無料電子ブック

学習

Xamarin.Android

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.an

droid

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

1: Xamarin.Androidをいめる

Xamarin.Androidをすると、のCのとさ、.NETベースクラスライブラリBCLのパワー、および2つのアプリケーションをいて、JavaとじUIコントロールをしてネイティブAndroidアプリケーションをできます。ファーストクラスのIDE - Xamarin StudioとVisual Studio - をにできます。

MacまたはWindowsマシンにXamarin.Androidをインストールするのについては、Xamarinデベロッパーセンターの「はじめに」ガイドをしてください。

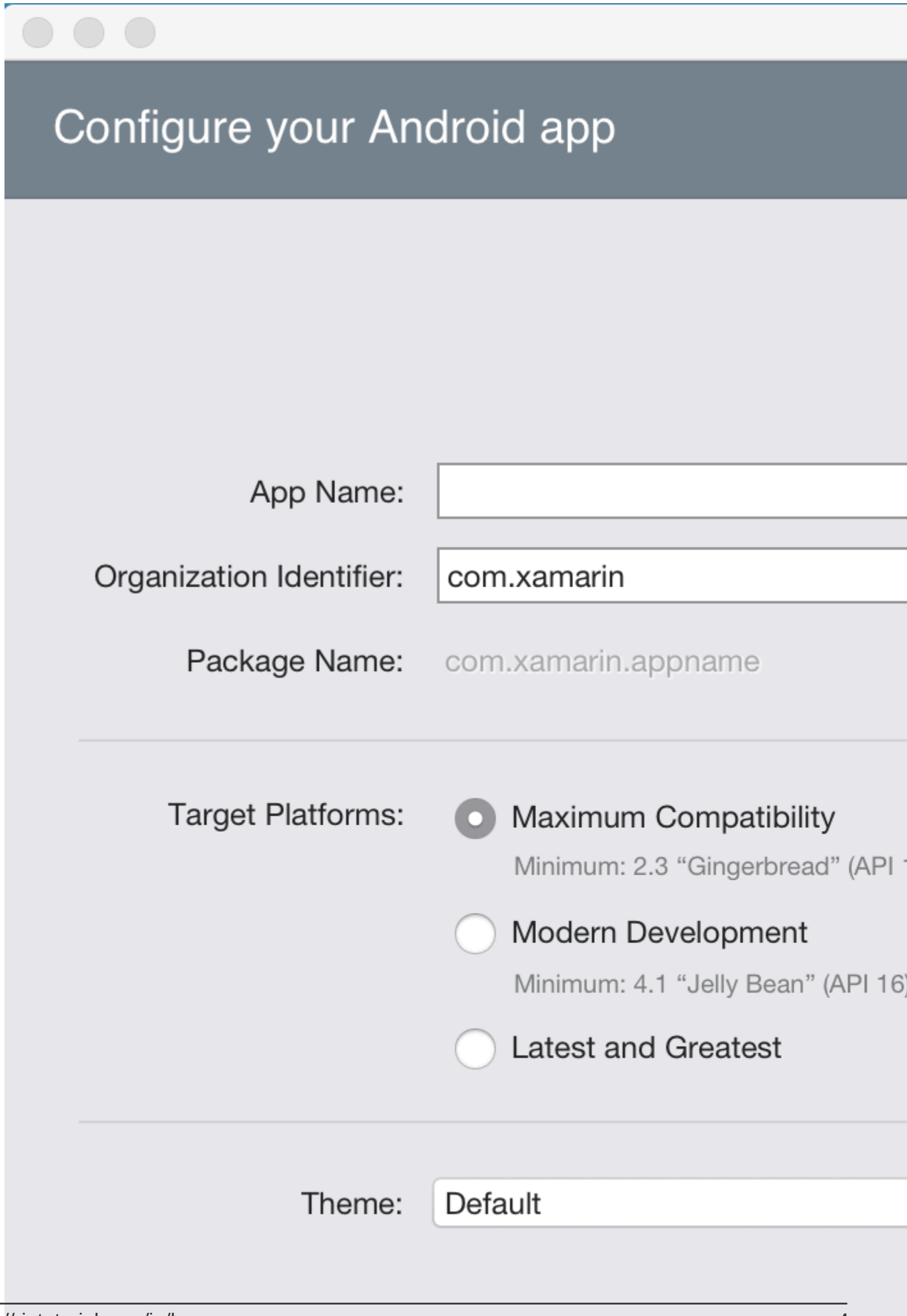
バージョン

バージョン	コード	APIレベル	
1.0	し	1	2008-09-23
1.1	し	2	2009-02-09
1.5	カップケーキ	3	2009-04-27
1.6	ドーナツ	4	2009-09-15
2.0-2.1	エクレア	5-7	2009-10-26
2.2-2.2.3	フロイヨ	8	2010-05-20
2.3-2.3.7	ジンジャーブレッド	9-10	2010-12-06
3.0-3.2.6	ハニカム	11-13	2011-02-22
4.0-4.0.4	アイスクリームサンドイッチ	14-15	20111018
4.1-4.3.1	ゼリービーン	16-18	2012-07-09
4.4-4.4.4,4.4W-4.4W.2	キットカット	19-20	20131031
5.0-5.1.1	ロリポップ	21-22	2014-11-12
6.0-6.0.1	マシュマロ	23	2015-10-05
7.0	ヌガー	24	2016-08-22

Examples

Xamarin Studioをいめる

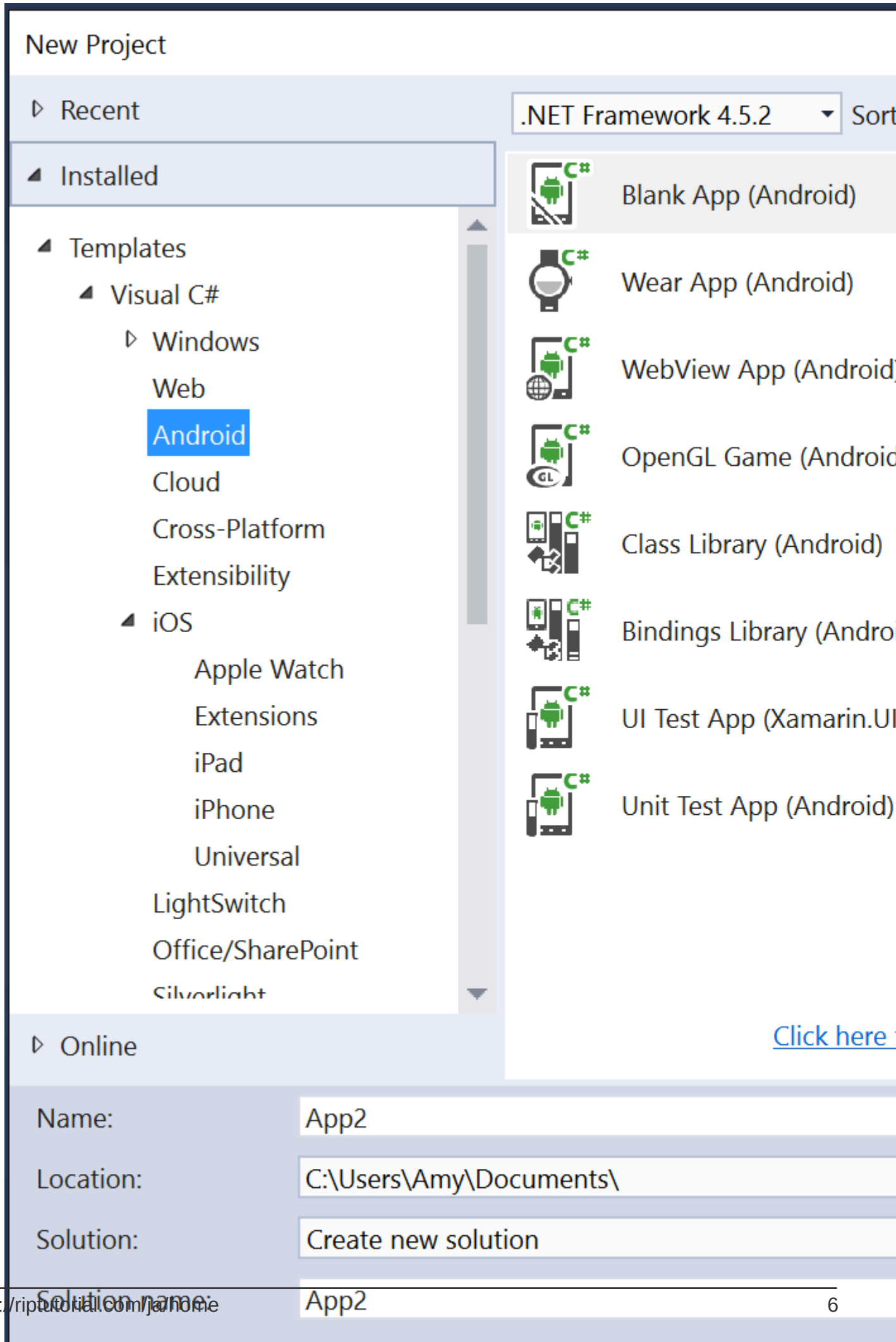
1. **[**ファイル**]> []> [**ソリューション**]**のにして、しいプロジェクトダィアログをします。
2. **Android App**をし、 へをします。
3. アプリとIDをしてアプリをします。ニーズにもしたターゲットプラットフォームをするか、デフォルトのままにしておきます。へ


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. プロジェクトとソリューションをするか、デフォルトのままにしておきます。 []をクリックしてプロジェクトをします。
   してプロジェクトをします。
5. デプロイメントにデバイスを するか、エミュレータをする
6. アプリケーションをするには、 デバッグをし、ボタンをします。

Visual Studioでする

1. 「 ファイル 」> 「 」> 「プロジェクト 」をして、「プロジェクト 」ダイアログをします。
2. **Visual C> Android**にし 、のアプリケーションをします。


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
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)
Online
Click here
Name:
App2
Location:
C:\Users\Amy\Documents\
Solution:
Create new solution
Solution name:
App2

3. あなたのアプリケーションにをけ 、 **OK**をしてプロジェクトをします。
4. デプロイメントにデバイスを するか、エミュレータをする
5. アプリケーションをするには、[デバッグ]をし、[ ]ボタンをします。

オンラインでXamarin.Androidをいめるをむ https://riptutorial.com/ja/xamarin-android/topic/403/xamarin-androidをいめる

2: Androidデバイスからキャプチャしたのきをする

1. このアプリサンプルはのGitHubでできます

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

2. Xamarin Mobileコンポーネントのドキュメントはのとおりです。

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

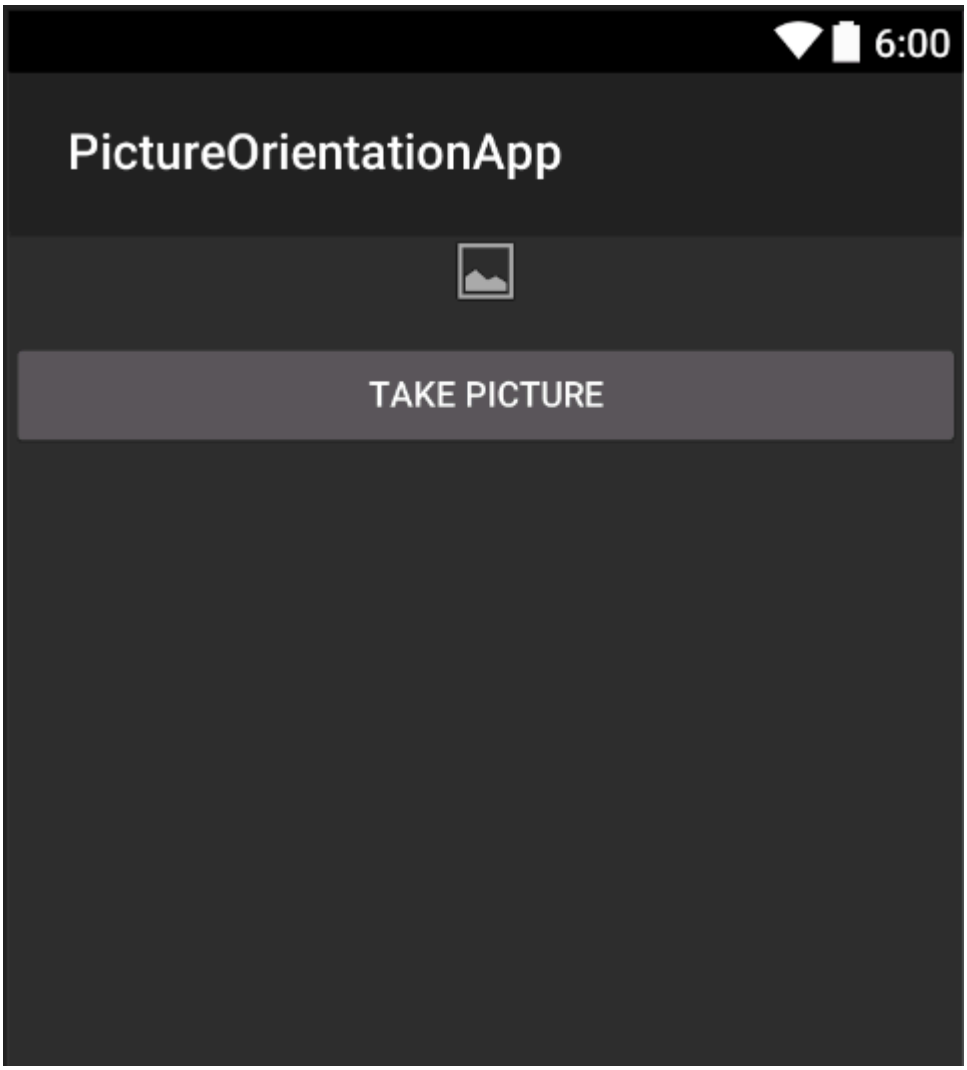
Examples

Androidデバイスからキャプチャしたのきをする

このでは、Androidデバイスでイメージをしてしくするをします。

まず、1つのボタンと1つのビューでサンプルアプリケーションをするがあります。ユーザーがボタンをクリックすると、カメラがし、ユーザーがをすると、になきでされます。

1. "TakePictureButton"というのボタンと "TakenPictureImageView"というのビューをします。

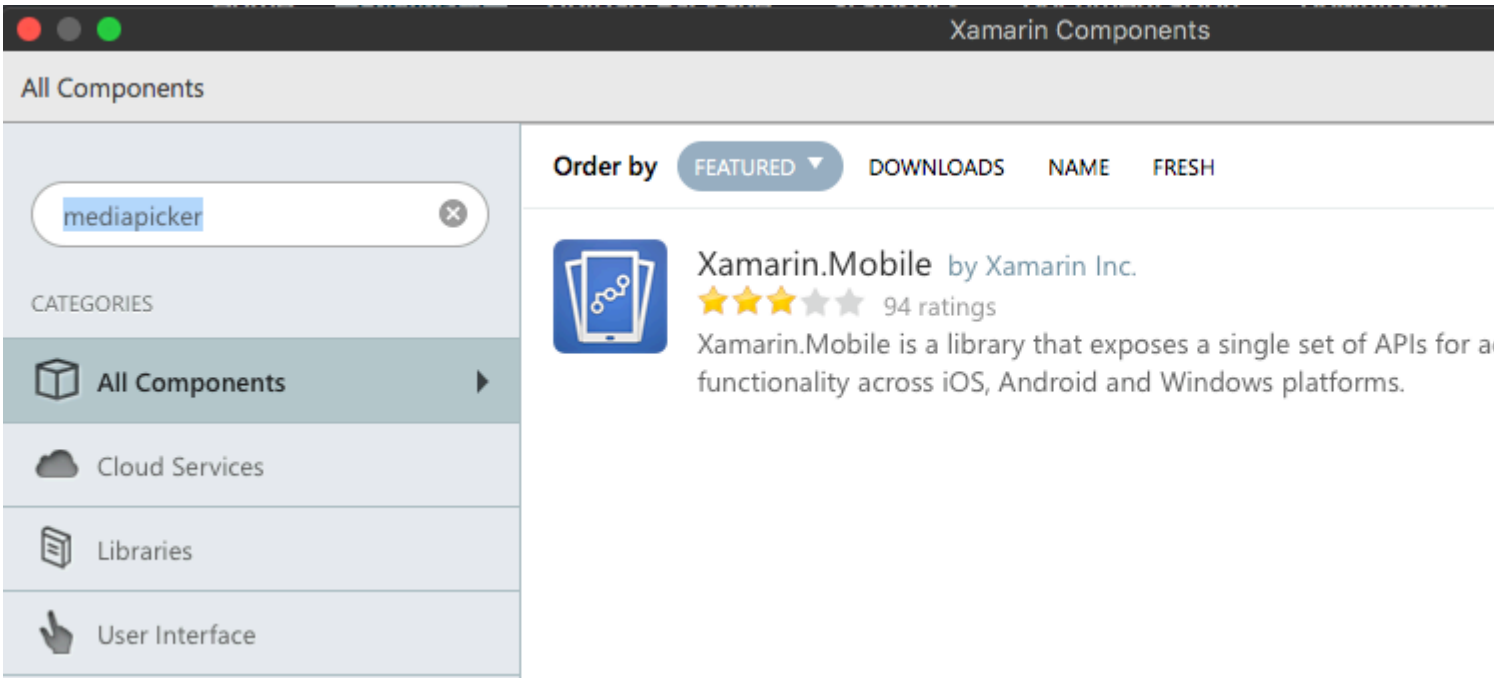


2. はアクティビティコーードをいてください

まず、あなたのコントロールへのをします。

```
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. アプリケーションでは、コンポーネントストアでできるXamarin Mobileコンポーネントをします。



4. それをプロジェクトにすると、たちはむことができます。カメラをするコードをにしてください。このメソッドは、のコードでられるように、ボタンのクリックでびされるがあります

```
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. ユーザーがをすると、なにするがあります。これをうにはのをします。これは、されたからexifをすることののけをむと、なきのビットマップをすることとをする。

```
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++;
     }
```

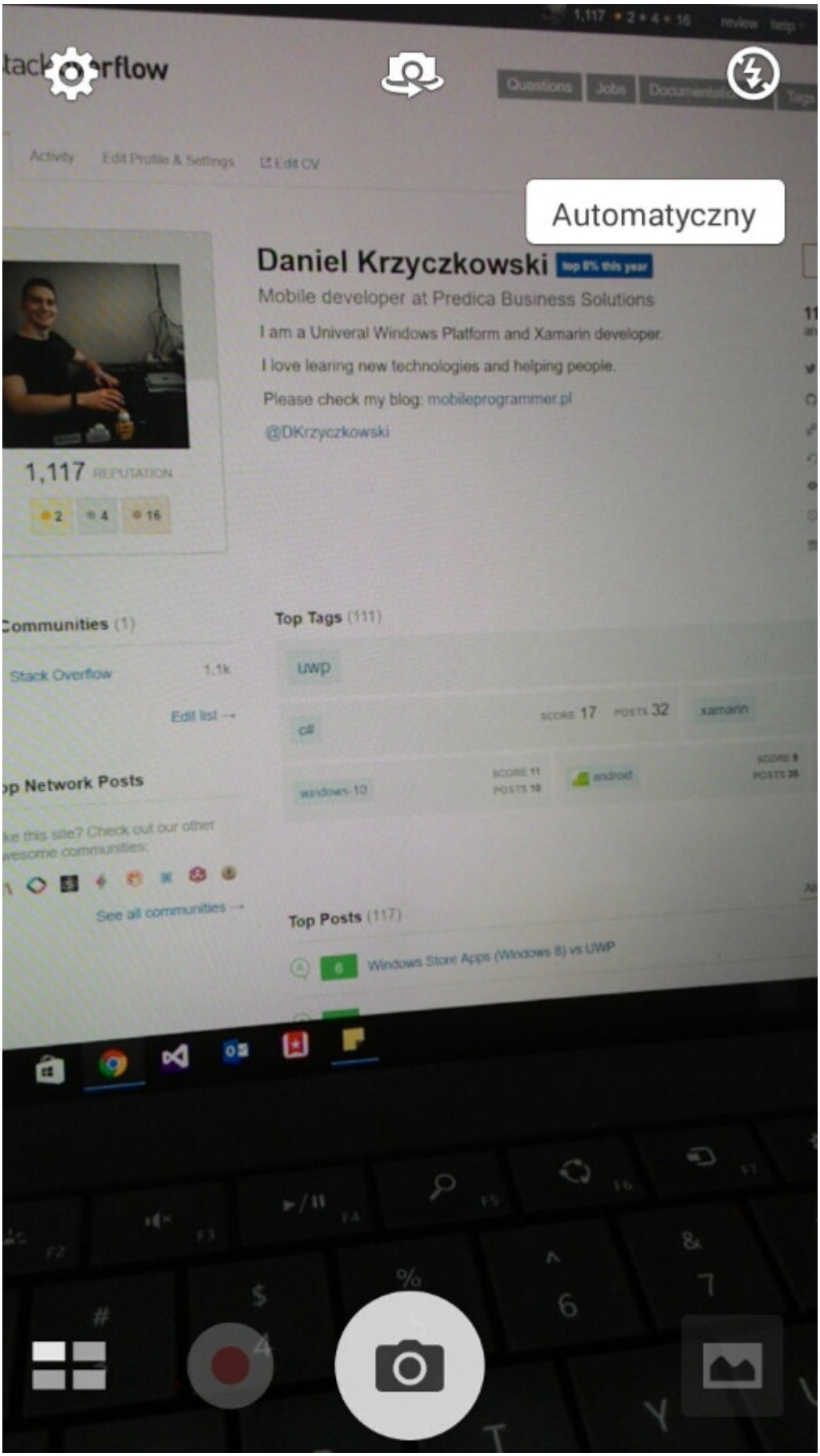
```
    options.InSampleSize = scale;
    options.InJustDecodeBounds = false;
    Bitmap resizedBitmap = BitmapFactory.DecodeFile(filePath, options);

    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;
    }
```

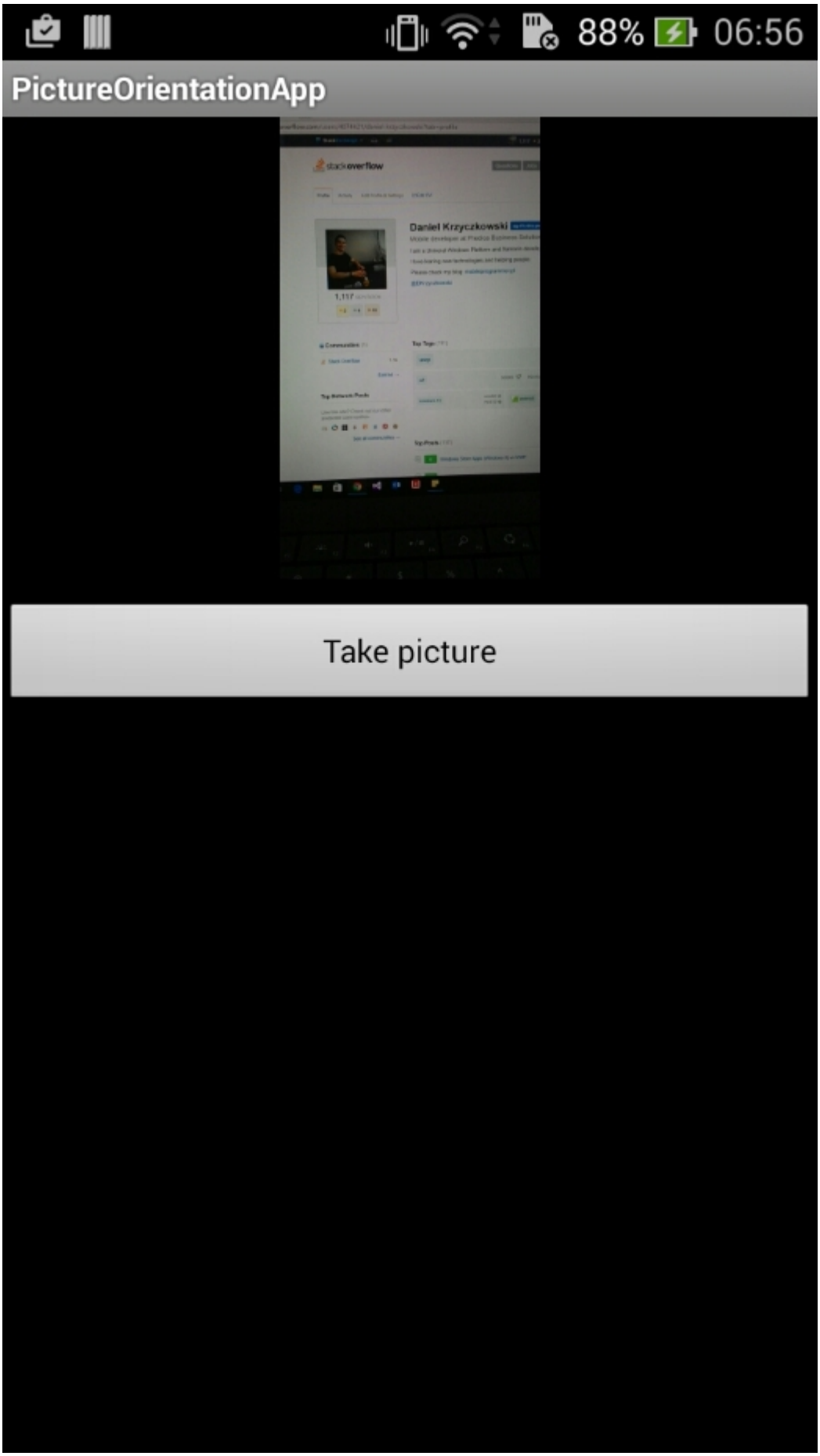
```
     catch (IOException ex)
     {
         Console.WriteLine("An exception was thrown when reading exif from media
file...:" + ex.Message);
         return null;
     }
 }
```

6. のメソッドは、ユーザーがをったにびされるOnActivityResultメソッドでびされます。

```
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. アプリケーションをします。をってをる


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
windows-10
Top Posts (117)
Windows Store Apps (Windows 8) vs UWP



それでおしまい。はあなたがったをすべてしいにさせます。

オンラインでAndroidデバイスからキャプチャしたのきをするをむ
https://riptutorial.com/ja/xamarin-android/topic/6683/androidデバイスからキャプチャしたのきをする

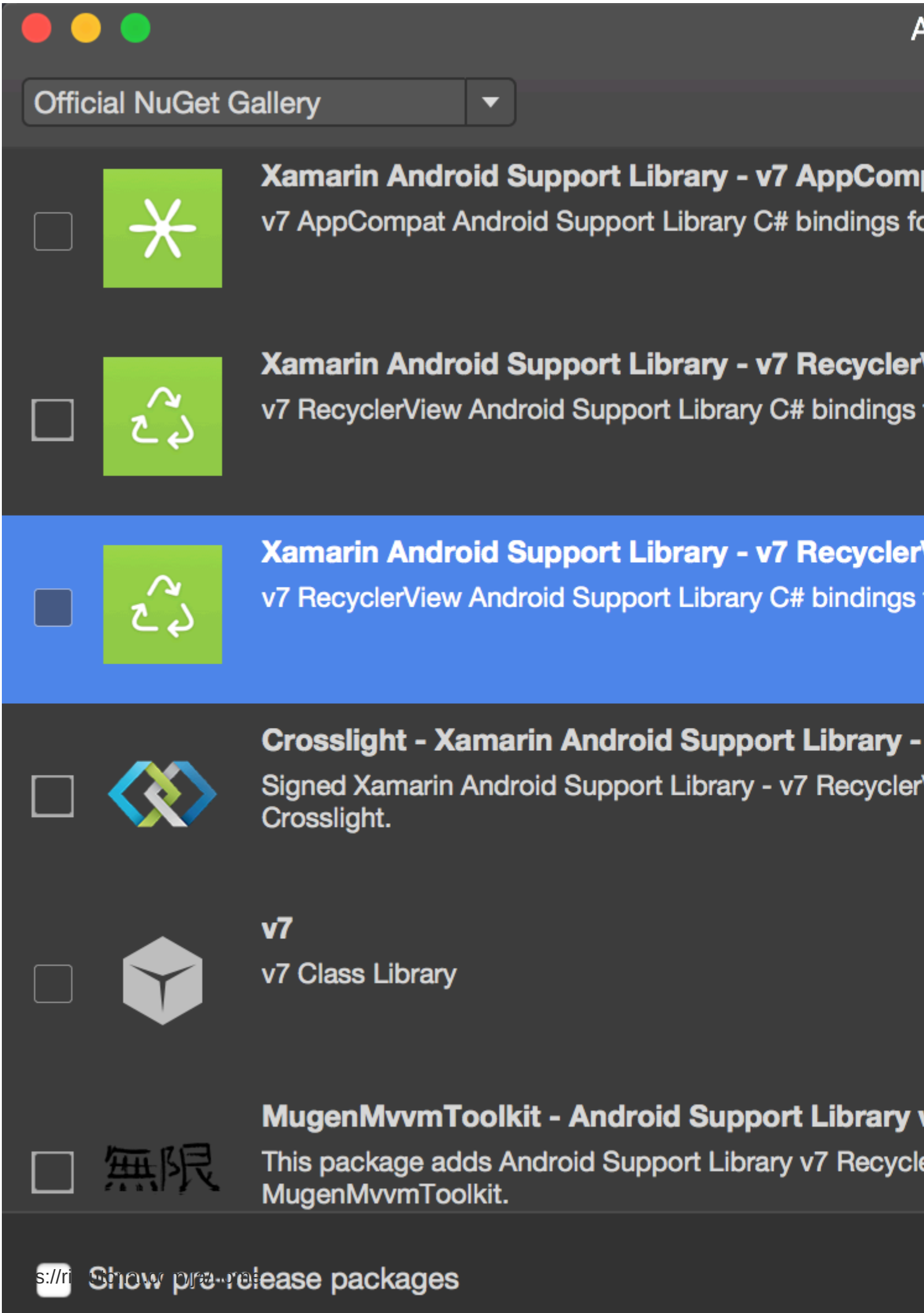
3: RecyclerView

Examples

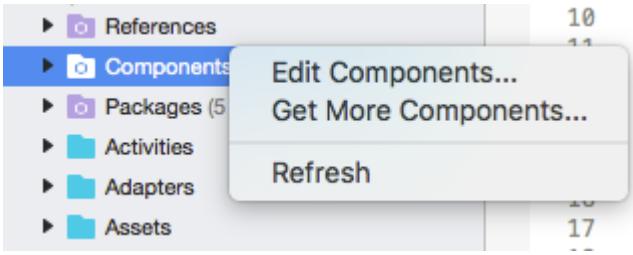
RecyclerViewの

これは、 `Android Support Library V7 RecyclerView`です。 サポートライブラリは、しいののあるバージョンをし、フレームワークにまれていないなUIをし、アプリケーションがきすことができるさまざまなユーティリティをするため、にされます。

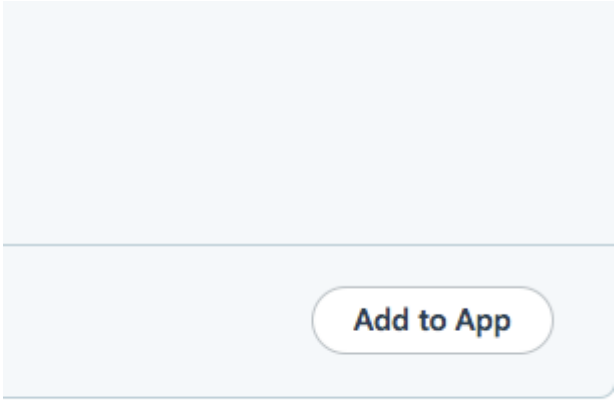
`RecyclerView`をするには、なNugetパッケージをインストールします。まず、 `v7 recyclerview`をします。 `Xamarin Android Support Library - v7 RecyclerView`されるまでスクロールダウンします。それをし、[パッケージの] をクリックします。


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

コンポーネントをするには、ソリューションエクスプローラでAndroidプロジェクトの`Components`クリックし、[`Get More Components`を`Get More Components`]をクリックし`Get More Components`。



されるComponent Storeウィンドウで、RecyclerViewをします。リストで、 `Android Support Library V7 RecyclerView`します。に、[`Add to App`に`Add to App`]をクリックします。コンポーネントがプロジェクトにされます。



のステップはRecyclerViewをページにすることです。 `axml` レイアウトファイルで、 `RecyclerView` をのようにすることができます。

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

RecyclerViewには、 `Adapter`と`ViewHolder`なのために、なくとも2つのヘルパークラスがされているがあり`ViewHolder`。 `Adapter`アイテムレイアウトをさせ、RecyclerViewにされるビューにデータをバインドします。 ViewHolderはビューをしてします。ビューホルダーは、アイテムビューのクリックをするのにもちます。

に、アダプタクラスのなをします

```
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;
        }
    }
}
```

`OnCreateViewHolder`メソッドでは、にViewをし、ViewHolderクラスのインスタンスをします。このインスタンスをすがあります。このメソッドは、ViewHolderのしいインスタンスがなときにアダプタによってびされます。このメソッドは、のセルごとにびされることはありません。RecyclerViewにビュ―をたすのになセルがあれば、Viewからスクロ―ルされているいセルをして、それのセルをします。

`OnBindViewHolder`コ―ルバックは、されたにデ―タをするために、Adapterによってびされます。このメソッドは、itemViewのをして、されたのをするがあります。

セルには`TextView`が1つしかまれていないため、のようなViewHolderをつことができます。

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


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

のステップは、 `Activity`のものをすることです。

```
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;
}
```

LayoutManagerクラスは、RecyclerViewのアイテムビュ―のとけ、およびユ―ザ―がえなくなったアイテムビュ―のリサイクルにするポリシ―のをします。 `RecyclerView`に、 `ListView`をしてにスクロ―ルするリストのようにセルをし、 `GridView`をしてアイテムを2のスクロ―ルなグリッドにするがありました。しかし、は、のLayoutMangerをすることで、RecyclerViewでをすることができます。 `LinearLayoutManager`はListViewのようにセルをし、 `GridLayoutManager`はセルのグリッドをします。

RecyclerView with Click イベント

このは、Xamarin.Android RecyclerViewでClick EventHandlerをするをしています。

Android Javaでは、Clickのリスナ―をするは、クリックされるビュ―にしてonClickListenerをしています。

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

ただし、Xamarin.Androidでは 、Clickイベントのリスナ―をするは、のでEventHandlerをすることです。

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
}
```

EventHandlerはされており、されていないことにしてください。 Click EventHandlerがGridView / ListViewアダプタのGetViewメソッド、またはRecyclerView.AdapterのOnBindViewHolderメソッドにされた、アイテムビュ―がされるたびにしいEventHandlerがされます。スクロ―ルすると、のEventHandlerがされ、ビュ―がクリックされるとすべてのイベントがします。

このをするには、EventHandlerのをし、GetViewまたはOnBindViewHolderメソッドできサブスクライブするがあります。また、EventHandlerをするには**3**のをするがあります。そうしないと、EventHandlerのをできません。

ClickイベントをむRecyclerView.Adapterのをにします。

```
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);
    }

}
```

オンラインでRecyclerViewをむ https://riptutorial.com/ja/xamarin-android/topic/3452/recyclerview

4: Xamarin.Android - Bluetooth

き

Xamarin.Androidで**BluetoothSocket.InputStream**と**BluetoothSocket.OutputStream**プロパティは、デザインによって**System.IO.Stream**にされます。いわゆるプロトコルの、サーバーがクライアントとするときだけサーバーがすると、をみるになバイトをするメソッドまたはプロパティーがないため、System.IO.Streamはではありません。

パラメーター

パラメータ	
ソケット	BluetoothSocketオブジェクトのインスタンスです。このメソッドをびすにソケットをくがあります。
cmd	BTデバイスにするバイトとしてのコマンド。
_mx	このメソッドはハードウェアリソースをするので、のワーカースレッドからびすほうがよいです。このパラメータは、System.Threading.Mutexオブジェクトのインスタンスであり、このメソッドをにびすのスレッドとスレッドをさせるためにされます。
タイムアウト	きみとみりののミリ。

Examples

ソケットをして**Bluetooth**デバイスとのでデータのをう

のでは、Android.Runtime.InputStreamInvokerとAndroid.Runtime.OutputStreamInvokerタイプがするjava.io.InputStreamとにjava.io.OutputStreamを 。たちがする**java.io.InputStream**インスタンスをっていたら、たちは**.Read**をすることができますなレスポンスのバイトをするためにその**.Available**メソッドをすることができます。

```
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;
}
```

オンラインでXamarin.Android - Bluetoothをむ https://riptutorial.com/ja/xamarin-android/topic/10844/xamarin-android-----bluetooth

5: Xamarin.Android - ツールバーをする

なるチーム、

は、ツールバーのコントロールがにされているのAndroidのドキュメンテーションについてするのはいことだとう

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

サンプルにされているAndroid.Support.v7ライブラリにするいコンテンツもあります

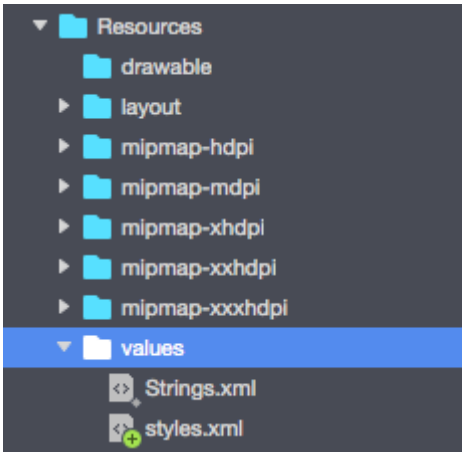
https://developer.android.com/training/appbar/index.html

Examples

Xamarin.Androidアプリケーションにツールバーをする

まず、NuGetのXamarin.Android.Support.V7.AppCompatライブラリをするがありま す。https://www.nuget.org/packages/Xamarin.Android.Support.v7.AppCompat/

"resources"のの "values"フォルダに、 "styles.xml"というしいxmlファイルをします。

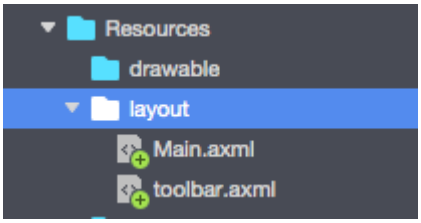


"styles.xml"ファイルにはのコードがまれているがあります

```
<?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. -->
```

のステップは、ツーールバーのコントローールをむ "toolbar.axml"ファイルを "layout"フォルダにすることです。



ツーールバーをするコードをにしてください

```
<?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" />
```

"Main.axml"ファイルをき、のレイアウトのタグのすぐにのコードをしてください。あなたのコードはのようになります

```
<?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>
```

これで、あなたのアプリがうテーマにするをするがあります。 "AndroidManifest"ファイルをき、テーマを "application"タグにします

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

のステップは、ツーールバーをアクティビティファイルにすることです。 "MainActivity.cs"ファイ

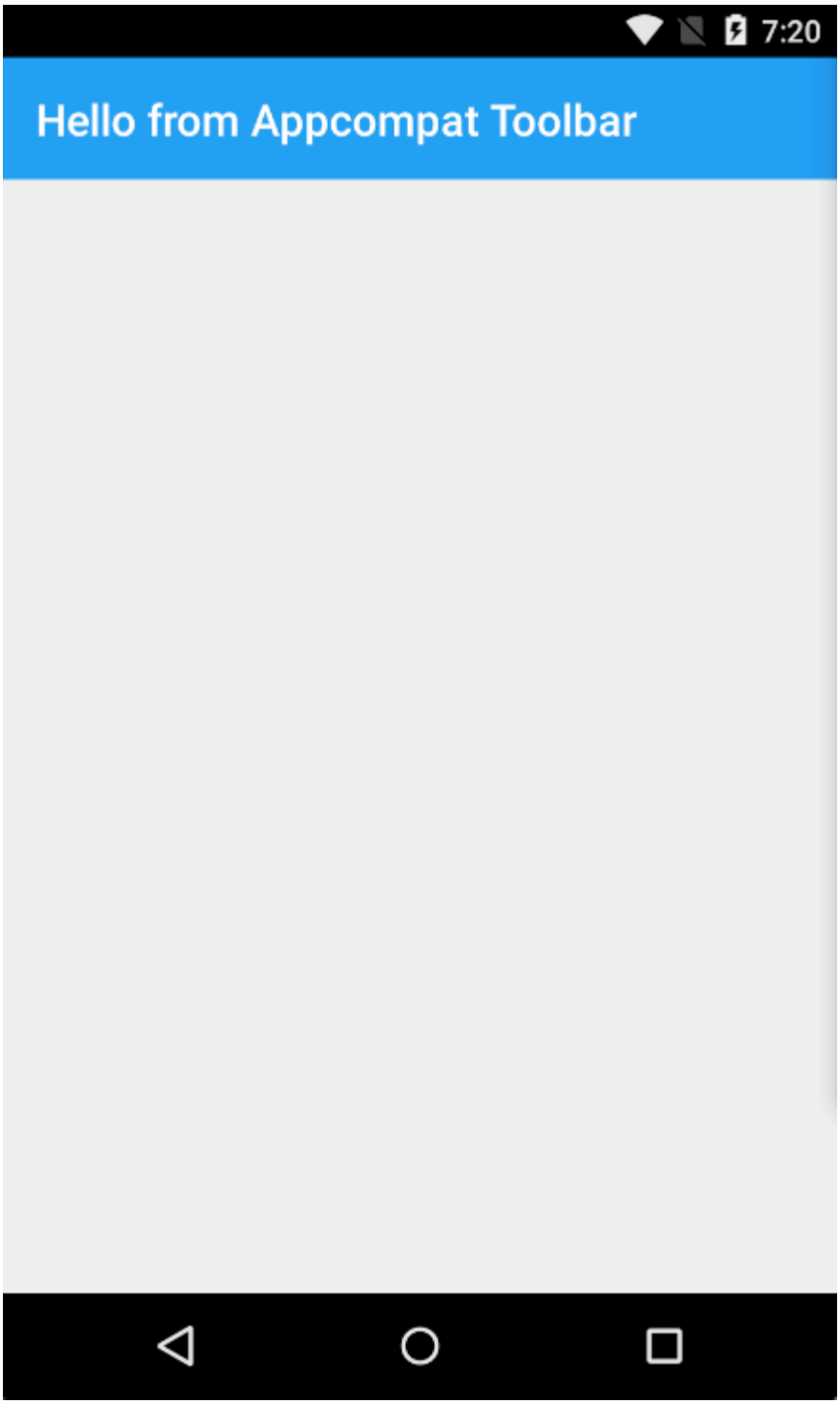
ルをきます。を「アクティビティ」から「AppCompatActivity」にするがあります。ツ—ルバ—へのをし、 "OnCreate"メソッドのアクティビティのデフォルトツ—ルバ—としてします。タイトルをすることもできます

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

のはのようになります

```
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";
    }
```

プロジェクトをビルドし、それをしてをする



オンラインでXamarin.Android - ツールバーをするをむ https://riptutorial.com/ja/xamarin-android/topic/4755/xamarin-android----ツールバーをする

6: Xamarin.Android APKをする

き

このトピックでは、Xamarin.Androidアプリのリリースモードのとについてします。

Examples

Visual StudioでのAPKの

あなたはあなたのアプリをし、デバッグモードでテストし、にしています。これで、Google Play ストアでするをえます。

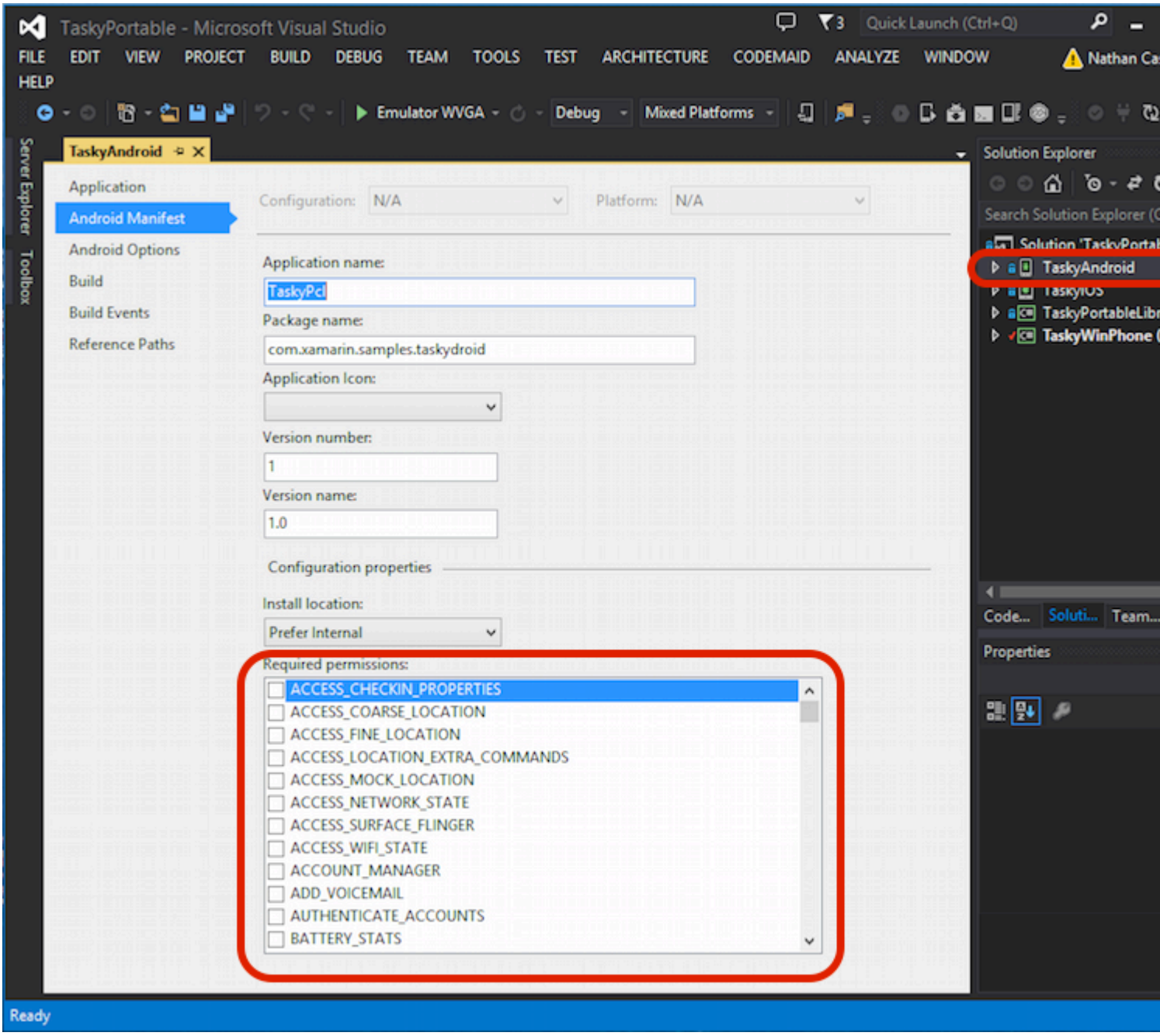
Xamarinのドキュメントは、ここにいをしています

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

---

Androidマニフェスト

まず、Visual Studioで、ソリューションエクスプローラでXamarin.Androidプロジェクトをクリックし、[プロパティ]をします。に、Android Manifestタブにしてこのをします



Android StudioやEclipseとはなり、AndroidManifest.xmlファイルをするはありません。 XamarinとVisual Studioはこれをいます。アクティビティ、ブロードキャスト、サービスは、クラスののをすることによって、Androidマニフェストにされます 。

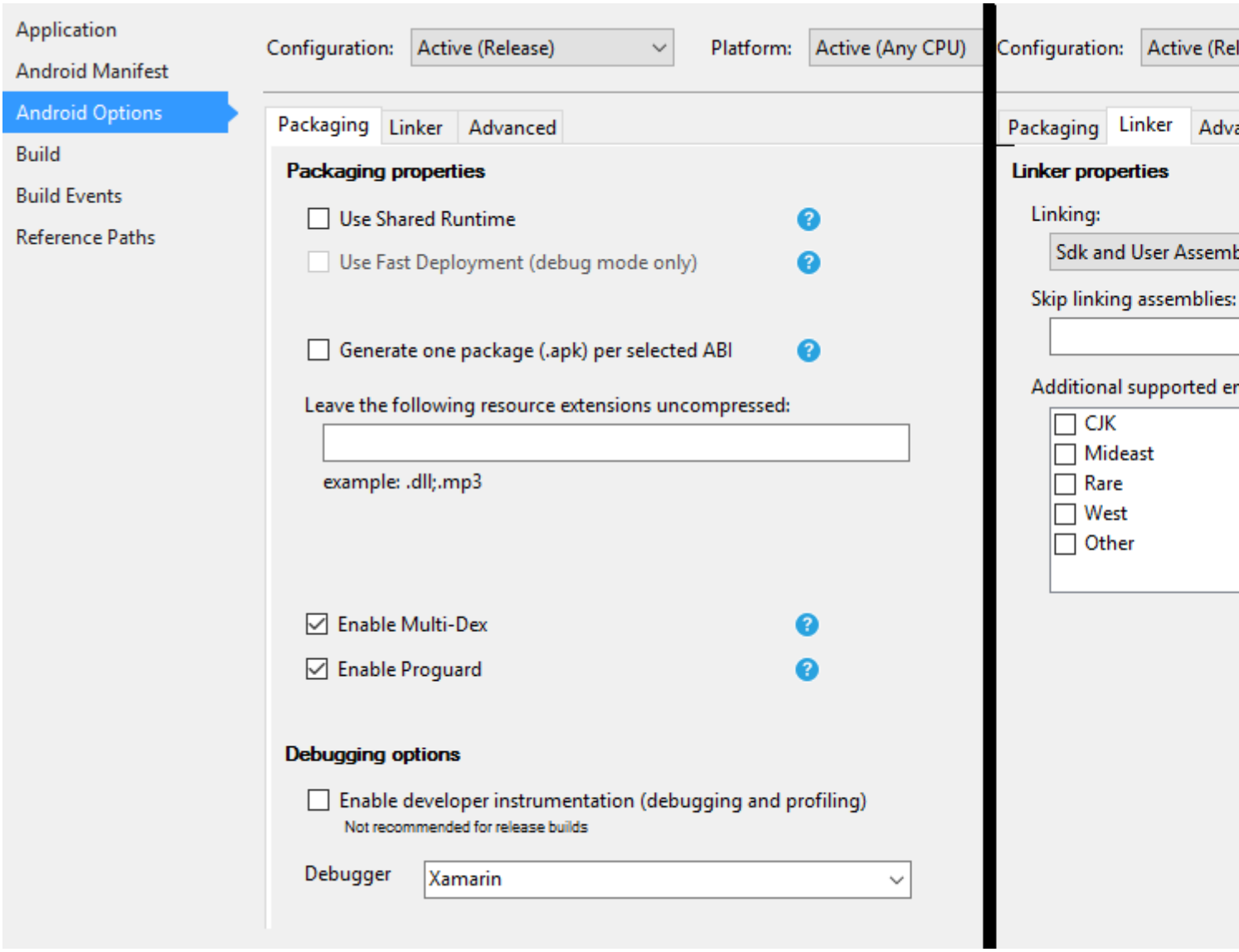
このでは、のオプションがあります。

- アプリケーション これはユーザーにされるアプリです。
- パッケージ これはパッケージです。これはであるがあります。つまり、Google Playストアののアプリとじパッケージをしてはいけません。
- アプリケーションアイコン これは、Androidスタジオまたはeclipseプロジェクトでされる@drawable / ic_launcherとの、ユーザーにされるアイコンです。
- バージョン バージョンはGoogle Playでバージョンにされます。アップデートされたバージョンのAPKをするは、しいアップグレードごとにこのに1をするがあります。
- バージョン これはユーザーにされるバージョンです。

- インストール APKのインストールをデバイスストレージまたはSDカードにします。
- な ここでは、あなたのアプリケーションになをします。

Androidオプション

のでは、コンパイラオプションをできます。ここでなオプションをすると、APKサイズをにし、エラーをぐことができます。



- アクティブリース 。
- プラットフォーム アクティブの**CPU** 。これらはGoogle Playストアけにapkをするためにです。が[デバッグ]にされているは、Google Playでけけられません。
- ランタイムをする **false** 。これをtrueにすると、APKはMono Runtimeをしてします。Monoランタイムは、USBでデバッグするとにインストールされますが、リリースAPKではインストールされません。 Mono Runtimeがデバイスにインストールされておらず、リリースAPKでこのオプションがtrueにされていると、アプリがクラッシュします。
- した**ABI**あたりつのパッケージの**.apk**をします。のから、なりくのプラットフォームにAPKをしてください。

- **Enable Multi-Dex** **true**ですが、アプリがあまりでないつまり、65536のメソッドがあるは、ここを 、falseにできます。
- **ProGuard**のにします 。これにより、アプリのJavaコードをするProguardツールがになります。 .NETコードにはされません。 .NETコードをするは、 Dotfuscatorをするがあります 。 Proguard for Xamarin.Androidのはこちらをください 。
- デバッグとプロファイリングをにする APKをリリースするは**false** 。
- リンク **SDK**とユーザーアセンブリこれにより、Xamarin LinkerはSDKおよびコードからのクラスをすべてし、APKサイズをします。

Xamarin.Linkerは、にプロジェクトのコアPCLライブラリにある、コードでされていないようなクラスをすることがあります。これをけるには、リンクを「Sdkアセンブリのみ」にするか、クラスのPreserveをします

PreserveAttribute.cs

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

クラス

```
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;
        }
    }
}
```

- サポートされているアーキテクチャ のから、 すべてをします 。

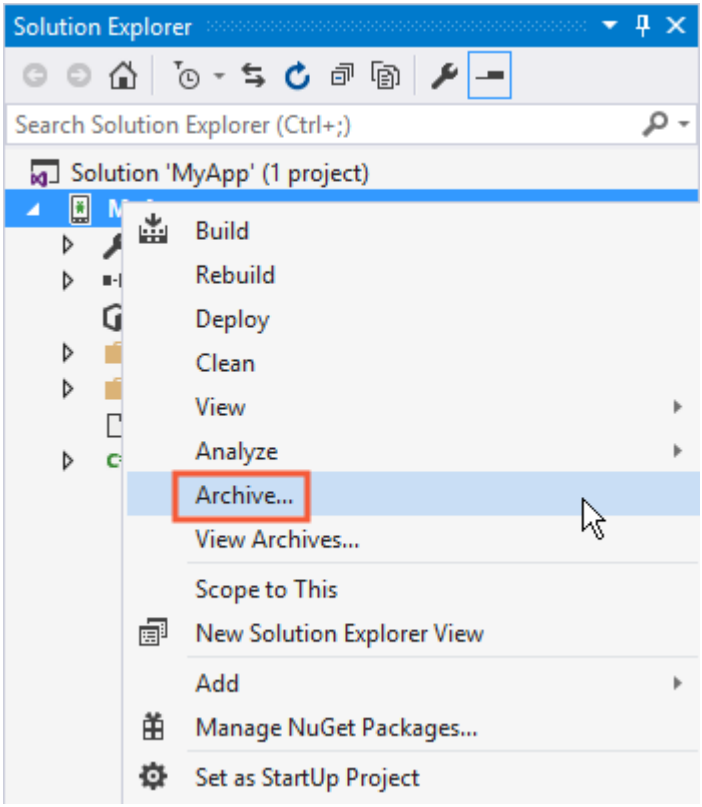
すべてをしたら、プロジェクトをビルドして、ビルドがにわれたことをします。

リリースモードでの**APK**の

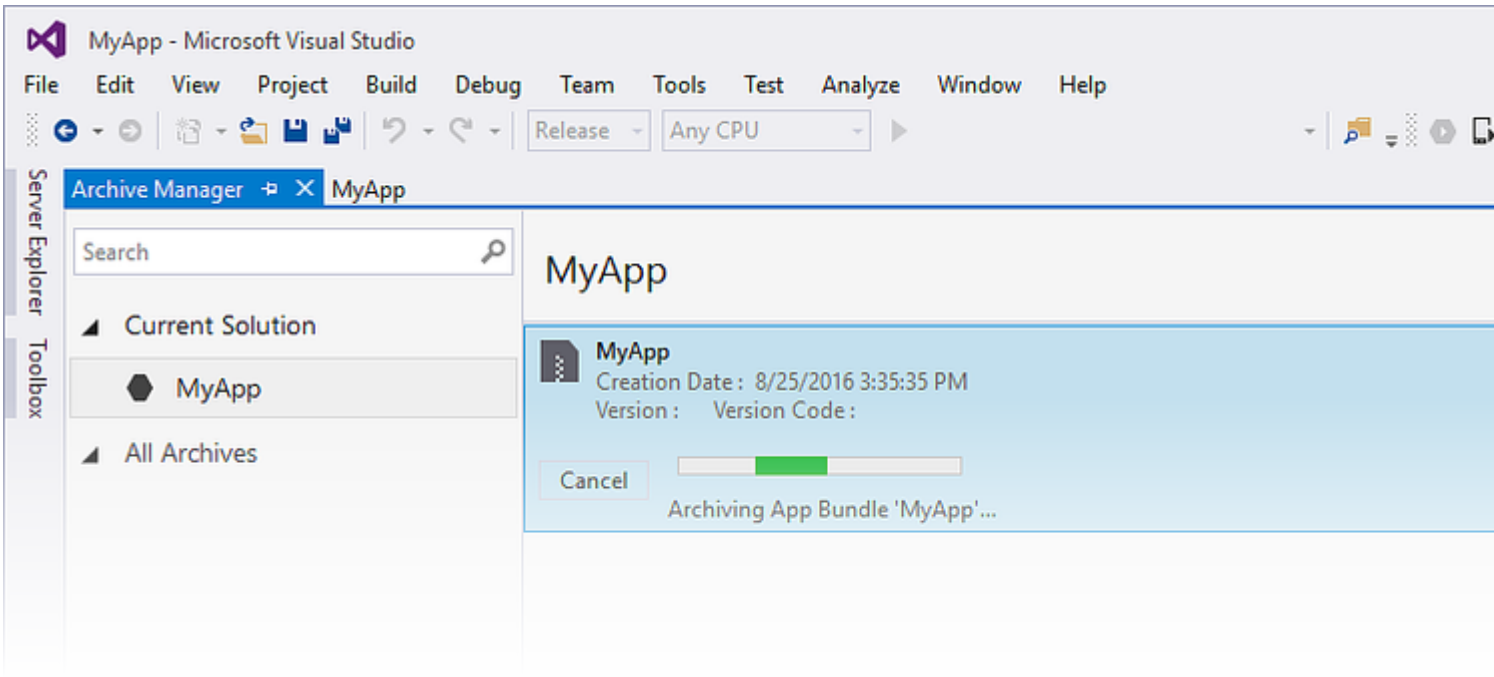
リリースのAndroidプロジェクトのがしました。のチュートリアルでは、Visual StudioでAPKをするをします。 Xamarinのドキュメンテーションのなチュートリアルはここにあります

https/ / / / /

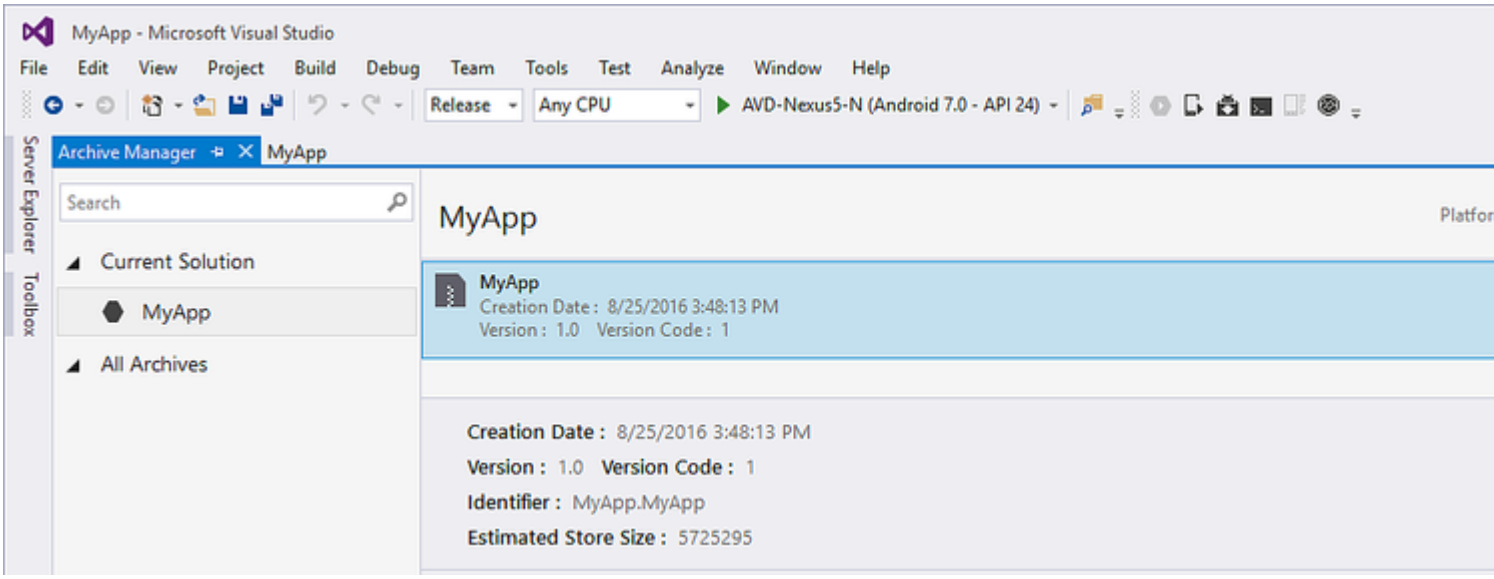
APKファイルをするには、ソリューションエクスプローラでXamarin.Androidプロジェクトをクリックし、[アーカイブ...]をします。



これにより、アーカイブマネージャがき、プロジェクトのアーカイブがされ、APKファイルのがされます。

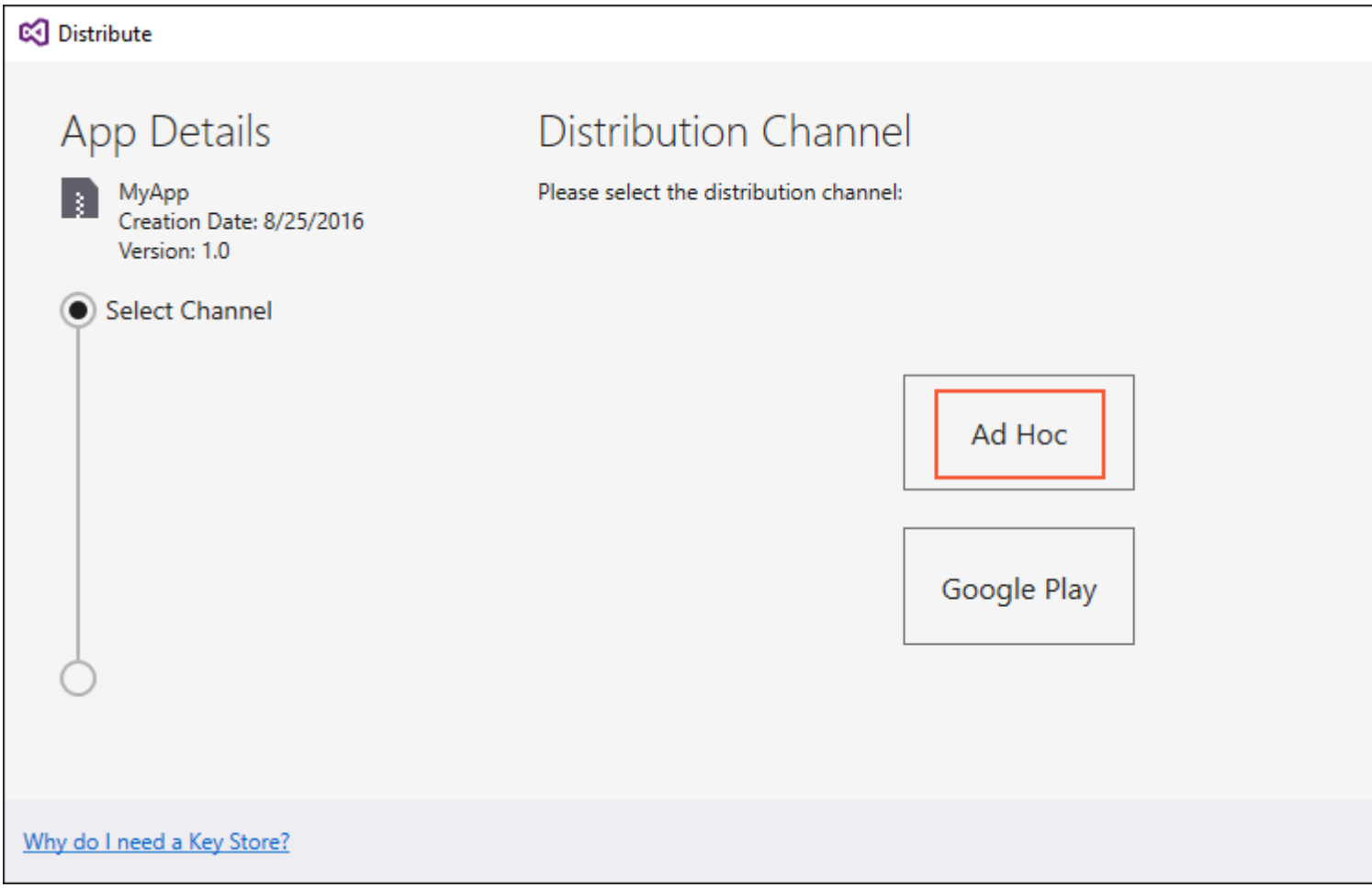


プロジェクトのアーカイブがしたら、[...]をクリックしてします。

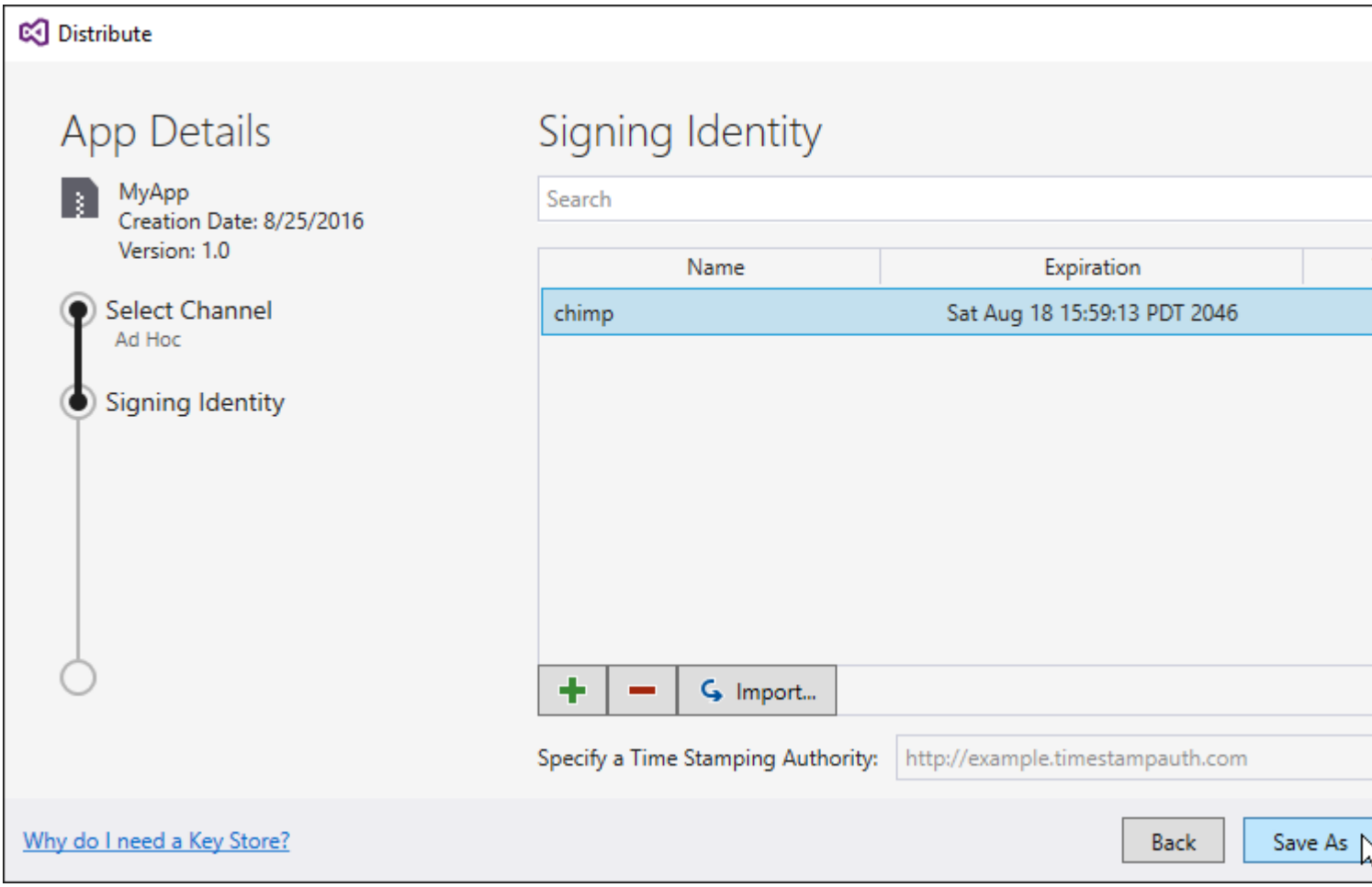


[]には、Ad-hocとGoogle Playという2つのオプションがされます。にAPKがされ、コンピュータにされます。 2はGoogle Playでアプリをします。

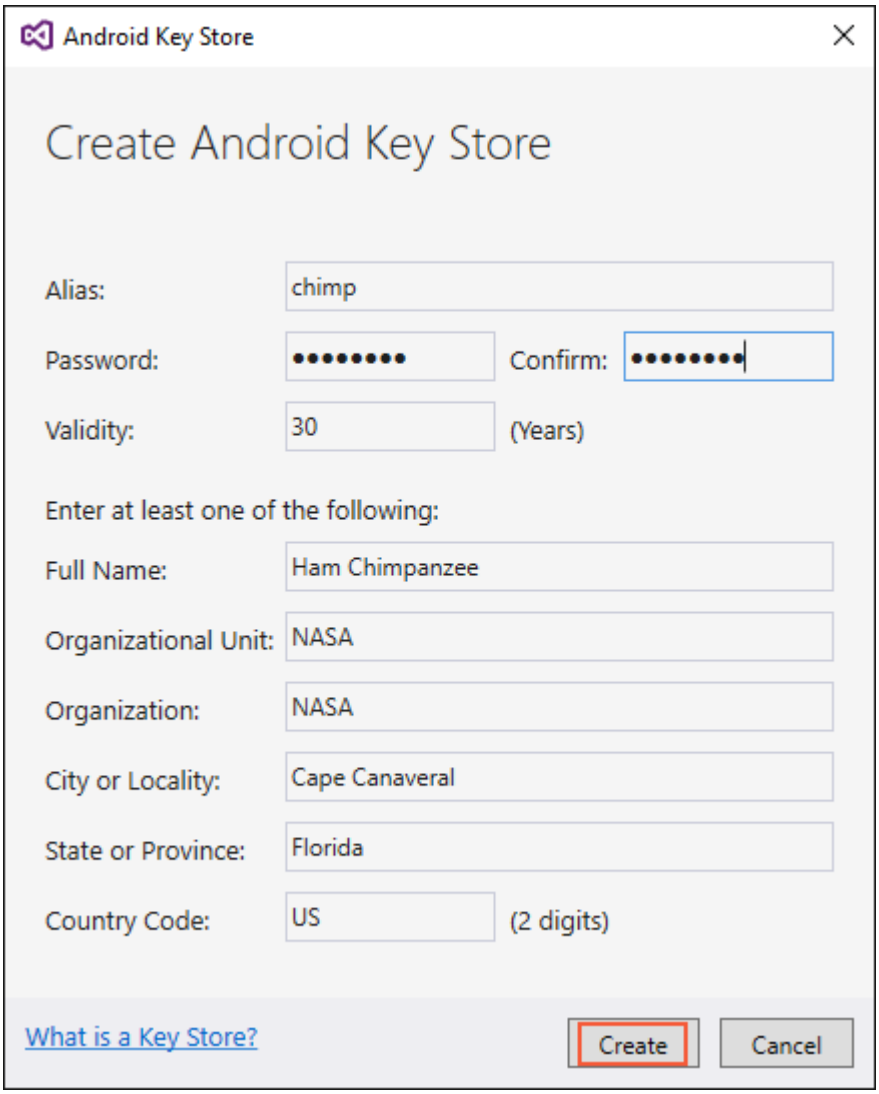
のものをすることをおめします。にじてのデバイスでAPKをテストできます。



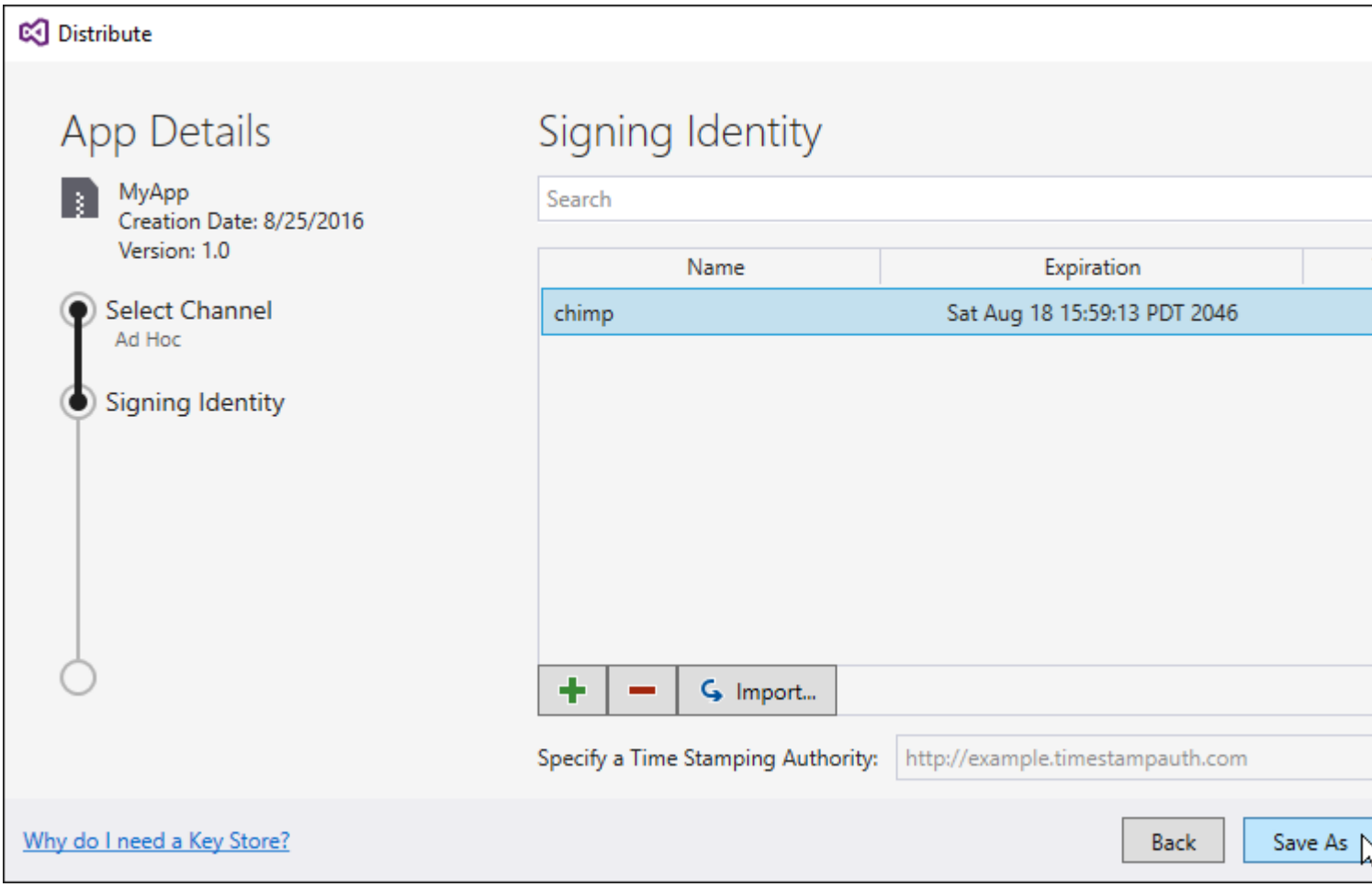
のでは、APKにするためにAndroid Key Storeがです。にっているは、[インポート...]をクリックしてできます。そうでないは、+をクリックしてしいAndroid Key Storeをすることができます。

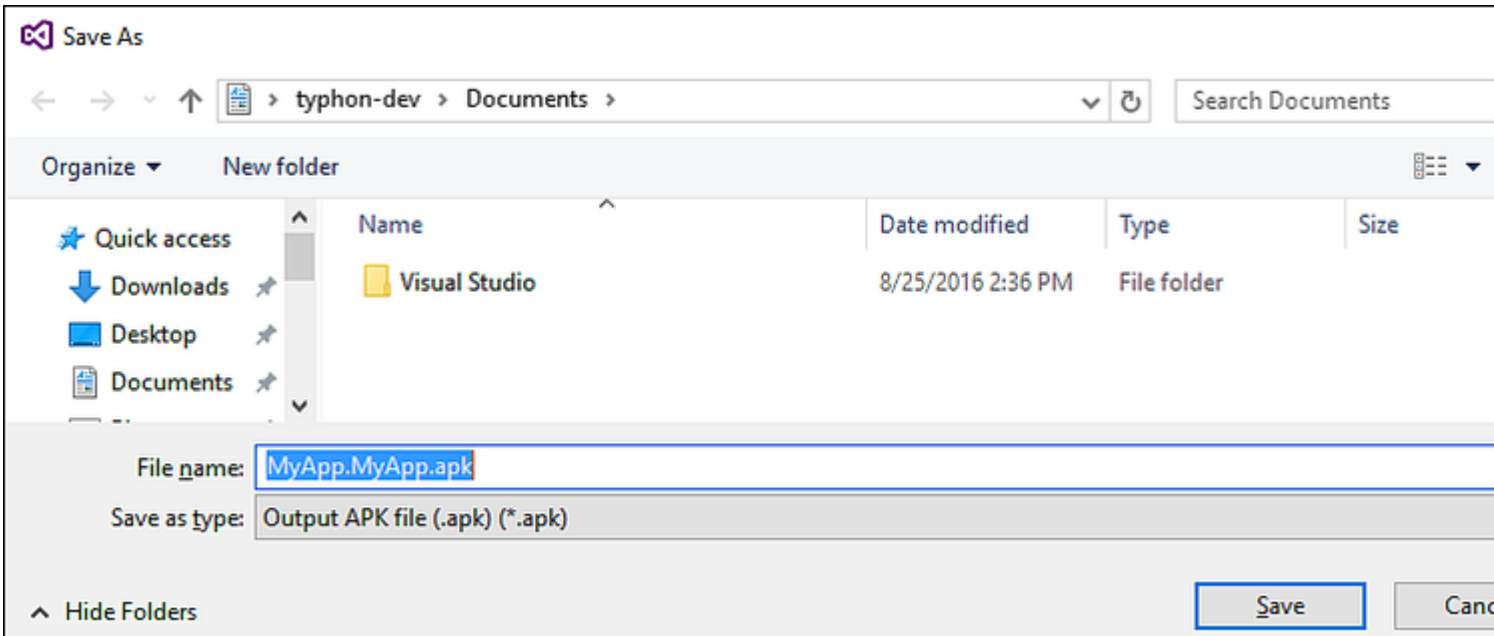


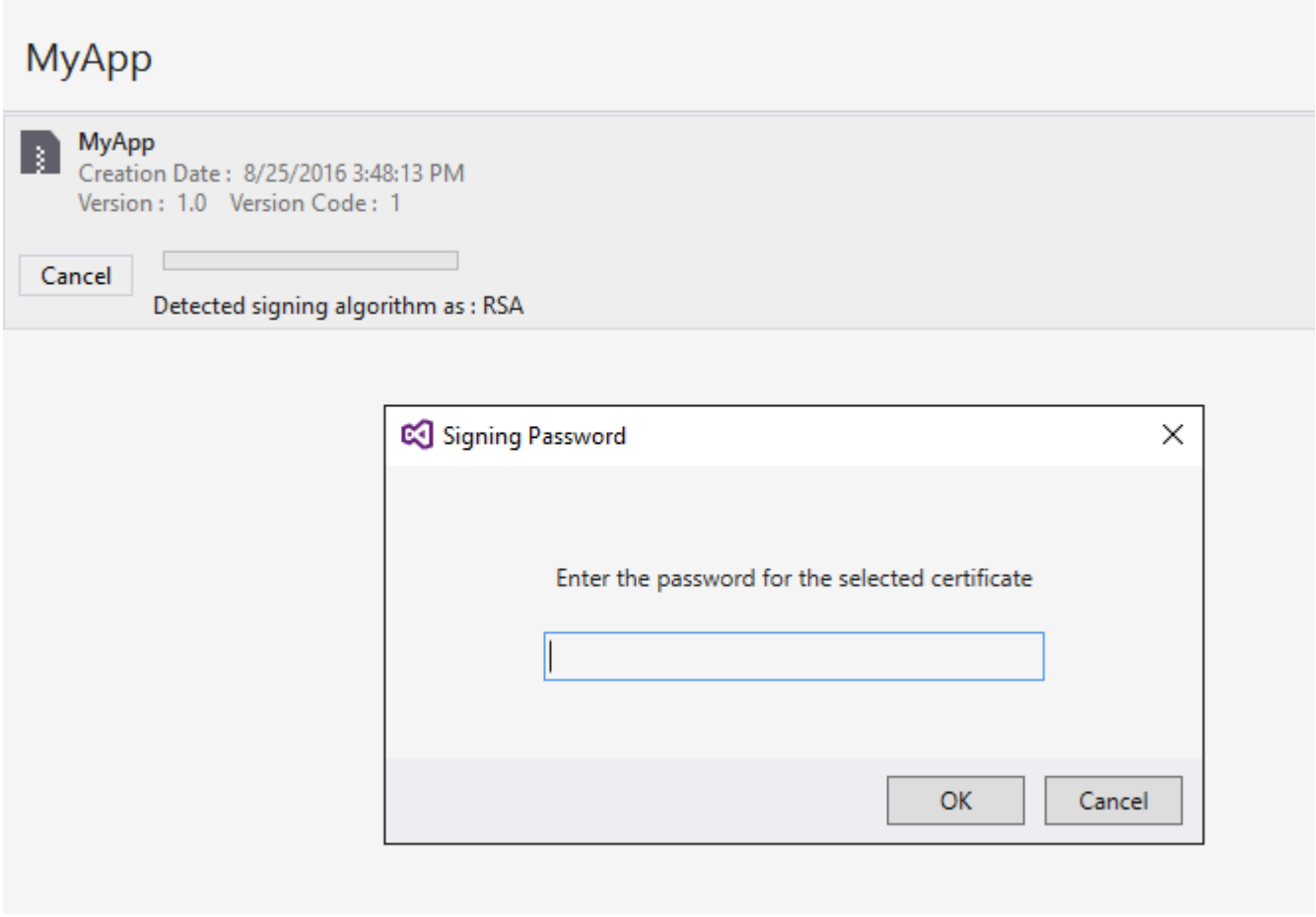
しいAndroid Key Storeの



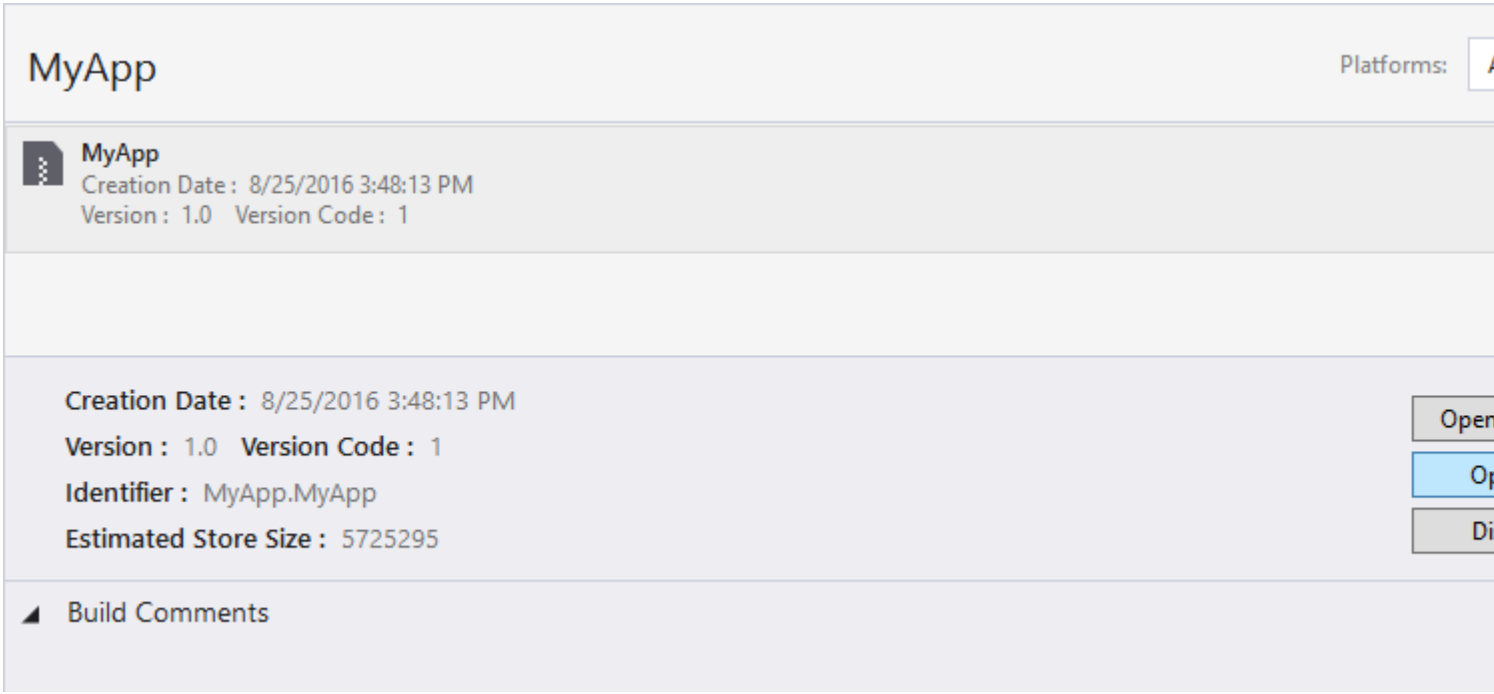
APKをするには、[をけて]をクリックします。キーストアパスワードのをめられることがあります。

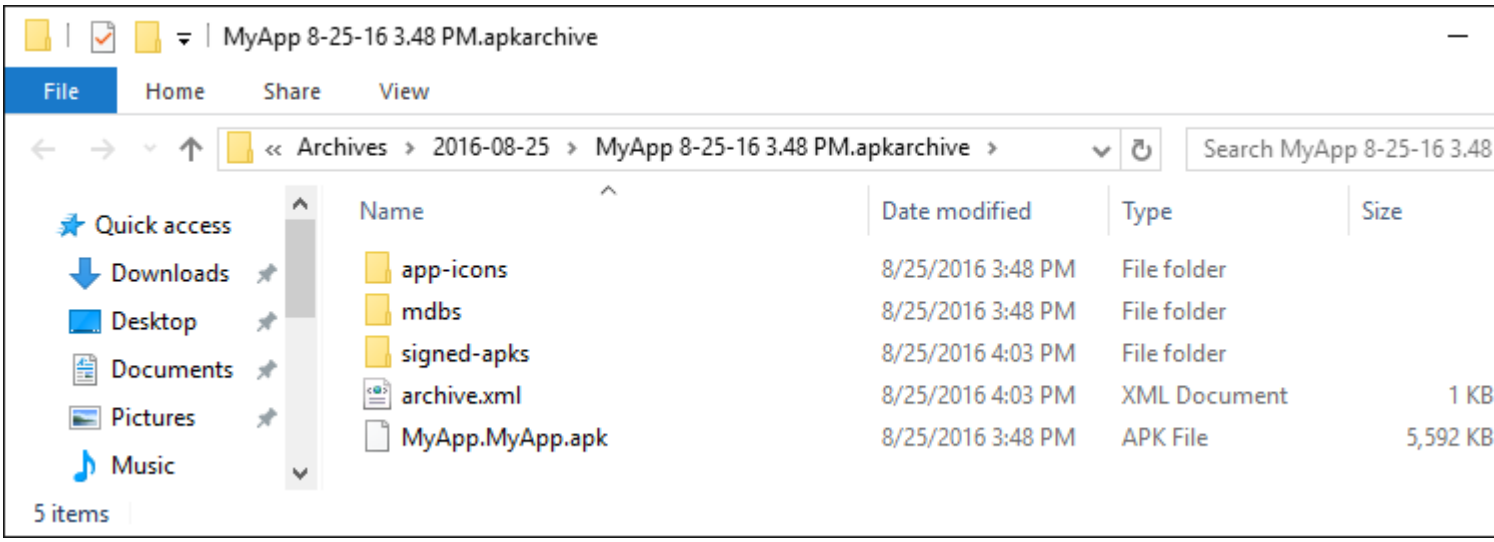

Distribute
App Details
MyApp
Creation Date: 8/25/2016
Version: 1.0
Select Channel
Ad Hoc
Signing Identity
Signing Identity
Search
Name
Expiration
chimp
Sat Aug 18 15:59:13 PDT 2046
Import...
Specify a Time Stamping Authority:
http://example.timestampauth.com
Why do I need a Key Store?
Back
Save As


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



したら、ArchivesのOpen Folderをクリックして、されたAPKファイルをすることができます。





Xamarin.Android APKでMultiDexをにする

MultiDexは、アプリが65,536のメソッドをつことをにするAndroid APKのライブラリです。

Android APKには、JavaコードからコンパイルされたされたバイトコードをむDalvikファイル.dexがあります。.dexファイルには、65,536のメソッド2＾16をめることができます。

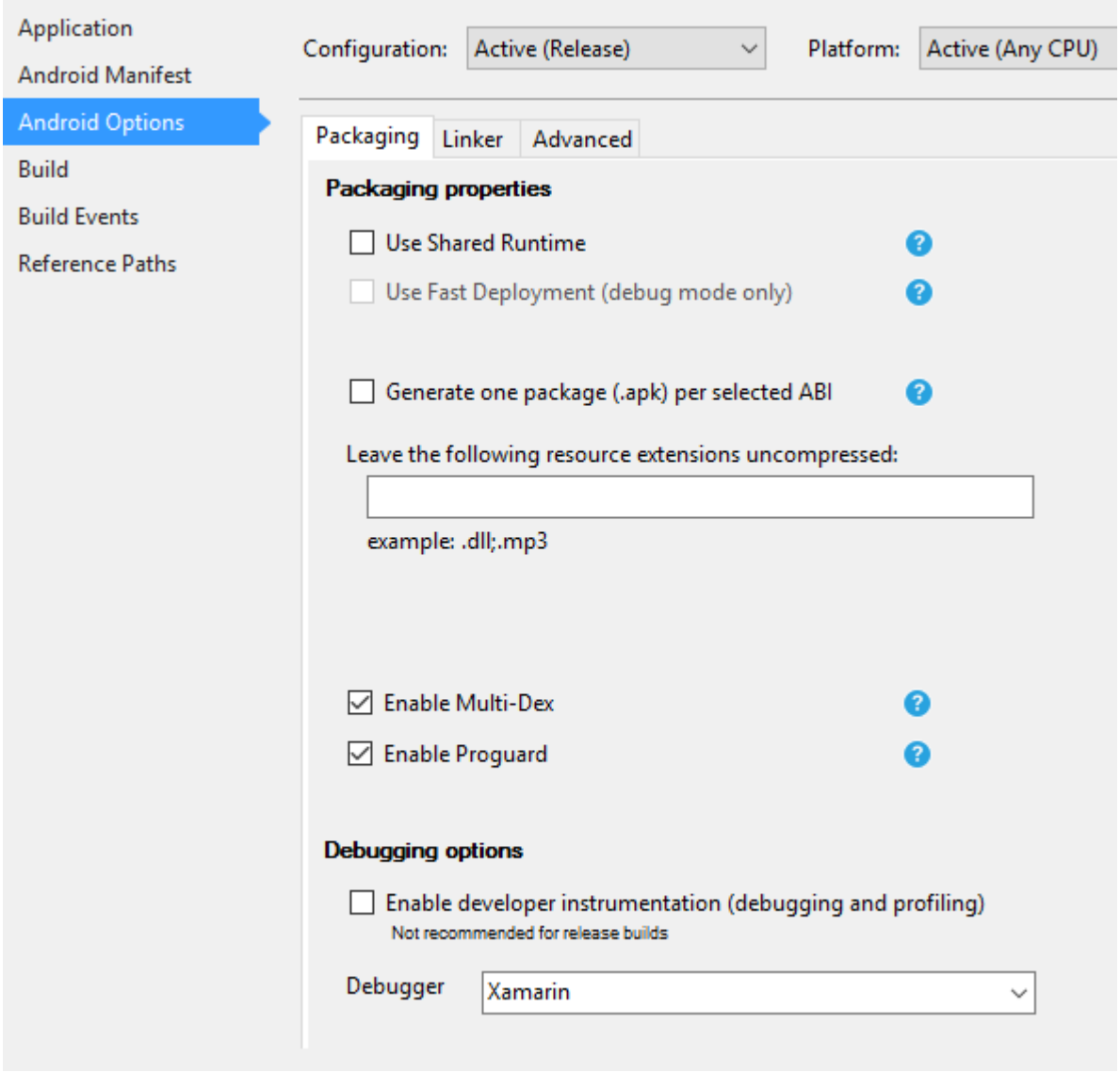
Android 5.0 LollipopAPI 21よりのAndroid OSのバージョンでは、APKあたり1つの.dexファイルしかサポートしないDalvikランタイムがされ、APKあたり65,536メソッドにされています。 Android 5.0、Android OSはARTランタイムをします.ARTランタイムは、APKごとにの.dexファイルをサポートし、をします。

API 21のAndroidバージョンで65kメソッドをえるには、はMultiDexサポートライブラリをするがあります。 MultiDexはclasses.dexファイルでそれらをするなclasses.dexファイルclasses2.dex、classes3.dex、...をします。アプリはみみをすると、MultiDexApplicationクラスをしてな.dexファイルをみみます。

あなたのAndroidアプリがAPI 21Android 5.0 LollipopのSDKバージョンをしているは、MultiDexライブラリをするはありません。これは、OSがネイティブにな.dexファイルをするためです。しかし、のから、がいAndroid OSをサポートしたいは、MultiDexライブラリをするがあります。

Xamarin.Androidアプリで MultiDexをする

まず、Xamarin.AndroidアプリでMultiDexをにするには、ののように、プロジェクトのプロパティ - > Androidオプション - >パッケージ - >マルチデクスをにします。



に、アプリでMultiDexApplicationクラスをするがあります。プロジェクトのルートで、しいクラスをしますソリューションエクスプローラで、プロジェクトをクリックし、Add - > Class、またはShift + Alt + C。しいクラスファイルで、のコードをコピーし、SampleをXamarin.Androidプロジェクトのにきえます。

```
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); }
        }
    }
}
```

コードソースはこちら。

Visual Studio for Windowsでしているは、プロジェクトをビルドするときにclasses.dexファイルをしくするためにするがあるAndroid SDKビルドツールにもバグがあります。

あなたのAndroid SDKフォルダにし、build-toolsフォルダをき、のようなAndroid SDKコンパイラ

ののフォルダがあります

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

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

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

これらのフォルダのそれぞれのでは、mainClassesDex.batとばれるファイル、classes.dexファイルをするためにされるバッチスクリプトがあります。mainClassesDex.batファイルをテキストエディタメモまたはメモ++でき、スクリプトでブロックをしてきえます。

```
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
```

ブロックの

```
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
```

ソースはこちら。

、mainClassesDex.batをテキストエディタにします。

のをすると、MultiDexでXamarin.Androidアプリをにできるはずです。

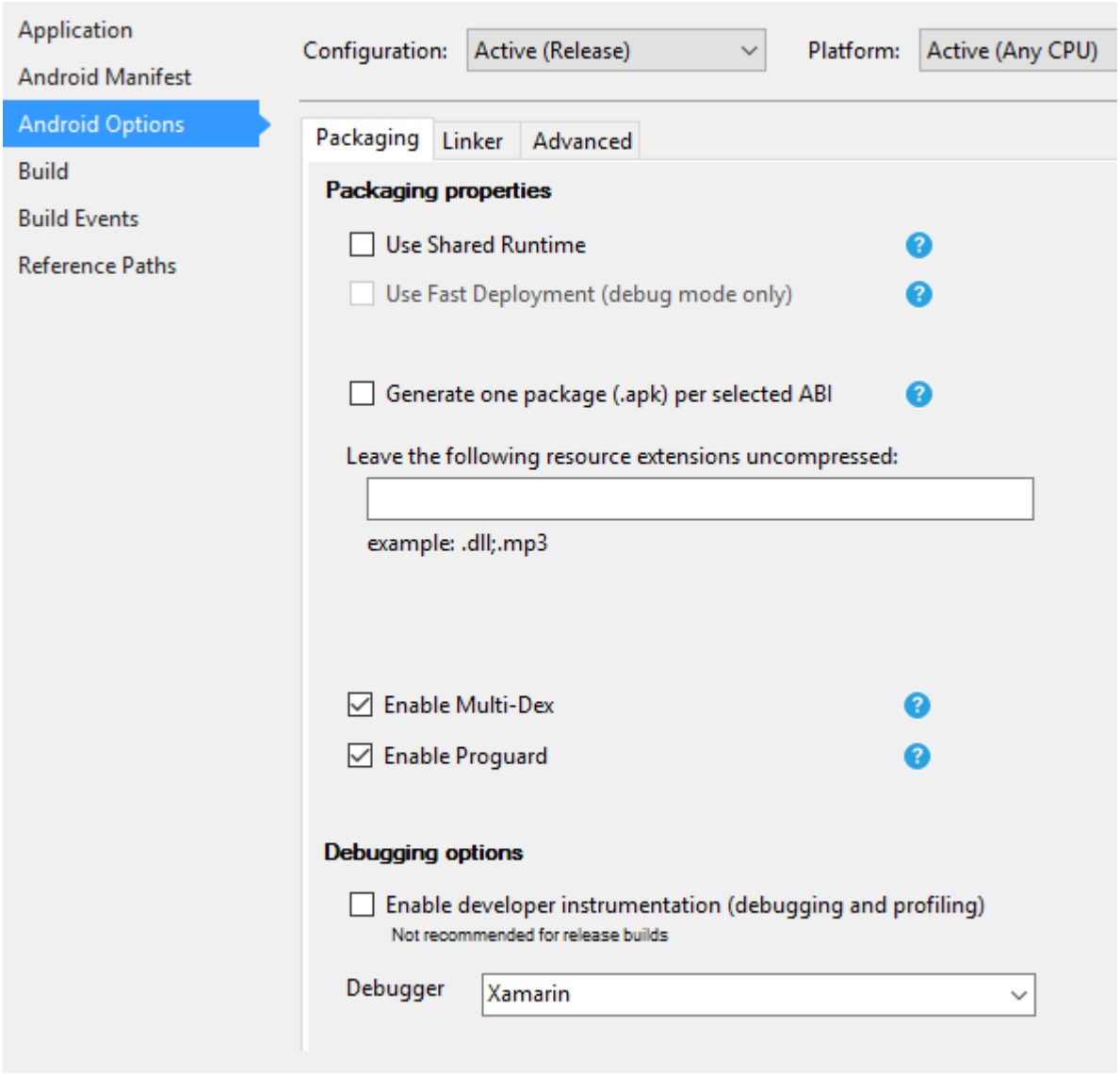
Xamarin.Android APKでProGuardをにする

ProGuardは、ビルドプロセスでAPKのJavaコードをおよびし、のクラスをするためにされるツールです。 ProGuardをすると、としてられるAPKのサイズがさくなり、リバースエンジニアリングデコンパイルがしくなります。

ProGuardは、Xamarin.Androidアプリでもできます。また、APKファイルサイズをし、Javaコードをします。ただし、ProGuardのはJavaコードにのみされます。 .NETコードをするために、はDotfuscatorまたはのツールをするがあります。

Xamarin.Androidアプリで**ProGuard**をする

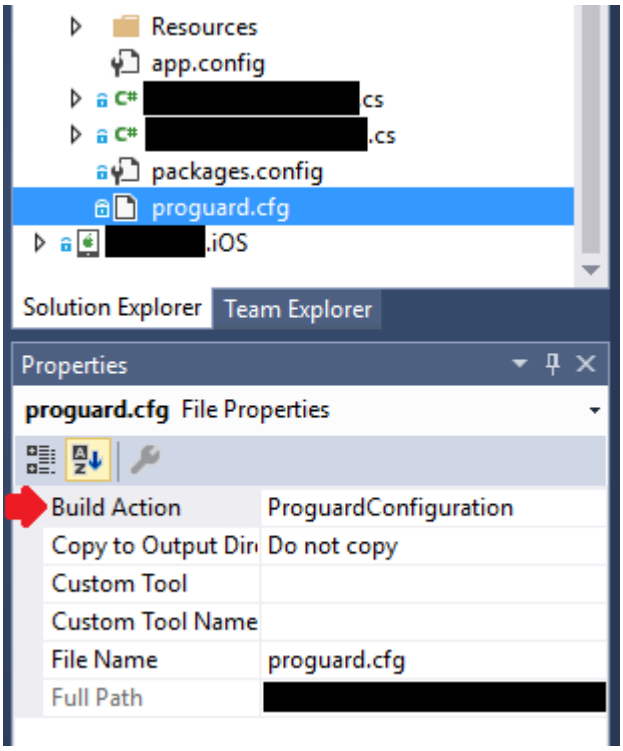
まず、Xamarin.AndroidアプリでProGuardをにするには、ののように、プロジェクトのプロパティ - > Androidオプション - >パッケージ - > ProGuardをにします。



これにより、アプリケーションをするにProGuardがになります。

デフォルトでXamarin.Androidは、 `obj/Debug/proguard`のをいます。これは、 `obj/Debug/proguard`または`obj/Release/proguard`フォルダの`proguard_project_primary.cfg`、`proguard_project_references.cfg`および`proguard_xamarin.cfg`ファイルにあります。 3つのファイルはProGuardのとしてされ、にXamarinによってにされます。

がProGuardオプションをさらにカスタマイズしたいは、プロジェクトのルート`proguard.cfg`ファイルをできますが.cfgであればのもです。ビルドアクションをProguardConfigurationにすると、ののように



ファイルには、カスタムProGuardのオプションは、のような、することができます`-dontwarn`、 `-keep class`および。

のところApril / 2017、ダウンロードされるAndroid SDKにはProGuardのバージョンがまれており、Java 1.8をしてアプリケーションをするときにエラーがするがあります。に、エラーリストにのメッセージがされます。

```
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]
```

ソースはこちら。

このをするには、ProGuardのバージョン ここ をダウンロードし、.zipファイルのを`android-sdk\tools\proguard\`コピーするがあります。これによりProGuardがされ、ビルドプロセスはなくされます。

その、ProGuardをしてXamarin.Androidアプリをにできるはずです。

ProGuardとLinkerにする "な"バグ

らしいアプリをり、デバッグでテストしたところ、いがられました。すべてがうまくいっていた

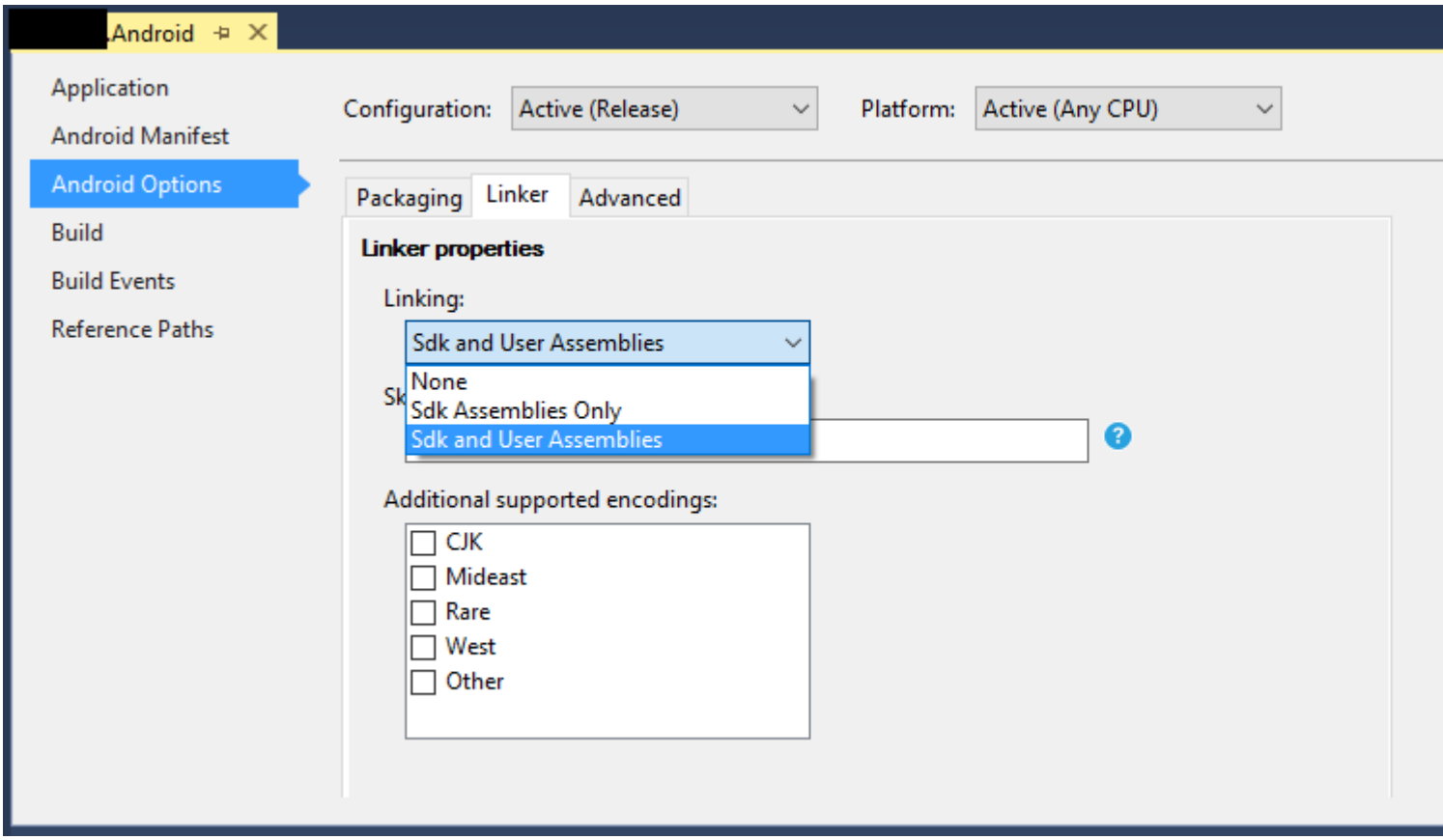
しかし、その、あなたのアプリをリリースするをすることにしました。 MultiDex、ProGuard、Linkerをした、をします。

このチュートリアルでは、なバグをきこすのあるProGuardやLinkerにするなをつけるのをするこ

とをとしています。

Xamarin.Linkerについて

Xamarin.Linkerは、 **Java**コードではなく、.NETコードからのコードとクラスをするビルドプロセスのツールです。プロジェクトの[プロパティ] - > [Androidオプション] - > [リンカ]に、オプションとリンクするボックスがされます。



なし コードはされません。

Sdkアセンブリのみ このオプションをすると、Xamarin.LinkerはXamarinライブラリのコードをチェックします。 このオプションはです。

Sdkとユーザーアセンブリ このオプションをすると、Xamarin.LinkerはXamarinライブラリとプロジェクトコードPCL、Xamarinコンポーネント、NuGetパッケージをむのコードをチェックします。 このオプションはずしもではありません

SdkとUser Assembliesオプションをする、Xamarin.Linkerはにコードがされているときにコードのがされていないとえるかもしれません。そのため、のライブラリがにしなくなり、アプリにバグがするがあります。

Xamarin.Linkerがコードをできないようにするには、3つのオプションがあります

1. リンクオプションを[なし]または[SDアセンブリのみ]にする。

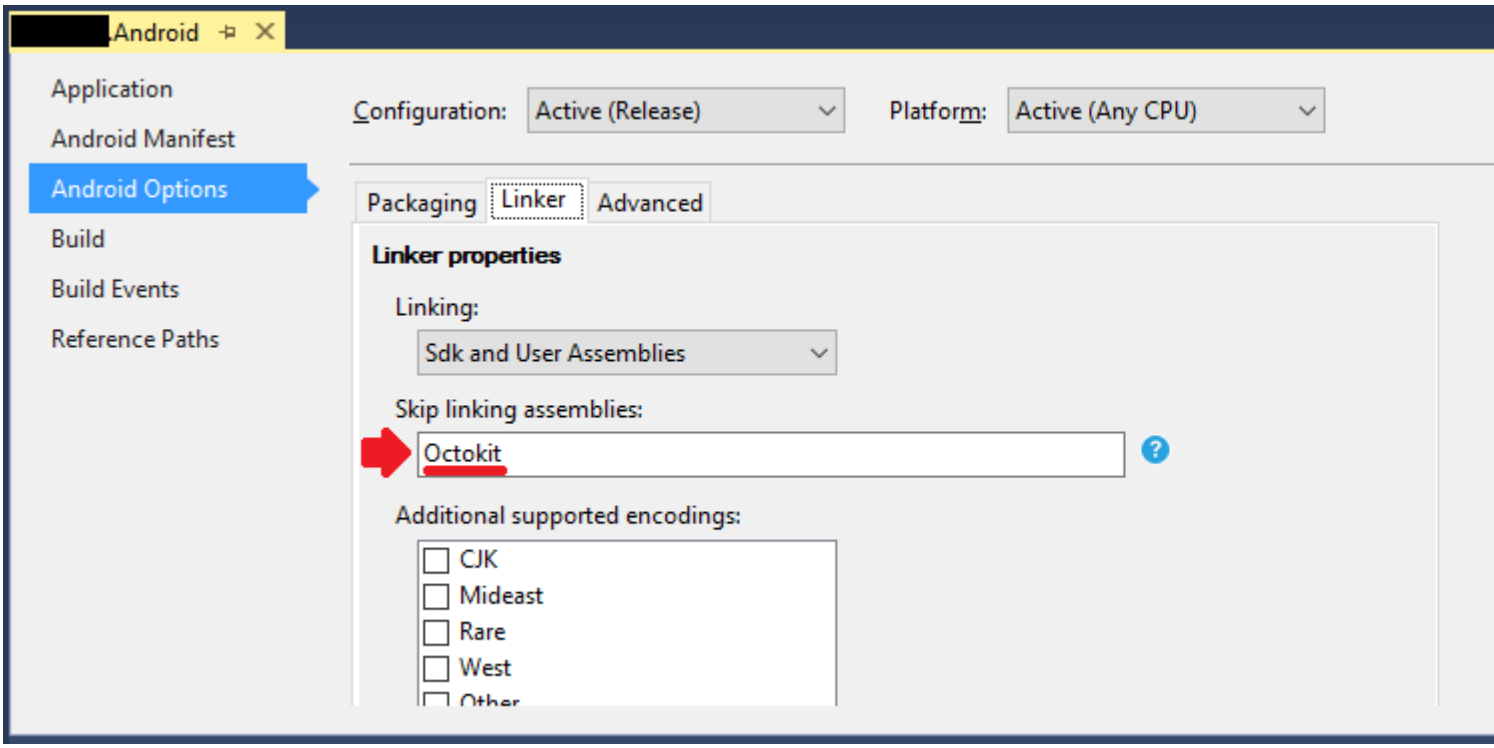
2. アセンブリをスキップする。
3. Preserveの。

2.アセンブリをスキップする

のでは、Xamarin.Linkerをすると、もうしなくなったNuGet Package Octokit がにしなくなりました。

```
[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 ---
```

ライブラリをびさせるためには、ののように、プロジェクト - >プロパティ - > Androidオプション - >リンカにあるアセンブリのスキップフィールドにパッケージをするがありました



その、はのもなくをめました。

3. Preserveをする

Xamarin.Linkerは、のコードがプロジェクトのコアにあるモデルクラスのコードのほとんどをします。

リンクにクラスをするには、Preserveをします。

まず、プロジェクトのコアに**PreserveAttribute.cs**というのクラスをし、のコードをしてをプロジェクトのにきえます。

PreserveAttribute.cs

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

プロジェクトのコアのモデルクラスに、ののようにPreserveをします。

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;
        }
    }
}
```

その、リンクではされたコードはされません。

ProGuardについて

ProGuardは、 **Java**コードからのコードとクラスをするビルドプロセスのツールです 。また、コ

ードをしてします。

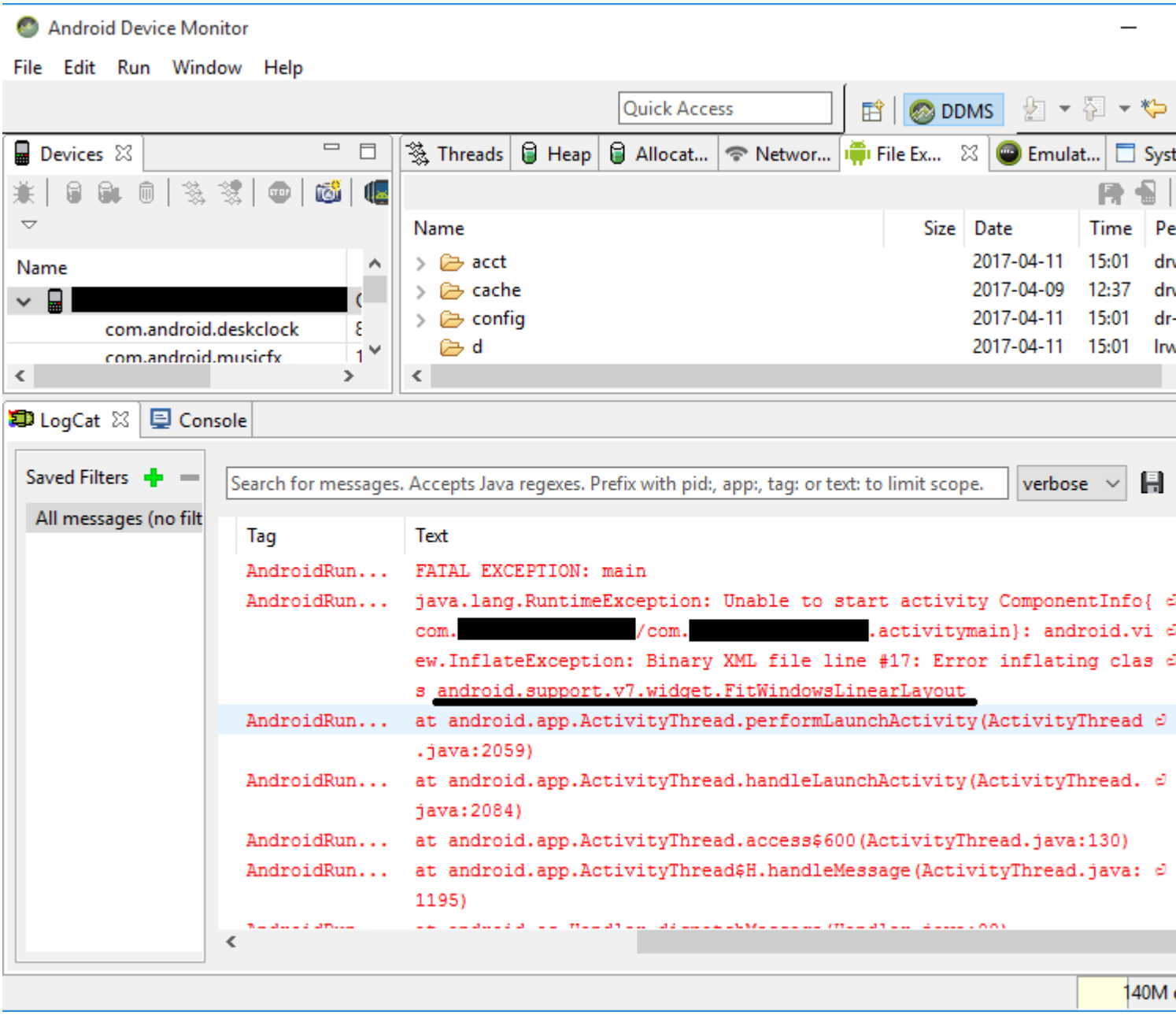
しかし、ProGuardはにはであるとするコードをすることがあります。これをけるには、はAndroidデバイスモニタとVisual Studio Debugでアプリをデバッグし、されたクラスをしてProGuardファイルをしてクラスをするがあります。

のでは、ProGuardはAXMLレイアウトファイルでされる2つのクラスAndroid.Support.V7.Widget.FitWindowsLinearLayoutおよびAndroid.Support.Design.Widget.AppBarLayoutをしましたが、コードではとされていました。アクティビティレイアウトのレンダリングに、JavaコードでClassNotFoundExceptionがされました。

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">
...
```

SetContentViewでレイアウトをするときのLogCatのエラー



このエラーをするには、プロジェクトのProGuardファイルにのをするがありました。

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

その、レイアウトをするときにエラーはされなくなりました。

ProGuardの

ProGuardは、プロジェクトをビルドした、エラーリストにをすることがあります。らはあなたのアプリがOKかどうかのをしますが、にあなたのアプリがうまくされているは、のすべてがをしているわけではありません。

するにそのためのつのは、ピカソのライブラリをProGuardのをしている、これはのようながされることがあり`okio.Okio: can't find referenced class (...)`か、`can't write resource [META-INF/MANIFEST.MF] (Duplicate zip entry [okhttp.jar:META-INF/MANIFEST.MF]) (...)`、しかし、アプリ

ケーションはビルドし、ライブラリはなくします。

オンラインでXamarin.Android APKをするをむ https://riptutorial.com/ja/xamarin-android/topic/9601/xamarin-android-apkをする

7: XamarinアプリケーションでZXingライブラリをしたバーコードスキャン

き

Zxingライブラリは、によくられています。 ZxingはJavaにづいており、.Netモジュールもで、xamarinアプリケーションでもできます。のをするにはここをクリックしてください http://zxingnet.codeplex.com/

は、このライブラリをしました。

ステップ1ソリューションにZXing.Net.Mobileコンポーネントをします。

ステップ2バーコードスキャナをするがあるアクティビティで、そのアクティビティでMobileBarcodeScannerをします。

ステップ3スキャンをするには、のビューでタップすると、コードのにいてください。

Examples

サンプルコード

```
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;
}
```

オンラインでXamarinアプリケーションでZXingライブラリをしたバーコードスキャンをむ https://riptutorial.com/ja/xamarin-android/topic/9526/xamarinアプリケーションでzxingライブラリをしたバーコードスキャン

8: アプリケーションのライフサイクル - Xamarin.Andorid

き

Xamarin.Androidアプリケーションのライフサイクルは、のAndroidアプリとじです。ライフサイクルについてえば、アプリケーションライフサイクル、アクティビティライフサイクル、およびフラグメントライフサイクルについてすがあります。

では、それらをするためのいとをしようとします。はのAndroidとXamarinのドキュメントとののセクションでされているくのにつWebリソースからこのドキュメントをしました。

Androidアプリケーションのライフサイクルにするいへのいリンク

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

アプリケーションライフサイクル

まず、Android.Applicationクラスをして、アプリのライフサイクルにする2つのなメソッドにアクセスできるようにするがあります。

- OnCreate - アプリケーションのに、のアプリケーションオブジェクトMainActivityなどがされるにびされます。
- OnTerminate - このメソッドは、エミュレートされたプロセスでするためのものです。これは、Androidデバイスではしてびされません。プロセスをにすだけでプロセスがされます。このとき、ユーザーコードこのコールバックをむはされません。ドキュメント https://developer.android.com/reference/android/app/Application.html#onTerminate

Xamarin.Androidアプリケーションでは、Applicationクラスをのようにできます。 "MyApplication.cs"というしいクラスをプロジェクトにします

```
[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();
    }
}
```

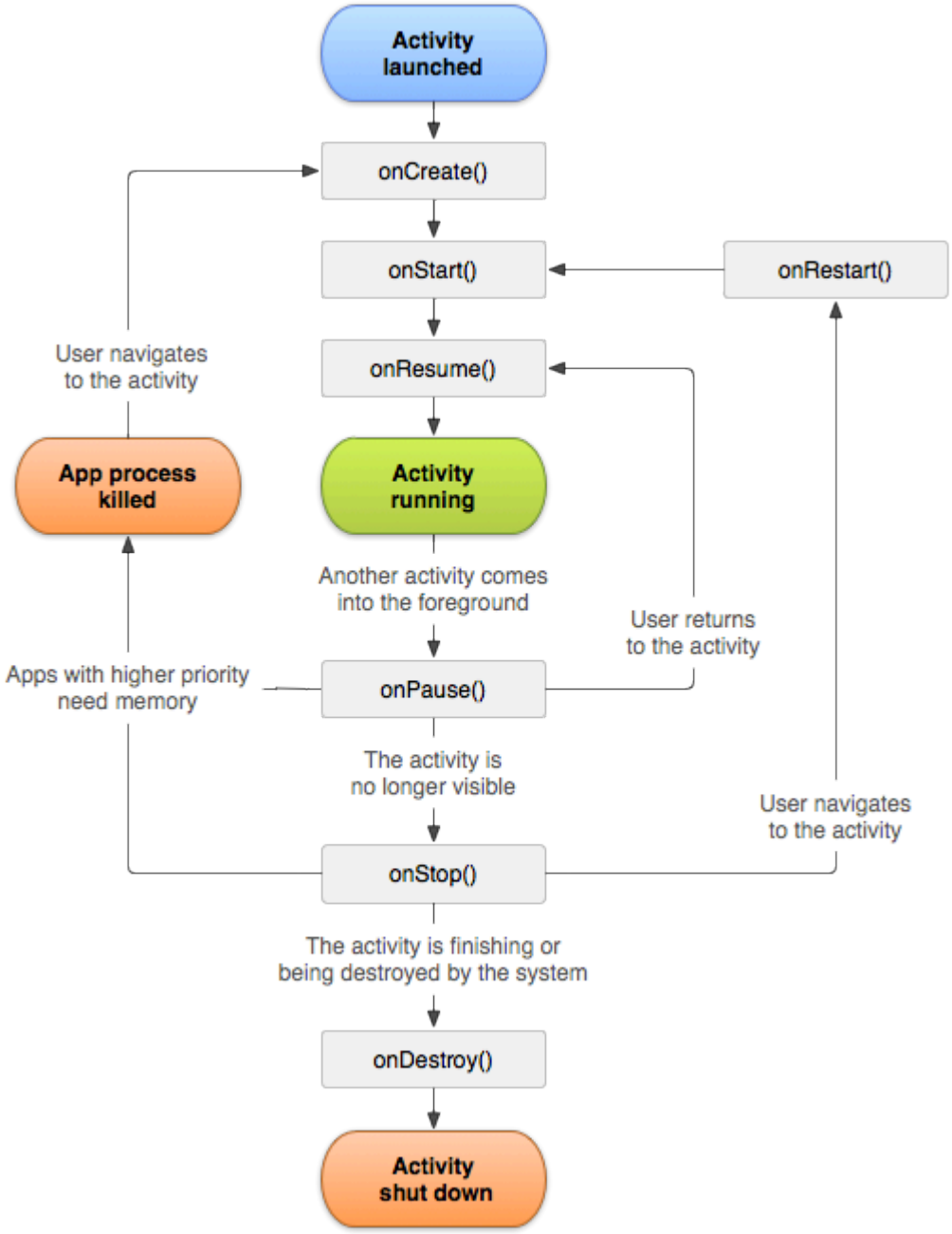
でいたように、OnCreateメソッドをうことができます。たとえば、ここでロ―カルデ―タベ―スをしたり、のをうことができます。

さらに、OnConfigurationChangedやOnLowMemoryのようにオ―バ―ライドできるメソッドがあります。

アクティビティライフサイクル

アクティビティのライフサイクルはかなりです。あなたがっているように、アクティビティはユ―ザ―がそれとのをできるAndroidアプリののペ―ジです。

のは、Android Activityライフサイクルのをしています。



ごのとおり、Activityライフサイクルのなれがあります。モバイルアプリケーションでは、のライフサイクルのをするアクティビティクラスのメソッドがあります。

```
[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");
    }
}
```

のAndroidのドキュメントにはいがあります

- アクティビティのライフタイムは、onCreateBundleののびしからonDestroyののびしまでのにします。アクティビティはonCreateで "グロ―バル"のすべてのをい、りのすべてのリソ―スをonDestroyでします。たとえば、ネットワ―クからデ―タをダウンロ―ドするスレッドがバックグラウンドでされている、onCreateでそのスレッドをしてから、onDestroyでスレッドをすることがあります。

- アクティビティのなは、onStartへのびしとするonStopへのびしのでします。この、ユ―ザはのアクティビティをることができますが、フォアグラウンドにはせず、ユ―ザとすることもできます。これらの2つののので、ユ―ザ―にアクティビティをするのになリソ―スをできます。たとえば、OnStartにBroadcastReceiverをして、UIにをえるをし、ユ―ザ―がしているものがされなくなったときにonStopでをできます。 onStartメソッドとonStopメソッドは、アクティビティがされてユ―ザ―にされなくなるため、びすことができます。

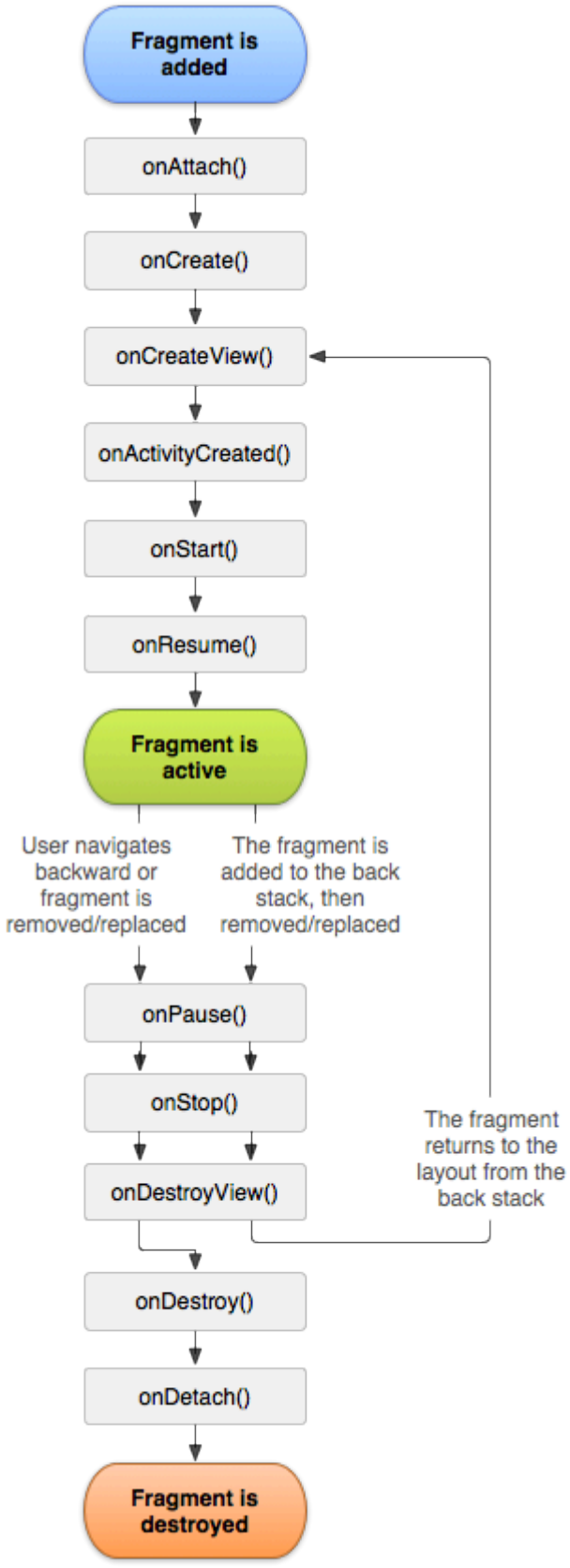
- onResumeのびしとonPauseへのするびしので、アクティビティのフォアグラウンドのがします。この、アクティビティはのすべてのアクティビティのにあり、ユ―ザとします。たとえば、デバイスがスリ―プになったとき、アクティビティがされたとき、しいインテントがされたときなど、アクティビティはにとのになることがあるため、これらのメソッドのコ―ドはかなりでなければなりません。

フラグメントライフサイクル

あなたがっているように、1つのアクティビティをつことができますが、そのにはなるフラグメン

トがめまそのため、フラグメントライフサイクルもにとってです。

のは、Androidのフラグメントライフサイクルがどのようになっているかをしています。



のAndroidのドキュメントにされているように、なくともの3つのをするがあります

- OnCreate - フラグメントをするときにシステムがこれをびします。では、フラグメントがまたはしてからするときにしたいフラグメントのなコンポ―ネントをするがあります。

- OnCreateView - フラグメントがめてユ―ザ―インタ―フェイスをするときにシステムがこれをびします。フラグメントのUIをするには、フラグメントのレイアウトのル―トであるこのメソッドからViewをすがあります。フラグメントがUIをしない、nullをすことができます。

- OnPause - このメソッドは、ユ―ザ―がフラグメントをれるのとしてコ―ルされますずしもフラグメントがされているとはりません。これは、ユ―ザ―がってこないがあるためのユ―ザ―セッションをえてするのあるをコミットするです。

ここにXamarin.Androidのサンプルがあります

```
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();
    }
}
```

もちろん、のをうは、ここにメソッドをすることもできます。

GitHubのなサンプル

のでプロジェクトをしたいは、のGitHubからXamarin.Androidアプリケ―ションテンプレ―トをダウンロ―ドできます。あなたはのをつけることができます

- アプリケ―ションライフサイクルメソッド

- アクティビティライフサイクルメソッド
- フラグメントライフサイクルメソッド

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

オンラインでアプリケーションのライフサイクル - Xamarin.Andoridをむ
https://riptutorial.com/ja/xamarin-android/topic/8842/アプリケーションのライフサイクル----xamarin-andorid

9: カスタムリストビュー

Examples

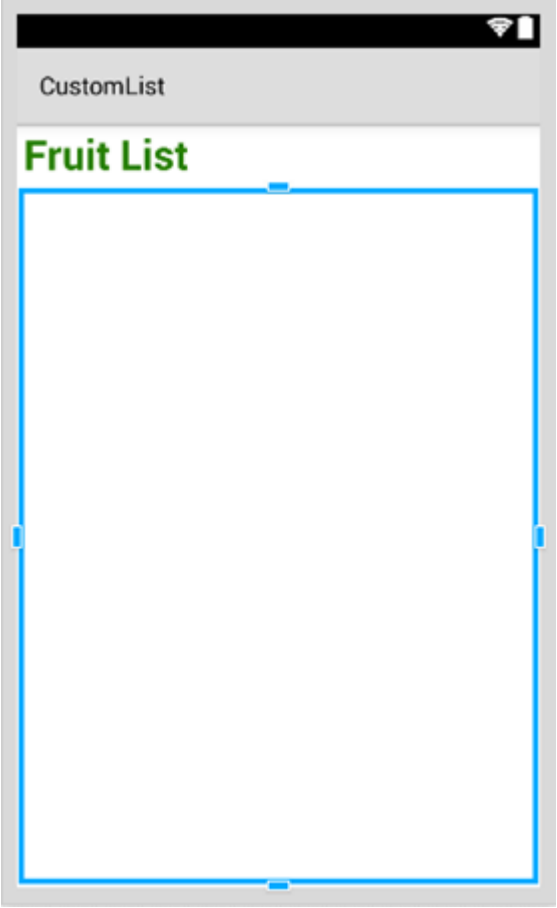
カスタムリストビューは、ユーザーのニーズにわせてされたでされています。

のレイアウトの、customrow.axmlファイルはのようになります

```
    <?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>
```

に、main.axmlをすることができます。このmain.axmlには、ヘッダーのテキストビューとリストビューがまれています。



はです...

に、オブジェクトをすData.csクラスをします。

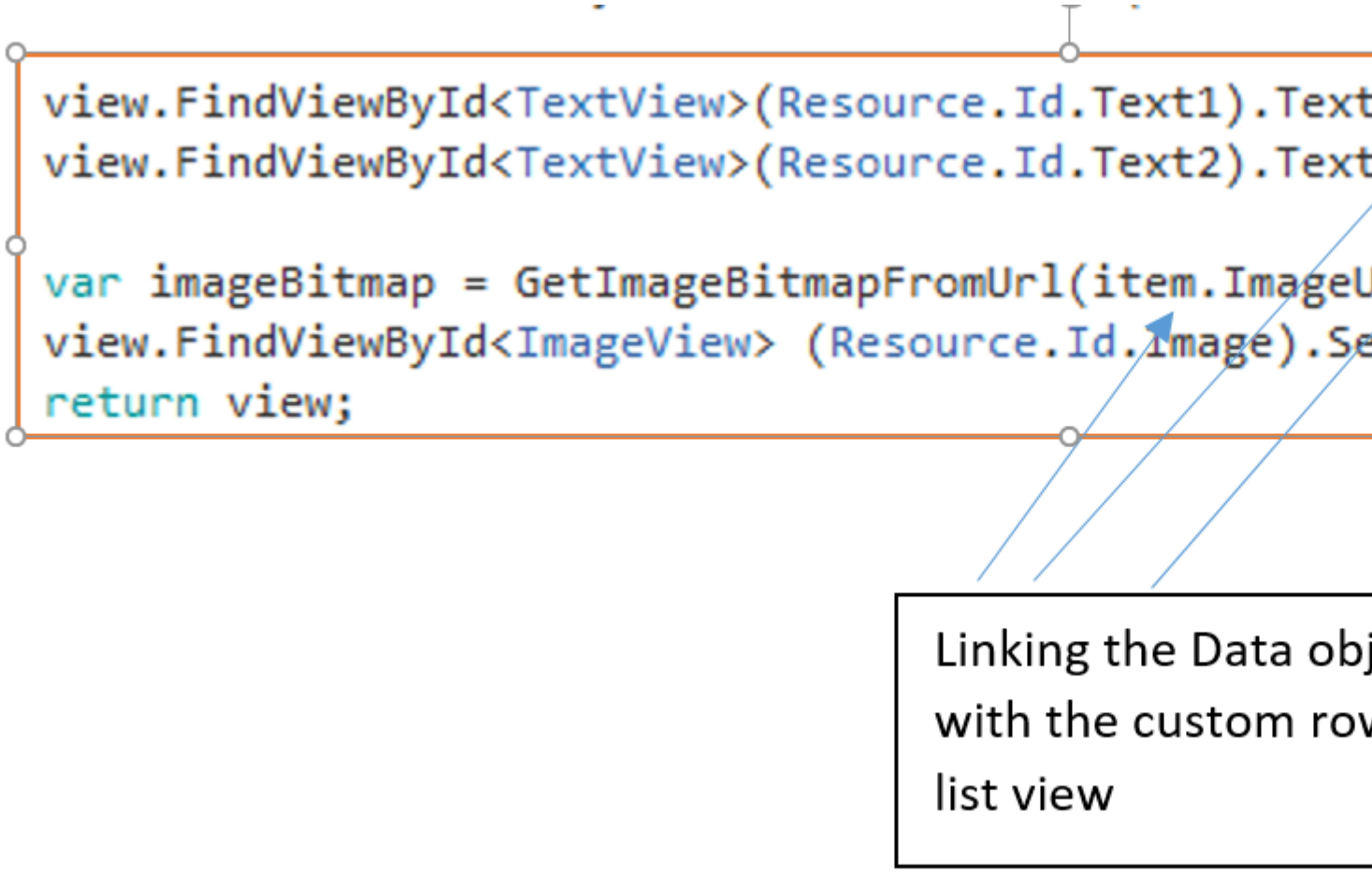
```
public class Data
{
    public string Heading;
    public string SubHeading;
    public string ImageURI;

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

に、DataAdapter.csクラスがです。アダプターは、データをになるビューにリンクします

```
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;
    }

}
```

もなはGetViewのにあります。これは、オブジェクトをカスタムにリンクするです。



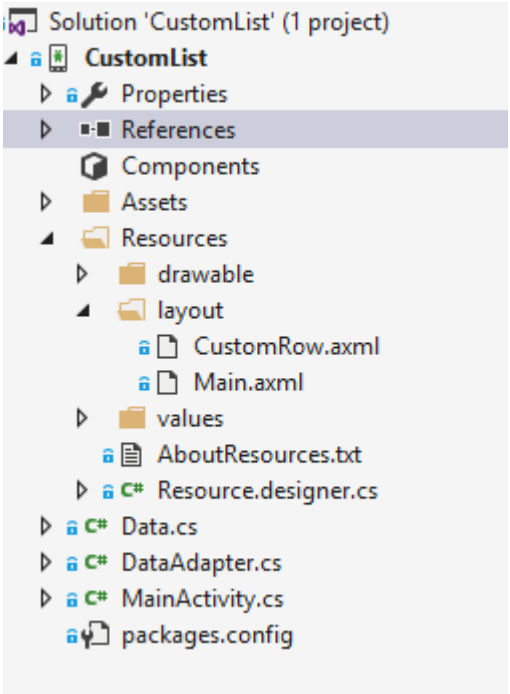
GetImageBitmapFromUrlはdataadapterのではありませんが、ここではにするためにここにします。

ようやくMainActivity.csにます

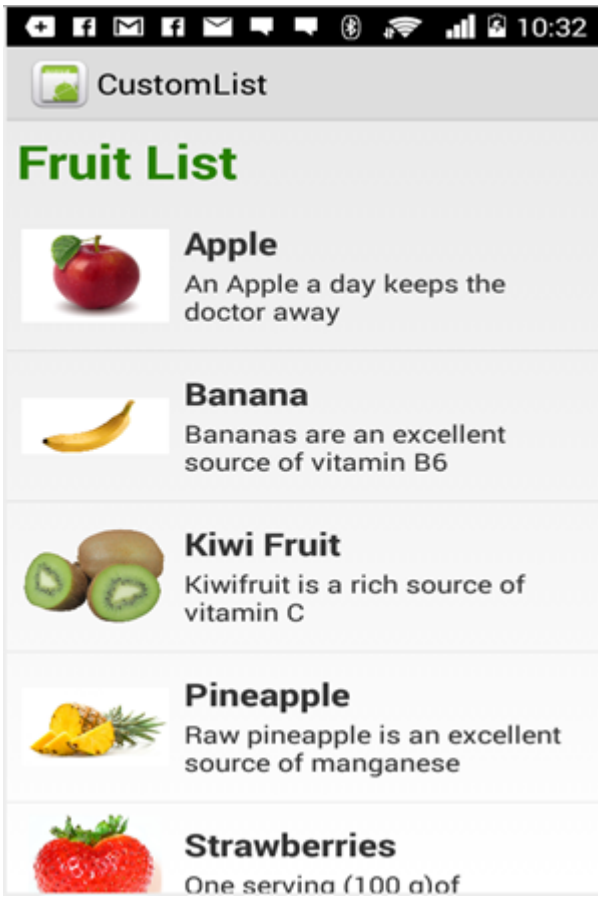
```
public class MainActivity : Activity
{

    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);

    }
```

なプロジェクトのはのとおりです。



すべてがうまくいけば、のようにがされます



オンラインでカスタムリストビューをむ https://riptutorial.com/ja/xamarin-android/topic/6406/カスタムリストビュー

10: ダイアログ

ダイアログの**Context**をする

`Activiy`から`Dialog`をするときに、 `this`をコンテキストとしてできます。

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

`Fragments`では、プロパティ`Context`をします。

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

ボタンの

`SetNeutralButton()`は、がみられるというなとにできます。 `SetPositiveButton()`は、たとえば「このアイテムをしますか」などのにできます。 `SetNegativeButton()`は、ダイアログを`SetNegativeButton()`アクションをキャンセルするためのものです。

るボタンからキャンセルをにする

るボタンでダイアログを`SetCanceable(false)`ことができないようにしたいは、 `SetCanceable(false)`びすことができます。これは、るボタンでのみします。

ダイアログがされているにをさせると、はえ、okアクションとcancelアクションはびされません。アクティビティーのでこれをし、アクティビティーがリロードされたにダイアログをするがあります。

これをするには、わりに`DialogFragment`をします。

Examples

ダイアログ

アラートダイアログの

```
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();
```

オンラインでダイアログをむ https://riptutorial.com/ja/xamarin-android/topic/2510/ダイアログ

11: ダイアログ

パラメーター

よくわれるPublicメソッド	つかいます
SetTitleString	ダイアログのタイトルをします。
SetIcon	アラートダイアログのアイコンをする
SetMessage	するメッセージをします。
SetNegativeButtonString、EventHandler	ダイアログのネガティブボタンがされたときにびされるリスナをします。
SetPositiveButtonString、EventHandler	ダイアログのポジティブボタンがされたときにびされるリスナをします。
SetNeutralButtonString、EventHandler	ダイアログのニュートラルボタンがされたときにびされるリスナをします。
SetOnCancelListener IDialogInterfaceOnCancelListener	ダイアログがキャンセルされたにびされるコールバックをします。
SetOnDismissListener IDialogInterfaceOnDismissListener	らかのでダイアログがじられたときにびされるコールバックをします。
	このビルダーにをしてAlertDialogをし、Dialog.Showにダイアログをします。

ネームスペースAndroid.App

アセンブリMono.AndroidMono.Android.dll

アセンブリバージョン0.0.0.0

パブリックコンストラクタ

AlertDialog.Builderコンテキスト -

このBuilderのコンテキストとそれがするAlertDialogをするコンストラクターです。

AlertDialog.Builderコンテキスト、Int32 -

このBuilderおよびそれによってされるAlertDialogのコンテキストおよびテーマをするコンストラクター。

マテリアルデザイン**AlertDialog**の

のAlertDialogをするには

1. NuGetパッケージからv7 AppCompatライブラリをインストールする
2. AlertDialogをAndroid.Support.V7.App.AlertDialogにきえるか、にのステートメントをしてダイアログをるくします。

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

Examples

AlertDialog

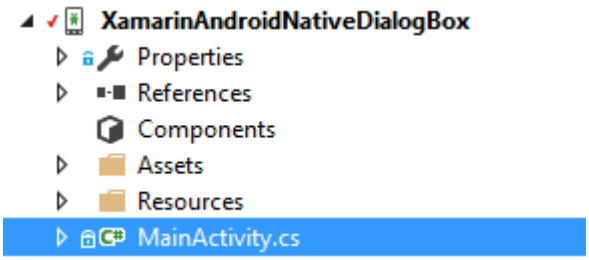
```
// 1. Instantiate an AlertDialog.Builder with its constructor
// the parameter this is the context (usually your activity)
AlertDialog.Builder builder = new AlertDialog.Builder(this);

// 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();
```

シンプルアラートダイアログの

Xamarin.Androidでなダイアログをします

では、ドキュメントからGetting Startedガイドをんでいるとえています。

のようなプロジェクトがです。



なはのようになっているはずです

```
public class MainActivity : Activity
```

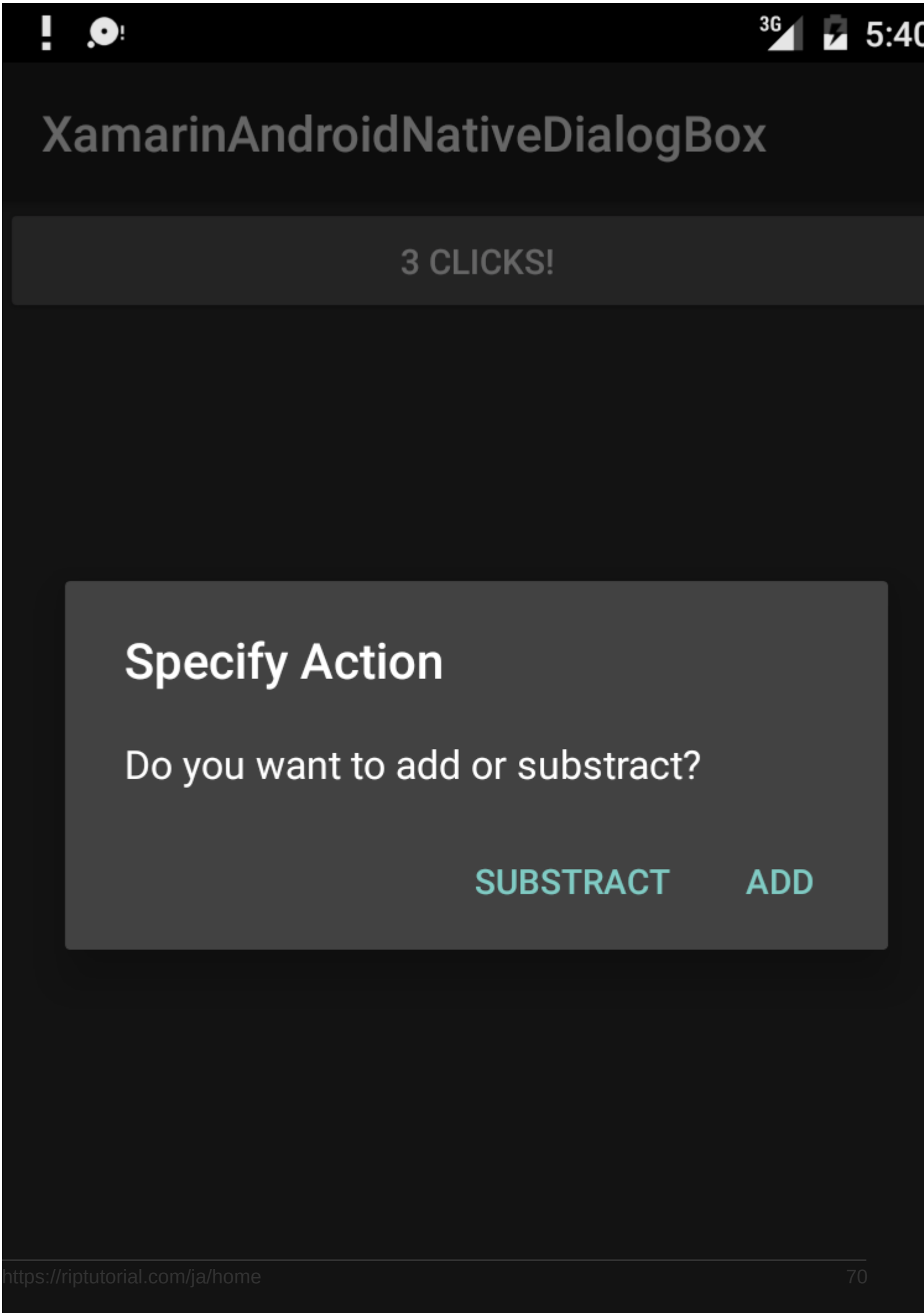
```
{
int count = 1;

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++); };
}
}
```

ここでは、ボタンクリックにカウンター—にするのではなく、なアラ—トダィアログでユ—ザ—にまたはするかどうかをねます

ポジティブボタンまたはネガティブボタンをクリックすると、アクションがされます。

```
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();
};
```

スクリ—ンショット


3G
5:40
XamarinAndroidNativeDialogBox
3 CLICKS!
Specify Action
Do you want to add or substract?
SUBSTRACT
ADD

12: トースト

Examples

トーストメッセージ

まず、 `MakeText()` メソッドの1つでToastオブジェクトをインスタンスします。このメソッドは、アプリケーション`Context` 、テキストメッセージ、およびトーストのという3つのパラメータをとります。これは、にされたToastオブジェクトをします。のにすように、 `Show()` をしてトーストを`Show()` できます。

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

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

このは、ほとんどのトーストになすべてをしています。にはほとんどがありません。しかし、トーストのをえたり、なテキストメッセージのわりにのレイアウトをしたりすることもできます。のセクションでは、これらのことをうについてします。

あなたのメソッドをさせて、1ライナーとしてびし、のようにトーストオブジェクトをしないようにすることもできます

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

については、トピックにするよりなAndroidドキュメントをしてください。

トーストのメッセージ

たちはトーストがたちにえるビューここではColorMatrixColorFilterをしていますにカラーフィルターをして、トーストメッセージのをすることができます

```
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();
```

また、がるいまたはいは、テキストのをすることもできます。

```
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
```

トーストのをする

SetGravityメソッドをってトーストをすることができます。このメソッドは、3つのパラメータをとります。はスクリーンのトーストので、2つはからオフセットされたトーストをしますのパラメータによってされます。

```
//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();
```

オンラインでトーストをむ https://riptutorial.com/ja/xamarin-android/topic/3550/トースト

13: バインディング

Examples

タイプの

Xamarin.Android Bindings GeneratorにJavaタイプをし、バインドしないようにすることはです。これは、 `remove-node` XMLをmetadata.xmlファイルにすることによってわれます。

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

Javaインタフェースの

Javaライブラリに、ユーザーがするのあるインタフェース `View.IOnClickListener`やコールバックなどのクリックリスナーなどがまれている、クラスは`Java.Lang.Object`または`Java.Lang.Throwable`からまたはにするがあります。これはなエラーです。なぜなら、パッケージングのでにとされるがされるからです。

> タイプ 'MyListener'はAndroid.Runtime.IJavaObjectをしますが、Java.Lang.Objectからしません。サポートされていません。

う

このをするとしないがします。

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

    public void Dispose()
    {
    }


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

しい

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

```
        // ...
    }
}
```

バインディングライブラリはメソッドとインタフェースのをすることがあります

バインディングライブラリのすべてがCでJavaとじをつわけではありません。

Cでは、インタフェースは「I」でまりますが、Javaではそのようなはありません。 Javaライブラリをインポートすると、 `ISomeInterface`というのインターフェイスが`SomeInterface`になり `ISomeInterface`。

に、JavaにはCのようなプロパティはありません。ライブラリがバインドされると、Javaのgetterメソッドとsetterメソッドがプロパティとしてリファクタリングされることがあります。たとえば、のJavaコード

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

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

リファクタリングすることができます

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

それがばれているとき。

オンラインでバインディングをむ https://riptutorial.com/ja/xamarin-android/topic/771/バインディング

クレジット

S. No		Contributors
1	Xamarin.Androidをいめる	Amy Burns, Community, Jon Douglas, Kevin Montrose, Ryan Weaver
2	Androidデバイスからキャプチャしたのきをする	Daniel Krzyczkowski
3	RecyclerView	Alexandre, Matthew, Ryan Alford, Sreeraj, Zverev Eugene
4	Xamarin.Android - Bluetooth	Ladislav
5	Xamarin.Android - ツールバーをする	Daniel Krzyczkowski, tylerjgarland
6	Xamarin.Android APKをする	Alexandre
7	XamarinアプリケーションでZXingライブラリをしたバーコードスキャン	GvSharma
8	アプリケーションのライフサイクル - Xamarin.Andorid	CDrosos, Daniel Krzyczkowski, Steven Mark Ford
9	カスタムリストビュー	user3814750
10	ダイアログ	JimBobBennett, Pilatus
11	トースト	GONeale, Matthew, Piet, user2912553
12	バインディング	EJoshuaS, Jon Douglas, jonp, Matthew, Prashant C, Sven-Michael Stübe