Kostenloses eBook

LERNEN Xamarin.Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.android

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xamarin-android

It is an unofficial and free Xamarin.Android ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Xamarin.Android.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Xamarin.Android

Bemerkungen

Mit Xamarin.Android können Sie native Android-Anwendungen mit den gleichen UI-Steuerelementen wie in Java erstellen, mit Ausnahme der Flexibilität und Eleganz einer modernen Sprache (C #), der Leistungsfähigkeit der .NET Base Class Library (BCL) und zwei erstklassige IDEs - Xamarin Studio und Visual Studio - an ihren Fingerspitzen.

Weitere Informationen zum Installieren von Xamarin.Android auf einem Mac- oder Windows-Computer finden Sie in den Handbüchern "Erste Schritte" im Xamarin Developer Center

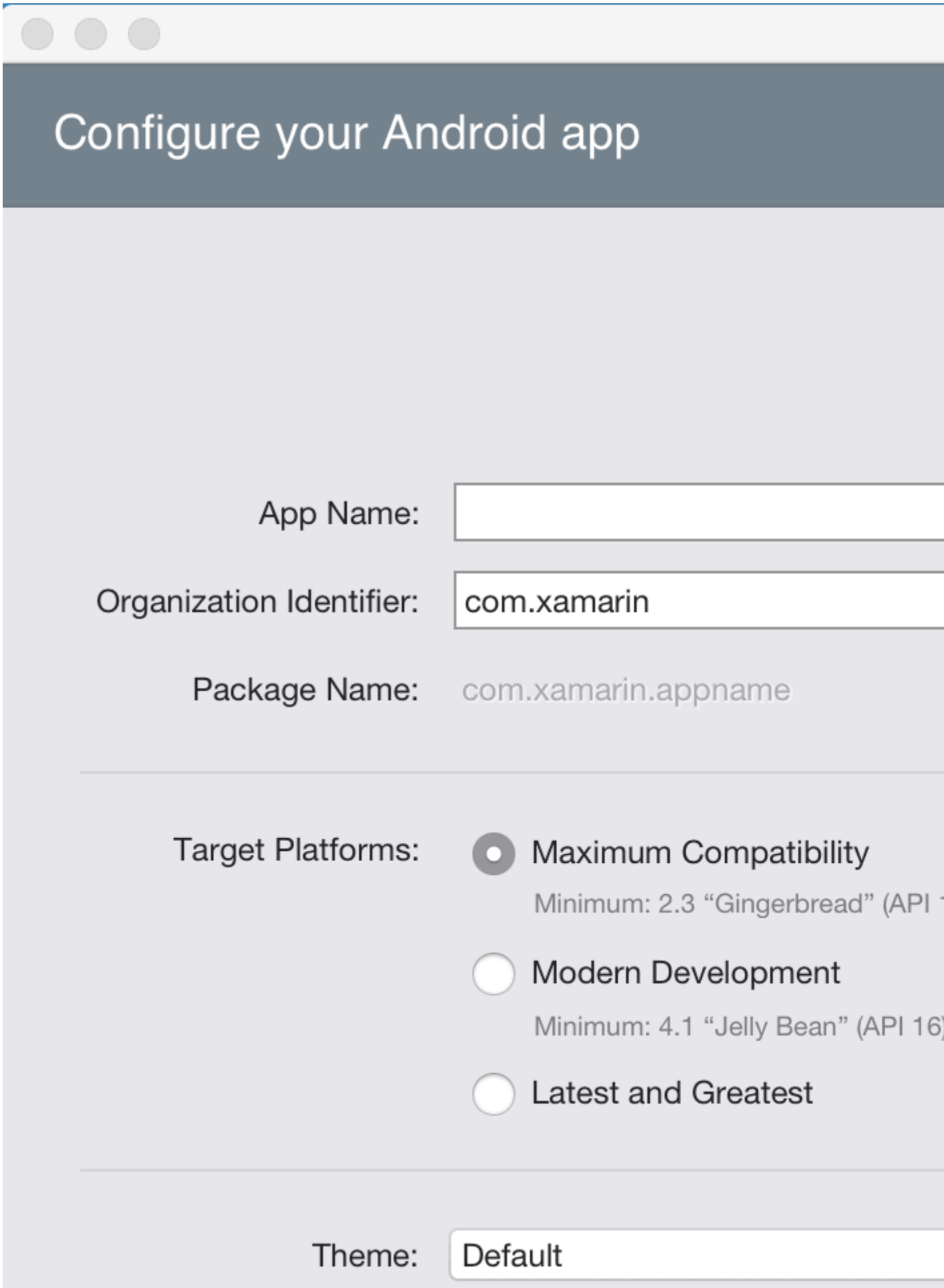
Versionen

Ausführung	Code Name	API-Ebene	Veröffentlichungsdatum
1,0	Keiner	1	2008-09-23
1.1	Keiner	2	2009-02-09
1,5	Cupcake	3	2009-04-27
1.6	Krapfen	4	2009-09-15
2,0-2,1	Eclair	5-7	2009-10-26
2.2-2.2.3	Froyo	8	2010-05-20
2.3-2.3.7	Lebkuchen	9-10	2010-12-06
3.0-3.2.6	Bienenwabe	11-13	2011-02-22
4,0-4,0,4	Eiscreme-Sandwich	14-15	2011-10-18
4.1-4.3.1	Geleebohne	16-18	2012-07-09
4,4-4,4,4, 4,4W-4,4W.2	KitKat	19-20	2013-10-31
5,0-5,1,1	Lutscher	21-22	2014-11-12
6,0-6,0,1	Marshmallow	23	2015-10-05
7,0	Nougat	24	2016-08-22

Examples

Beginnen Sie in Xamarin Studio

1. Navigieren Sie zu **Datei> Neu> Lösung** , um das Dialogfeld für das neue Projekt aufzurufen.
2. Wählen Sie **Android App** und klicken Sie auf **Weiter** .
3. Konfigurieren Sie Ihre App, indem Sie Ihren App-Namen und Ihre Organisations-ID festlegen. Wählen Sie die für Ihre Anforderungen am besten geeignete Zielplattform aus oder übernehmen Sie sie als Standard. Weiter drücken:

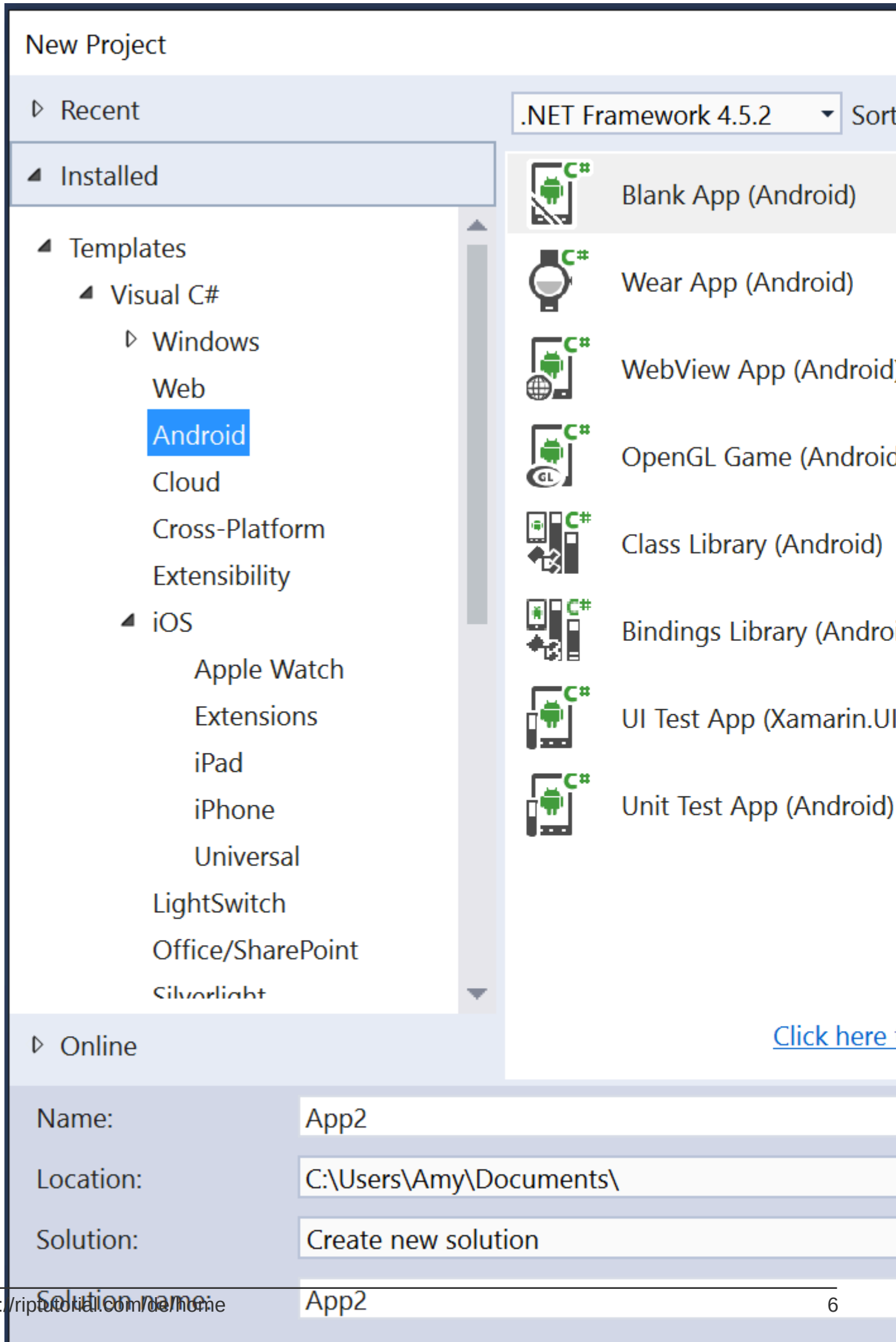

Configure your Android app
App Name:
Organization Identifier:
com.xamarin
Package Name:
com.xamarin.appname
Target Platforms:
Maximum Compatibility
Minimum: 2.3 “Gingerbread” (API
Modern Development
Minimum: 4.1 “Jelly Bean” (API 16)
Latest and Greatest
Theme:
Default

4. Legen Sie den Projektnamen und den Lösungsnamen fest oder übernehmen Sie den Standardnamen. Klicken Sie auf Erstellen, um Ihr Projekt zu erstellen.
5. Richten Sie Ihr Gerät für die Bereitstellung ein oder konfigurieren Sie einen Emulator
6. Um Ihre Anwendung auszuführen, wählen Sie die **Debug-** Konfiguration aus und drücken Sie die Wiedergabetaste:

Konfiguration aus und drücken Sie die Wiedergabetaste:

Starten Sie in Visual Studio

1. Navigieren Sie zu **Datei> Neu> Projekt** , um das Dialogfeld Neues Projekt aufzurufen.
2. Navigieren Sie zu **Visual C #> Android** und wählen Sie Leere App:


New Project
Recent
.NET Framework 4.5.2
Sort
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
Apple Watch
Extensions
iPad
iPhone
Universal
LightSwitch
Office/SharePoint
Silverlight
Online
Blank App (Android)
Wear App (Android)
WebView App (Android)
OpenGL Game (Android
Class Library (Android)
Bindings Library (Andro
UI Test App (Xamarin.UI
Unit Test App (Android)
Click here
Name:
App2
Location:
C:\Users\Amy\Documents\
Solution:
Create new solution
Solution name:
App2

3. Geben Sie Ihrer App einen **Namen** und drücken Sie **OK** , um Ihr Projekt zu erstellen.
4. Richten Sie Ihr Gerät für die Bereitstellung ein oder konfigurieren Sie einen Emulator
5. Um Ihre Anwendung auszuführen, wählen Sie die **Debug-** Konfiguration aus und klicken Sie

   auf die Schaltfläche **Start** :
   Konfiguration aus und klicken Sie auf die Schaltfläche **Start** :

Erste Schritte mit Xamarin.Android online lesen: https://riptutorial.com/de/xamarin-android/topic/403/erste-schritte-mit-xamarin-android

Kapitel 2: App-Lebenszyklus - Xamarin.Andorid

Einführung

Der Lebenszyklus von Xamarin.Android-Anwendungen entspricht der normalen Android-App. Wenn wir über den Lebenszyklus sprechen, müssen wir über den Anwendungslebenszyklus, den Aktivitätslebenszyklus und den Fragment-Lebenszyklus sprechen.

Im Folgenden werde ich versuchen, eine gute Beschreibung und Verwendung zu geben. Ich habe diese Dokumentation von der offiziellen Android- und Xamarin-Dokumentation und vielen hilfreichen Webressourcen erhalten, die im Abschnitt "Anmerkungen" unten aufgeführt sind.

Bemerkungen

Interessante Links zu einem umfassenden Wissen über den Lebenszyklus von Android-Anwendungen:

https://developer.android.com/reference/android/app/Activity.html

http://www.vogella.com/tutorials/AndroidLifeCycle/article.html

https://github.com/xxv/android-lifecycle

https://developer.android.com/guide/components/fragments.html

https://developer.xamarin.com/guides/android/platform_features/fragments/part_1__creating_a_fragment/

https://developer.android.com/guide/components/activities/activity-lifecycle.html

Examples

Lebenszyklus der Anwendung

Zunächst sollten Sie wissen, dass Sie die Android.Application-Klasse erweitern können, sodass Sie auf zwei wichtige Methoden zugreifen können, die sich auf den App-Lebenszyklus beziehen:

- OnCreate - Wird beim Start der Anwendung aufgerufen, bevor andere Anwendungsobjekte erstellt wurden (wie MainActivity).
- OnTerminate - Diese Methode wird in emulierten Prozessumgebungen verwendet. Es wird niemals auf einem Android-Produktionsgerät aufgerufen, bei dem Prozesse durch einfaches Abschalten entfernt werden. Dabei wird kein Benutzercode (einschließlich dieses Callbacks) ausgeführt. Aus der Dokumentation: https://developer.android.com/reference/android/app/Application.html#onTerminate ()

In der Xamarin.Android-Anwendung können Sie die Anwendungsklasse auf die unten dargestellte Weise erweitern. Fügen Sie Ihrem Projekt die neue Klasse "MyApplication.cs" hinzu:

```
[Application]
public class MyApplication : Application
{
    public MyApplication(IntPtr handle, JniHandleOwnership ownerShip) : base(handle,
ownerShip)
    {
    }

    public override void OnCreate()
    {
        base.OnCreate();
    }

    public override void OnTerminate()
    {
        base.OnTerminate();
    }
}
```

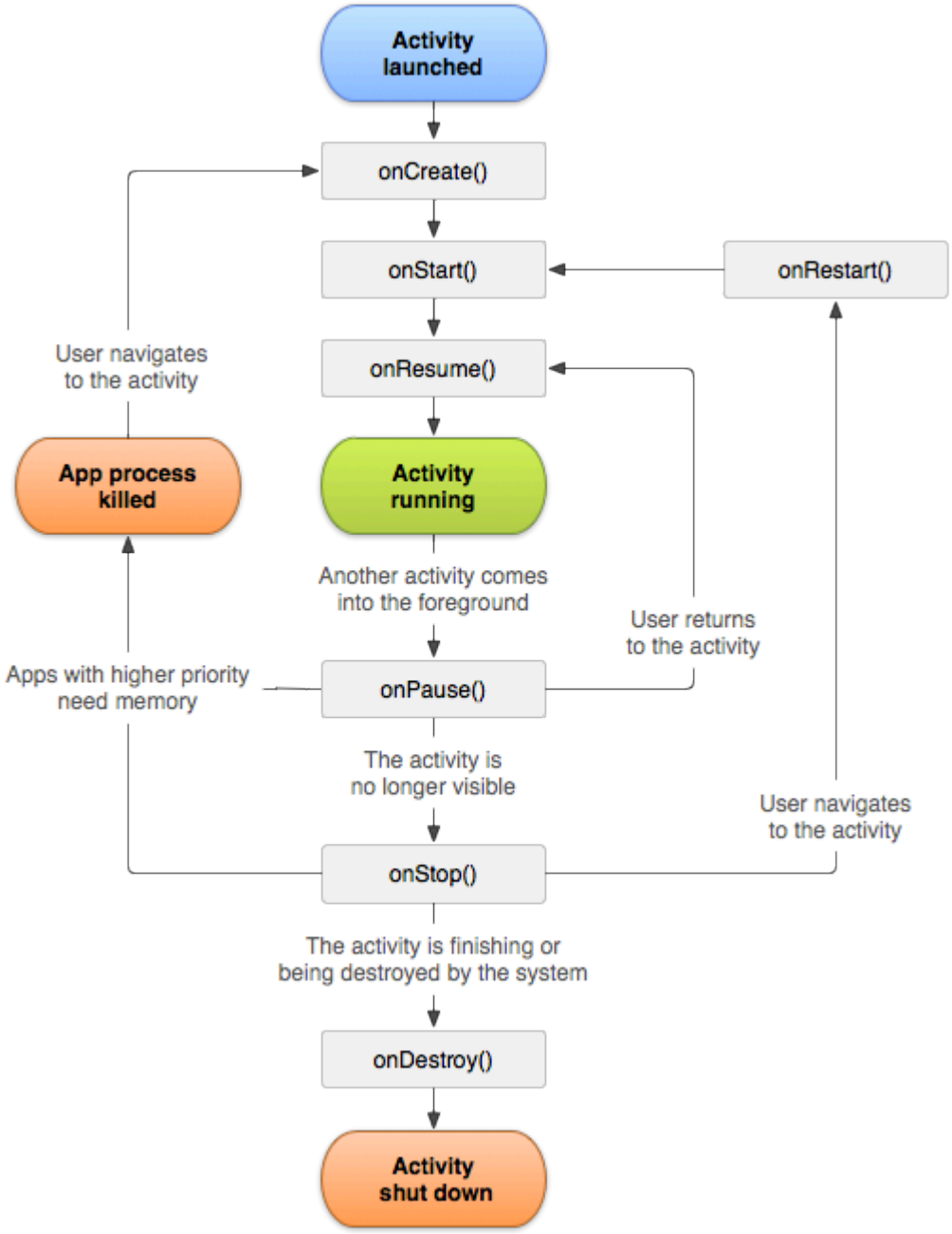
Wie Sie oben geschrieben haben, können Sie die OnCreate-Methode verwenden. Sie können hier beispielsweise die lokale Datenbank initialisieren oder eine zusätzliche Konfiguration einrichten.

Es gibt auch mehr Methoden, die überschrieben werden können: OnConfigurationChanged oder OnLowMemory.

Lebenszyklus der Aktivität

Der Lebenszyklus von Aktivitäten ist sehr viel komplexer. Wie Sie wissen, ist Aktivität eine einzige Seite in der Android-App, auf der Benutzer Interaktionen ausführen können.

In der folgenden Abbildung können Sie sehen, wie der Lebenszyklus von Android-Aktivitäten aussieht:



Wie Sie sehen, gibt es einen bestimmten Ablauf des Aktivitätslebenszyklus. In der mobilen Anwendung gibt es natürlich in jeder Aktivitätsklasse Methoden, die ein bestimmtes Lebenszyklusfragment behandeln:

```
[Activity(Label = "LifecycleApp", MainLauncher = true, Icon = "@mipmap/icon")]
public class MainActivity : Activity
{
    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        Log.Debug("OnCreate", "OnCreate called, Activity components are being created");

        // Set our view from the "main" layout resource
        SetContentView(Resource.Layout.MainActivity);
    }

    protected override void OnStart()
    {
```

```
        Log.Debug("OnStart", "OnStart called, App is Active");
        base.OnStart();
    }

    protected override void OnResume()
    {
        Log.Debug("OnResume", "OnResume called, app is ready to interact with the user");
        base.OnResume();
    }

    protected override void OnPause()
    {
        Log.Debug("OnPause", "OnPause called, App is moving to background");
        base.OnPause();
    }

    protected override void OnStop()
    {
        Log.Debug("OnStop", "OnStop called, App is in the background");
        base.OnStop();
    }

    protected override void OnDestroy()
    {
        base.OnDestroy();
        Log.Debug("OnDestroy", "OnDestroy called, App is Terminating");
    }
}
```

Es gibt eine gute Beschreibung in der offiziellen Android-Dokumentation:

- Die gesamte Lebensdauer einer Aktivität geschieht zwischen dem ersten Aufruf von onCreate (Bundle) bis zu einem einzigen letzten Aufruf von onDestroy (). Eine Aktivität führt alle Einstellungen des "globalen" Status in onCreate () durch und gibt alle verbleibenden Ressourcen in onDestroy () frei. Wenn beispielsweise ein Thread im Hintergrund ausgeführt wird, um Daten aus dem Netzwerk herunterzuladen, kann er diesen Thread in onCreate () erstellen und dann den Thread in onDestroy () stoppen.

- Die sichtbare Lebensdauer einer Aktivität geschieht zwischen einem Aufruf von onStart () bis zu einem entsprechenden Aufruf von onStop (). Während dieser Zeit kann der Benutzer die Aktivität auf dem Bildschirm sehen, obwohl sie sich möglicherweise nicht im Vordergrund befindet und mit dem Benutzer interagiert. Zwischen diesen beiden Methoden können Sie Ressourcen verwalten, die benötigt werden, um dem Benutzer die Aktivität anzuzeigen. Sie können beispielsweise einen BroadcastReceiver in onStart () registrieren, um auf Änderungen zu prüfen, die Auswirkungen auf Ihre Benutzeroberfläche haben, und die Registrierung in onStop () aufheben, wenn der Benutzer nicht mehr sieht, was Sie anzeigen. Die Methoden onStart () und onStop () können mehrmals aufgerufen werden, wenn die Aktivität für den Benutzer sichtbar und verborgen ist.

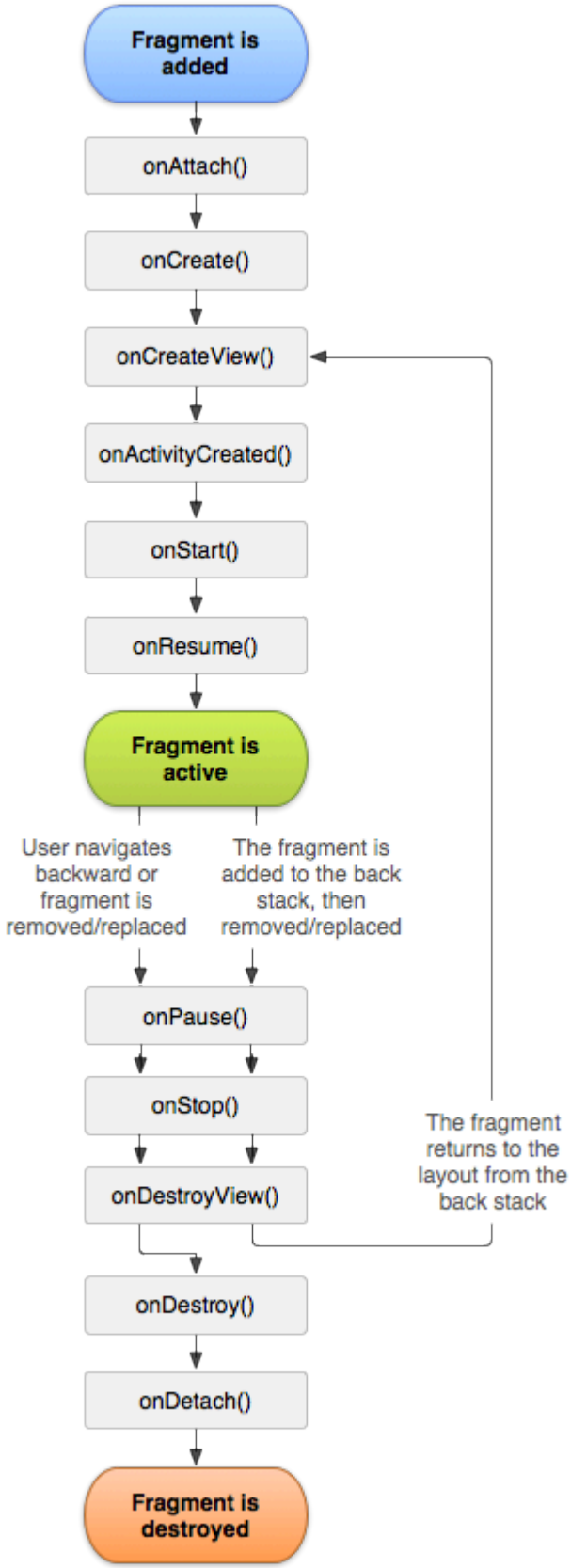
- Die Vordergrundlebensdauer einer Aktivität geschieht zwischen einem Aufruf von onResume () bis zu einem entsprechenden Aufruf von onPause (). Während dieser Zeit befindet sich die Aktivität vor allen anderen Aktivitäten und interagiert mit dem Benutzer. Eine Aktivität kann häufig zwischen den wiederaufgenommenen und den pausierten Zuständen wechseln - zum Beispiel, wenn das Gerät in den Ruhezustand versetzt wird, wenn ein Aktivitätsergebnis

geliefert wird, wenn eine neue Absicht geliefert wird - der Code in diesen Methoden sollte daher recht leicht sein.

Fragment Lebenszyklus

Wie Sie wissen, können Sie eine Aktivität haben, in die jedoch verschiedene Fragmente eingebettet sind. Deshalb ist der Fragment-Lebenszyklus auch für Entwickler wichtig.

In der folgenden Abbildung können Sie sehen, wie der Lebenszyklus eines Android-Fragments aussieht:



Wie in der offiziellen Android-Dokumentation beschrieben, sollten Sie mindestens drei Methoden implementieren:

- OnCreate - das System ruft dies beim Erstellen des Fragments auf. Innerhalb Ihrer Implementierung sollten Sie die wesentlichen Komponenten des Fragments initialisieren, die Sie beibehalten möchten, wenn das Fragment angehalten oder angehalten und

anschließend fortgesetzt wird.

- OnCreateView - Das System ruft dies auf, wenn das Fragment zum ersten Mal seine Benutzeroberfläche zeichnen soll. Um eine Benutzeroberfläche für Ihr Fragment zu zeichnen, müssen Sie eine View aus dieser Methode zurückgeben, die der Stamm des Layouts Ihres Fragments ist. Sie können null zurückgeben, wenn das Fragment keine Benutzeroberfläche bereitstellt.

- OnPause - Das System ruft diese Methode als ersten Hinweis auf, dass der Benutzer das Fragment verlässt (obwohl dies nicht immer bedeutet, dass das Fragment zerstört wird). Hier sollten Sie normalerweise Änderungen vornehmen, die über die aktuelle Benutzersitzung hinaus beibehalten werden sollten (da der Benutzer möglicherweise nicht zurückkommt).

Beispielimplementierung in Xamarin.Android:

```
public class MainFragment : Fragment
{
    public override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);

        // Create your fragment here
        // You should initialize essential components of the fragment
        // that you want to retain when the fragment is paused or stopped, then resumed.
    }

    public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState)
    {
        // Use this to return your custom view for this Fragment
        // The system calls this when it's time for the fragment to draw its user interface
for the first time.

        var mainView = inflater.Inflate(Resource.Layout.MainFragment, container, false);
        return mainView;
    }

    public override void OnPause()
    {
        // The system calls this method as the first indication that the user is leaving the
fragment

        base.OnPause();
    }
}
```

Natürlich können Sie hier zusätzliche Methoden hinzufügen, wenn Sie mit unterschiedlichen Zuständen umgehen möchten.

Vollständiges Beispiel auf GitHub

Wenn Sie ein Basisprojekt mit den unten beschriebenen Methoden erhalten möchten, können Sie die Xamarin.Android-Anwendungsvorlage von meinem GitHub herunterladen. Sie finden Beispiele für:

- Anwendungslebenszyklus-Methoden
- Aktivitätslebenszyklus-Methoden
- Fragment-Lebenszyklusmethoden

https://github.com/Daniel-Krzyczkowski/XamarinAndroid/tree/master/AndroidLifecycle/LifecycleApp

App-Lebenszyklus - Xamarin.Andorid online lesen: https://riptutorial.com/de/xamarin-android/topic/8842/app-lebenszyklus---xamarin-andorid

Kapitel 3: Barcode-Scannen mit der ZXing-Bibliothek in Xamarin-Anwendungen

Einführung

Die Zxing-Bibliothek ist für die Bildverarbeitung bekannt. Zxing basierte auf Java und das .Net-Modul ist ebenfalls verfügbar und kann in Xamarin-Anwendungen verwendet werden. Klicken Sie hier, um die offizielle Dokumentation zu überprüfen. http://zxingnet.codeplex.com/

Ich habe diesen Libarry kürzlich benutzt.

Schritt 1: Fügen Sie die ZXing.Net.Mobile-Komponente zur Lösung hinzu.

Schritt 2: In welcher Aktivität wir den Barcode-Scanner anzeigen müssen, initialisieren Sie den MobileBarcodeScanner.

Schritt 3: Schreiben Sie den folgenden Code, wenn Sie auf eine Ansicht tippen, um den Scanvorgang zu starten.

Examples

Beispielcode

```
button.Click +=async delegate
{
var MScanner = new MobileBarcodeScanner();
var Result = await MScanner.Scan();
if(Result == null)
{
    return;
}
//get the bar code text here
string BarcodeText = Result.text;
}
```

Barcode-Scannen mit der ZXing-Bibliothek in Xamarin-Anwendungen online lesen: https://riptutorial.com/de/xamarin-android/topic/9526/barcode-scannen-mit-der-zxing-bibliothek-in-xamarin-anwendungen

Kapitel 4: Benutzerdefinierte ListView

Examples

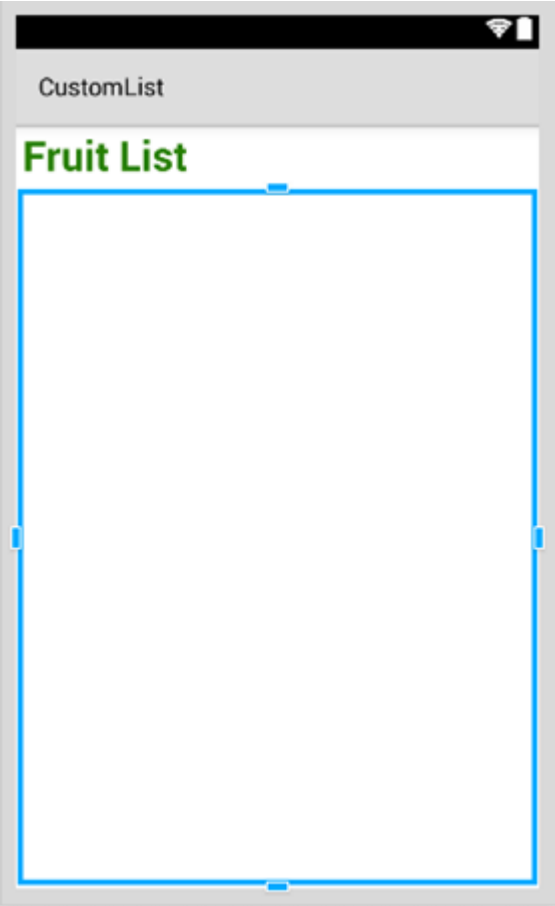
Benutzerdefinierte Listenansicht besteht aus Zeilen, die entsprechend den Anforderungen der Benutzer gestaltet werden.

Für das Layout oben ist Ihre customrow.axml-Datei wie unten gezeigt

```
    <?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">
    <ImageView
        android:id="@+id/Image"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_alignParentLeft="true"
        android:layout_marginRight="8dp"
        android:src="@drawable/icon" />
    <TextView
        android:id="@+id/Text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/Image"
        android:layout_toRightOf="@id/Image"
        android:layout_marginTop="5dip"
        android:text="This is Line1"
        android:textSize="20dip"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/Text2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/Text1"
        android:layout_marginTop="1dip"
```

```
        android:text="This is line2"
        android:textSize="15dip"
        android:layout_toRightOf="@id/Image" />
</RelativeLayout>
```

Dann können Sie Ihre main.axml entwerfen, die eine Textansicht für die Kopfzeile und eine Listenansicht enthält.



Hoffe das ist einfach ...

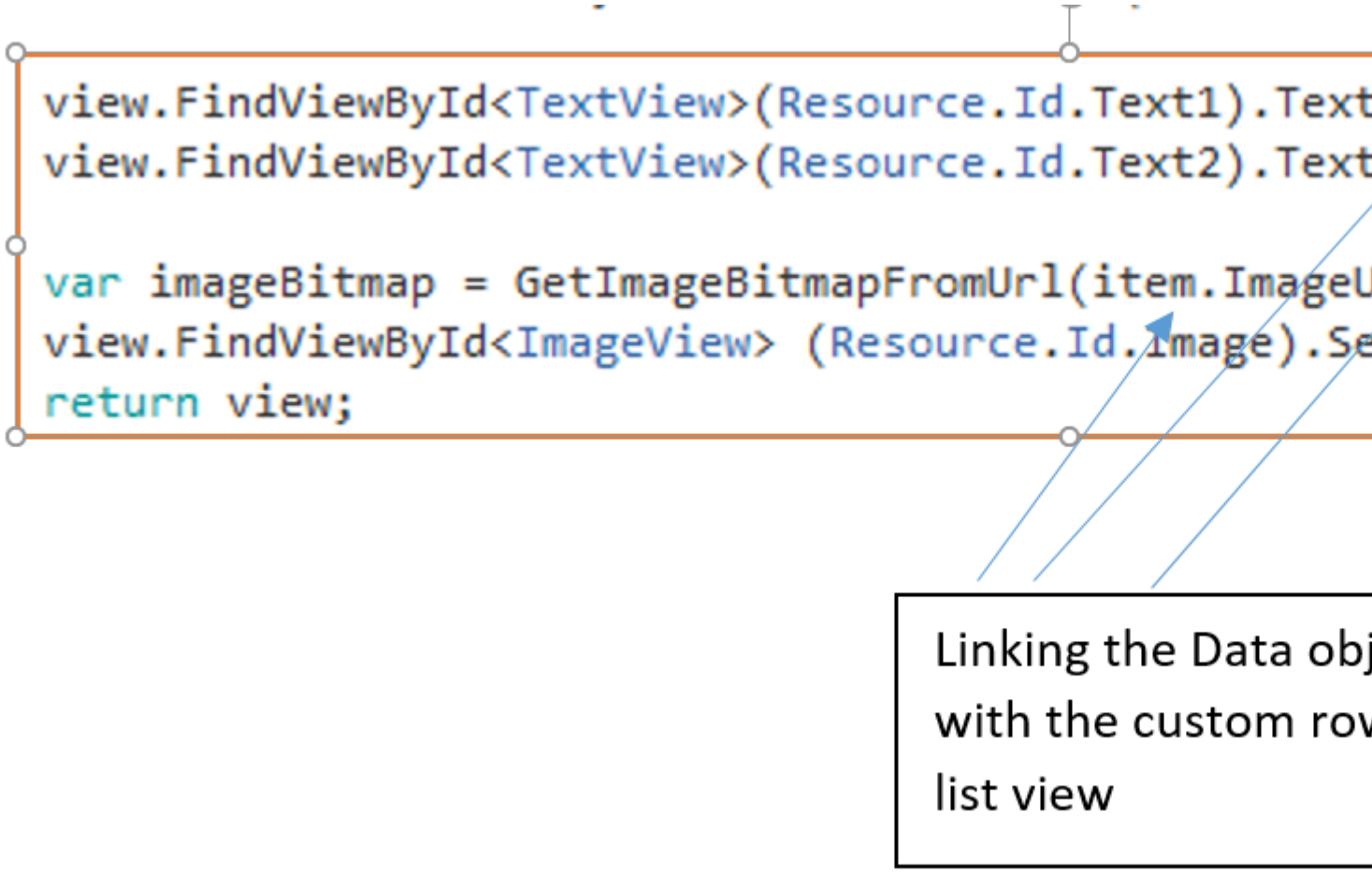
Erstellen Sie als Nächstes Ihre Data.cs-Klasse, die Ihre Zeilenobjekte darstellt

```
public class Data
{
    public string Heading;
    public string SubHeading;
    public string ImageURI;

    public Data ()
    {
        Heading = "";
        SubHeading = "";
        ImageURI = "";
    }
}
```

Als Nächstes benötigen Sie die Klasse DataAdapter.cs. Adapter verknüpfen Ihre Daten mit der zugrunde liegenden Ansicht

```
public class DataAdapter : BaseAdapter<Data> {

    List<Data> items;

    Activity context;
    public DataAdapter(Activity context, List<Data> items)
        : base()
    {
        this.context = context;
        this.items = items;
    }
    public override long GetItemId(int position)
    {
        return position;
    }
    public override Data this[int position]
    {
        get { return items[position]; }
    }
    public override int Count
    {
        get { return items.Count; }
    }
    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        var item = items[position];
        View view = convertView;
        if (view == null) // no view to re-use, create new
            view = context.LayoutInflater.Inflate(Resource.Layout.CustomRow, null);

        view.FindViewById<TextView>(Resource.Id.Text1).Text = item.Heading;
        view.FindViewById<TextView>(Resource.Id.Text2).Text = item.SubHeading;

        var imageBitmap = GetImageBitmapFromUrl(item.ImageURI);
        view.FindViewById<ImageView> (Resource.Id.Image).SetImageBitmap (imageBitmap);
        return view;
    }

    private Bitmap GetImageBitmapFromUrl(string url)
    {
        Bitmap imageBitmap = null;
        if(!(url=="null"))
            using (var webClient = new WebClient())
            {
                var imageBytes = webClient.DownloadData(url);
                if (imageBytes != null && imageBytes.Length > 0)
                {
                    imageBitmap = BitmapFactory.DecodeByteArray(imageBytes, 0,
imageBytes.Length);
                }
            }

        return imageBitmap;
    }

}
```

Der wichtigste Teil befindet sich in der GetView-Funktion. Hier verknüpfen Sie Ihr Objekt mit Ihrer benutzerdefinierten Zeile.



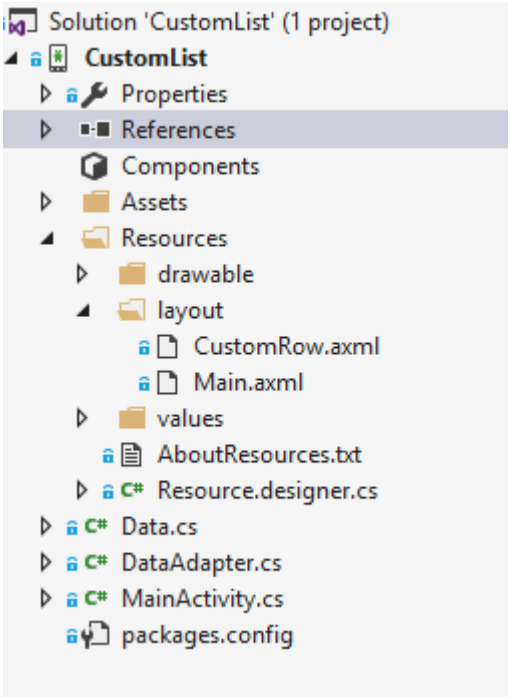
Die GetImageBitmapFromUrl ist nicht Teil des Datenadapters, aber ich habe dies der Einfachheit halber hierher gestellt.

Endlich kommen wir zu den MainActivity.cs

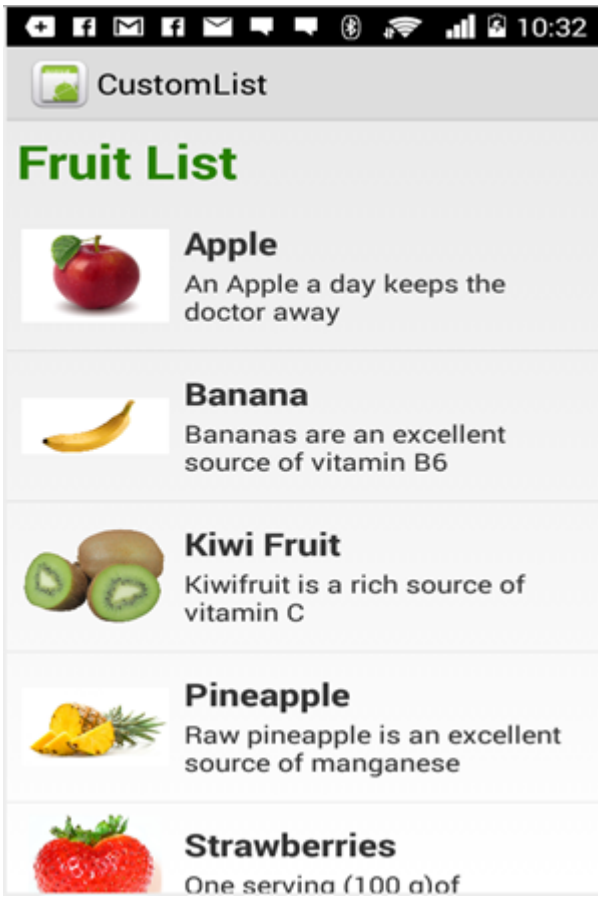
```
public class MainActivity : Activity
{

    ListView listView;

    protected override void OnCreate (Bundle bundle)
    {
        base.OnCreate (bundle);

        // Set our view from the "main" layout resource
        SetContentView (Resource.Layout.Main);
        listView = FindViewById<ListView>(Resource.Id.List);

        List<Data> myList = new List<Data> ();

        Data obj = new Data ();
        obj.Heading = "Apple";
        obj.SubHeading = "An Apple a day keeps the doctor away";
        obj.ImageURI =
"http://www.thestar.com/content/dam/thestar/opinion/editorials/star_s_view_/2011/10/12/an_apple_a_day_


        myList.Add (obj);

        Data obj1 = new Data();
```

```
        obj1.Heading = "Banana";
        obj1.SubHeading = "Bananas are an excellent source of vitamin B6 ";
        obj1.ImageURI =
"http://www.bbcgoodfood.com/sites/bbcgoodfood.com/files/glossary/banana-crop.jpg";

        myList.Add(obj1);

        Data obj2 = new Data();
        obj2.Heading = "Kiwi Fruit";
        obj2.SubHeading = "Kiwifruit is a rich source of vitamin C";
        obj2.ImageURI = "http://www.wiffens.com/wp-content/uploads/kiwi.png";

        myList.Add(obj2);

        Data obj3 = new Data();
        obj3.Heading = "Pineapple";
        obj3.SubHeading = "Raw pineapple is an excellent source of manganese";
        obj3.ImageURI =
"http://www.medicalnewstoday.com/images/articles/276/276903/pineapple.jpg";

        myList.Add(obj3);

        Data obj4 = new Data();
        obj4.Heading = "Strawberries";
        obj4.SubHeading = "One serving (100 g)of strawberries contains approximately 33
kilocalories";
        obj4.ImageURI = "https://ecs3.tokopedia.net/newimg/product-
1/2014/8/18/5088/5088_8dac78de-2694-11e4-8c99-6be54908a8c2.jpg";

        myList.Add (obj4);
        listView.Adapter = new DataAdapter(this,myList);

    }
```

Ihre endgültige Projektstruktur sieht wie folgt aus.



Wenn alles in Ordnung ist, sollten Sie die Ausgabe wie gezeigt sehen



Benutzerdefinierte ListView online lesen: https://riptutorial.com/de/xamarin-android/topic/6406/benutzerdefinierte-listview

Kapitel 5: Bindungen

Examples

Typen entfernen

Es ist möglich, den Xamarin.Android Bindings Generator anzuweisen, einen Java-Typ zu ignorieren und nicht zu binden. Dazu fügen Sie der Datei metadata.xml ein XML-Element zum `remove-node` eines `remove-node` hinzu:

```
<remove-node path="/api/package[@name='{package_name}']/class[@name='{name}']" />
```

Java-Schnittstellen implementieren

Wenn eine Java-Bibliothek Schnittstellen enthält, die vom Benutzer implementiert werden sollen (z. B. `View.IOnClickListener` wie `View.IOnClickListener` oder Callbacks), muss die implementierende Klasse direkt oder indirekt von `Java.Lang.Object` oder `Java.Lang.Throwable` . Dies ist ein häufiger Fehler, da die Verpackungsschritte lediglich eine Warnung ausgeben, die leicht übersehen wird:

> Typ 'MyListener' implementiert Android.Runtime.IJavaObject, erbt jedoch nicht von Java.Lang.Object. Es wird nicht unterstützt.

Falsch

Die Verwendung dieser Implementierung führt zu unerwartetem Verhalten.

```
class MyListener : View.IOnClickListener
{
    public IntPtr Handle { get; }

    public void Dispose()
    {
    }


    public void OnClick(View v)
    {
        // ...
    }
}
```

Richtig

```
class MyListener :
    Java.Lang.Object, // this is the important part
    View.IOnClickListener
{
    public void OnClick(View v)
    {
        // ...
```

```
    }
}
```

Bindungsbibliotheken können Methoden und Schnittstellen umbenennen

Nicht alles in einer Bindungsbibliothek hat in C # denselben Namen wie in Java.

In C # beginnen Schnittstellennamen mit "I", Java hat jedoch keine solche Konvention. Wenn Sie eine Java-Bibliothek importieren, wird eine Schnittstelle namens `SomeInterface` zu `ISomeInterface` .

Ebenso verfügt Java nicht über Eigenschaften wie C #. Wenn eine Bibliothek gebunden ist, können Java-Getter- und Setter-Methoden als Eigenschaften umgestaltet werden. Zum Beispiel der folgende Java-Code

```
public int getX() { return someInt; }

public int setX(int someInt) { this.someInt = someInt; }
```

kann als umgestaltet werden

```
public int X { get; set; }
```

wenn es gebunden ist

Bindungen online lesen: https://riptutorial.com/de/xamarin-android/topic/771/bindungen

Kapitel 6: Dialoge

Bemerkungen

Festlegen des `Context` des Dialogs

Wenn Sie einen `Dialog` aus einer `Activiy` , können wir `this` als Kontext verwenden.

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
```

Bei `Fragments` wir die Eigenschaft `Context` .

```
AlertDialog.Builder builder = new AlertDialog.Builder(Context);
```

Schaltflächentypen

`SetNeutralButton()` kann für eine einfache Benachrichtigung und Bestätigung verwendet werden, dass die Benachrichtigung gelesen wird. `SetPositiveButton()` kann beispielsweise für eine Bestätigung verwendet werden: " `SetPositiveButton()` löschen?" `SetNegativeButton()` dient zum Abbrechen des Dialogs und zum Abbrechen seiner Aktion.

Deaktivieren Sie die Schaltfläche "Abbrechen"

Wenn Sie sicherstellen möchten, dass der Benutzer den Dialog nicht mit der Zurück-Schaltfläche schließen kann, können Sie `SetCanceable(false)` aufrufen. Dies funktioniert nur für die Zurück-Taste.

Drehung

Wenn der Bildschirm gedreht wird, während ein Dialog sichtbar ist, wird er abgebrochen und die OK- und Abbrechen-Aktionen werden nicht aufgerufen. Sie müssen dies in Ihrer Aktivität erledigen und das Dialogfeld erneut anzeigen, nachdem die Aktivität neu geladen wurde.

Um dies zu `DialogFragment` stattdessen ein `DialogFragment` .

Examples

Benachrichtigungsdialog

Alarmdialog erstellen

```
AlertDialog.Builder builder = new AlertDialog.Builder(Context);
builder.SetIcon(Resource.Drawable.Icon);
builder.SetTitle(title);
```

```
builder.SetMessage(message);

builder.SetNeutralButton("Neutral", (evt, args) => {
    // code here for handling the Neutral tap
});

builder.SetPositiveButton("Ok", (evt, args) => {
    // code here for handling the OK tap
});

builder.SetNegativeButton("Cancel", (evt, args) => {
    // code here for handling the Cancel tap
});

builder.SetCancelable(false);
builder.Show();
```

Dialoge online lesen: https://riptutorial.com/de/xamarin-android/topic/2510/dialoge

Kapitel 7: Dialoge

Parameter

häufig verwendete öffentliche Methode	Benutzen
SetTitle (String)	Legt den Titel für den Dialog fest
SetIcon (Drawable)	Symbol für den Alarmdialog festlegen
SetMessage (String)	Stellen Sie die anzuzeigende Nachricht ein.
SetNegativeButton (String, EventHandler)	Legen Sie einen Listener fest, der aufgerufen wird, wenn die negative Schaltfläche des Dialogs gedrückt wird.
SetPositiveButton (String, EventHandler)	Legt fest, dass ein Listener aufgerufen wird, wenn die positive Schaltfläche des Dialogs gedrückt wird.
SetNeutralButton (String, EventHandler)	Legt fest, dass ein Listener aufgerufen wird, wenn die neutrale Schaltfläche des Dialogfelds gedrückt wird.
SetOnCancelListener (IDialogInterfaceOnCancelListener)	Legt den Rückruf fest, der aufgerufen wird, wenn der Dialog abgebrochen wird.
SetOnDismissListener (IDialogInterfaceOnDismissListener)	Legt den Rückruf fest, der aufgerufen wird, wenn der Dialog aus irgendeinem Grund geschlossen wird.
Show()	Erstellt einen AlertDialog mit den an diesen Builder gelieferten Argumenten und den Dialog von Dialog.Show.

Bemerkungen

Bedarf

Namensraum: Android.App

Assembly: Mono.Android (in Mono.Android.dll)

Montageversionen: 0.0.0.0

Öffentliche Konstruktoren

AlertDialog.Builder (Kontext): -

Konstruktor, der einen Kontext für diesen Builder und den von ihm erstellten AlertDialog verwendet.

AlertDialog.Builder (Context, Int32): -

Konstruktor, der einen Kontext und ein Design für diesen Builder und den von ihm erstellten AlertDialog verwendet.

Verwenden des Material Design AlertDialogs

Um den modernen AlertDialog zu nutzen:

1. Installieren Sie die Support v7 AppCompat-Bibliothek aus den NuGet-Paketen
2. Ersetzen Sie AlertDialog durch Android.Support.V7.App.AlertDialog, oder fügen Sie die folgende Anweisung oben hinzu, um den Dialog zum Leuchten zu bringen.

```
        using AlertDialog = Android.Support.V7.App.AlertDialog;
```

Examples

AlertDialog

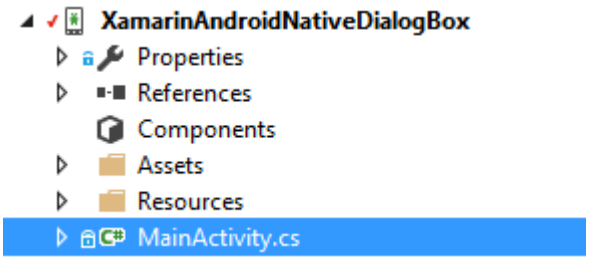
```
// 1. Instantiate an AlertDialog.Builder with its constructor
// the parameter this is the context (usually your activity)
AlertDialog.Builder builder = new AlertDialog.Builder(this);

// 2. Chain together various setter methods to set the dialog characteristics
builder.SetMessage(Resource.String.dialog_message)
       .SetTitle(Resource.String.dialog_title);

// 3. Get the AlertDialog from create()
AlertDialog dialog = builder.Create();

dialog.Show();
```

Beispiel eines einfachen Alert-Dialogs

Wir erstellen einen einfachen Alert-Dialog in Xamarin.Android

Bedenken Sie, dass Sie den Leitfaden zur ersten Inbetriebnahme aus der Dokumentation gelesen haben.

> Sie müssen die Projektstruktur folgendermaßen haben:



Ihre Hauptaktivität muss so aussehen:

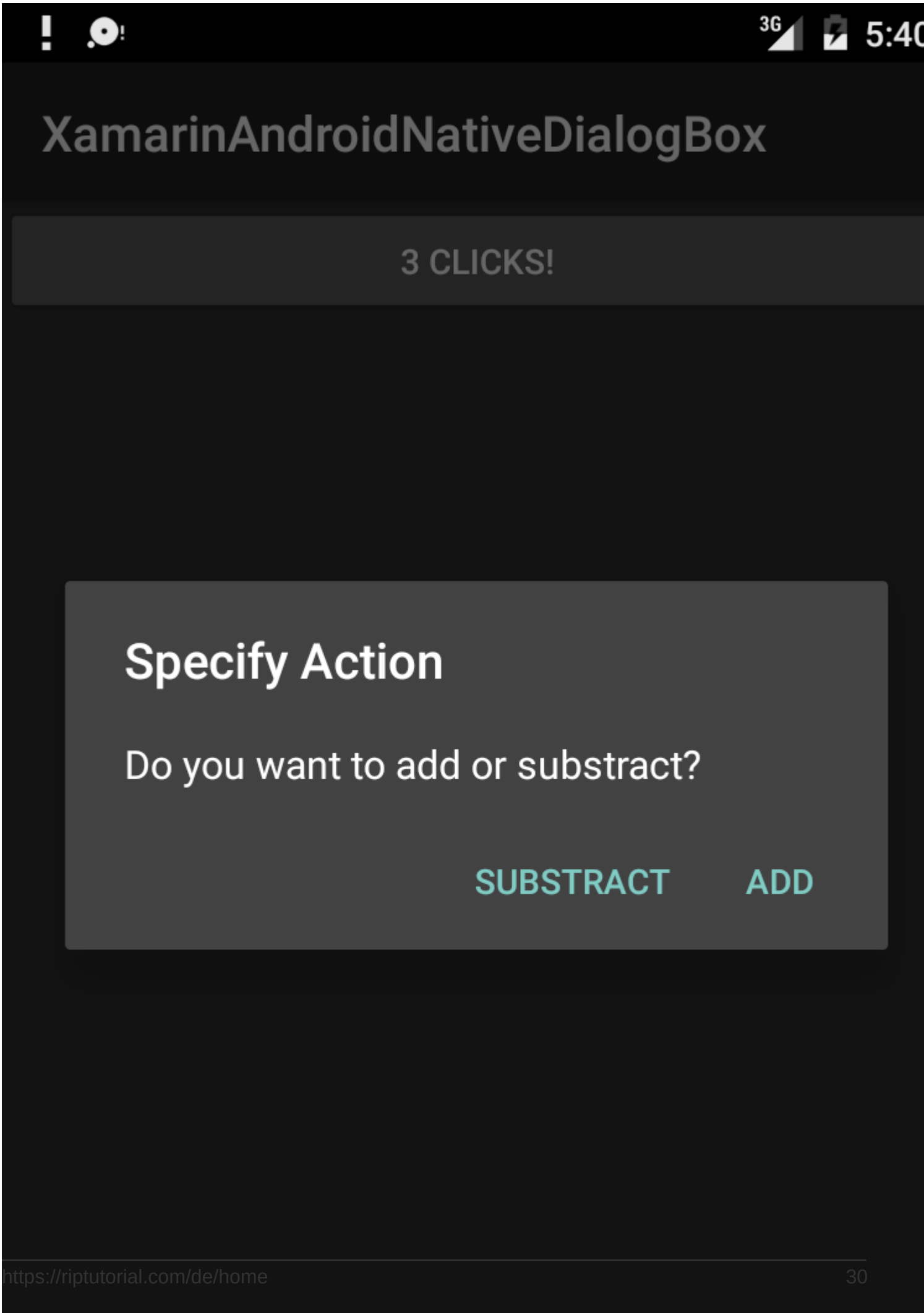
```
public class MainActivity : Activity
{
int count = 1;

protected override void OnCreate(Bundle bundle)
{
base.OnCreate(bundle);

// Set our view from the "main" layout resource
SetContentView(Resource.Layout.Main);

// Get our button from the layout resource,
// and attach an event to it
Button button = FindViewById<Button>(Resource.Id.MyButton);

button.Click += delegate { button.Text = string.Format("{0} clicks!", count++); };
}
}
```

Was wir jetzt tun, ist, anstatt dem Zähler beim Klicken eines Schalters einen Zähler hinzuzufügen, werden wir den Benutzer fragen, ob er einen in einem einfachen Alert-Dialog hinzufügen oder subtrahieren möchte

Und durch Klicken auf die Schaltfläche "Positiv" oder "Negativ" führen wir die Aktion aus.

```
button.Click += delegate {
AlertDialog.Builder alert = new AlertDialog.Builder(this);
alert.SetTitle("Specify Action");
alert.SetMessage("Do you want to add or substract?");

alert.SetPositiveButton("Add", (senderAlert, args) =>
{
count++;
button.Text = string.Format("{0} clicks!", count);
 });

 alert.SetNegativeButton("Substract", (senderAlert, args) =>
 {
 count--;
 button.Text = string.Format("{0} clicks!", count);
 });

 Dialog dialog = alert.Create();
     dialog.Show();
};
```

Bildschirmfoto:


XamarinAndroidNativeDialogBox
3 CLICKS!
Specify Action
Do you want to add or substract?
SUBSTRACT
ADD

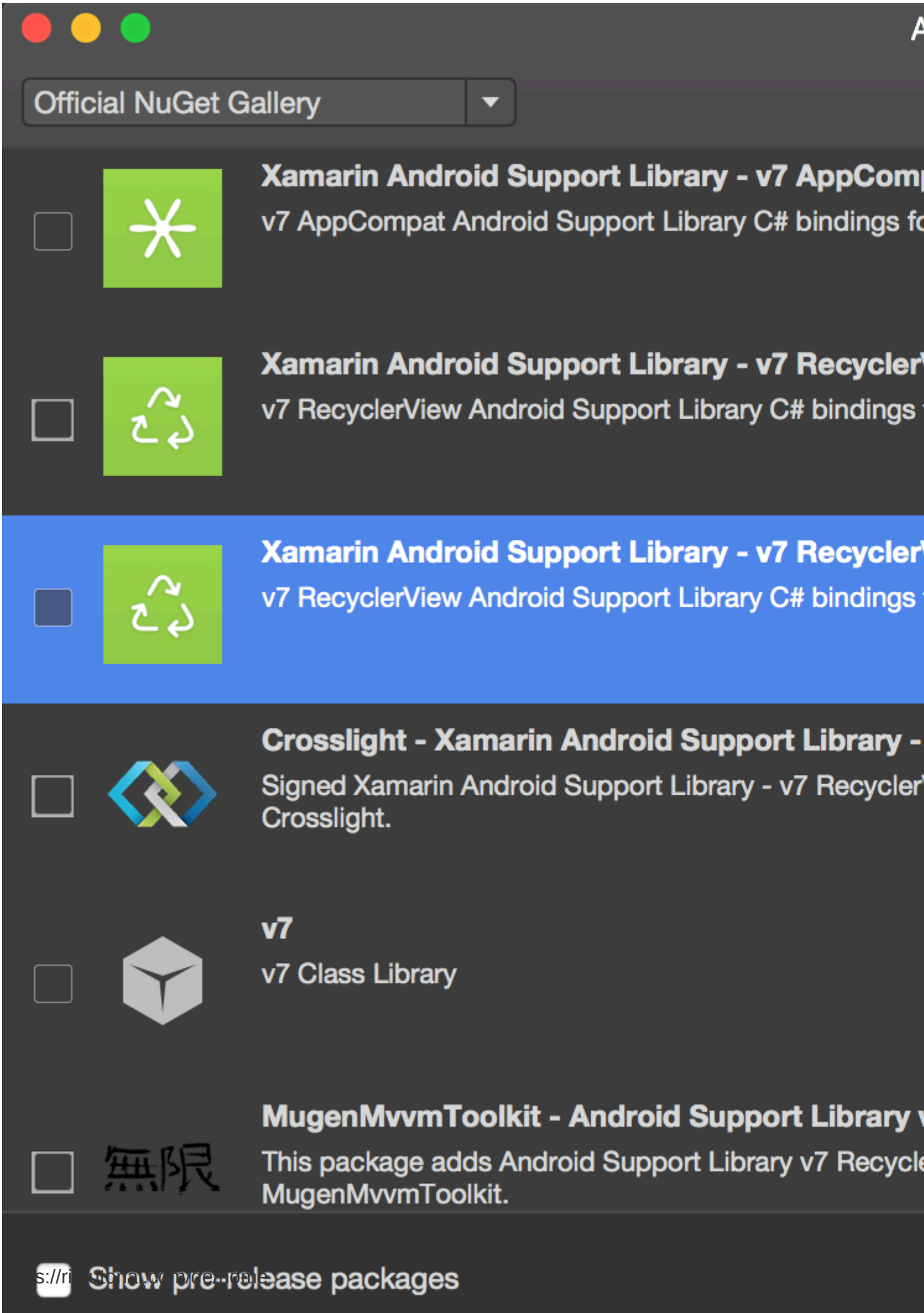
Kapitel 8: RecyclerView

Examples

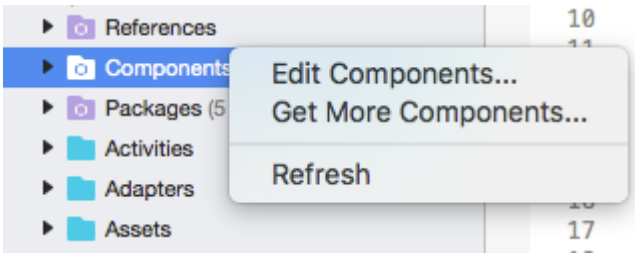
RecyclerView-Grundlagen

Dies ist ein Beispiel für die Verwendung der `Android Support Library V7 RecyclerView` . Unterstützungsbibliotheken werden im Allgemeinen empfohlen, da sie abwärtskompatible Versionen neuer Funktionen bereitstellen, nützliche Elemente der Benutzeroberfläche enthalten, die nicht im Framework enthalten sind, und eine Reihe von Dienstprogrammen bereitstellen, auf die Apps zurückgreifen können.

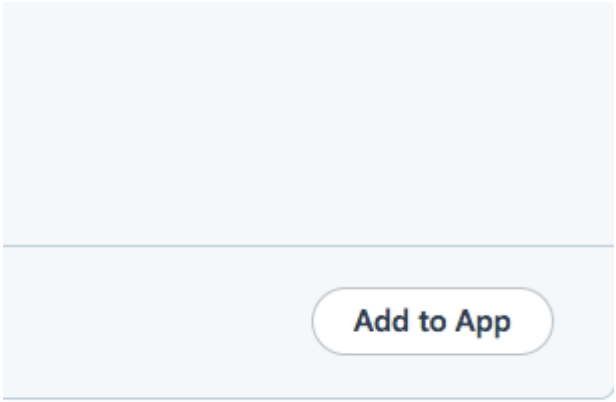
Um `RecyclerView` , installieren wir die erforderlichen Nuget-Pakete. Zuerst suchen wir nach `v7 recyclerview` . Scrollen Sie nach unten, bis die `Xamarin Android Support Library – v7 RecyclerView` . Wählen Sie es aus und klicken **Sie auf Paket hinzufügen** .


Official NuGet Gallery
Xamarin Android Support Library - v7 AppCom
v7 AppCompat Android Support Library C# bindings fo
Xamarin Android Support Library - v7 Recycler
v7 RecyclerView Android Support Library C# bindings
Xamarin Android Support Library - v7 Recycler
v7 RecyclerView Android Support Library C# bindings
Crosslight - Xamarin Android Support Library -
Signed Xamarin Android Support Library - v7 Recycler
Crosslight.
v7
v7 Class Library
MugenMvvmToolkit - Android Support Library
This package adds Android Support Library v7 Recycle
MugenMvvmToolkit.
Show pre-release packages

als Xamarin-Komponente verfügbar. Um die Komponente hinzuzufügen, klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf `Components` im Android-Projekt, und klicken Sie auf `Get More Components` .



Suchen Sie im angezeigten Fenster "Component Store" nach RecyclerView. `Android Support Library V7 RecyclerView` in der Suchliste `Android Support Library V7 RecyclerView` . Klicken Sie dann auf `Add to App` . Die Komponente wird dem Projekt hinzugefügt.



Der nächste Schritt ist das Hinzufügen der RecyclerView zu einer Seite. In der `axml` Datei (Layoutdatei) können Sie wie `axml RecyclerView` hinzufügen.

```
<android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

Für RecyclerView müssen mindestens zwei `ViewHolder` für die grundlegende Standardimplementierung eingerichtet werden: `Adapter` und `ViewHolder` . `Adapter` bläst Elementlayouts auf und bindet Daten an Ansichten, die in einer RecyclerView angezeigt werden. ViewHolder sucht und speichert Ansichtsreferenzen. Der Ansichtshalter hilft auch beim Erkennen von Elementklicks.

Hier ist ein grundlegendes Beispiel für die Adapterklasse

```
public class MyAdapter : RecyclerView.Adapter
{
    string [] items;

    public MyAdapter (string [] data)
    {
```

```
        items = data;
    }

    // Create new views (invoked by the layout manager)
    public override RecyclerView.ViewHolder OnCreateViewHolder (ViewGroup parent, int
viewType)
    {
        // set the view's size, margins, paddings and layout parameters
        var tv = new TextView (parent.Context);
        tv.SetWidth (200);
        tv.Text = "";

        var vh = new MyViewHolder (tv);
        return vh;
    }

    // Replace the contents of a view (invoked by the layout manager)
    public override void OnBindViewHolder (RecyclerView.ViewHolder viewHolder, int position)
    {
        var item = items [position];

        // Replace the contents of the view with that element
        var holder = viewHolder as MyViewHolder;
        holder.TextView.Text = items[position];
    }

    public override int ItemCount {
        get {
            return items.Length;
        }
    }
}
```

In der `OnCreateViewHolder` Methode `OnCreateViewHolder` wir zuerst eine View auf und erstellen eine Instanz der ViewHolder-Klasse. Diese Instanz muss zurückgegeben werden. Diese Methode wird vom Adapter aufgerufen, wenn eine neue Instanz von ViewHolder erforderlich ist. Diese Methode wird nicht für jede einzelne Zelle aufgerufen. Wenn RecyclerView genug Zellen zum Füllen der Ansicht hat, werden die alten Zellen, die aus der Ansicht gescrollt werden, für weitere Zellen wiederverwendet.

Der `OnBindViewHolder` Callback wird vom Adapter aufgerufen, um die Daten an der angegebenen Position anzuzeigen. Diese Methode sollte den Inhalt der itemView aktualisieren, um das Element an der angegebenen Position wiederzugeben.

Da die Zelle nur eine einzige `TextView` , können wir einen einfachen ViewHolder wie folgt verwenden.

```
public class MyViewHolder : RecyclerView.ViewHolder
{
    public TextView TextView { get; set; }


    public MyViewHolder (TextView v) : base (v)
    {
        TextView = v;
    }
}
```

Der nächste Schritt besteht darin, Dinge in `Activity` .

```
RecyclerView mRecyclerView;
MyAdapter mAdapter;
protected override void OnCreate (Bundle bundle)
{
    base.OnCreate (bundle);
    SetContentView (Resource.Layout.Main);
    mRecyclerView = FindViewById<RecyclerView> (Resource.Id.recyclerView);

    // Plug in the linear layout manager:
    var layoutManager = new LinearLayoutManager (this) { Orientation =
LinearLayoutManager.Vertical };
    mRecyclerView.SetLayoutManager (layoutManager);
    mRecyclerView.HasFixedSize = true;

    var recyclerViewData = GetData();
    // Plug in my adapter:
    mAdapter = new MyAdapter (recyclerViewData);
    mRecyclerView.SetAdapter (mAdapter);
}

string[] GetData()
{
     string[] data;
     .
     .
     .
     return data;
}
```

Die LayoutManager-Klasse ist für das Messen und Positionieren von Elementansichten in einer RecyclerView sowie für die Festlegung der Richtlinie dafür verantwortlich, wann Elementansichten wieder verwendet werden sollen, die für den Benutzer nicht mehr sichtbar sind. Vor der `RecyclerView` mussten wir `ListView` , um Zellen in einer vertikal scrollenden Liste `GridView` , und `GridView` , um Elemente in einem zweidimensionalen, scrollbaren Raster anzuzeigen. Beides können wir jetzt mit RecyclerView erreichen, indem wir einen anderen LayoutManger einstellen. `LinearLayoutManager` ordnet Zellen wie in einem ListView an, und `GridLayoutManager` ordnet Zellen im Grid- `GridLayoutManager` .

RecyclerView mit Click-Ereignissen

Dieses Beispiel zeigt, wie Sie Click EventHandlers in einer Xamarin.Android RecyclerView festlegen.

In Android Java können Sie einen Listener für einen Click einrichten, indem Sie einen onClickListener für die Ansicht verwenden, auf die Sie klicken.

```
ImageView picture = findViewById(R.id.item_picture);
picture.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // do stuff
    }
});
```

In Xamarin.Android können Sie jedoch einen Listener für ein Click-Ereignis einrichten, **indem Sie** einen EventHandler auf folgende Weise **hinzufügen** :

1

```
ImageView picture = FindViewById<ImageView>(Resource.Id.item_picture);
picture.Click += delegate {
    // do stuff
};
```

2

```
ImageView picture = FindViewById<ImageView>(Resource.Id.item_picture);
picture.Click += async delegate {
    // await DoAsyncMethod();
    // do async stuff
};
```

3.

```
ImageView picture = FindViewById<ImageView>(Resource.Id.item_picture);
picture.Click += Picture_Click;
... // rest of your method

private void Picture_Click(object sender, EventArgs e)
{
    // do stuff
}
```

Beachten Sie, dass **der EventHandler hinzugefügt und nicht festgelegt wird.** Wenn der Click-EventHandler aus einer GridView / ListView-Adapter oder einer OnBindViewHolder-Methode aus einem RecyclerView.Adapter in eine GetView-Methode eingefügt wird, wird jedes Mal, wenn die Elementansicht erstellt wird, ein neuer EventHandler hinzugefügt. Nach mehrmaligem Scrollen werden mehrere EventHandler hinzugefügt. Wenn Sie auf die Ansicht klicken, werden alle ausgelöst.

Um dieses Problem zu vermeiden, müssen die EventHandlers abbestellt und anschließend in den Methoden GetView oder OnBindViewHolder abonniert werden. Sie müssen außerdem die Nummer **3. verwenden** , um den EventHandler festzulegen. Andernfalls können die EventHandlers nicht abbestellt werden.

Ein Beispiel eines RecyclerView.Adapter mit Click-Ereignissen wird unten gezeigt:

```
public class ViewHolderPerson : Android.Support.V7.Widget.RecyclerView.ViewHolder
{
    public View Item { get; private set; }
    public ImageView Picture { get; private set; }
    public TextView Name { get; private set; }

    public ViewHolderPerson(View itemView) : base(itemView)
    {
        this.Item = itemView;
        this.Picture = itemView.FindViewById<ImageView>(Resource.Id.Item_Person_Picture);
```

```
        this.Name = itemView.FindViewById<TextView>(Resource.Id.Item_Person_Name);
    }
}

public class AdapterPersons : Android.Support.V7.Widget.RecyclerView.Adapter
{
    private Context context;
    private Android.Support.V7.Widget.RecyclerView recyclerView;
    private List<Person> persons;

    public AdapterPersons(Context context, Android.Support.V7.Widget.RecyclerView
recyclerView, List<Person> persons)
    {
        this.context = context;
        this.recyclerView = recyclerView;
        this.persons = persons;
    }

    public override int ItemCount => persons.Count;

    public override void OnBindViewHolder(RecyclerView.ViewHolder holder, int position)
    {
        Person person = this.persons[position];
        ((ViewHolderPerson)holder).Name.Text = person.Name;
        ((ViewHolderPerson)holder).Picture.SetImageBitmap(person.Picture);

        // Unsubscribe and subscribe the method, to avoid setting multiple times.
        ((ViewHolderPerson)holder).Item.Click -= Person_Click;
        ((ViewHolderPerson)holder).Item.Click += Person_Click;
    }

    private void Person_Click(object sender, EventArgs e)
    {
        int position = this.recyclerView.GetChildAdapterPosition((View)sender);
        Person personClicked = this.persons[position];
        if(personClicked.Gender == Gender.Female)
        {
            Toast.MakeText(this.context, "The person clicked is a female!",
ToastLength.Long).Show();
        }
        else if(personClicked.Gender == Gender.Male)
        {
            Toast.MakeText(this.context, "The person clicked is a male!",
ToastLength.Long).Show();
        }
    }

    public override RecyclerView.ViewHolder OnCreateViewHolder(ViewGroup parent, int viewType)
    {
        View itemView =
LayoutInflater.From(parent.Context).Inflate(Resource.Layout.item_person, parent, false);
        return new ViewHolderPerson(itemView);
    }

}
```

RecyclerView online lesen: https://riptutorial.com/de/xamarin-android/topic/3452/recyclerview

Kapitel 9: So korrigieren Sie die Ausrichtung eines mit einem Android-Gerät aufgenommenen Bildes

Bemerkungen

1. Dieses App-Beispiel ist auf meinem GitHub verfügbar:

https://github.com/Daniel-Krzyczkowski/XamarinAndroid/tree/master/AndroidPictureOrientation/PictureOrientationApp

2. Die Dokumentation zu Xamarin Mobile-Komponenten ist unten verfügbar:

https://components.xamarin.com/view/xamarin.mobile

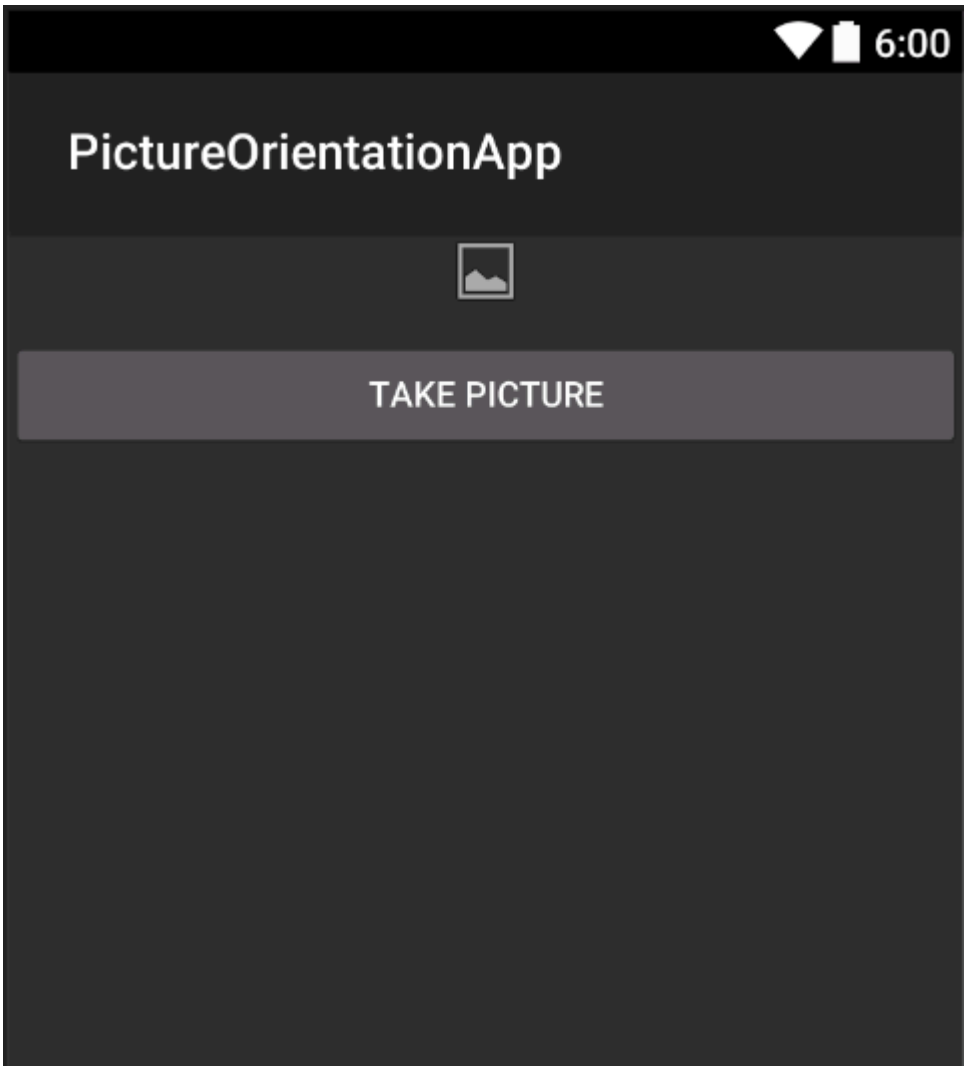
Examples

So korrigieren Sie die Ausrichtung eines mit einem Android-Gerät aufgenommenen Bildes

Dieses Beispiel zeigt, wie ein Bild aufgenommen und auf dem Android-Gerät korrekt angezeigt wird.

Zunächst müssen wir eine Beispielanwendung mit einer Schaltfläche und einer Bildansicht erstellen. Sobald der Benutzer auf die Schaltfläche klickt, wird die Kamera gestartet, und nachdem der Benutzer das Bild ausgewählt hat, wird diese mit der richtigen Ausrichtung auf dem Bildschirm angezeigt.

1. Fügen Sie die Schaltfläche "TakePictureButton" und die Bildansicht "TakenPictureImageView" hinzu:

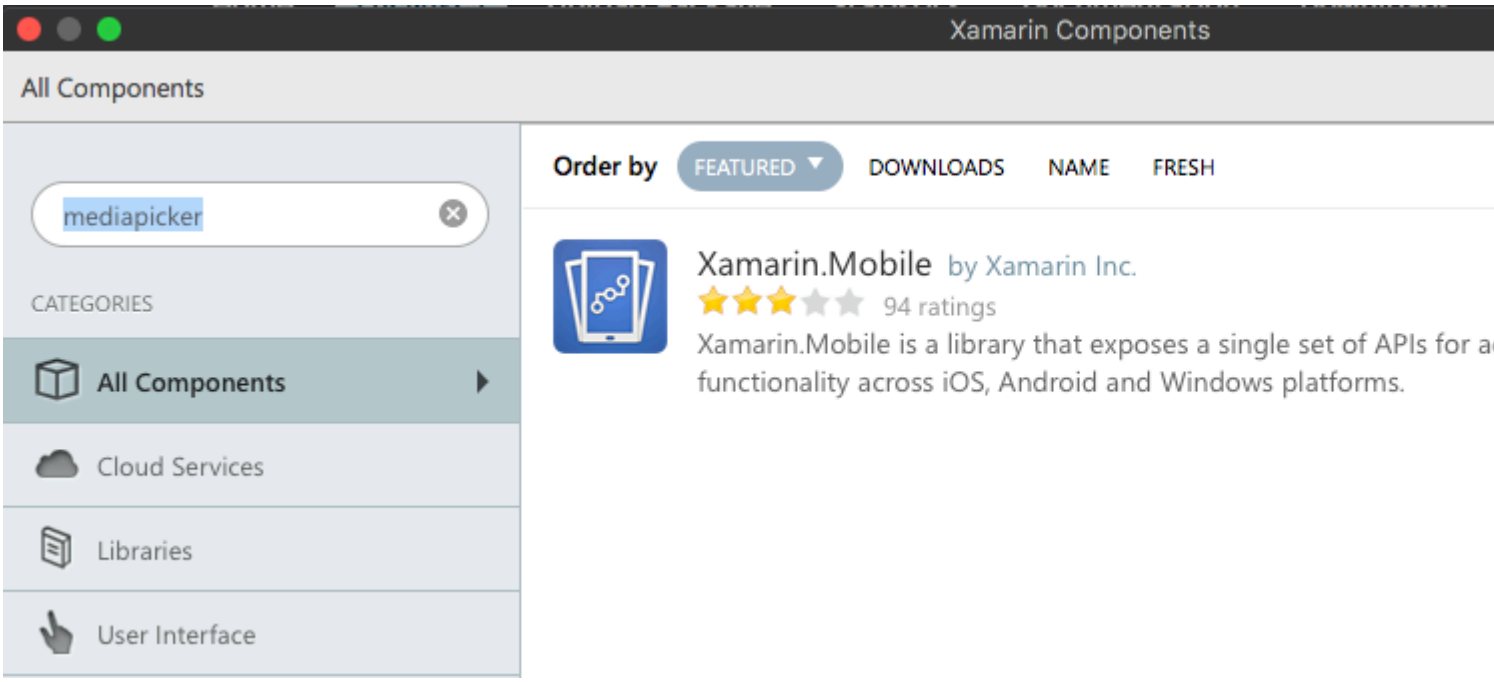


2. Öffnen Sie jetzt den Aktivitätscode hinter:

Hier erhalten Sie zunächst einen Hinweis auf Ihre Steuerelemente:

```
ImageView _takenPictureImageView;
Button _takePictureButton;

    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.Main);

        _takenPictureImageView = FindViewById<ImageView>(Resource.Id.TakenPictureImageView);
        _takePictureButton = FindViewById<Button>(Resource.Id.TakePictureButton);

        _takePictureButton.Click += delegate
        {
            takePicture();
        };
    }
```

3. In unserer Anwendung verwenden wir die im Components Store verfügbare Xamarin Mobile-Komponente:



4. Sobald Sie es dem Projekt hinzufügen, können wir fortfahren. Fügen Sie unten den Code hinzu, der für das Starten der Kamera verantwortlich ist. Diese Methode sollte beim Klicken auf die Schaltfläche aufgerufen werden, wie Sie im obigen Code sehen können:

```
void takePicture()
 {
     var picker = new MediaPicker(this);
     DateTime now = DateTime.Now;
     var intent = picker.GetTakePhotoUI(new StoreCameraMediaOptions
     {
         Name = "picture_" + now.Day + "_" + now.Month + "_" + now.Year + ".jpg",
         Directory = null
     });
     StartActivityForResult(intent, 1);
 }
```

5. Sobald der Benutzer ein Foto aufgenommen hat, sollte er in der richtigen Ausrichtung angezeigt werden. Verwenden Sie dazu die Methode unten. Es ist dafür verantwortlich, Exif-Informationen aus dem aufgenommenen Bild abzurufen (einschließlich der Ausrichtung während der Aufnahme), und dann eine Bitmap mit der richtigen Ausrichtung zu erstellen:

```
Bitmap loadAndResizeBitmap(string filePath)
 {
     BitmapFactory.Options options = new BitmapFactory.Options { InJustDecodeBounds =
true };
     BitmapFactory.DecodeFile(filePath, options);

     int REQUIRED_SIZE = 100;
     int width_tmp = options.OutWidth, height_tmp = options.OutHeight;
     int scale = 4;
     while (true)
     {
         if (width_tmp / 2 < REQUIRED_SIZE || height_tmp / 2 < REQUIRED_SIZE)
             break;
         width_tmp /= 2;
```

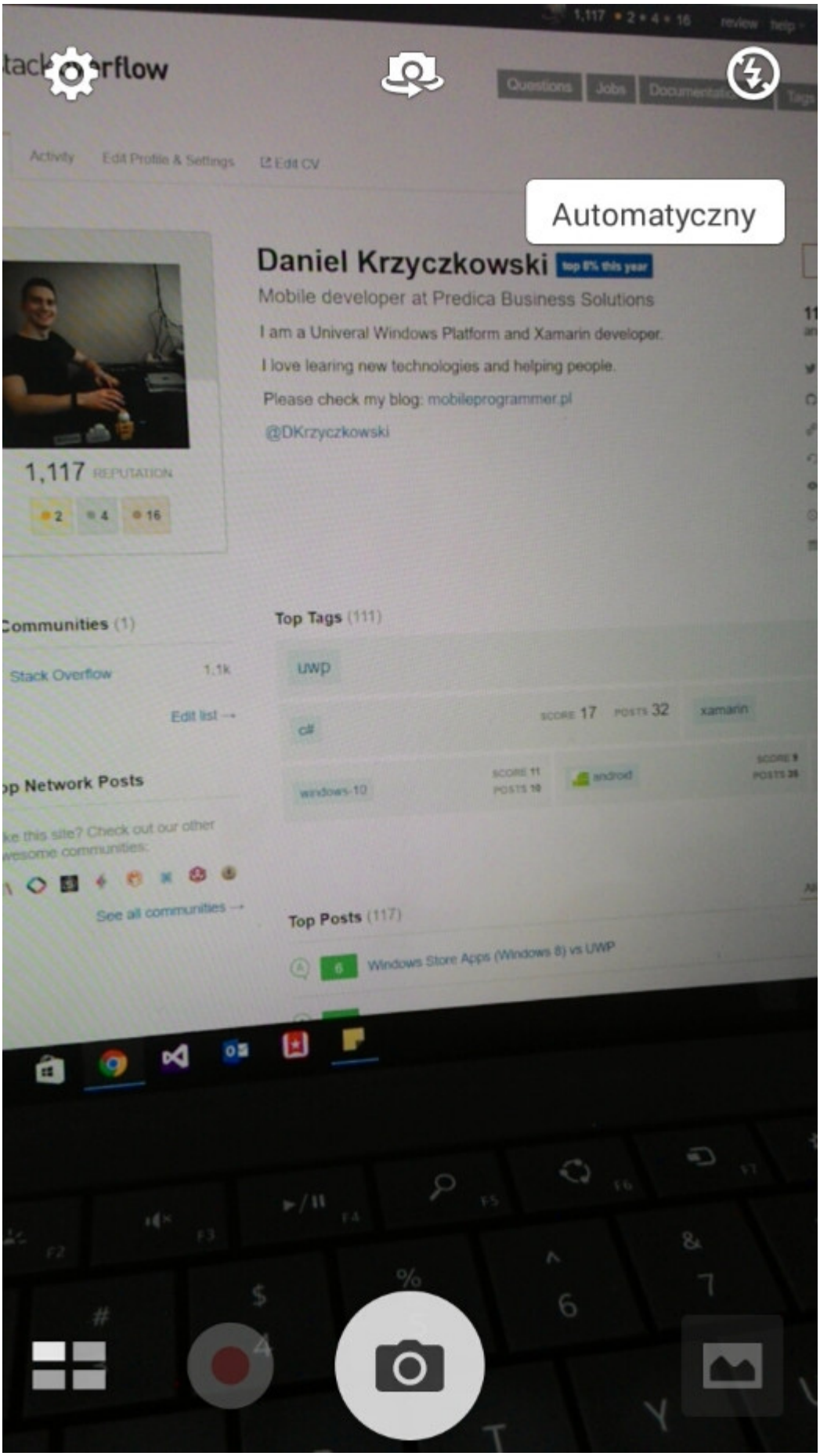
```
        height_tmp /= 2;
        scale++;
    }

    options.InSampleSize = scale;
    options.InJustDecodeBounds = false;
    Bitmap resizedBitmap = BitmapFactory.DecodeFile(filePath, options);

    ExifInterface exif = null;
    try
    {
        exif = new ExifInterface(filePath);
        string orientation = exif.GetAttribute(ExifInterface.TagOrientation);

        Matrix matrix = new Matrix();
        switch (orientation)
        {
            case "1": // landscape
                break;
            case "3":
                matrix.PreRotate(180);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
            case "4":
                matrix.PreRotate(180);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
            case "5":
                matrix.PreRotate(90);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
            case "6": // portrait
                matrix.PreRotate(90);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
            case "7":
                matrix.PreRotate(-90);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
            case "8":
                matrix.PreRotate(-90);
                resizedBitmap = Bitmap.CreateBitmap(resizedBitmap, 0, 0,
resizedBitmap.Width, resizedBitmap.Height, matrix, false);
                matrix.Dispose();
                matrix = null;
                break;
```

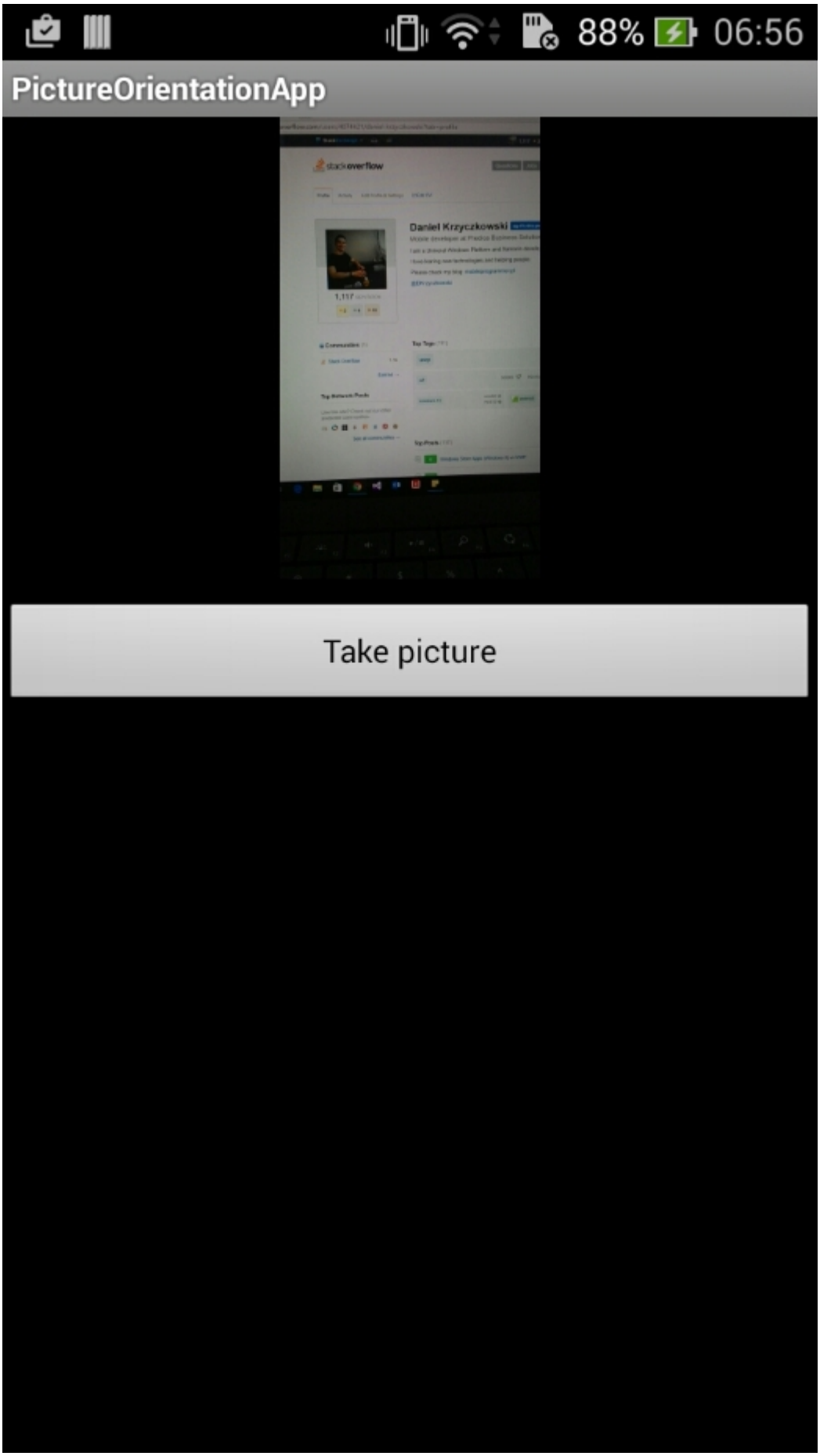
```
        }

        return resizedBitmap;
    }

    catch (IOException ex)
    {
        Console.WriteLine("An exception was thrown when reading exif from media
file...:" + ex.Message);
        return null;
    }
 }
```

6. Die obige Methode sollte in der OnActivityResult-Methode aufgerufen werden, nachdem der Benutzer das Bild aufgenommen hat:

```
protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)
 {
     base.OnActivityResult(requestCode, resultCode, data);

     if (requestCode == 1)
     {
         if (resultCode == Result.Ok)
         {
             data.GetMediaFileExtraAsync(this).ContinueWith(t =>
             {
                 using (Bitmap bmp = loadAndResizeBitmap(t.Result.Path))
                 {
                     if (bmp != null)
                     _takenPictureImageView.SetImageBitmap(bmp);
                 }

             }, TaskScheduler.FromCurrentSynchronizationContext());
         }
     }
 }
```

7. Starte die Anwendung. Machen Sie ein Foto und sehen Sie das Ergebnis:


Automatyczny
Daniel Krzyczkowski
Mobile developer at Predica Business Solutions
I am a Univeral Windows Platform and Xamarin developer.
I love learing new technologies and helping people.
Please check my blog: mobileprogrammer.pl
@DKrzyczkowski
1,117 REPUTATION
Communities (1)
Stack Overflow
Edit list
Top Tags (111)
uwp
c#
xamarin
android
Top Posts (117)
Windows Store Apps (Windows 8) vs UWP



Das ist es. Jetzt werden alle Fotos in der richtigen Ausrichtung angezeigt.

So korrigieren Sie die Ausrichtung eines mit einem Android-Gerät aufgenommenen Bildes online lesen: https://riptutorial.com/de/xamarin-android/topic/6683/so-korrigieren-sie-die-ausrichtung-eines-mit-einem-android-gerat-aufgenommenen-bildes

Kapitel 10: Toast

Examples

Grundlegende Toast-Nachricht

Instanziieren Sie zunächst ein Toast-Objekt mit einer der `MakeText()` Methoden. Diese Methode nimmt drei Parameter: die Anwendung `Context` , die Textnachricht und die Dauer für den Toast. Es gibt ein ordnungsgemäß initialisiertes Toast-Objekt zurück. Sie können die Toastbenachrichtigung mit `Show()` anzeigen, wie im folgenden Beispiel gezeigt:

```
Context context = Application.Context;
string text = "Hello toast!";
ToastLength duration = ToastLength.Short;

var toast = Toast.MakeText(context, text, duration);
toast.Show();
```

Dieses Beispiel zeigt alles, was Sie für die meisten Toastbenachrichtigungen benötigen. Sie sollten selten etwas anderes brauchen. Möglicherweise möchten Sie jedoch den Toast anders positionieren oder sogar Ihr eigenes Layout anstelle einer einfachen Textnachricht verwenden. In den folgenden Abschnitten wird beschrieben, wie Sie diese Funktionen ausführen können.

Sie können Ihre Methoden auch verketten, als Einzeiler aufrufen und vermeiden, dass Sie das Toast-Objekt wie folgt halten:

```
Toast.MakeText(Application.Context, "Hello toast!", ToastLength.Short).Show();
```

Weitere Informationen finden Sie in der umfassenderen Android-Dokumentation zum Thema.

Farbige Toast-Meldungen

Manchmal möchten wir unseren Benutzern zusätzliche Informationen mit Farben geben (z. B. bedeutet Rot, dass ein Fehler aufgetreten ist). Wir können die Hintergrundfarbe der Toast-Nachricht ändern, indem Sie einen Farbfilter auf die Ansicht setzen, die unser Toast uns gibt (hier verwende ich einen ColorMatrixColorFilter):

```
Toast t = Toast.MakeText(context, message, duration);
Color c = */your color/*;
ColorMatrixColorFilter CM = new ColorMatrixColorFilter(new float[]
    {
        0,0,0,0,c.R,
        0,0,0,0,c.G,
        0,0,0,0,c.B,
        0,0,0,1,0
    });
t.View.Background.SetColorFilter(CM);
t.Show();
```

Und wir können auch die Textfarbe ändern, wenn der Hintergrund hell oder dunkel ist:

```
if ((((float)(c.R) + (float)(c.G) + (float)(c.B)) / 3) >= 128)
    t.View.FindViewById<TextView>(Android.Resource.Id.Message).SetTextColor(Color.Black);
else
//text color is white by default
```

Toastposition ändern

Wir können unseren Toast mit der SetGravity-Methode ändern. Für diese Methode sind drei Parameter erforderlich: erstens die Schwere des Toasts auf dem Bildschirm und zwei weitere, die Toastversatz von der Startposition (die durch den ersten Parameter festgelegt wird) festlegen:

```
//Toast at bottom left corner of screen
Toast t = Toast.MakeText(context, message, duration);
t.SetGravity(GravityFlags.Bottom | GravityFlags.Left, 0, 0);
t.Show();

//Toast at a custom position on screen
Toast t = Toast.MakeText(context, message, duration);
t.SetGravity(GravityFlags.Top | GravityFlags.Left, x, y);
t.Show();
```

Toast online lesen: https://riptutorial.com/de/xamarin-android/topic/3550/toast

Kapitel 11: Veröffentlichen Ihres Xamarin.Android APK

Einführung

In diesem Thema erfahren Sie, wie Sie Ihre Xamarin.Android-App für den Freigabemodus vorbereiten und wie Sie sie optimieren können.

Examples

Vorbereiten der APK im Visual Studio

Sie haben Ihre App fertiggestellt, im Debug-Modus getestet und funktionieren einwandfrei. Nun möchten Sie es für die Veröffentlichung im Google Play Store vorbereiten.

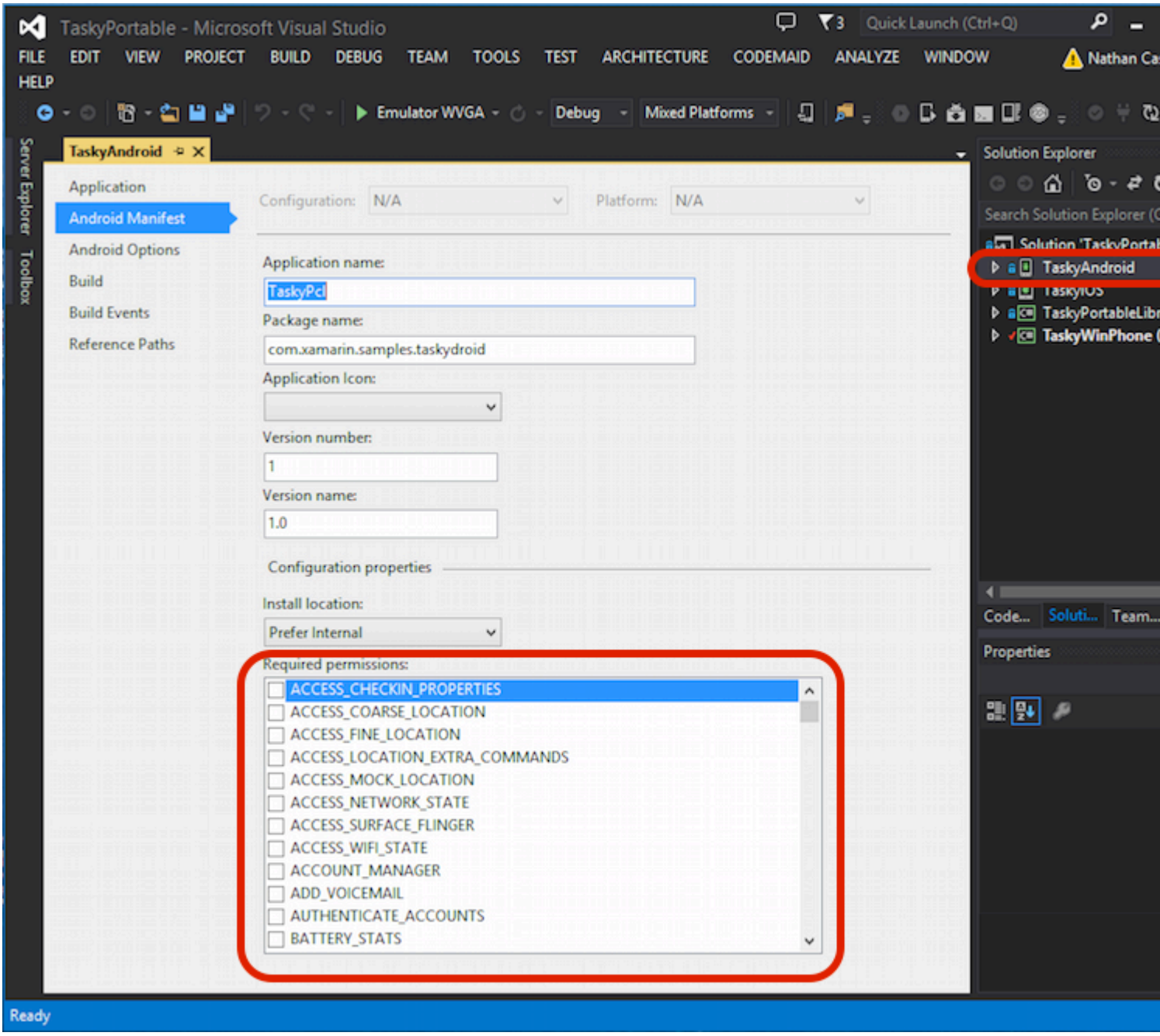
Die Xamarin-Dokumentation bietet hier gute Informationen:

https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_applicati

---

Android Manifest

Klicken Sie mit der rechten Maustaste in Visual Studio auf Ihr Xamarin.Android-Projekt im Projektmappen-Explorer, und wählen Sie Eigenschaften aus. Wechseln Sie dann zur Registerkarte Android Manifest, um diesen Bildschirm anzuzeigen:



Im Gegensatz zu Android Studio oder Eclipse müssen Sie die Datei AndroidManifest.xml nicht durch Schreiben festlegen. Xamarin und Visual Studio erledigen das für Sie. Aktivitäten, BroadcastReceiver und Services werden in Android Manifest eingefügt, indem bestimmte Attribute in ihren Klassen deklariert werden .

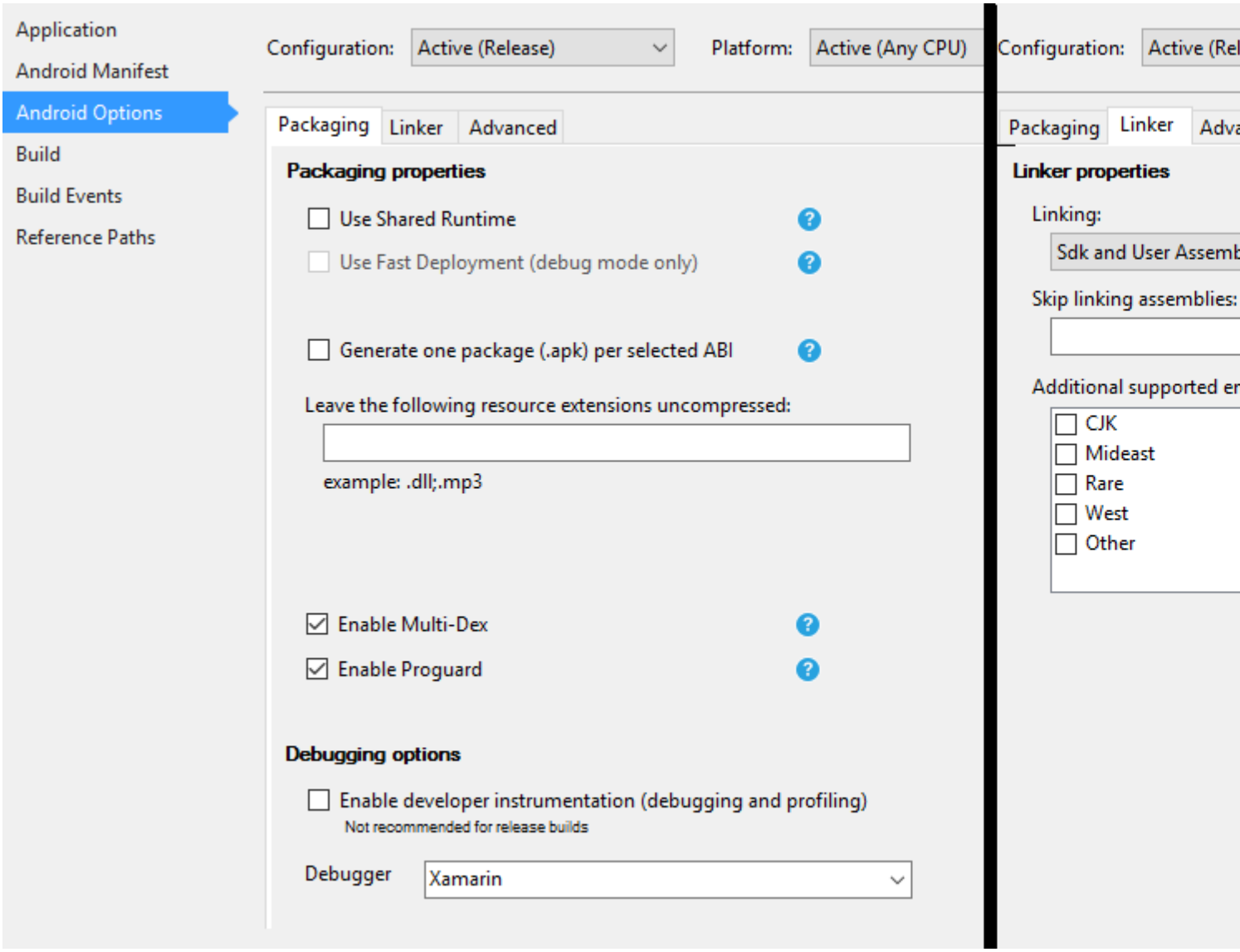
In diesem Bildschirm stehen folgende Optionen zur Verfügung:

- **Anwendungsname** : Dies ist der **Anwendungsname** , der für den Benutzer sichtbar ist.
- **Paketname:** Dies ist der Name des Pakets. Es muss eindeutig sein, dh es darf nicht den gleichen Paketnamen anderer Apps im Google Play Store verwenden.
- **Anwendungssymbol** : Dies ist das Symbol, das für den Benutzer sichtbar ist. Dies entspricht dem in Android Studio- oder Eclipse-Projekten verwendeten @ drawable / ic_launcher.
- **Versionsnummer** : Die Versionsnummer wird von Google Play zur Versionskontrolle verwendet. Wenn Sie ein APK für eine aktualisierte Version Ihrer App veröffentlichen

möchten, müssen Sie diese Nummer für jedes neue Upgrade um 1 erhöhen.
- **Versionsname** : Dies ist der Versionsname, der dem Benutzer angezeigt wird.
- **Installationsort:** Dies bestimmt , wo Ihre APK, im Gerätespeicher oder auf der SD - Karte installiert wird.
- **Erforderliche Berechtigungen** : Hier legen Sie fest, welche Berechtigungen für Ihre App erforderlich sind.

Android-Optionen

Im folgenden Bildschirm können Sie die Compiler-Optionen konfigurieren. Wenn Sie hier die richtigen Optionen verwenden, können Sie Ihre APK-Größe erheblich reduzieren und Fehler vermeiden.



- **Konfiguration** : **Aktiv (Freigabe)** .
- **Plattform** : **Aktiv (beliebige CPU)** . Diese sind erforderlich, um Ihr APK für den Google Play Store zu erstellen. Wenn die Konfiguration auf "Debuggen" eingestellt ist, wird sie von Google Play nicht akzeptiert.
- **Shared Runtime verwenden** : **false** . Wenn Sie den Wert auf true setzen, verwendet der APK Mono Runtime zur Ausführung. Die Mono Runtime wird beim Debuggen über USB

automatisch installiert, nicht jedoch in der Release-APK. Wenn Mono Runtime nicht auf dem Gerät installiert ist und diese Option in Release APK auf true gesetzt ist, stürzt die App ab.

- **Generieren Sie ein Paket (.apk) pro ausgewähltem ABI** : **false** . Erstellen Sie aus Gründen der Kompatibilität Ihr APK für so viele Plattformen wie möglich.
- **Aktivieren Sie Multi-Dex** : **true** , aber Sie können es auf false setzen, wenn Ihre App nicht sehr komplex ist (dh weniger als 65536 Methoden enthält, siehe hier).
- **Proguard aktivieren** : **true** . Dadurch wird das Proguard-Tool aktiviert, das Java-Code in Ihrer App verschleiert. Beachten Sie, dass es nicht für .NET-Code gilt. Wenn Sie .NET-Code verschleiern möchten, müssen Sie Dotfuscator verwenden . Weitere Informationen zu Proguard for Xamarin.Android finden Sie hier .
- **Aktivieren Sie die Entwicklerinstrumentierung (Debugging und Profiling)** : **false** für Release APK.
- **Verknüpfen** : **SDK und Benutzerbaugruppen** . Dadurch entfernt der Xamarin Linker alle nicht verwendeten Klassen aus dem SDK und Ihren Code, wodurch die APK-Größe reduziert wird.

Wichtig

Xamarin.Linker entfernt möglicherweise Klassen, die scheinbar von Ihrem Code nicht verwendet werden, insbesondere wenn sie sich im Kern des Projekts (PCL-Bibliothek) befinden. Um dies zu vermeiden, können Sie entweder die Verknüpfung auf "Nur Sdk-Assemblies" setzen oder das Attribut "Preserve" in Ihren Klassen verwenden. Beispiel:

PreserveAttribute.cs

```
namespace My_App_Core.Models
{
    public sealed class PreserveAttribute : System.Attribute
    {
        public bool AllMembers;
        public bool Conditional;
    }
}
```

In einer Klasse:

```
using System;

namespace My_App_Core.Models
{
    [Preserve(AllMembers = true)]
    public class ServiceException : Exception
    {
        public int errorCode;

        [Preserve(AllMembers = true)]
        public ServiceException() { }

        [Preserve(AllMembers = true)]
        public ServiceException(int errorCode)
        {
            this.errorCode = errorCode;
```

```
        }
    }
}
```

- **Unterstützte Architekturen** : **Wählen Sie alle** aus Kompatibilitätsgründen aus.

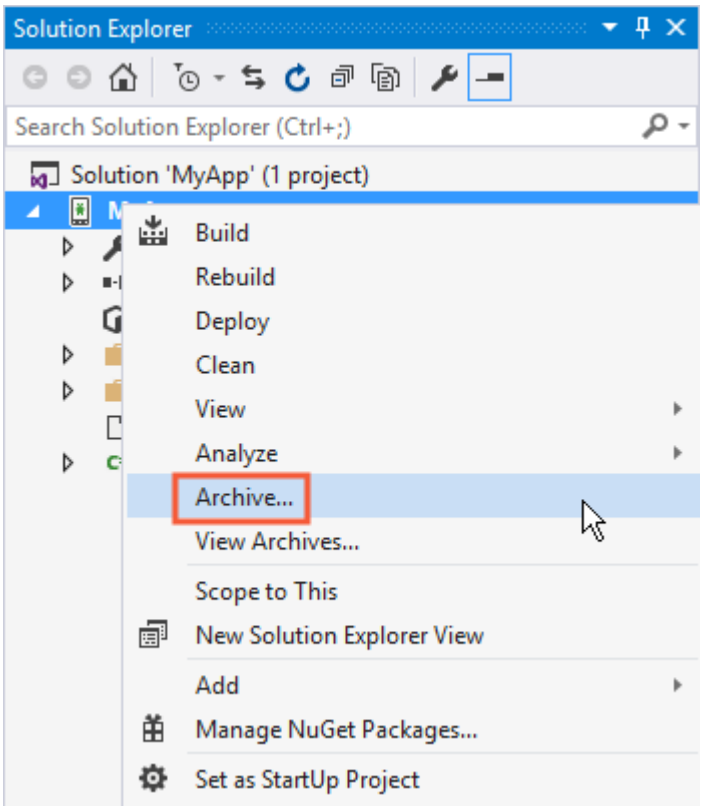
Erstellen Sie nach dem Konfigurieren alles neu, um sicherzustellen, dass es erfolgreich erstellt wird.

Erstellen der APK für den Freigabemodus

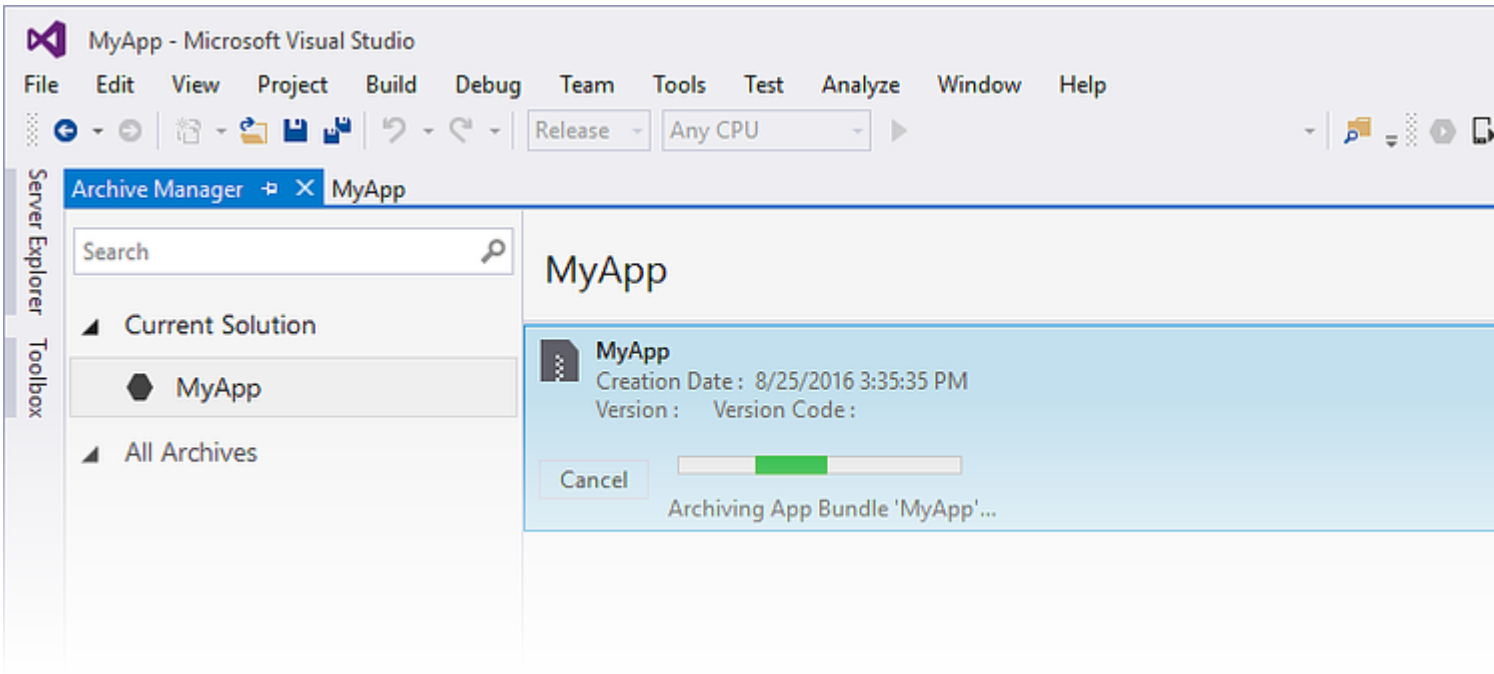
Sie haben die Konfiguration Ihres Android-Projekts für Release abgeschlossen. Das folgende Tutorial zeigt, wie Sie den APK in Visual Studio generieren. Ein vollständiges Tutorial aus der Xamarin-Dokumentation finden Sie hier:

https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_applicati
_signing_the_android_application_package/

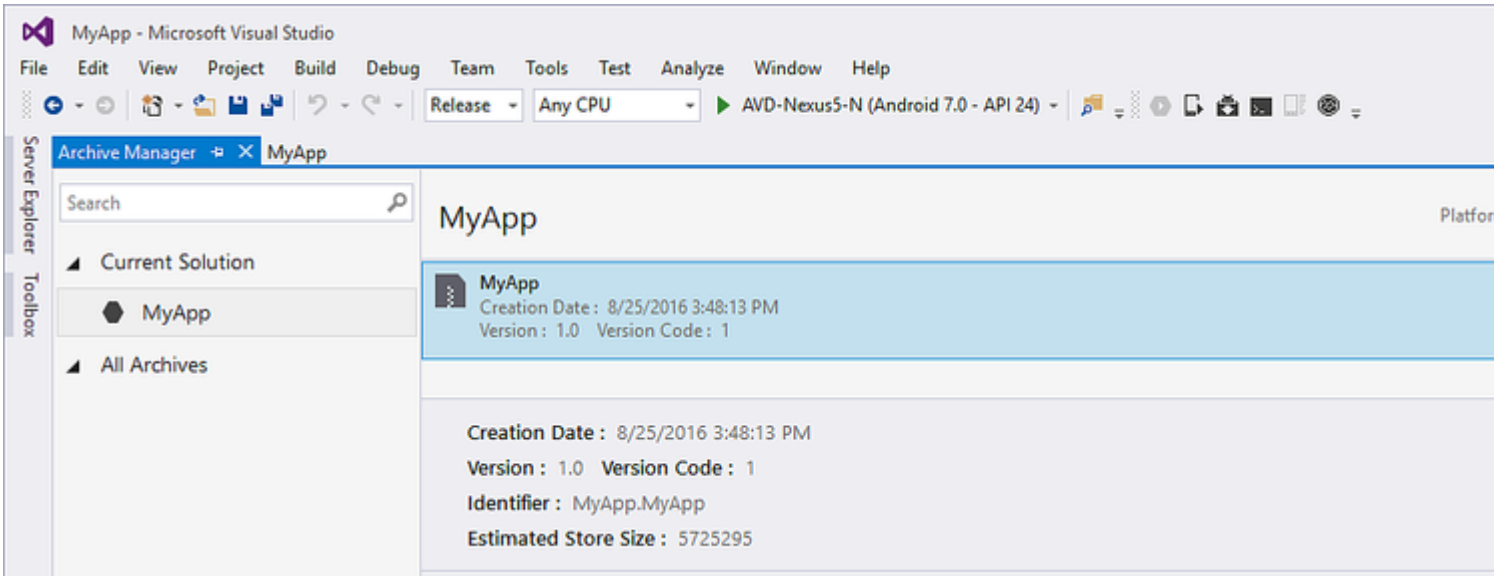
Um die APK-Datei zu erstellen, klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf das Projekt Xamarin.Android und wählen Sie Archiv ...



Dadurch wird der Archivmanager geöffnet, das Projekt wird archiviert und die Erstellung der APK-Datei vorbereitet.

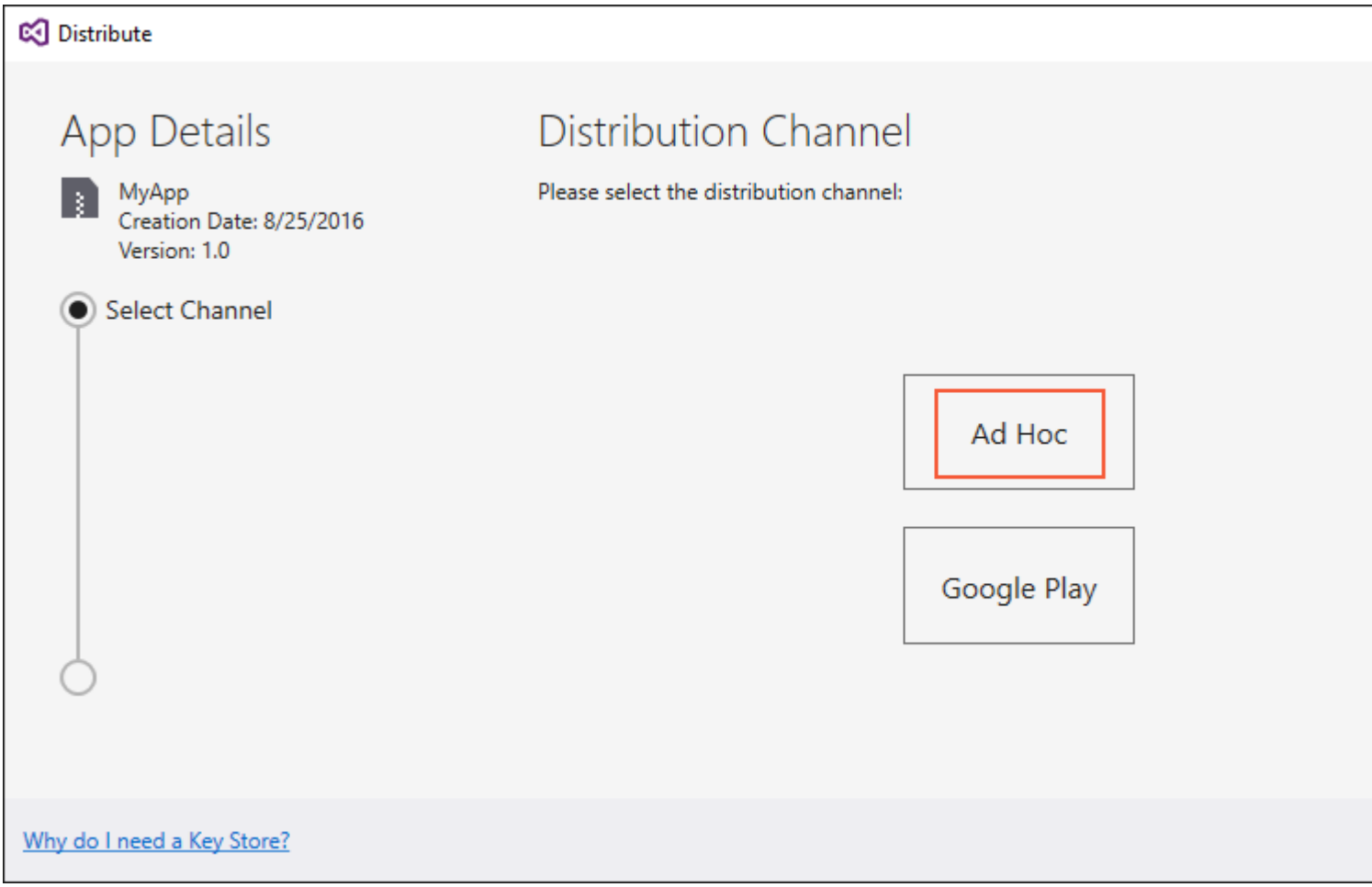


Klicken Sie nach Abschluss der Archivierung des Projekts auf Verteilen, um fortzufahren.

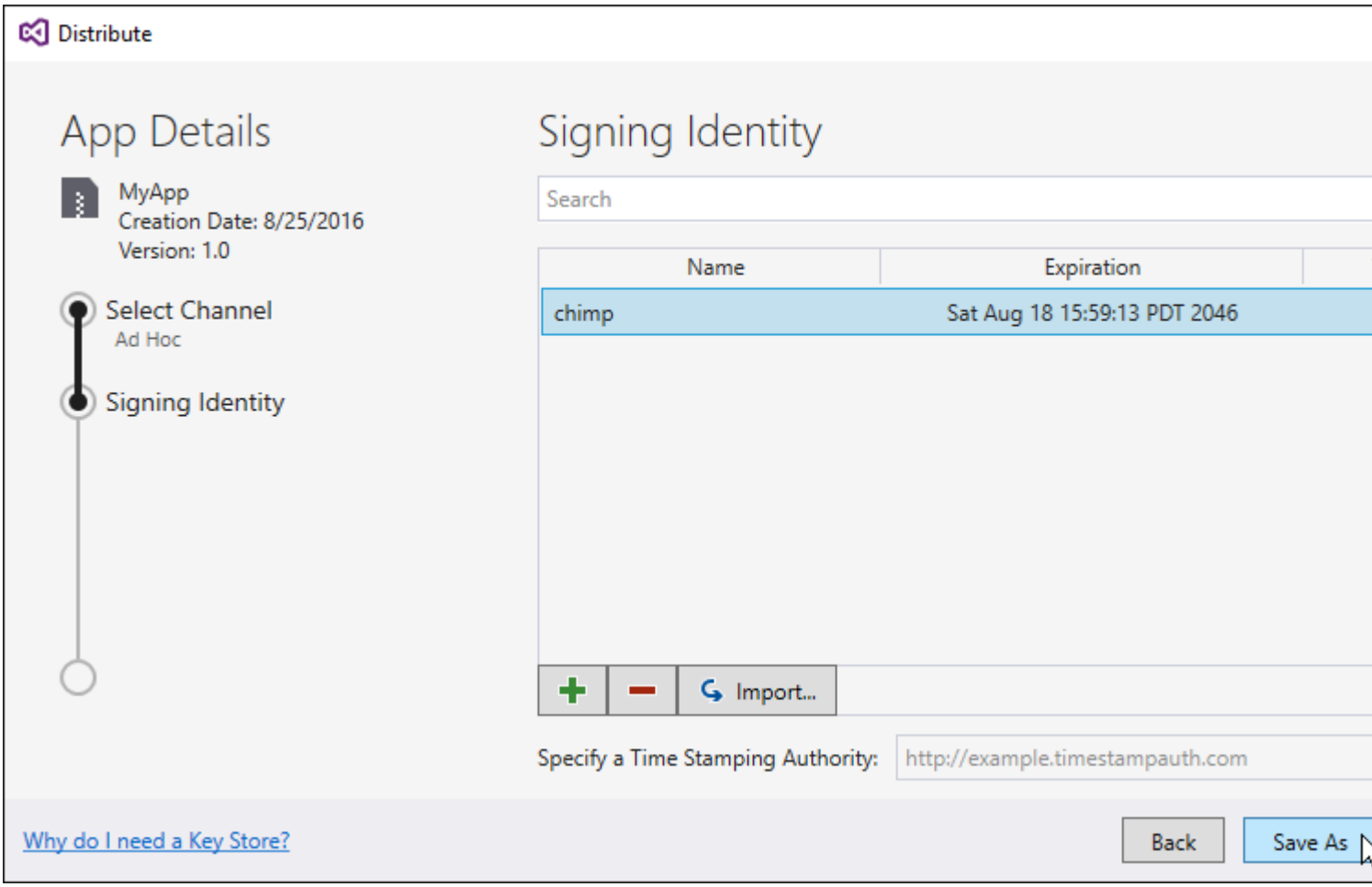


Auf dem Bildschirm "Verteilen" werden zwei Optionen angezeigt: Ad-hoc und Google Play. Beim ersten wird ein APK erstellt und auf Ihrem Computer gespeichert. Der zweite wird die App direkt in Google Play veröffentlichen.

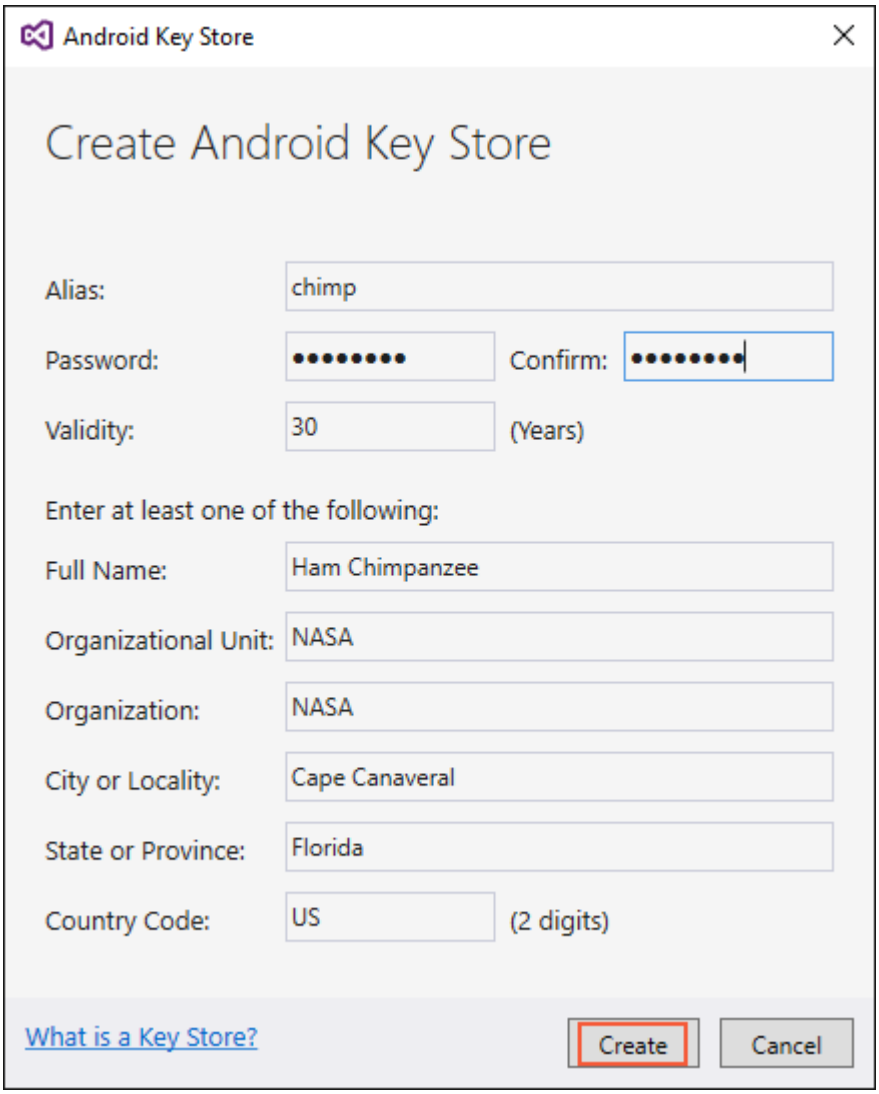
Es wird empfohlen, die erste zu wählen, damit Sie den APK bei Bedarf in anderen Geräten testen können.



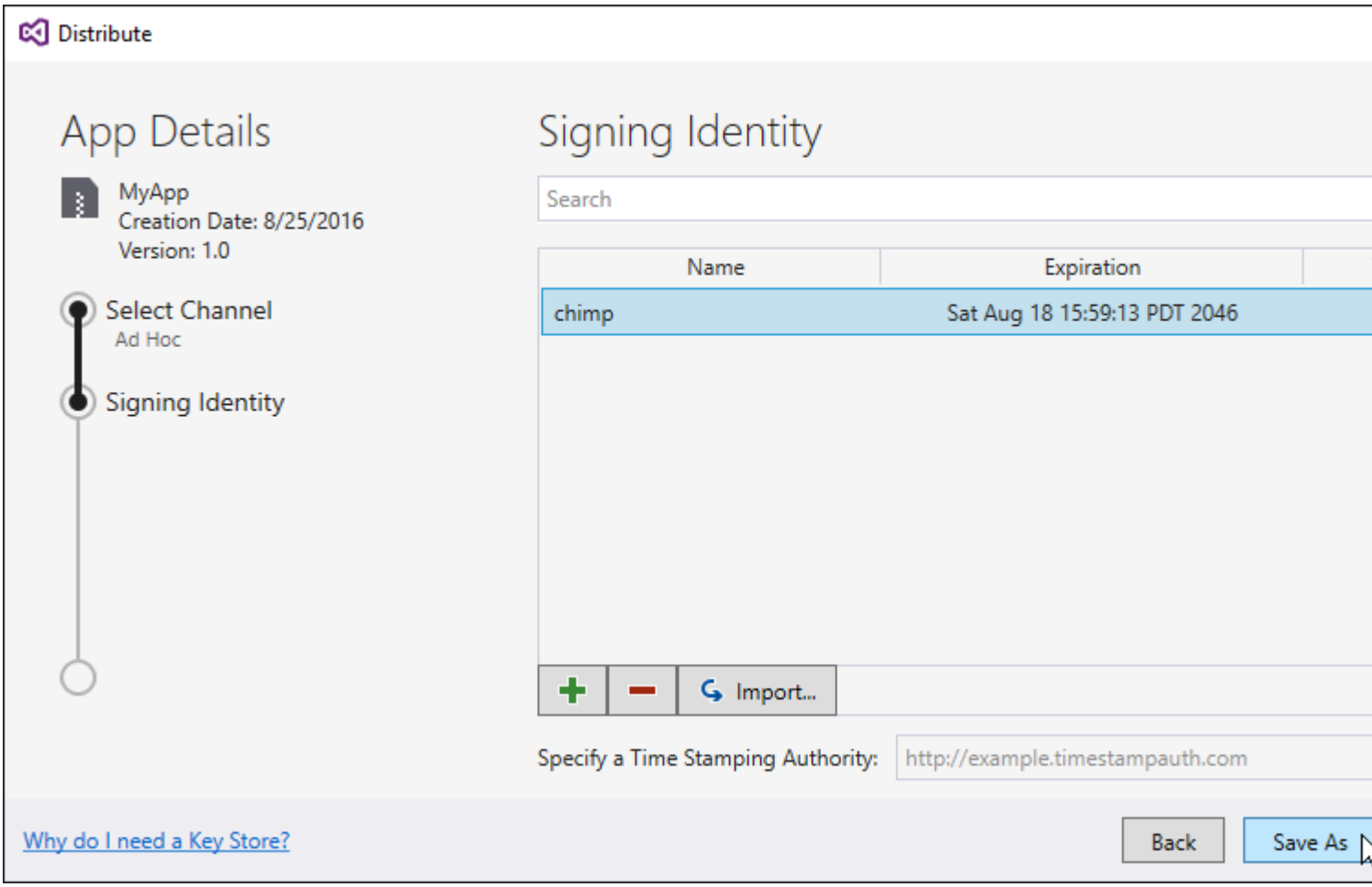
Im folgenden Bildschirm wird ein Android Key Store benötigt, um den APK zu signieren. Wenn Sie bereits über eines verfügen, klicken Sie auf Importieren ...; Wenn Sie dies nicht tun, können Sie einen neuen Android Key Store erstellen, indem Sie auf + klicken.

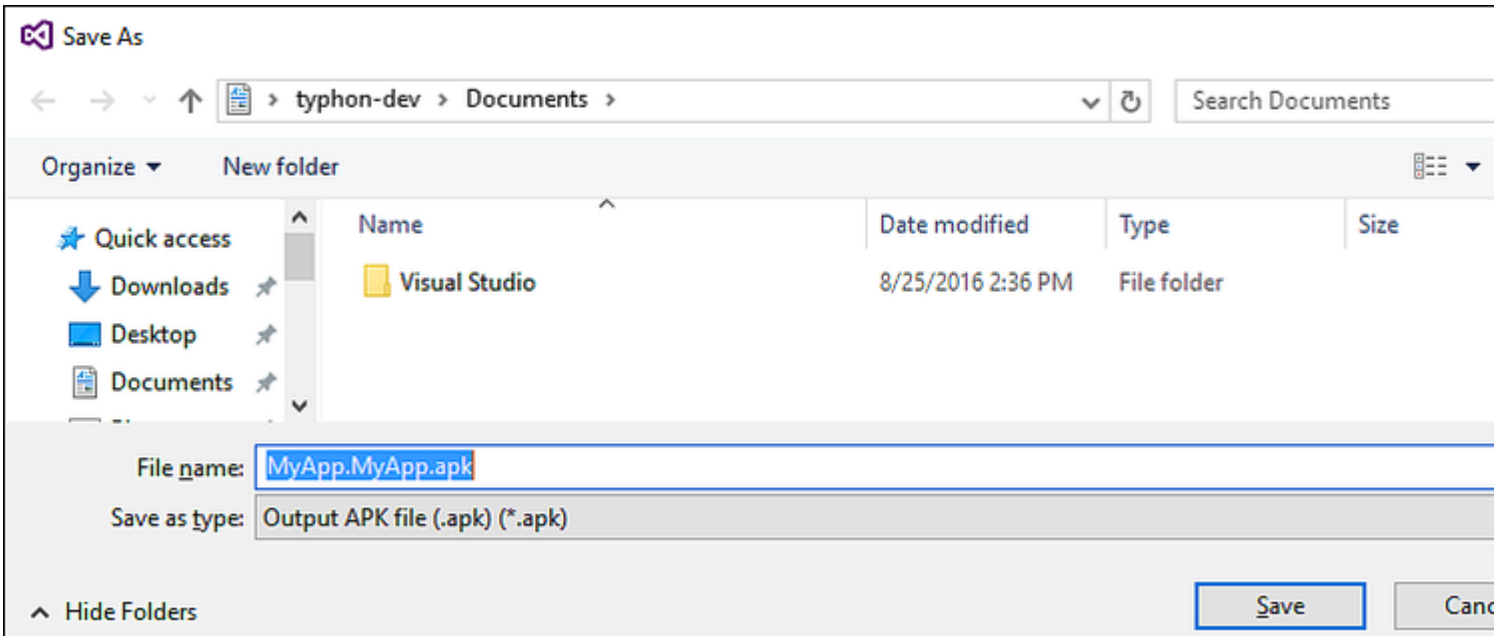


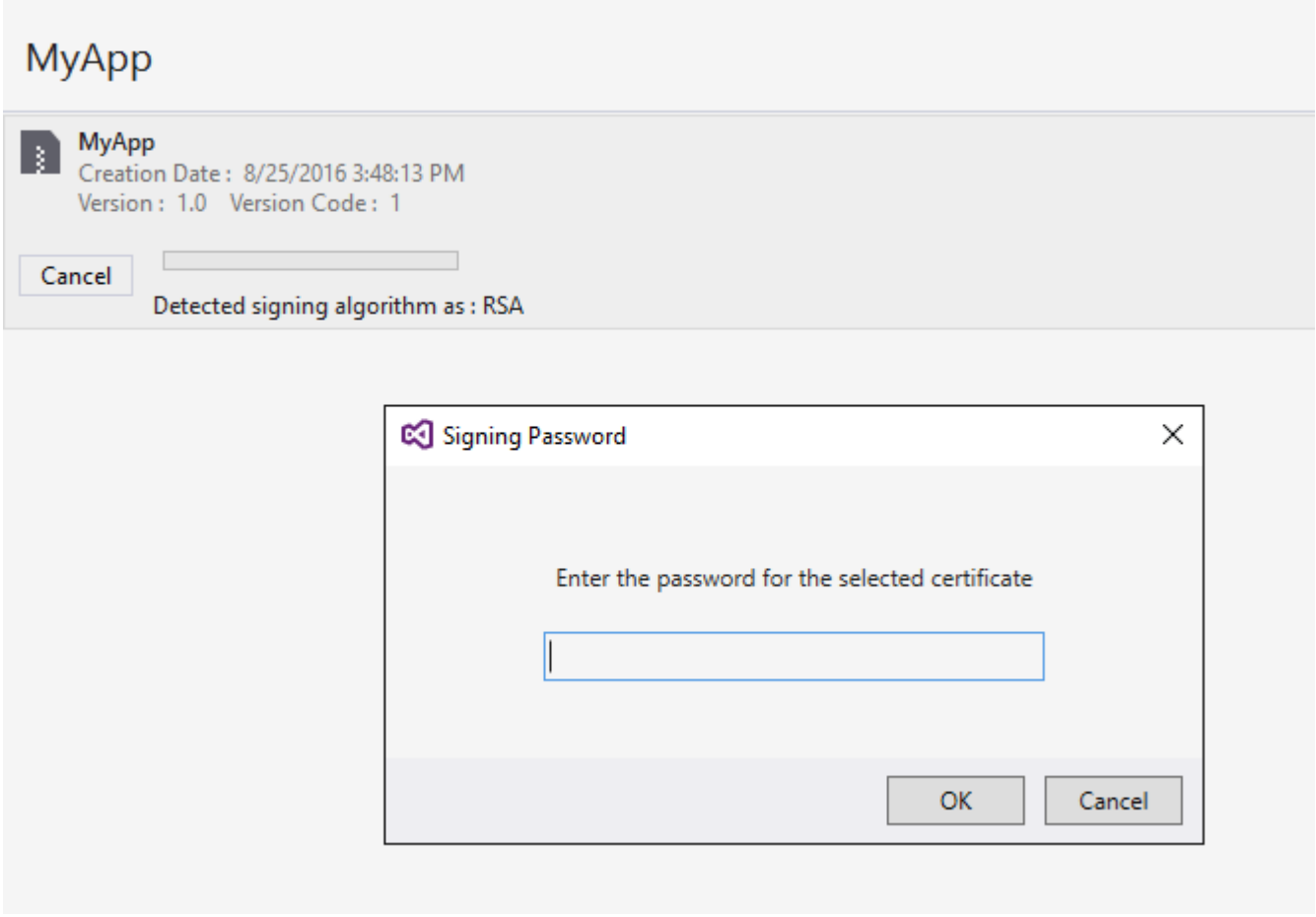
Erstellen eines neuen Android Key Store-Bildschirms:



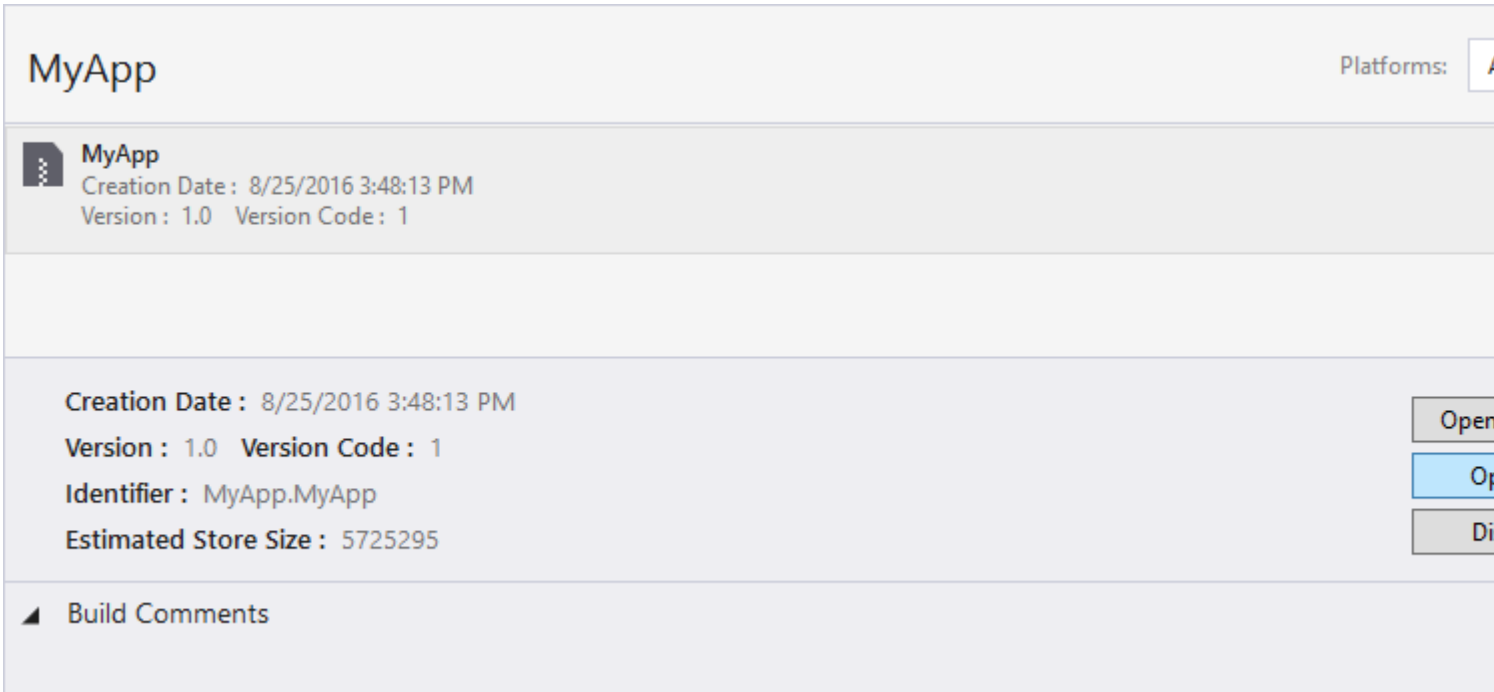
Um die APK zu erstellen, klicken Sie auf Speichern unter. Sie werden möglicherweise aufgefordert, das Schlüsselspeicher-Kennwort einzugeben.

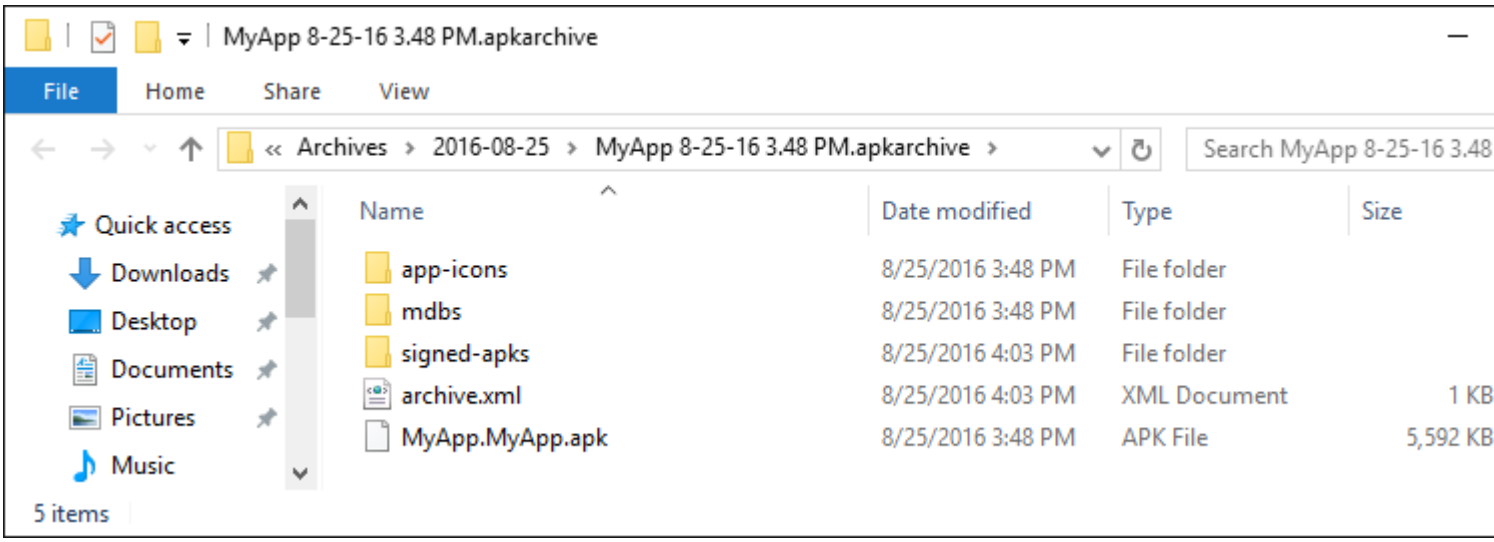

Distribute
App Details
MyApp
Creation Date: 8/25/2016
Version: 1.0
Select Channel
Ad Hoc
Signing Identity
Signing Identity
Search
Name
Expiration
chimp
Sat Aug 18 15:59:13 PDT 2046
Import...
Specify a Time Stamping Authority:
http://example.timestampauth.com
Why do I need a Key Store?
Back
Save As


Save As
typhon-dev
Documents
Search Documents
Organize
New folder
Quick access
Downloads
Desktop
Documents
Name
Date modified
Type
Size
Visual Studio
8/25/2016 2:36 PM
File folder
File name:
MyApp.MyApp.apk
Save as type:
Output APK file (.apk) (*.apk)
Hide Folders
Save



Wenn der Vorgang abgeschlossen ist, können Sie im Ordner "Archiv" auf "Ordner öffnen" klicken, um die generierte APK-Datei anzuzeigen.





Aktivieren von MultiDex in Ihrem Xamarin.Android APK

MultiDex ist eine Bibliothek im Android APK, mit der die App über 65.536 Methoden verfügt.

Die Android-APKs verfügen über ausführbare Dalvik-Dateien (.dex), die die generierten Bytecodes enthalten, die aus Ihrem Java-Code kompiliert wurden. Jede DEX-Datei kann bis zu 65.536 Methoden enthalten (2 ^ 16).

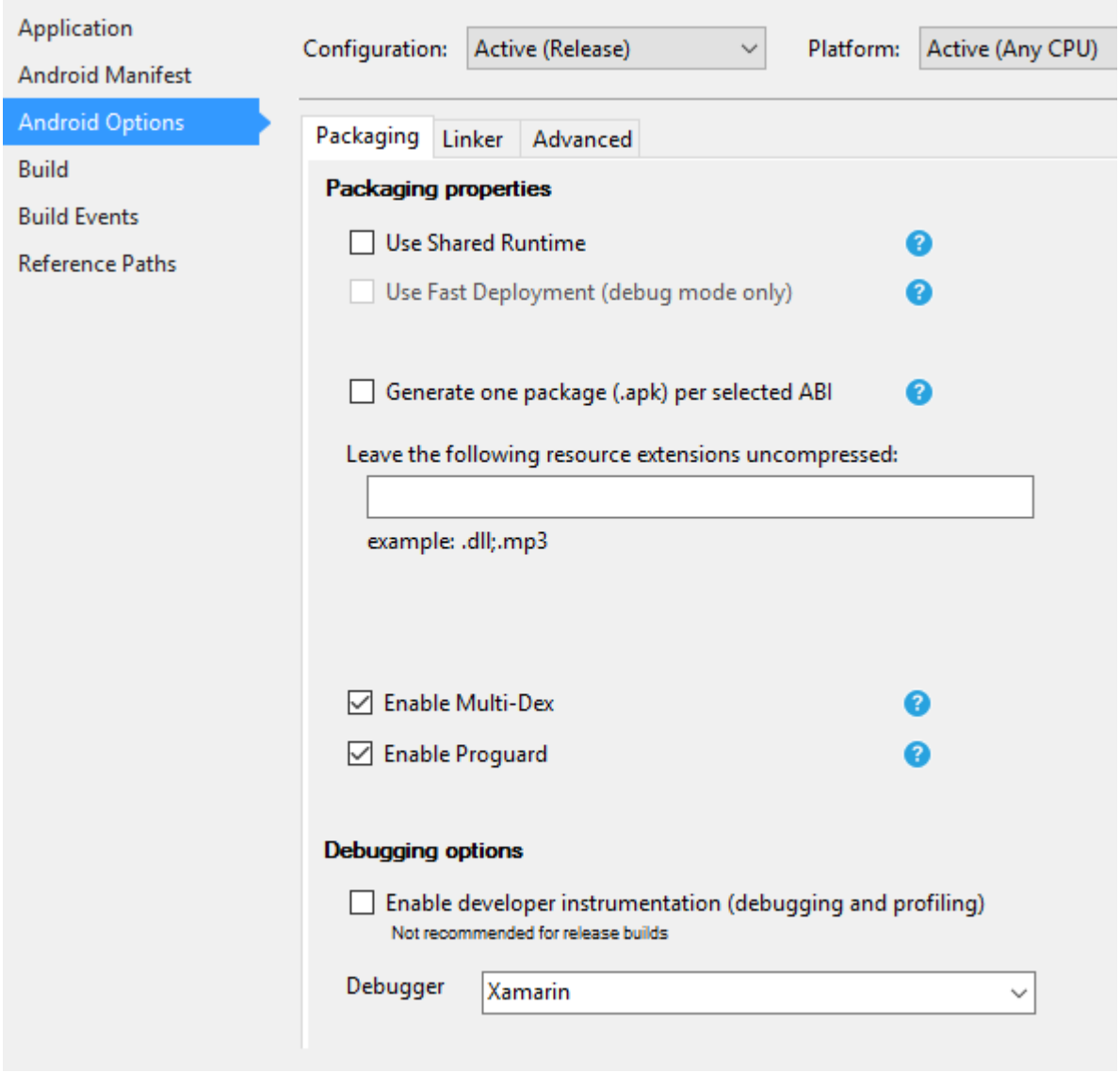
Android-Betriebssystemversionen vor Android 5.0 Lollipop (API 21) verwenden die Dalvik-Laufzeitumgebung, die nur eine .dex-Datei pro APK unterstützt, wobei die APK auf 65.536 Methoden beschränkt ist. Ab Android 5.0 wird für das Android-Betriebssystem die ART-Laufzeitumgebung verwendet, die mehr als eine .dex-Datei pro APK unterstützt, wodurch das Limit vermieden wird.

Um das 65k-Methodenlimit in Android-Versionen unter API 21 zu überschreiten, müssen die Entwickler die MultiDex-Unterstützungsbibliothek verwenden. Der MultiDex erstellt zusätzliche classes.dex-Dateien (classes2.dex, classes3.dex, ...), die diese in der classes.dex-Datei referenzieren. Wenn die App mit dem Laden beginnt, verwendet sie eine MultiDexApplication-Klasse, um die zusätzlichen .dex-Dateien zu laden.

Wenn Ihre Android-App eine Mindest-SDK-Version über oder gleich API 21 (Android 5.0 Lollipop) anstrebt, ist es nicht erforderlich, die MultiDex-Bibliothek zu verwenden, da das Betriebssystem die zusätzlichen .dex-Dateien nativ verarbeitet. Wenn der Entwickler jedoch aus Kompatibilitätsgründen ältere Android-Betriebssysteme unterstützen möchte, sollte er die MultiDex-Bibliothek verwenden.

So verwenden Sie MultiDex in Ihrer Xamarin.Android-App

Um MultiDex in Ihrer Xamarin.Android-App zu aktivieren, gehen Sie zu Ihrem Projekt Eigenschaften -> Android-Optionen -> Paketierung -> Multi-Dex aktivieren, wie im folgenden Druckbildschirm:



Dann müssen Sie eine MultiDexApplication-Klasse in Ihrer App erstellen. Erstellen Sie im Stammverzeichnis des Projekts eine neue Klasse (klicken Sie im Projektmappen-Explorer mit der rechten Maustaste in das Projekt, Hinzufügen ... -> Klasse oder Umschalt + Alt + C). Kopieren Sie in der neuen Klassendatei den folgenden Code und ersetzen Sie den Namespace Sample durch den Namen Ihres Xamarin.Android-Projekt-Namespaces.

```
using System;
using Android.App;
using Android.Runtime;
using Java.Interop;

namespace Sample
{
    [Register("android/support/multidex/MultiDexApplication", DoNotGenerateAcw = true)]
    public class MultiDexApplication : Application
    {
        internal static readonly JniPeerMembers _members =
        new XAPeerMembers("android/support/multidex/MultiDexApplication", typeof
(MultiDexApplication));

        internal static IntPtr java_class_handle;

        private static IntPtr id_ctor;
```

```
        [Register(".ctor", "()V", "", DoNotGenerateAcw = true)]
        public MultiDexApplication()
        : base(IntPtr.Zero, JniHandleOwnership.DoNotTransfer)
        {
            if (Handle != IntPtr.Zero)
                return;

            try
            {
                if (GetType() != typeof (MultiDexApplication))
                {
                    SetHandle(
                        JNIEnv.StartCreateInstance(GetType(), "()V"),
                        JniHandleOwnership.TransferLocalRef);
                        JNIEnv.FinishCreateInstance(Handle, "()V");
                    return;
                }

                if (id_ctor == IntPtr.Zero)
                    id_ctor = JNIEnv.GetMethodID(class_ref, "<init>", "()V");
                SetHandle(
                    JNIEnv.StartCreateInstance(class_ref, id_ctor),
                    JniHandleOwnership.TransferLocalRef);
                JNIEnv.FinishCreateInstance(Handle, class_ref, id_ctor);
            }
            finally
            {
            }
        }

        protected MultiDexApplication(IntPtr javaReference, JniHandleOwnership transfer)
            : base(javaReference, transfer)
        {
        }

        internal static IntPtr class_ref
        {
            get { return JNIEnv.FindClass("android/support/multidex/MultiDexApplication", ref
java_class_handle); }
        }

        protected override IntPtr ThresholdClass
        {
            get { return class_ref; }
        }

        protected override Type ThresholdType
        {
            get { return typeof (MultiDexApplication); }
        }
    }
}
```

Quellcode hier.

Wenn Sie in Visual Studio für Windows entwickeln, gibt es auch einen Fehler in den Android SDK-Build-Tools, die Sie beheben müssen, um die classes.dex-Dateien beim Erstellen Ihres Projekts ordnungsgemäß zu erstellen.

Gehen Sie zu Ihrem Android SDK-Ordner, öffnen Sie den Build-Tools-Ordner, und es werden Ordner mit den Nummern der Android SDK-Compiler angezeigt, z.

C: \ android-sdk \ build-tools \ 23.0.3 \

C: \ android-sdk \ build-tools \ 24.0.1 \

C: \ android-sdk \ build-tools \ 25.0.2 \

In jedem dieser Ordner befindet sich eine Datei namens **mainClassesDex.bat** , ein Batch-Skript, das zum Erstellen der classes.dex-Dateien verwendet wird. Öffnen Sie jede mainClassesDex.bat-Datei mit einem Texteditor (Notepad oder Notepad ++) und suchen und ersetzen Sie den Block in seinem Skript:

```
if DEFINED output goto redirect
call "%java_exe%" -Djava.ext.dirs="%frameworkdir%" com.android.multidex.MainDexListBuilder
"%disableKeepAnnotated%" "%tmpJar%" "%params%"
goto afterClassReferenceListBuilder
:redirect
call "%java_exe%" -Djava.ext.dirs="%frameworkdir%" com.android.multidex.MainDexListBuilder
"%disableKeepAnnotated%" "%tmpJar%" "%params%" 1>"%output%"
:afterClassReferenceListBuilder
```

Mit dem Block:

```
SET params=%params:'=%
if DEFINED output goto redirect
call "%java_exe%" -Djava.ext.dirs="%frameworkdir%" com.android.multidex.MainDexListBuilder
%disableKeepAnnotated% "%tmpJar%" %params%
goto afterClassReferenceListBuilder
:redirect
call "%java_exe%" -Djava.ext.dirs="%frameworkdir%" com.android.multidex.MainDexListBuilder
%disableKeepAnnotated% "%tmpJar%" %params% 1>"%output%"
:afterClassReferenceListBuilder
```

Quelle hier.

Speichern Sie jede mainClassesDex.bat nach Änderungen im Texteditor.

Nach den obigen Schritten sollten Sie in der Lage sein, Ihre Xamarin.Android-App mit MultiDex erfolgreich zu erstellen.

Aktivieren von ProGuard in Ihrem Xamarin.Android APK

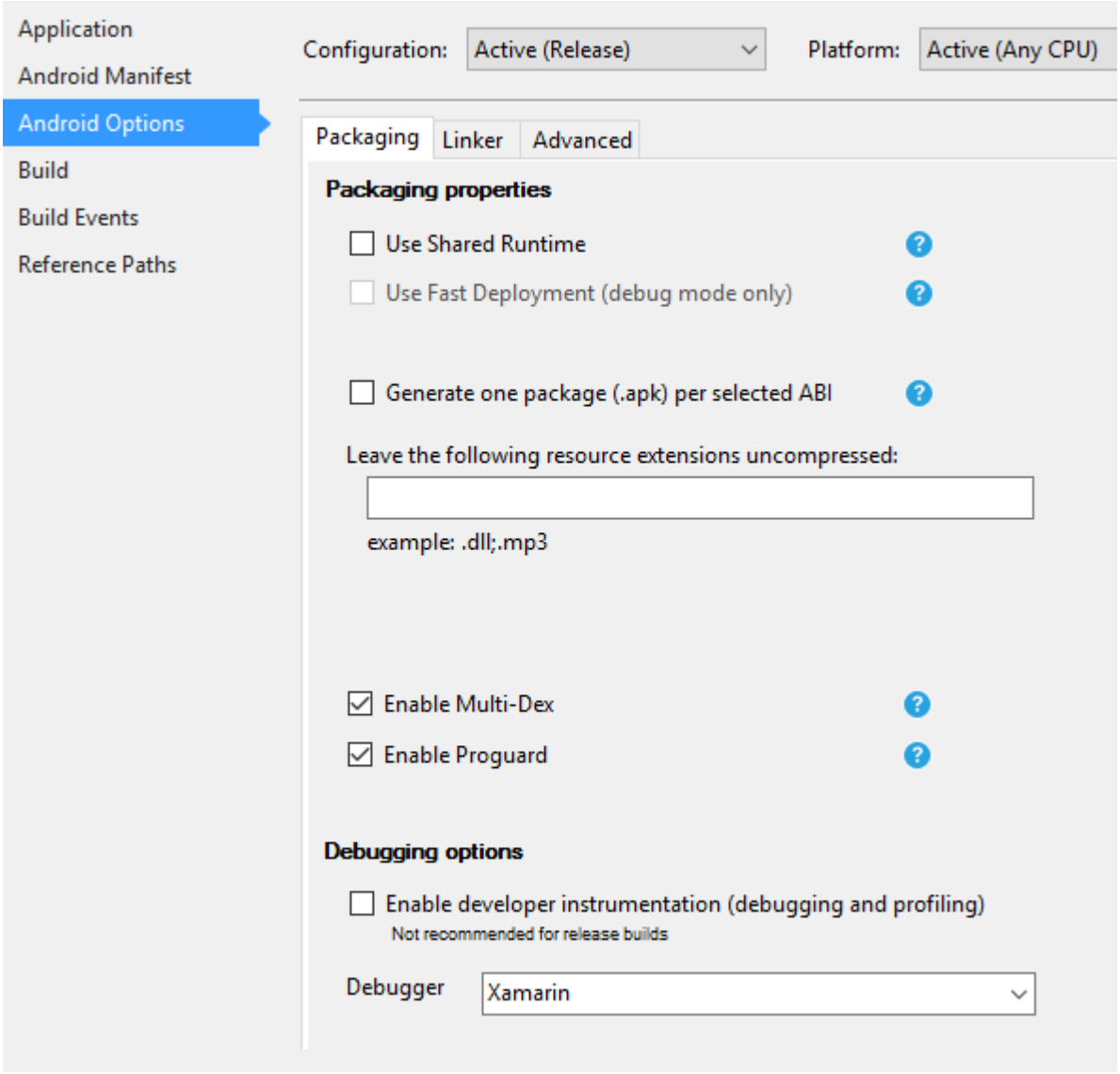
ProGuard ist ein Werkzeug, das im Erstellungsprozess verwendet wird, um den Java-Code Ihres APK zu optimieren und zu verschleiern sowie nicht verwendete Klassen zu entfernen. Die resultierende APK bei Verwendung von ProGuard hat eine geringere Größe und ist schwieriger zu entwickeln (Dekompilierung).

ProGuard kann auch in Xamarin.Android-Apps verwendet werden. Außerdem wird die APK-Dateigröße reduziert und der Java-Code verschleiert. Beachten Sie jedoch, dass die ProGuard-Verschleierung nur für Java-Code gilt. Um .NET-Code zu verschleiern, sollte der Entwickler

Dotfuscator oder ähnliche Tools verwenden.

So verwenden Sie ProGuard in Ihrer Xamarin.Android-App

Um ProGuard in Ihrer Xamarin.Android-App zu aktivieren, gehen Sie zu Ihrem Projekt Eigenschaften -> Android-Optionen -> Paketierung -> ProGuard aktivieren, wie im folgenden Druckbildschirm:

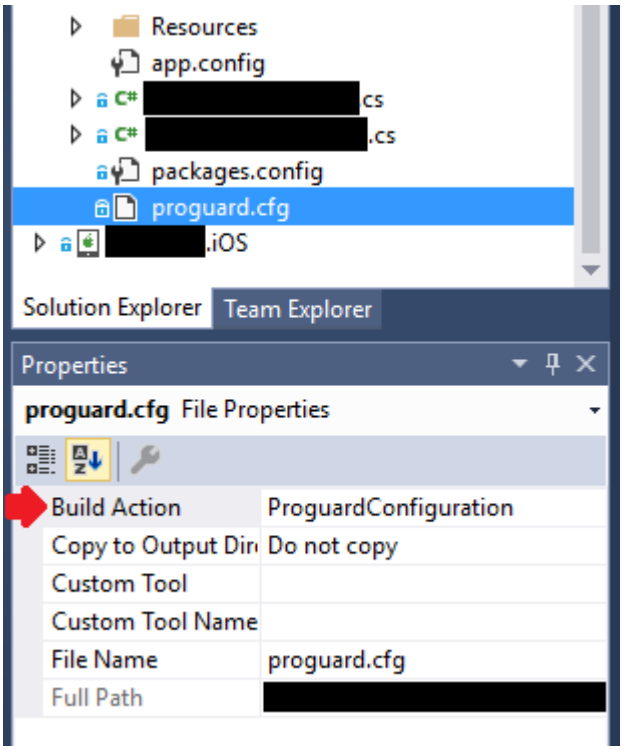


Dies ermöglicht ProGuard beim Erstellen Ihrer App.

Xamarin.Android legt standardmäßig seine eigenen Konfigurationen für ProGuard fest, die sich in den Ordnern `obj/Debug/proguard` oder `obj/Release/proguard` in den Dateien `proguard_project_primary.cfg` , `proguard_project_references.cfg` und `proguard_xamarin.cfg` . Die drei Dateien werden als Konfigurationen für ProGuard zusammengefasst und beim Erstellen automatisch von Xamarin erstellt.

Wenn der Entwickler die ProGuard-Optionen weiter anpassen möchte, kann er im Projektstamm eine Datei mit dem Namen `proguard.cfg` (andere Namen sind ebenfalls gültig, sofern die

Erweiterung .cfg ist) und die Build-Aktion auf ProguardConfiguration setzen. wie im Bild unten:



In die Datei können benutzerdefinierte ProGuard-Optionen eingefügt werden, z. B. `-dontwarn` , `-keep class` und andere .

Wichtig

Wie üblich (April / 2017) enthält das normalerweise heruntergeladene Android SDK eine ältere ProGuard-Version, die zu Fehlern beim Erstellen der App mit Java 1.8 führen kann. Beim Erstellen zeigt die Fehlerliste die folgende Meldung an:

```
Error
Can't read [C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\MonoAndroid\v7.0\mono.android.jar]
(Can't process class [android/app/ActivityTracker.class]  (Unsupported class version number
[52.0] (maximum 51.0, Java 1.7))) [CREATEMULTIDEXMAINDEXCLASSLIST]
```

Quelle hier.

Um dieses Problem zu beheben, müssen Sie die neueste Version von ProGuard (hier) herunterladen und den Inhalt der ZIP-Datei nach `android-sdk\tools\proguard\` . Dadurch wird der ProGuard aktualisiert und der Bauprozess sollte problemlos ablaufen.

Danach sollten Sie in der Lage sein, Ihre Xamarin.Android-App mit ProGuard erfolgreich zu erstellen.

"Rätselhafte" Fehler im Zusammenhang mit ProGuard und Linker

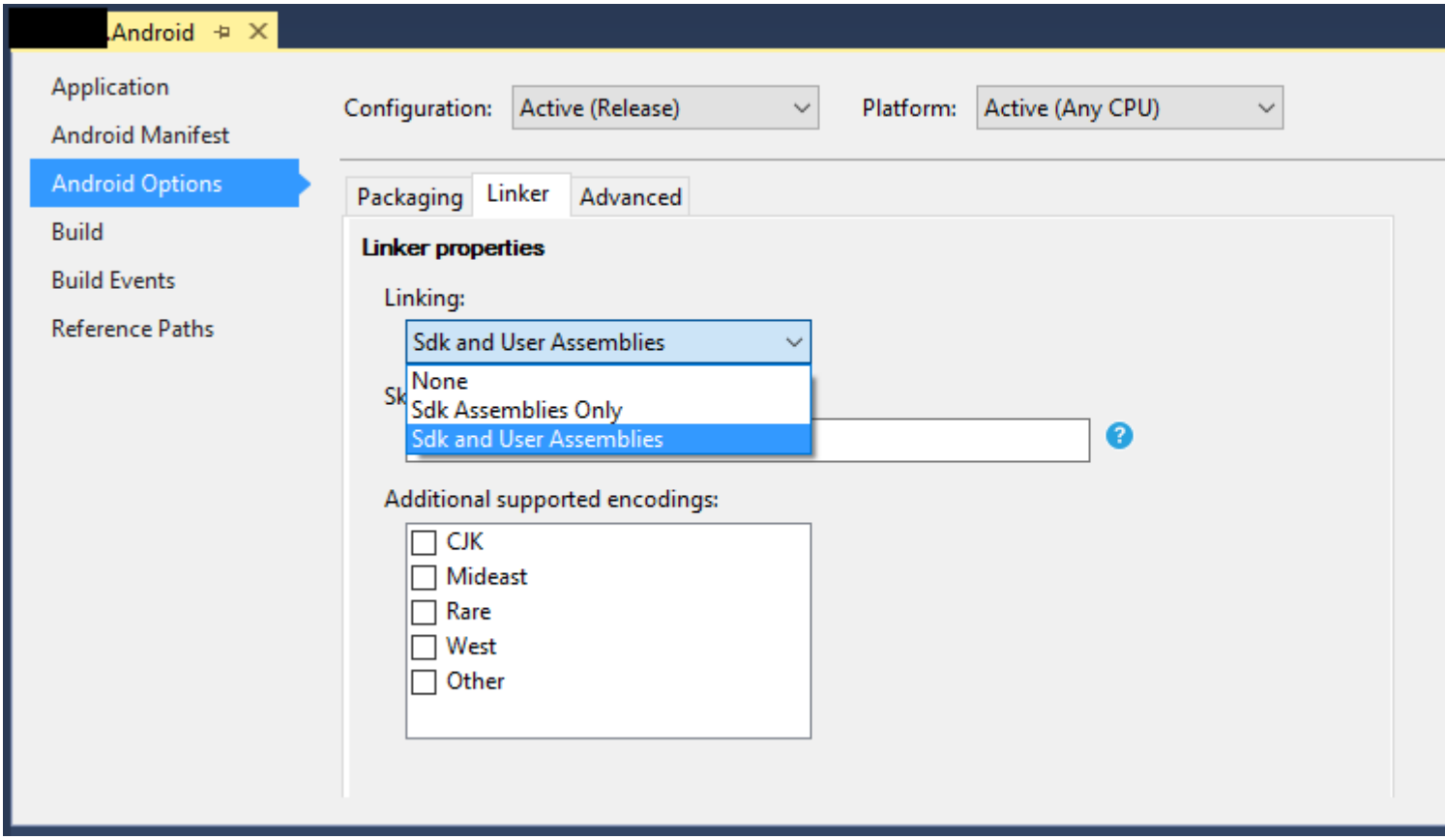
Sie haben eine großartige App erstellt und in Debug getestet, mit guten Ergebnissen. Alles hat gut funktioniert!

Dann haben Sie sich jedoch entschieden, Ihre App für die Veröffentlichung vorzubereiten. Sie haben MultiDex, ProGuard und Linker eingerichtet und dann funktioniert es nicht mehr.

Dieses Tutorial soll Ihnen helfen, allgemeine Probleme im Zusammenhang mit ProGuard und Linker herauszufinden, die mysteriöse Fehler verursachen können.

Informationen zu Xamarin.Linker

Xamarin.Linker ist ein Werkzeug im Erstellungsprozess, das nicht verwendeten Code und Klassen **aus Ihrem .NET-Code (nicht Java-Code) entfernt** . In den Eigenschaften Ihres Projekts -> Android-Optionen -> Linker wird ein Auswahlfeld angezeigt, das mit den Optionen verknüpft wird:



Keine : Es wird kein Code entfernt.

Nur Sdk-Assemblys : Diese Option veranlasst den Xamarin.Linker, nur in den Xamarin-Bibliotheken nach nicht verwendetem Code zu **suchen** . **Diese Option ist sicher.**

Sdk- und Benutzerbaugruppen : Mit dieser Option wird der Xamarin.Linker in den Xamarin-Bibliotheken und im Projektcode (einschließlich PCLs, Xamarin-Komponenten und NuGet-Paketen) auf nicht verwendeten Code überprüft. **Diese Option ist nicht immer sicher!**

Bei Verwendung der Option "Sdk and User Assemblies" kann Xamarin.Linker denken, dass Teile des Codes nicht verwendet werden, wenn sie tatsächlich verwendet werden! Dies kann dazu führen, dass einige Bibliotheken nicht mehr ordnungsgemäß funktionieren und Fehler in Ihrer App verursachen.

Damit der Xamarin.Linker keinen Code entfernt, gibt es 3 Optionen:

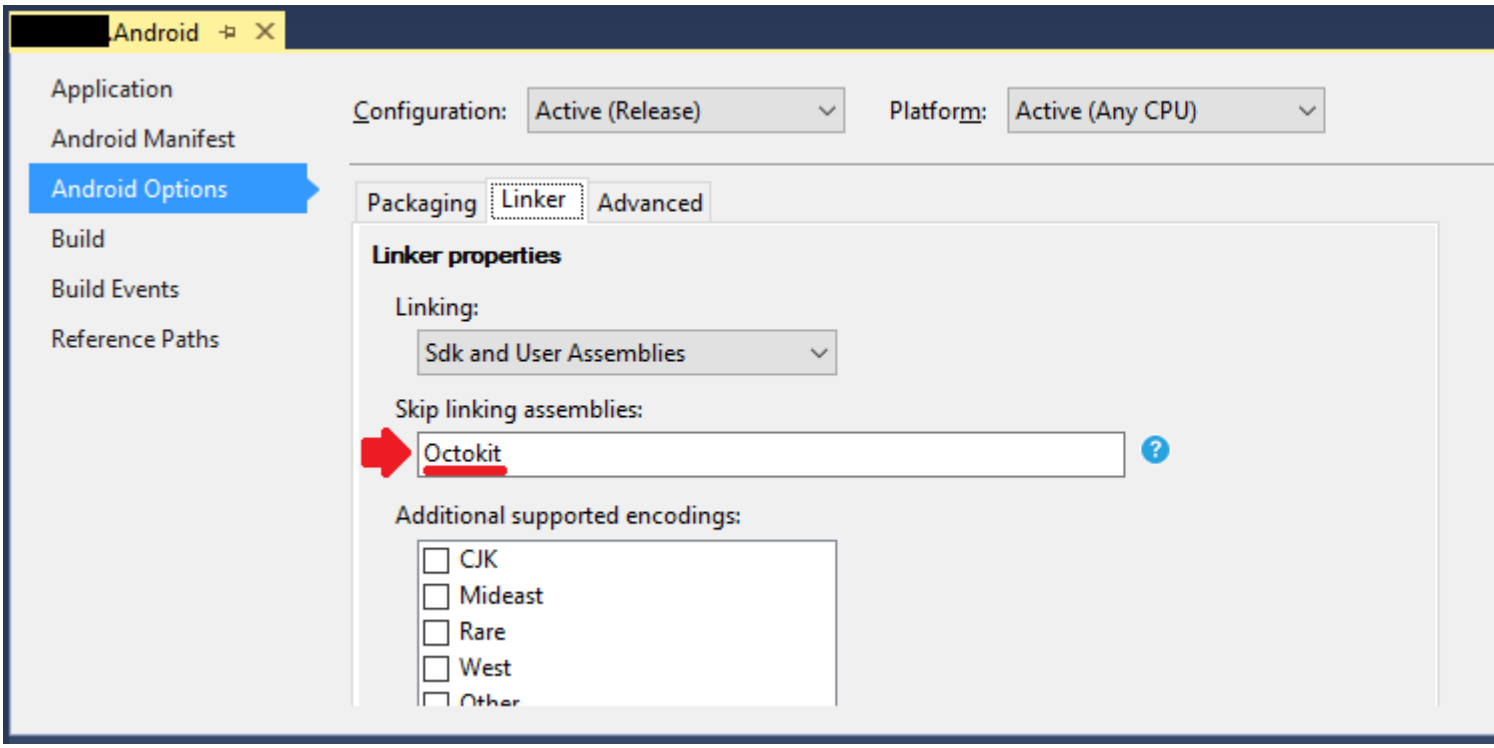
1. Festlegen der Linking-Option auf Keine oder Nur Sdk-Assemblies;
2. Überspringen Sie Verknüpfungsanordnungen.
3. Verwenden des Attributs "Preserve".

Beispiel für 2. Überspringen von Verknüpfungsbaugruppen

In dem folgenden Beispiel verursachte die Verwendung von Xamarin.Linker ein NuGet-Paket (Octokit), das funktioniert, um nicht mehr zu funktionieren, da es keine Verbindung zum Internet mehr herstellen konnte:

```
[0:] ERROR
[0:] SOURCE: mscorlib
[0:] MESSAGE: Object reference not set to an instance of an object.
[0:] STACK TRACE:   at Octokit.PocoJsonSerializerStrategy.DeserializeObject (System.Object
value, System.Type type) [0x003d8] in D:\repos\octokit.net\Octokit\SimpleJson.cs:1472
  at Octokit.Internal.SimpleJsonSerializer+GitHubSerializerStrategy.DeserializeObject
(System.Object value, System.Type type) [0x001c3] in
D:\repos\octokit.net\Octokit\Http\SimpleJsonSerializer.cs:165
  at Octokit.SimpleJson.DeserializeObject (System.String json, System.Type type,
Octokit.IJsonSerializerStrategy jsonSerializerStrategy) [0x00007] in
D:\repos\octokit.net\Octokit\SimpleJson.cs:583
  at Octokit.SimpleJson.DeserializeObject[T] (System.String json,
Octokit.IJsonSerializerStrategy jsonSerializerStrategy) [0x00000] in
D:\repos\octokit.net\Octokit\SimpleJson.cs:595
  at Octokit.Internal.SimpleJsonSerializer.Deserialize[T] (System.String json) [0x00000] in
D:\repos\octokit.net\Octokit\Http\SimpleJsonSerializer.cs:21
  at Octokit.Internal.JsonHttpPipeline.DeserializeResponse[T] (Octokit.IResponse response)
[0x000a7] in D:\repos\octokit.net\Octokit\Http\JsonHttpPipeline.cs:62
  at Octokit.Connection+<Run>d__54`1[T].MoveNext () [0x0009c] in
D:\repos\octokit.net\Octokit\Http\Connection.cs:574
--- End of stack trace from previous location where exception was thrown ---
```

Damit die Bibliothek wieder funktioniert, musste der Paketreferenzname im Feld Verknüpfungsassetten überspringen in Projekt -> Eigenschaften -> Android-Optionen -> Linker wie in der folgenden Abbildung hinzugefügt werden:



Danach begann die Bibliothek ohne Probleme zu arbeiten.

Beispiel für 3. Verwendung des Attributs "Preserve":

Xamarin.Linker erkennt als ungenutzten Code hauptsächlich Code aus Modellklassen im Kern Ihres Projekts.

Um die Klasse während des Verknüpfungsprozesses beizubehalten, können Sie das Attribut Preserve verwenden.

Erstellen Sie zunächst in Ihrem Projektkern eine Klasse namens **PreserveAttribute.cs** , fügen Sie den folgenden Code ein und ersetzen Sie den Namespace durch den Namespace Ihres Projekts:

PreserveAttribute.cs:

```
namespace My_App_Core.Models
{
    public sealed class PreserveAttribute : System.Attribute
    {
        public bool AllMembers;
        public bool Conditional;
    }
}
```

Fügen Sie in jede Modellklasse Ihres Projektkerns das Attribut "Preserve" wie im folgenden Beispiel ein:

Country.cs:

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

```
namespace My_App_Core.Models
{
    [Preserve(AllMembers = true)]
    public class Country
    {
        public String name { get; set; }
        public String ISOcode { get; set; }

        [Preserve(AllMembers = true)]
        public Country(String name, String ISOCode)
        {
            this.name = name;
            this.ISOCode = ISOCode;
        }
    }
}
```

Danach entfernt der Verknüpfungsvorgang den erhaltenen Code nicht mehr.

Grundlegendes zu ProGuard

ProGuard ist ein Werkzeug im Aufbauprozess, das nicht verwendeten Code und Klassen **aus Ihrem Java-Code entfernt** . Es verschleiert und optimiert den Code.

Es kann jedoch vorkommen, dass ProGuard möglicherweise Code entfernt, den es als nicht verwendet erkennt, wenn dies nicht der Fall ist. Um dies zu vermeiden, muss der Entwickler die App debuggen (in Android Device Monitor und in Visual Studio Debug) und feststellen, welche Klasse entfernt wurde, um dann die ProGuard-Konfigurationsdatei so zu konfigurieren, dass die Klasse beibehalten wird.

Beispiel

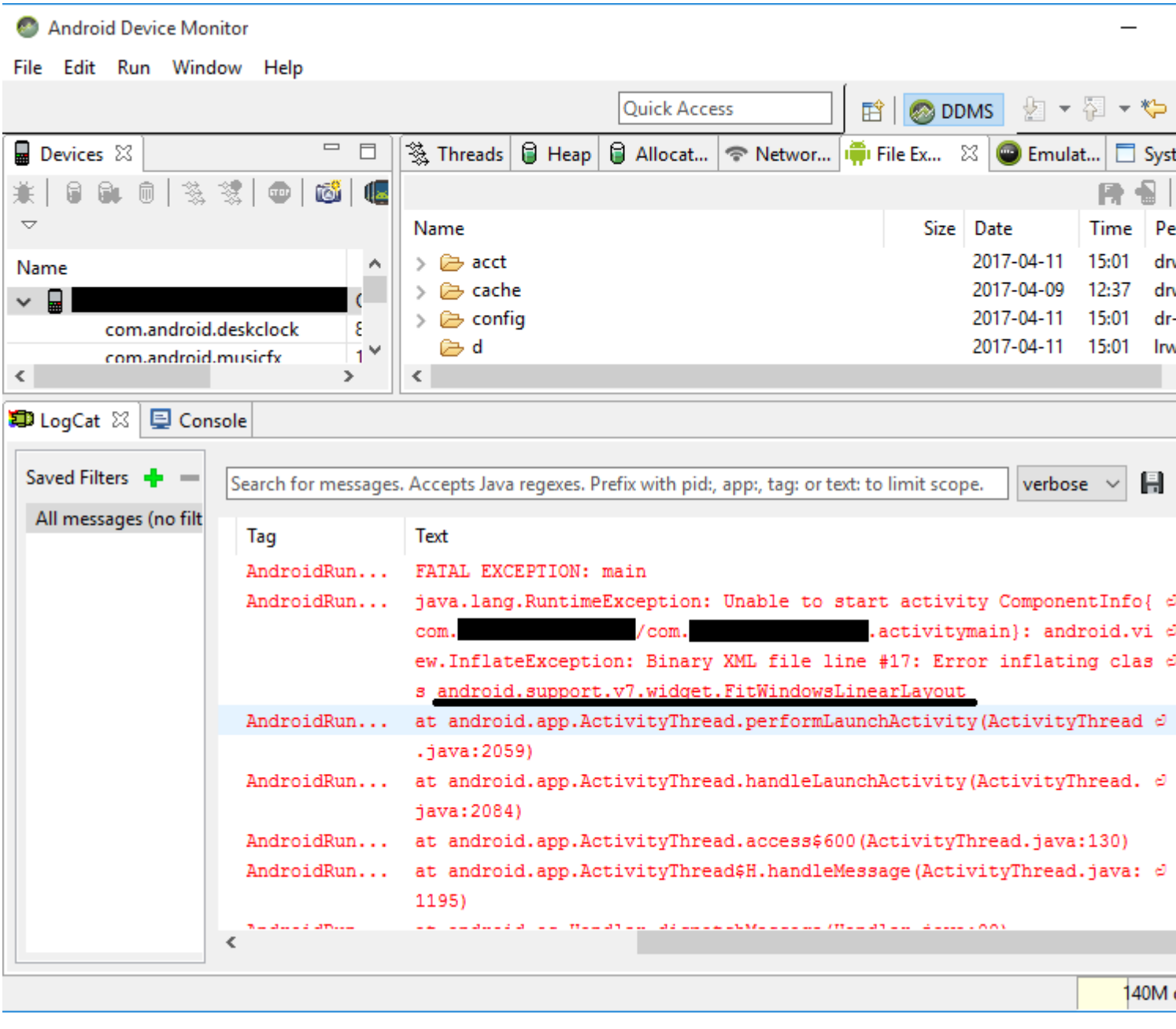
Im nachstehenden Beispiel hat ProGuard zwei Klassen (Android.Support.V7.Widget.FitWindowsLinearLayout und Android.Support.Design.Widget.AppBarLayout) entfernt, die in AXML-Layoutdateien verwendet werden, aber im Code als nicht verwendet angesehen wurden. Die Entfernung verursachte beim Rendern des Aktivitätslayouts ClassNotFoundException im Java-Code:

layout_activitymain.axml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activitymain_drawerlayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true" <!-- ### HERE ### -->
    tools:openDrawer="start">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:fitsSystemWindows="true">
        <!-- ### HERE ## -->
        <android.support.design.widget.AppBarLayout
            android:id="@+id/activitymain_appbarlayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/AppTheme.AppBarOverlay">
...
```

LogCat zeigt einen Fehler beim Erstellen des Layouts in SetContentView:



Um diesen Fehler zu beheben, mussten der ProGuard-Konfigurationsdatei des Projekts die folgenden Zeilen hinzugefügt werden:

```
-keep public class android.support.v7.widget.FitWindowsLinearLayout
-keep public class android.support.design.widget.AppBarLayout
```

Danach wurden beim Erstellen des Layouts keine Fehler mehr angezeigt.

ProGuard-Warnungen

ProGuard zeigt manchmal Warnungen in der Fehlerliste an, nachdem Sie Ihr Projekt erstellt haben. Obwohl sie die Frage aufwerfen, ob Ihre App in Ordnung ist oder nicht, weisen nicht alle Warnungen auf Probleme hin, insbesondere wenn Ihre App erfolgreich erstellt wurde.

Ein Beispiel dafür ist , wenn die Verwendung von Picasso Bibliothek: wenn ProGuard Verwendung dieser Warnungen zeigen kann wie `okio.Okio: can't find referenced class (...)` oder `can't write resource [META-INF/MANIFEST.MF] (Duplicate zip entry [okhttp.jar:META-INF/MANIFEST.MF]) (...)` , aber die App baut und die Bibliothek funktioniert ohne Probleme.

Veröffentlichen Ihres Xamarin.Android APK online lesen: https://riptutorial.com/de/xamarin-android/topic/9601/veroffentlichen-ihres-xamarin-android-apk

Kapitel 12: Xamarin.Android - Bluetooth-Kommunikation

Einführung

In **Xamarin.Android werden** die Eigenschaften **BluetoothSocket.InputStream** und **BluetoothSocket.OutputStream** automatisch in **System.IO.Stream** konvertiert. Im Falle des sogenannten interaktiven Kommunikationsprotokolls, wenn der Server nur dann antwortet, wenn der Client mit ihm spricht, ist System.IO.Stream nicht geeignet, da er keine Methode oder Eigenschaft hat, um die Anzahl der verfügbaren Antwortbytes vor dem Lesen der Antwort zu erhalten.

Parameter

Parameter	Einzelheiten
Steckdose	Eine Instanz des BluetoothSocket-Objekts. Socket muss vor dem Aufruf dieser Methode geöffnet werden.
cmd	Befehl als Byte-Array zum Senden an das BT-Gerät.
_mx	Da diese Methode eine Hardwareressource verwendet, ist es besser, sie von einem separaten Arbeitsthread aus aufzurufen. Dieser Parameter ist eine Instanz des System.Threading.Mutex-Objekts und wird verwendet, um den Thread mit anderen Threads zu synchronisieren, die diese Methode optional aufrufen.
Auszeit	Wartezeit in Millisekunden zwischen Schreib- und Lesevorgängen.

Examples

Senden und empfangen Sie Daten von und zu einem Bluetooth-Gerät über Socket

Im folgenden Beispiel werden die Typen Android.Runtime.InputStreamInvoker und Android.Runtime.OutputStreamInvoker verwendet, um Java.IO.InputStream und Java.IO.OutputStream zu erhalten . Sobald wir eine **Java.IO.InputStream-** Instanz haben, können wir die **.Available ()** -Methode verwenden, um die Anzahl der verfügbaren Antwortbytes **abzurufen,** die wir in der **.Read ()** -Methode verwenden können:

```
byte[] Talk2BTsocket(BluetoothSocket socket, byte[] cmd, Mutex _mx, int timeOut = 150)
{
    var buf = new byte[0x20];
```

```
    _mx.WaitOne();
    try
    {
        using (var ost = socket.OutputStream)
        {
            var _ost = (ost as OutputStreamInvoker).BaseOutputStream;
            _ost.Write(cmd, 0, cmd.Length);
        }

        // needed because when skipped, it can cause no or invalid data on input stream
        Thread.Sleep(timeOut);

        using (var ist = socket.InputStream)
        {
            var _ist = (ist as InputStreamInvoker).BaseInputStream;
            var aa = 0;
            if ((aa = _ist.Available()) > 0)
            {
                var nn = _ist.Read(buf, 0, aa);
                System.Array.Resize(ref buf, nn);
            }
        }
    }
    catch (System.Exception ex)
    {
        DisplayAlert(ex.Message);
    }
    finally
    {
        _mx.ReleaseMutex();     // must be called here !!!
    }

    return buf;
}
```

Xamarin.Android - Bluetooth-Kommunikation online lesen: https://riptutorial.com/de/xamarin-android/topic/10844/xamarin-android---bluetooth-kommunikation

Kapitel 13: Xamarin.Android - So erstellen Sie eine Symbolleiste

Bemerkungen

Liebes Team,

Ich denke, dass es gut ist, über die offizielle Android-Dokumentation zu erwähnen, in der die Symbolleistensteuerung detailliert erklärt wird:

https://developer.android.com/reference/android/support/v7/widget/Toolbar.html

Es gibt auch interessante Inhalte über die Android.Support.v7-Bibliothek, die im Beispiel verwendet wird:

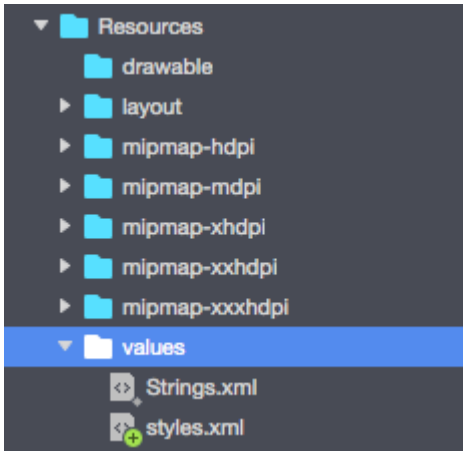
https://developer.android.com/training/appbar/index.html

Examples

Fügen Sie der Xamarin.Android-Anwendung eine Symbolleiste hinzu

Zunächst müssen Sie die Bibliothek Xamarin.Android.Support.V7.AppCompat für NuGet hinzufügen: https://www.nuget.org/packages/Xamarin.Android.Support.v7.AppCompat/

Fügen Sie im Ordner "values" unter "Resources" eine neue XML-Datei mit dem Namen "styles.xml" hinzu:

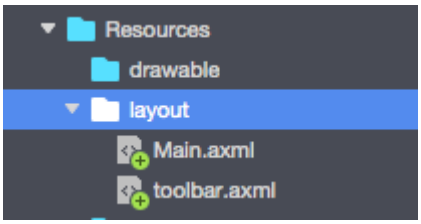


Die Datei "styles.xml" sollte folgenden Code enthalten:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
<style name="MyTheme" parent="MyTheme.Base">
</style>

<!-- Base theme applied no matter what API -->
<style name="MyTheme.Base" parent="Theme.AppCompat.Light.DarkActionBar">
```

```
<item name="windowNoTitle">true</item>
<!--We will be using the toolbar so no need to show ActionBar-->
<item name="windowActionBar">false</item>
<!-- Set theme colors from http://www.google.com/design/spec/style/color.html#color-color-
palette-->
<!-- colorPrimary is used for the default action bar background -->
<item name="colorPrimary">#2196F3</item>
<!-- colorPrimaryDark is used for the status bar -->
<item name="colorPrimaryDark">#1976D2</item>
<!-- colorAccent is used as the default value for colorControlActivated
     which is used to tint widgets -->
<item name="colorAccent">#FF4081</item>

<item name="colorControlHighlight">#FF4081</item>
<!-- You can also set colorControlNormal, colorControlActivated
     colorControlHighlight and colorSwitchThumbNormal. -->
```

Im nächsten Schritt fügen Sie dem Ordner "Layout" die Datei "toolbar.axml" hinzu, die die Definition der Symbolleisten-Steuerelemente enthält:



Fügen Sie folgenden Code hinzu, um die Symbolleiste zu definieren:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:minHeight="?attr/actionBarSize"
android:background="?attr/colorPrimary"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
```

Öffnen Sie nun die Datei "Main.axml" und fügen Sie den Code unterhalb des schließenden Tags für das erste Layout hinzu. Ihr Code sollte wie folgt aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <include android:id="@+id/toolbar" layout="@layout/toolbar" />

</LinearLayout>
```

Jetzt müssen Sie Informationen zu einem Thema hinzufügen, das Ihre App verwendet. Öffnen Sie die Datei "AndroidManifest" und fügen Sie dem Tag "application" Themeninformationen hinzu:

```
<application android:theme="@style/MyTheme" android:allowBackup="true"
android:icon="@mipmap/icon" android:label="@string/app_name">
```

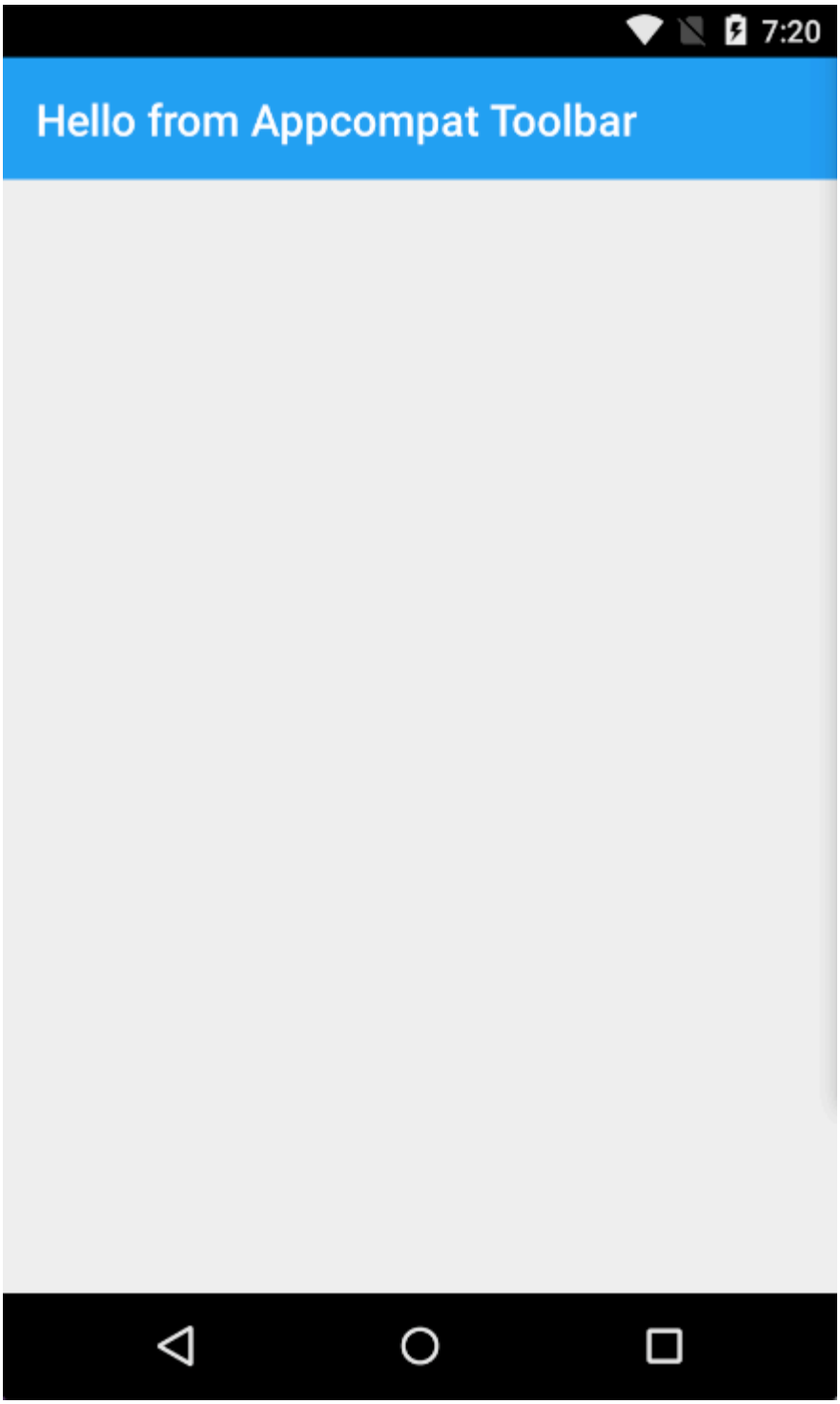
Der letzte Schritt ist das Verbinden der Symbolleiste in der Aktivitätsdatei. Öffnen Sie die Datei "MainActivity.cs". Sie müssen die Ableitung von "Aktivität" in "AppCompatActivity" ändern. Rufen Sie nun den Verweis auf die Symbolleiste ab und legen Sie sie als Standard-Symbolleiste für die Aktivität in der Methode "OnCreate" fest. Sie können auch den Titel definieren:

```
var toolbar = FindViewById<Android.Support.V7.Widget.Toolbar>(Resource.Id.toolbar);
        SetSupportActionBar(toolbar);
        SupportActionBar.Title = "Hello from Appcompat Toolbar";
```

Die ganze Methode sollte wie folgt aussehen:

```
protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.Main);

        var toolbar = FindViewById<Android.Support.V7.Widget.Toolbar>(Resource.Id.toolbar);
        SetSupportActionBar(toolbar);
        SupportActionBar.Title = "Hello from Appcompat Toolbar";
    }
```

Projekt neu erstellen und starten, um das Ergebnis zu sehen:



Xamarin.Android - So erstellen Sie eine Symbolleiste online lesen: https://riptutorial.com/de/xamarin-android/topic/4755/xamarin-android---so-erstellen-sie-eine-symbolleiste

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Xamarin.Android	Amy Burns, Community, Jon Douglas, Kevin Montrose, Ryan Weaver
2	App-Lebenszyklus - Xamarin.Andorid	CDrosos, Daniel Krzyczkowski, Steven Mark Ford
3	Barcode-Scannen mit der ZXing-Bibliothek in Xamarin-Anwendungen	GvSharma
4	Benutzerdefinierte ListView	user3814750
5	Bindungen	EJoshuaS, Jon Douglas, jonp, Matthew, Prashant C, Sven-Michael Stübe
6	Dialoge	JimBobBennett, Pilatus
7	RecyclerView	Alexandre, Matthew, Ryan Alford, Sreeraj, Zverev Eugene
8	So korrigieren Sie die Ausrichtung eines mit einem Android-Gerät aufgenommenen Bildes	Daniel Krzyczkowski
9	Toast	GONeale, Matthew, Piet, user2912553
10	Veröffentlichen Ihres Xamarin.Android APK	Alexandre
11	Xamarin.Android - Bluetooth-Kommunikation	Ladislav
12	Xamarin.Android - So erstellen Sie eine Symbolleiste	Daniel Krzyczkowski, tylerjgarland