

};

View = layout;

}

}

Add the list to the stack instead of the Image and Label.

In the Android project at the end of On Create, we will set the List View’s Item Source (change XamCam to whatever project name was used):

// normally this should be a call to Android.OS.Environment.GetExternalETC to find right location

string photoPath = "/storage/emulated/0/Android/data/com.companyname.XamCam/files/Pictures/";

## (Xamarin.Forms.Application.Current as App).lFiles.ItemsSource = System.IO.Directory.GetFiles(photoPath);

## Refreshing the List View

What is happening in our app while the camera is running? It is sleeping. We can add code to the On Resume method of the App class to deal with it resuming. In this case, we’ll refresh the List View:

lFiles.BeginRefresh(); //(place in OnResume)

Then we should define what happens in that event for the list view:

lFiles.Refreshing += (sender, e) =>

{

string photoPath = "/storage/emulated/0/Android/data/com.companyname.XamCam/files/Pictures/";

lFiles.ItemsSource = System.IO.Directory.GetFiles(photoPath);

lFiles.EndRefresh(); // if missing the “refreshing” icon will stay

};