無料電子ブック

学習

Xamarin.Forms

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.fo

rms

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

It is an unofficial and free Xamarin.Forms 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.Forms.

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.Formsをいめる

Xamarin.Formsでは、UIコードやXAML UIマークアップなど、のコードをしてiOS、Android、およびWindowsアプリケーションをすることができます。アプリケーションページとビューはプラットフォームのネイティブコントロールにマッピングされますが、プラットフォームのUIをするようにカスタマイズすることも、プラットフォームのにアクセスすることもできます。

バージョン

バージョン	
2.3.1	2016-08-03
2.3.0-hotfix1	2016629
2.3.0	2016616
2.2.0-hotfix1	2016-05-30
2.2.0	2016-04-27
2.1.0	2016-03-13
2.0.1	2016-01-20
2.0.0	2015-11-17
1.5.1	2016-10-20
1.5.0	2016-09-25
1.4.4	2015-07-27
1.4.3	2015-06-30
1.4.2	2015-04-21
1.4.1	2015-03-30
1.4.0	2015-03-09
1.3.5	2015-03-02
1.3.4	2015-02-17
1.3.3	2015-02-09
1.3.2	2015-02-03

バージョン	
1.3.1	2015-01-04
1.3.0	2014-12-24
1.2.3	2014-10-02
1.2.2	2014-07-30
1.2.1	2014-07-14
1.2.0	2014-07-11
1.1.1	2014-06-19
1.1.0	2014-06-12
1.0.1	20140604

Examples

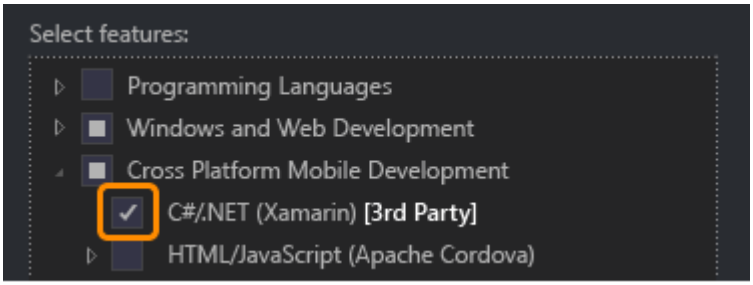
インストールVisual Studio

Xamarin.Formsは、がAndroid、iOS、Windows、およびWindows Phoneでできるユーザーインターフェイスをにできる、クロスプラットフォームのネイティブにサポートされたUIツールキットのツールです。ユーザーインターフェイスは、ターゲットプラットフォームのネイティブコントロールをしてレンダリングされるため、Xamarin.Formsアプリケーションはプラットフォームのなルックアンドフィールをできます。

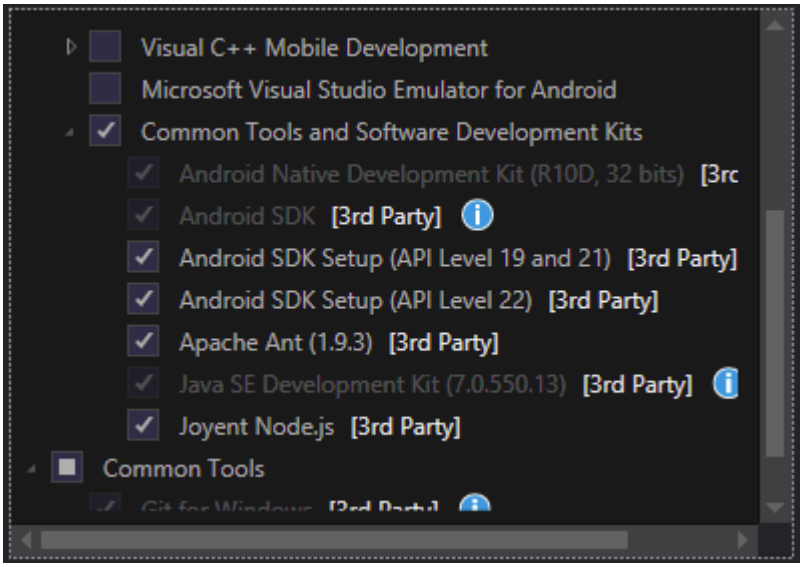
Visual StudioのXamarinプラグイン

Visual StudioのXamarin.Formsをいめるには、Xamarinプラグインがです。それをインストールするもなは、のVisual Studioをダウンロードしてインストールすることです。

のVisual Studioがにインストールされているは、[コントロールパネル]> [プログラムと]にし、Visual Studioをクリックして[]をクリックします。インストーラがいたら、[]をクリックし、クロスプラットフォームのモバイルツールをします。



Android SDKのインストールをすることもできます



すでにSDKがインストールされているは、チェックをしてください。でのAndroid SDKをするようにXamarinをすることができます。

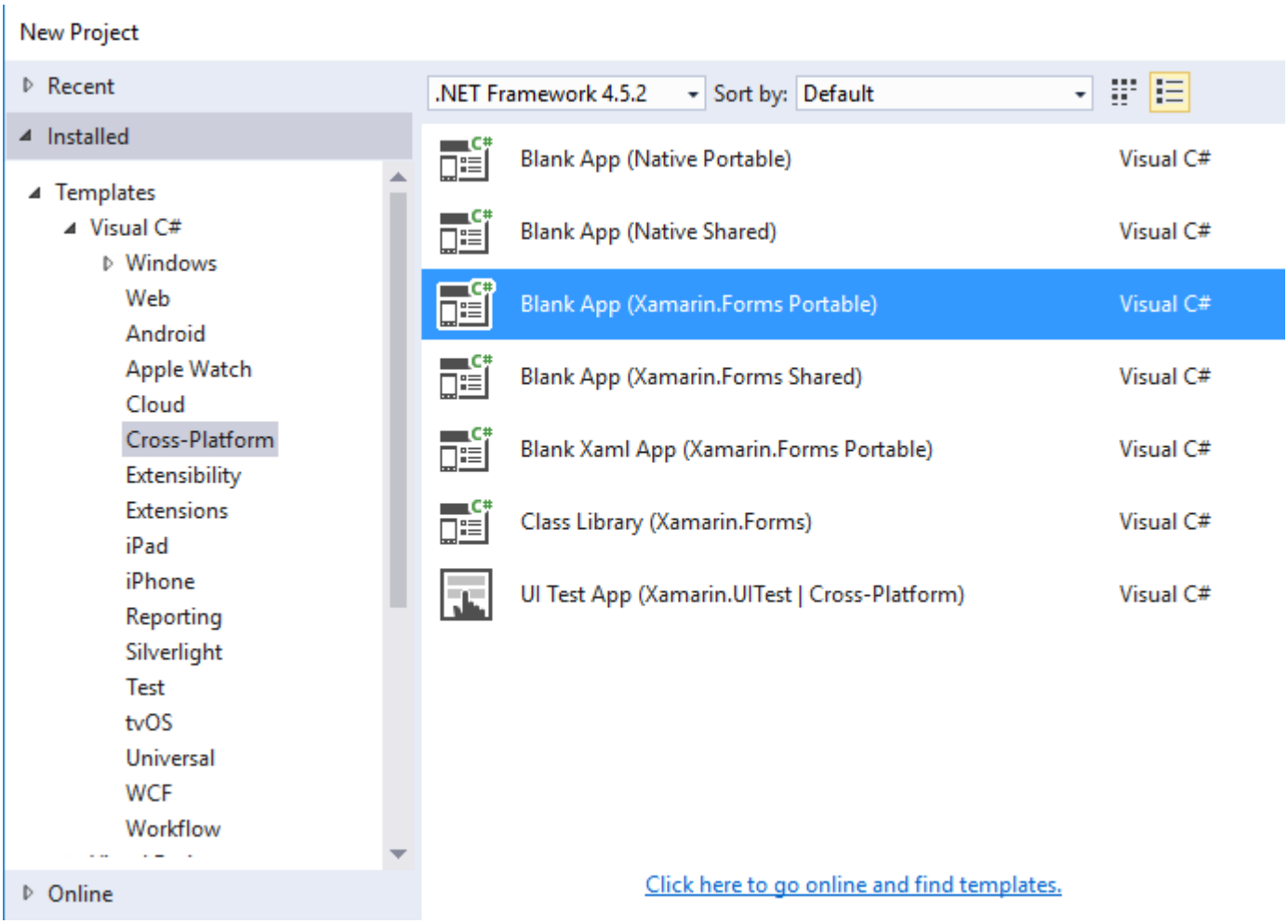
Xamarin.Forms

Xamarin.Formsは、ポータブルクラスライブラリとネイティブアセンブリのライブラリセットです。 Xamarin.Formsライブラリは、NuGetパッケージとしてできます。これをプロジェクトにするには、パッケージマネージャコンソールののの`Install-Package`コマンドをします。

```
Install-Package Xamarin.Forms
```

すべてのアセンブリMyProject、MyProject.Droid、MyProject.iOSなどにされます。

Xamarin.Formsをいめるもなは、Visual Studioでのプロジェクトをすることです。



ごのとおり、のアプリをするための2つのオプション、ポータブルとがあります。はPortableのものをいめることをおめします。なぜなら、でもにされているからですやをするがあります。

プロジェクトをした、のテンプレートにいXamarin.Formsバージョンがまれているがあるので、のXamarin.Formsバージョンをしていることをしてください。のXamarin.Formsにアップグレードするには、パッケージマネージャコンソールまたはNuGetパッケージのオプションをしますNuGetパッケージのみ。

Visual Studio Xamarin.FormsテンプレートはiOSプラットフォームプロジェクトをしますが、これらのプロジェクトをiOSシミュレータまたはデバイスでできるようにするには、XamarinをMacビルドホストにするがあります。

Hello World Xamarinフォーム Visual Studio

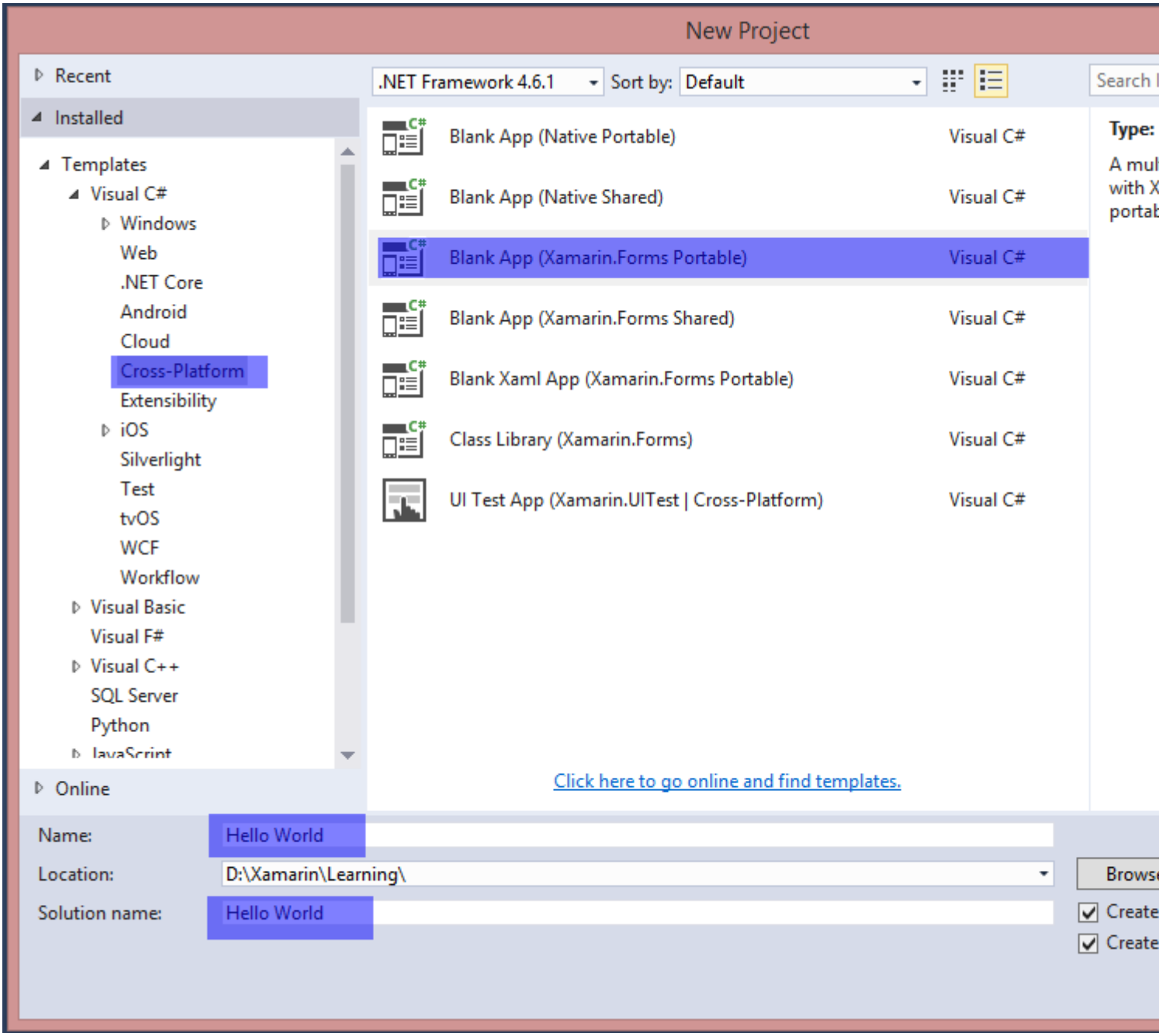
のでしたようにXamarinをにインストールしたら、のサンプルアプリケーションをします。

ステップ1しいプロジェクトをする。

Visual Studioで、「」->「プロジェクト」->「Visual C」->「クロスプラットフォーム」->「のアプリケーション」Xamarin.Forms Portableをします。

アプリケーションに「Hello World」というをけてプロジェクトをするをし、「OK」をクリックします。これは3つのプロジェクトをむあなたのためのソリューションをします

1. HelloWorldこれはロジックとビューがされているポータルプロジェクトです
2. HelloWorld.DroidAndroidプロジェクト
3. HelloWorld.iOSiOSプロジェクト



ステップ2サンプルの

ソリューションがされると、サンプルアプリケーションをすぐにデプロイできるようになります。ポータブルプロジェクトのルートにある`App.cs`き、コードをべます。にすように、サンプルの`Content`は`Label`をむ`StackLayout`です。

```
using Xamarin.Forms;

namespace Hello_World
{
    public class App : Application
    {
        public App()
        {
            // The root page of your application
            MainPage = new ContentPage
            {
                Content = new StackLayout
                {
                    VerticalOptions = LayoutOptions.Center,
                    Children = {
                        new Label {
                            HorizontalTextAlignment = TextAlignment.Center,
                            Text = "Welcome to Xamarin Forms!"
                        }
                    }
                }
            };
        }
        protected override void OnStart()
        {
            // Handle when your app starts
        }
        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }
        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```

ステップ3アプリケーションをする

ここで、するプロジェクト `HelloWorld.Droid`または`HelloWorld.iOS` をにクリックし、「 `Set as StartUp Project`クリックします。に、Visual Studioのツールバーで、[`Start`]ボタン[]ボタンにたのボタンをクリックして、ターゲットのシミュレータ/エミュレータでアプリケーションをします。

オンラインでXamarin.Formsをいめるをむ https://riptutorial.com/ja/xamarin-forms/topic/908/xamarin-formsをいめる

2: CarouselView - プレリリース

CarouselViewはののビューをむことができるXamarinコントロールです。このプレリリースコントロールは、Xamarinフォームプロジェクトでのみできます。

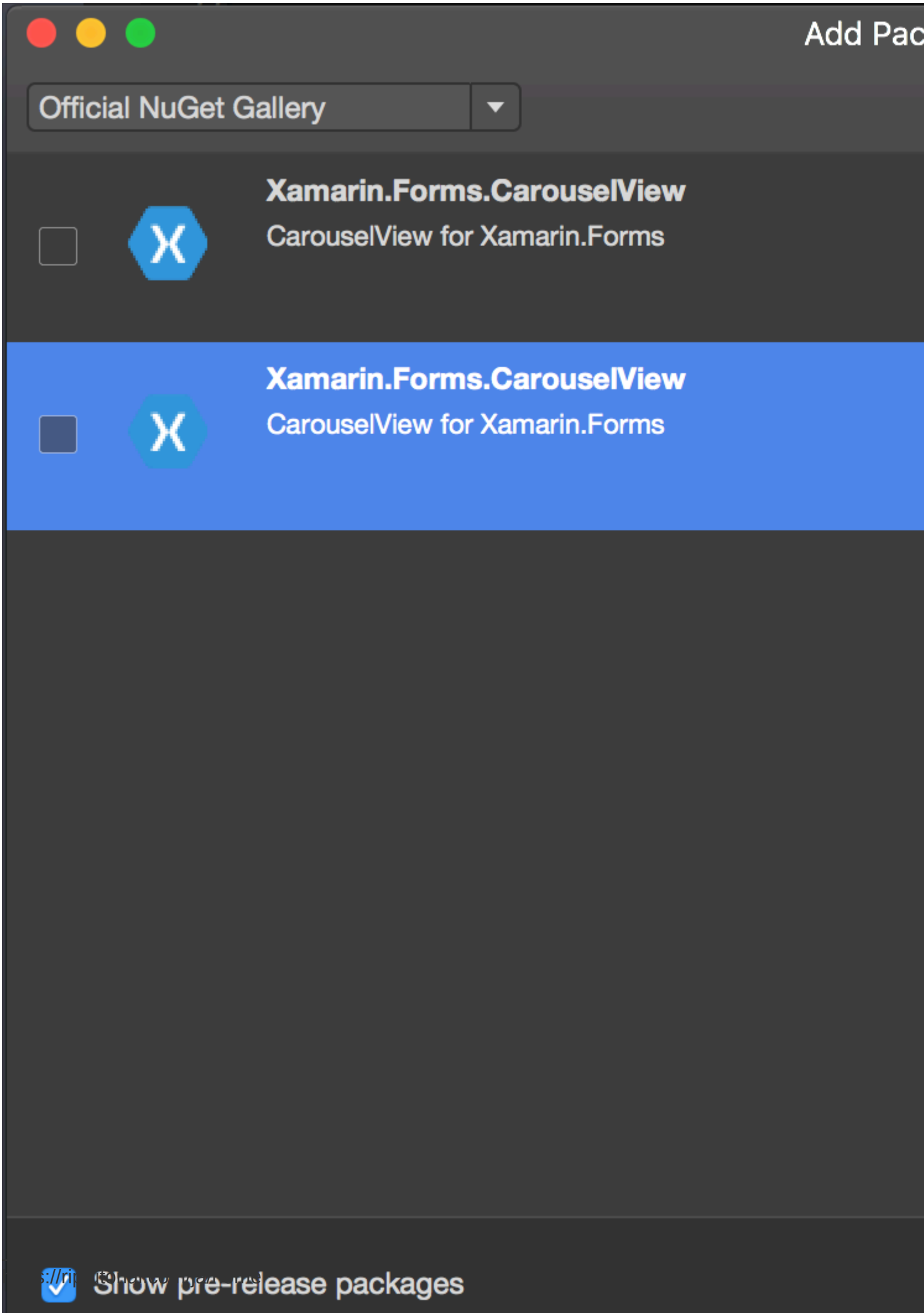
James MontemagnoがするXamarinのブログでは、をするためにCarouselViewがされています。

このでCarouselViewはXamarin.Formsにされていません。プロジェクトでこれをするには、NuGet-Packageをするがありますのを。

Examples

CarouselViewのインポート

CarouselViewをインポートするもなは、Xamarin / Visual StudioのNuGet-Packages Managerをすることです。


Add Pac
Official NuGet Gallery
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Show pre-release packages

にしてください。

サブプロジェクト.iOS / .droid ./ WinPhoneはこのパッケージをインポートするがあります。

CarouselViewをXAMLページにインポートする

ContentPageのしに、のをします。

```
xmlns:cv="clr-namespace:Xamarin.Forms;assembly=Xamarin.Forms.CarouselView"
```

<ContentPage.Content>タグのにCarouselView

```
<cv:CarouselView x:Name="DemoCarouselView">
</cv:CarouselView>
```

xNameはCarouselViewにをけ、Cのコードビハインドファイルでできます。これは、CarouselViewをビューにするためになです。 CarouselViewがであるため、このではもされません。

バインドなソースの

ItemSourceのとして、ObservableCollectionというをします。

```
public ObservableCollection<TechGiant> TechGiants { get; set; }
```

TechGiantはTechnology Giantsのをホストするクラスです

```
public class TechGiant
{
    public string Name { get; set; }

    public TechGiant(string Name)
    {
        this.Name = Name;
    }
}
```

ページのInitializeComponentのに、ObservableCollectionをしてりつぶします

```
TechGiants = new ObservableCollection<TechGiant>();
TechGiants.Add(new TechGiant("Xamarin"));
TechGiants.Add(new TechGiant("Microsoft"));
TechGiants.Add(new TechGiant("Apple"));
TechGiants.Add(new TechGiant("Google"));
```

に、TechGiantsをDemoCarouselViewのItemSourceにします

```
DemoCarouselView.ItemsSource = TechGiants;
```

DataTemplates

XAMLファイルでCarouselViewにDataTemplateをえます

```
<cv:CarouselView.ItemTemplate>
</cv:CarouselView.ItemTemplate>
```

DataTemplateをします。この、これは、itemsourceとのにバインドされたテキストをつラベルになります。

```
<DataTemplate>
    <Label Text="{Binding Name}" BackgroundColor="Green"/>
</DataTemplate>
```

それでおしまいプログラムをし、をしてください

オンラインでCarouselView - プレリリースをむ https://riptutorial.com/ja/xamarin-forms/topic/6094/carouselview----プレリリース

3: DependencyService

`DependencyService`をする、は3つのがです。

- インタフェース - したいをします。
- プラットフォーム - されたインタフェースをするプラットフォームのプロジェクトのクラス。
- - プラットフォームのクラスは、メタデータをして`DependencyService`するがあります。これにより、 `DependencyService`はにをつけることができます。

`DependencyService`をするは、ターゲットプラットフォームごとにをするがあります。がされていない、アプリケーションはにします。

Examples

インタフェース

インタフェースは、 `DependencyService`してするをします。 `DependencyService` 1つのは、テキスト・ツー・スピーチ・サービスである。のところ、Xamarin.Formsではこののがないため、にするがあります。ビヘイビアのインタフェースをすることからめます。

```
public interface ITextToSpeech
{
	void Speak (string whatToSay);
}
```

インタフェースをするので、コードからそのインタフェースをコーディングすることができます。

インタフェースをするクラスでは、 `DependencyService`をするためのパラメータのないコンストラクタがです。

iOSの

したインタフェースは、すべてのプラットフォームでするがあります。 iOSの、これは`AVFoundation`フレームワークによってわれます。の`ITextToSpeech`インターフェイスのでは、のテキストをスピーチします。

```
using AVFoundation;

public class TextToSpeechiOS : ITextToSpeech
{
    public TextToSpeechiOS () {}

    public void Speak (string whatToSay)
    {
```

```
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (whatToSay) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

クラスをしたら、に`DependencyService`をできるようにするがあります。これは、 `[assembly]`をクラスのにし、ののにすることによってわれます。

```
using AVFoundation;
using DependencyServiceSample.iOS;

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechiOS))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechiOS : ITextToSpeech
...
```

このはクラスを`DependencyService`し、 `ITextToSpeech`インターフェイスのインスタンスがなときにできるようにします。

コード

プラットフォームのクラスをしてしたら、それらをコードにフックすることができます。のページには、みのをしてテキストみげをトリガーするボタンがまれています。これは、ネイティブSDKをして、に`DependencyService`をして`ITextToSpeech`プラットフォームのをします。

```
public MainPage ()
{
    var speakButton = new Button {
        Text = "Talk to me baby!",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };

    speakButton.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Xamarin Forms likes eating cake by the
ocean.");
    };

    Content = speakButton;
}
```

このアプリケーションをiOSまたはAndroidデバイスでし、ボタンをタップすると、アプリケーションがのをすのがこえます。

Androidの

Androidのは、ネイティブの`Java.Lang.Object`からするがあり、 `IOnInitListener`インタフエースをするがあるため、 `IOnInitListener`です。 Androidでは、するくのSDKメソッドにしてなAndroidコンテキストをするがあります。 Xamarin.Formsは、そのようなにできるAndroidコンテキストをする`Forms.Context`オブジェクトをします。

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechAndroid : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech _speaker;

    public TextToSpeechAndroid () {}

    public void Speak (string whatToSay)
    {
        var ctx = Forms.Context;

        if (_speaker == null)
        {
            _speaker = new TextToSpeech (ctx, this);
        }
        else
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (whatToSay, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation

    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success))
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }

    #endregion
}
```

クラスをしたら、に`DependencyService`をできるようにするがあります。これは、 `[assembly]`をクラスのにし、ののにすることによってわれます。

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechAndroid))]
namespace DependencyServiceSample.Droid {
```

```
    ...
```

このはクラスを`DependencyService`し、 `ITextToSpeech`インターフェイスのインスタンスがなときにできるようにします。

オンラインでDependencyServiceをむ https://riptutorial.com/ja/xamarin-forms/topic/2508/dependencyservice

4: DependencyServiceをしたネイティブへのアクセス

インプリメンテーションがつからないときにコードをしないようにするには、なインプリメンテーションがあるはまず`DependencyService`します。

`null`でない、これをなチェックでうことができ`null`。

```
var speaker = DependencyService.Get<ITextToSpeech>();

if (speaker != null)
{
    speaker.Speak("Ready for action!");
}
```

IDEがC6をサポートしていて、NULLきをしている

```
var speaker = DependencyService.Get<ITextToSpeech>();

speaker?.Speak("Ready for action!");
```

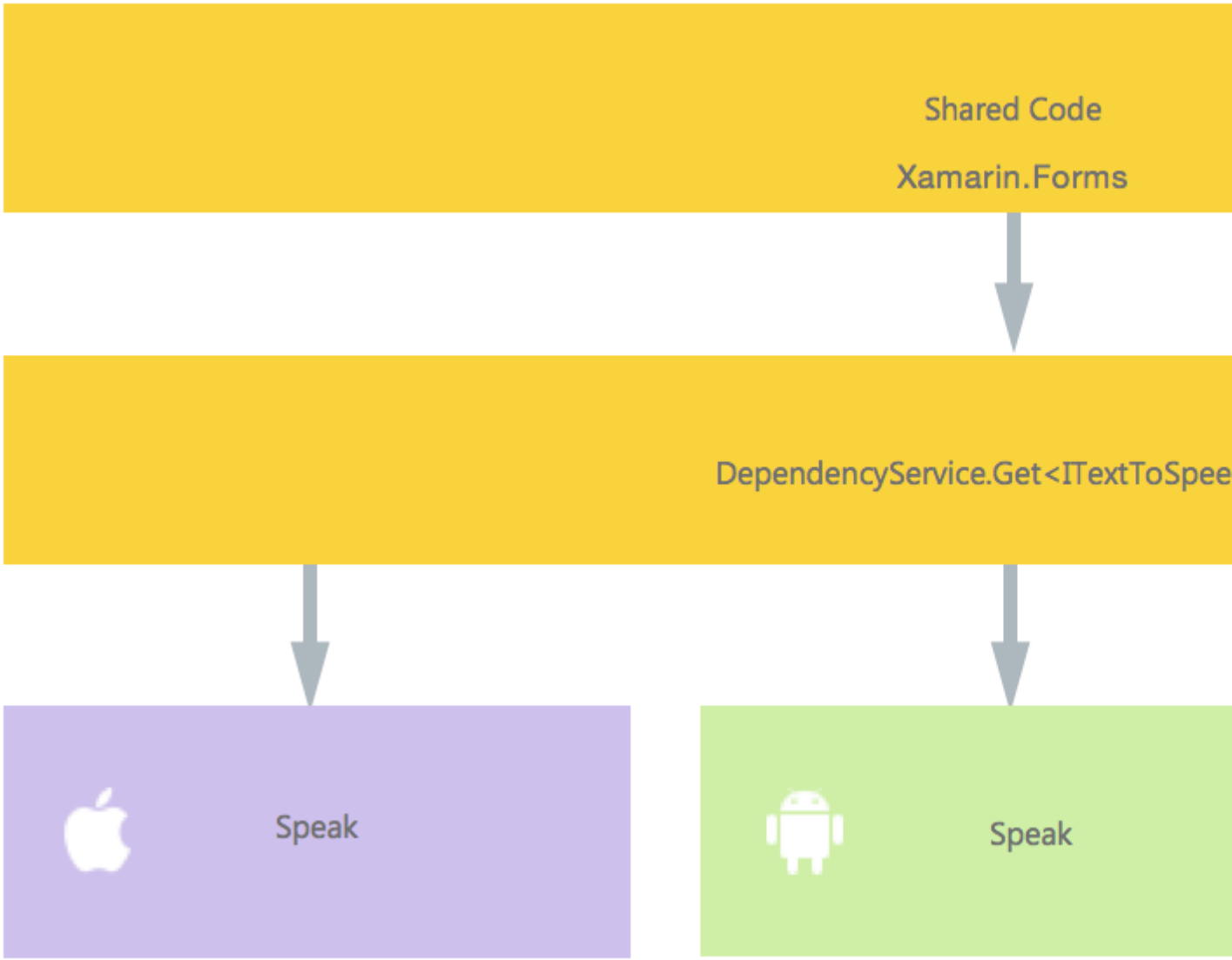
これをわず、にがつからないは、コードによってがされます。

Examples

テキストみげの

プラットフォームのコードをするのいは、テキストみげttsをするです。このでは、PCLライブラリのコードをしていることをとしています。

たちのソリューションのは、ののようになります。



コードでは、 `DependencyService` されているインタフェースをします。これがたちがをするです。にあるようなインターフェースをする。

```
public interface ITextToSpeech
{
    void Speak (string text);
}
```

プラットフォームで、このインタフェースのをするがあります。 iOSのからめましょう。

iOSの

```
using AVFoundation;

public class TextToSpeechImplementation : ITextToSpeech
{
    public TextToSpeechImplementation () {}
```

```
    public void Speak (string text)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (text) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

のコードでは、iOSにのコードがあることにきました。 `AVSpeechSynthesizer`ようなタイプ。これらはコードではしません。

このをXamarin `DependencyService`するには、このをのにします。

```
using AVFoundation;
using DependencyServiceSample.iOS;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechImplementation : ITextToSpeech
//... Rest of code
```

これでコードでこのようなびしをうと、アプリケーションをしているプラットフォームのながわれます。

`DependencyService.Get<ITextToSpeech>()` 。 これについてはでしくします。

Android

このコードのAndroidのは、のようになります。

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechImplementation : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech speaker;
    string toSpeak;

    public TextToSpeechImplementation () {}

    public void Speak (string text)
    {
        var ctx = Forms.Context; // useful for many Android SDK features
```

```
        toSpeak = text;
        if (speaker == null) {
            speaker = new TextToSpeech (ctx, this);
        } else {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation
    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success)) {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }
    #endregion
}
```

もう`DependencyService`にすることをれないでください。

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.Droid{
    //... Rest of code
```

Windows Phoneの

に、Windows Phoneの、このコードをすることができます。

```
public class TextToSpeechImplementation : ITextToSpeech
{
    public TextToSpeechImplementation() {}

    public async void Speak(string text)
    {
        MediaElement mediaElement = new MediaElement();

        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

        SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync("Hello World");

        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
        await synth.SynthesizeTextToStreamAsync(text);
    }
}
```

そしてもうそれをすることをれないでください。

```
using Windows.Media.SpeechSynthesis;
using Windows.UI.Xaml.Controls;
using DependencyServiceSample.WinPhone;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.WinPhone{
    //... Rest of code
```

コードでの

はそれをさせるためにすべてのにありますに、コードで、このをインターフェイスをしてびすことができます。に、されているのプラットフォームにするがされます。

このコードでは、Xamarinフォームプロジェクトにまれるページがされます。 `DependencyService` をして`Speak()`メソッドをびすボタンをします。

```
public MainPage ()
{
    var speak = new Button {
        Text = "Hello, Forms !",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };
    speak.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");
    };
    Content = speak;
}
```

は、アプリケーションがされ、ボタンがクリックされると、されたテキストがされることになります。

コンパイラのヒントなどのようなハードなことをするはありません。これで、プラットフォームにしないコードをして、プラットフォームのにアクセスするされたがしました。

アプリケーションとデバイスのOSバージョンの - AndroidiOS - PCL

のでは、Androidのバージョンと iOSのバージョンにされたデバイスのOSバージョンとアプリケーションのバージョンプロジェクトのプロパティでされていますをします。

まず、PCLプロジェクトでインターフェイスをします。

```
public interface INativeHelper {
    /// <summary>
    /// On iOS, gets the <c>CFBundleVersion</c> number and on Android, gets the
<c>PackageInfo</c>'s <c>VersionName</c>, both of which are specified in their respective
project properties.
    /// </summary>
    /// <returns><c>string</c>, containing the build number.</returns>
    string GetAppVersion();
```

```
    /// <summary>
    /// On iOS, gets the <c>UIDevice.CurrentDevice.SystemVersion</c> number and on Android,
gets the <c>Build.VERSION.Release</c>.
    /// </summary>
    /// <returns><c>string</c>, containing the OS version number.</returns>
    string GetOsVersion();
}
```

AndroidとiOSプロジェクトにインターフェイスをします。

アンドロイド

```
[assembly: Dependency(typeof(NativeHelper_Android))]

namespace YourNamespace.Droid{
    public class NativeHelper_Android : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() {
            Context context = Forms.Context;
            return context.PackageManager.GetPackageInfo(context.PackageName, 0).VersionName;
        }

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return Build.VERSION.Release; }
    }
}
```

iOS

```
[assembly: Dependency(typeof(NativeHelper_iOS))]

namespace YourNamespace.iOS {
    public class NativeHelper_iOS : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() { return
Foundation.NSBundle.MainBundle.InfoDictionary[new
Foundation.NSString("CFBundleVersion")].ToString(); }

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return UIDevice.CurrentDevice.SystemVersion; }
    }
}
```

コードをメソッドでするようになりました。

```
public string GetOsAndAppVersion {
    INativeHelper helper = DependencyService.Get<INativeHelper>();
```

```
    if(helper != null) {
        string osVersion  = helper.GetOsVersion();
        string appVersion = helper.GetBuildNumber()
    }
}
```

オンラインでDependencyServiceをしたネイティブへのアクセスをむ
https://riptutorial.com/ja/xamarin-forms/topic/2409/dependencyserviceをしたネイティブへのアクセス

5: MessagingCenter

き

Xamarin.Formsには、されたコードをするメッセージングメカニズムがみまれています。このように、ビューモデルとのコンポーネントはおいをるはありません。らはなメッセージングでることができます。

`MessagingCenter`をするためのなは、に2つあり`MessagingCenter`。

する;のをつメッセージをリッスンし、メッセージがされたときにコードをします。メッセージにはのサブスクライバをめることができます。

します。がするためのメッセージをする。

Examples

な

ここでは、Xamarin.FormsでMessagingCenterをするなをていきます。

まず、メッセージをしてみましょう。で`FooMessaging`モデル々はからたメッセージをサブスクライブ`MainPage`。メッセージは「Hi」でなければならず、けったときにプロパティ`Greeting`をするハンドラをします。に、の`FooMessaging`インスタンスがこのメッセージにして`this`ことをします。

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage> (this, "Hi", (sender) => {
            this.Greeting = "Hi there!";
        });
    }
}
```

このをするメッセージをするには、`MainPage`というページがあり、そのにコードをするがあります。

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage> (this, "Hi");
    }
}
```

たちの`MainPage`には、メッセ―ジをするハンドラがあるボタンがあります。 `this`は`MainPage`インスタンスでなければなりません。

をす

また、メッセ―ジをとしてすこともできます。

ののクラスをってします。では、 `Subscribe`メソッドびしのに、あなたがしているのをします。また、ハンドラシグニチャのもしてください。

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage, string> (this, "Hi", (sender, arg) => {
            this.Greeting = arg;
        });
    }
}
```

メッセ―ジをするは、ずのをめてください。また、 `Send`メソッドのすぐろにをし、のをします。

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage, string> (this, "Hi", "Hi there!");
    }
}
```

このではながされていますが、のタイプのなオブジェクトもできます。

サブスクライブ

メッセ―ジをけるがなくなったときに、にすることができます。あなたはのようにすることができます

```
MessagingCenter.Unsubscribe<MainPage> (this, "Hi");
```

をするときは、のようになをするがあります。

```
MessagingCenter.Unsubscribe<MainPage, string> (this, "Hi");
```

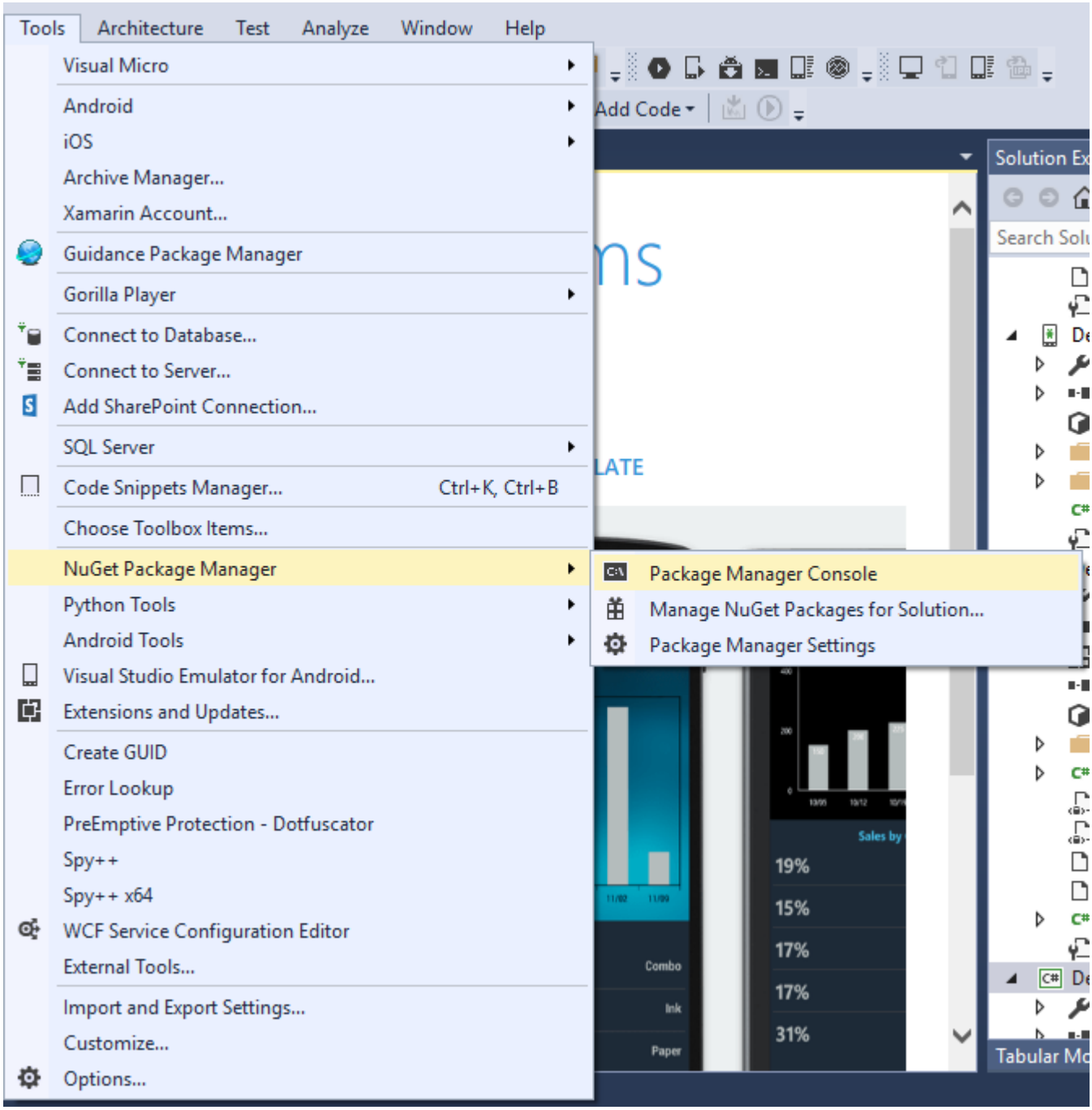
オンラインでMessagingCenterをむ https://riptutorial.com/ja/xamarin-forms/topic/9672/messagingcenter

6: OAuth2

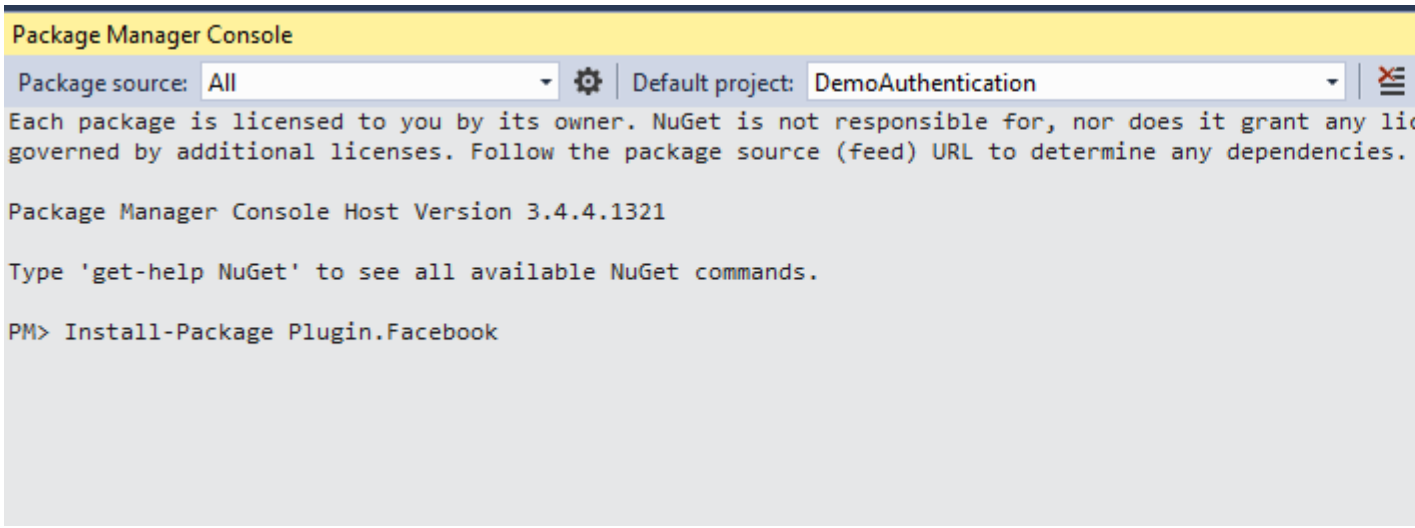
Examples

プラグインによる

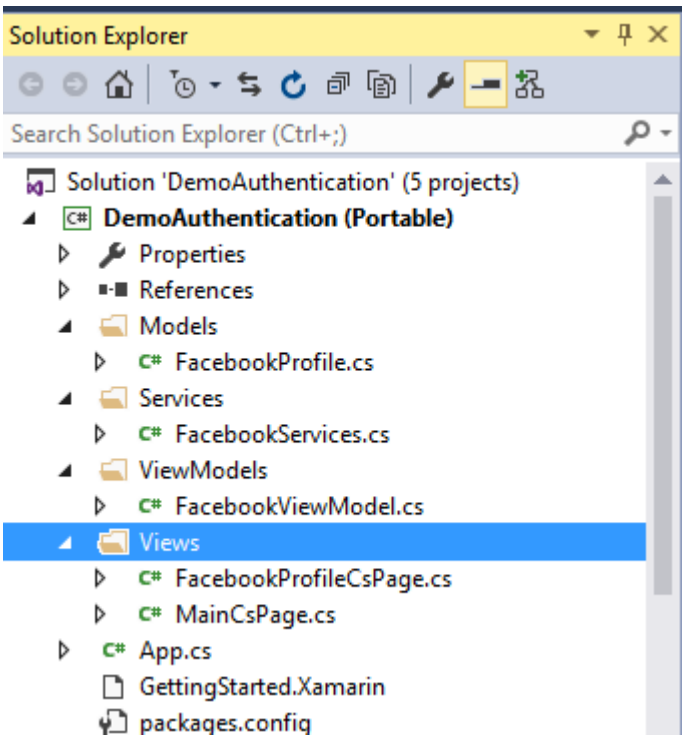
1. まず、「ツール」>「**NuGet**パッケージマネージャー」>「パッケージマネージャーコンソール」にアクセスします。



2. Package Manger Consoleでこのコマンド " **Install-Package Plugin.Facebook** " をします。



3. これですべてのファイルがにされます。



ビデオ XamarinフォームでFacebookでログイン

プラグインによるその2にすように、パッケージマネージャコンソールでコマンドをしてください。

1. **Youtube** インストールパッケージプラグイン。Youtube
2. **Twitter** Install-Package Plugin.Twitter
3. フォースクエア Install-Package Plugin.Foursquare
4. **Google** Install-Package Plugin.Google
5. **Instagram** Install-Package Plugin.Instagram
6. **Eventbrite** Install-Package Plugin.Eventbrite

オンラインでOAuth2をむ https://riptutorial.com/ja/xamarin-forms/topic/8828/oauth2

7: Xamarin Plugin

Examples

プラグイン

メッセージやリンクをしたり、テキストをクリップボードにコピーしたり、XamarinやWindowsアプリケーションでブラウザをいたりするな。

NuGetで https : //www.nuget.org/packages/Plugin.Share/

XAML

```
<StackLayout Padding="20" Spacing="20">
      <Button StyleId="Text" Text="Share Text" Clicked="Button_OnClicked"/>
      <Button StyleId="Link" Text="Share Link" Clicked="Button_OnClicked"/>
      <Button StyleId="Browser" Text="Open Browser" Clicked="Button_OnClicked"/>
      <Label Text=""/>

 </StackLayout>
```

C

```
 async void Button_OnClicked(object sender, EventArgs e)
        {
            switch (((Button)sender).StyleId)
            {
                case "Text":
                    await CrossShare.Current.Share("Follow @JamesMontemagno on Twitter",
"Share");
                    break;
                case "Link":
                    await CrossShare.Current.ShareLink("http://motzcod.es", "Checkout my
blog", "MotzCod.es");
                    break;
                case "Browser":
                    await CrossShare.Current.OpenBrowser("http://motzcod.es");
                    break;
            }
        }
```

ExternalMaps

プラグインをいて、のやにします。 iOSでナビゲーションオプションをしてするオプション。

NuGetで[https://www.nuget.org/packages/Xam.Plugin.ExternalMaps/] [1]

XAML

```
  <StackLayout Spacing="10" Padding="10">
```

```
      <Button x:Name="navigateAddress" Text="Navigate to Address"/>
      <Button x:Name="navigateLatLong" Text="Navigate to Lat|Long"/>
      <Label Text=""/>

    </StackLayout>
```

コード

```
namespace PluginDemo
{
    public partial class ExternalMaps : ContentPage
    {
        public ExternalMaps()
        {
            InitializeComponent();
            navigateLatLong.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Space Needle", 47.6204, -122.3491);
            };

            navigateAddress.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Xamarin", "394 pacific ave.", "San
Francisco", "CA", "94111", "USA", "USA");
            };
        }
    }
}
```

ジオロケーターープラグイン

Xamarin.iOS、Xamarin.Android、Windowsのジオロケーションににアクセスできます。

なナゲット[https://www.nuget.org/packages/Xam.Plugin.Geolocator/] [1]

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Button x:Name="buttonGetGPS" Text="Get GPS"/>
      <Label x:Name="labelGPS"/>
      <Button x:Name="buttonTrack" Text="Track Movements"/>
      <Label x:Name="labelGPSTrack"/>
      <Label Text=""/>

    </StackLayout>
```

コード

```
namespace PluginDemo
{
    public partial class GeolocatorPage : ContentPage
    {
        public GeolocatorPage()
        {
            InitializeComponent();
```

```
            buttonGetGPS.Clicked += async (sender, args) =>
            {
                try
                {
                    var locator = CrossGeolocator.Current;
                    locator.DesiredAccuracy = 1000;
                    labelGPS.Text = "Getting gps";

                    var position = await locator.GetPositionAsync(timeoutMilliseconds: 10000);

                    if (position == null)
                    {
                        labelGPS.Text = "null gps :(";
                        return;
                    }
                    labelGPS.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2}
\nAltitude: {3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                        position.Timestamp, position.Latitude, position.Longitude,
                        position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            buttonTrack.Clicked += async (object sender, EventArgs e) =>
            {
                try
                {
                    if (CrossGeolocator.Current.IsListening)
                    {
                        await CrossGeolocator.Current.StopListeningAsync();
                        labelGPSTrack.Text = "Stopped tracking";
                        buttonTrack.Text = "Stop Tracking";
                    }
                    else
                    {
                        if (await CrossGeolocator.Current.StartListeningAsync(30000, 0))
                        {
                            labelGPSTrack.Text = "Started tracking";
                            buttonTrack.Text = "Track Movements";
                        }
                    }
                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
```

```
            try
            {
                CrossGeolocator.Current.PositionChanged +=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError +=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }

        void CrossGeolocator_Current_PositionError(object sender,
Plugin.Geolocator.Abstractions.PositionErrorEventArgs e)
        {

            labelGPSTrack.Text = "Location error: " + e.Error.ToString();
        }

        void CrossGeolocator_Current_PositionChanged(object sender,
Plugin.Geolocator.Abstractions.PositionEventArgs e)
        {
            var position = e.Position;
            labelGPSTrack.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2} \nAltitude:
{3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                position.Timestamp, position.Latitude, position.Longitude,
                position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);


        }

        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            try
            {
                CrossGeolocator.Current.PositionChanged -=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError -=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }
    }
}
```

メディアプラグイン

クロスプラットフォームAPIからやをするか、します。

なナゲット[https://www.nuget.org/packages/Xam.Plugin.Media/] [1]

XAML

```
<StackLayout Spacing="10" Padding="10">
```

```
      <Button x:Name="takePhoto" Text="Take Photo"/>
      <Button x:Name="pickPhoto" Text="Pick Photo"/>
      <Button x:Name="takeVideo" Text="Take Video"/>
      <Button x:Name="pickVideo" Text="Pick Video"/>
      <Label Text="Save to Gallery"/>
      <Switch x:Name="saveToGallery" IsToggled="false" HorizontalOptions="Center"/>
      <Label Text="Image will show here"/>
      <Image x:Name="image"/>
      <Label Text=""/>

    </StackLayout>
```

コード

```
namespace PluginDemo
{
    public partial class MediaPage : ContentPage
    {
        public MediaPage()
        {
            InitializeComponent();
            takePhoto.Clicked += async (sender, args) =>
            {

                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.TakePhotoAsync(new
Plugin.Media.Abstractions.StoreCameraMediaOptions
                    {
                        Directory = "Sample",
                        Name = "test.jpg",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("File Location", (saveToGallery.IsToggled ?
file.AlbumPath : file.Path), "OK");

                    image.Source = ImageSource.FromStream(() =>
                    {
                        var stream = file.GetStream();
                        file.Dispose();
                        return stream;
                    });
                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
```

```
            pickPhoto.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickPhotoSupported)
                {
                    await DisplayAlert("Photos Not Supported", ":( Permission not granted to
photos.", "OK");
                    return;
                }
                try
                {
                    Stream stream = null;
                    var file = await CrossMedia.Current.PickPhotoAsync().ConfigureAwait(true);


                    if (file == null)
                        return;

                    stream = file.GetStream();
                    file.Dispose();

                    image.Source = ImageSource.FromStream(() => stream);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            takeVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakeVideoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }

                try
                {
                    var file = await CrossMedia.Current.TakeVideoAsync(new
Plugin.Media.Abstractions.StoreVideoOptions
                    {
                        Name = "video.mp4",
                        Directory = "DefaultVideos",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("Video Recorded", "Location: " +
(saveToGallery.IsToggled ? file.AlbumPath : file.Path), "OK");

                    file.Dispose();

                }
                catch //(Exception ex)
```

```
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            pickVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickVideoSupported)
                {
                    await DisplayAlert("Videos Not Supported", ":( Permission not granted to
videos.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.PickVideoAsync();

                    if (file == null)
                        return;

                    await DisplayAlert("Video Selected", "Location: " + file.Path, "OK");
                    file.Dispose();

                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                    //await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }
    }
}
```

メッセージングプラグイン

XamarinとWindowsのためのメッセージングプラグインで、をかけたり、SMSをしたり、さまざまなモバイルプラットフォームのデフォルトのメッセージングアプリケーションをってメールをしたりできます。

なナゲット[https://www.nuget.org/packages/Xam.Plugins.Messaging/] [1]

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Entry Placeholder="Phone Number" x:Name="phone"/>
      <Button x:Name="buttonSms" Text="Send SMS"/>
      <Button x:Name="buttonCall" Text="Call Phone Number"/>
      <Entry Placeholder="E-mail Address" x:Name="email"/>
      <Button x:Name="buttonEmail" Text="Send E-mail"/>
      <Label Text=""/>

    </StackLayout>
```

コード

```
namespace PluginDemo
{
    public partial class MessagingPage : ContentPage
    {
        public MessagingPage()
        {
            InitializeComponent();
            buttonCall.Clicked += async (sender, e) =>
            {
                try
                {
                    // Make Phone Call
                    var phoneCallTask = MessagingPlugin.PhoneDialer;
                    if (phoneCallTask.CanMakePhoneCall)
                        phoneCallTask.MakePhoneCall(phone.Text);
                    else
                        await DisplayAlert("Error", "This device can't place calls", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonSms.Clicked += async (sender, e) =>
            {
                try
                {

                    var smsTask = MessagingPlugin.SmsMessenger;
                    if (smsTask.CanSendSms)
                        smsTask.SendSms(phone.Text, "Hello World");
                    else
                        await DisplayAlert("Error", "This device can't send sms", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonEmail.Clicked += async (sender, e) =>
            {
                try
                {
                    var emailTask = MessagingPlugin.EmailMessenger;
                    if (emailTask.CanSendEmail)
                        emailTask.SendEmail(email.Text, "Hello there!", "This was sent from
the Xamrain Messaging Plugin from shared code!");
                    else
                        await DisplayAlert("Error", "This device can't send emails", "OK");
                }
                catch
                {
//await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };
        }
    }
```

```
}
```

パーミッションプラグイン

ユーザーがiOSとAndroidのなグループのをまたはしているかどうかをします。

さらに、なクロスプラットフォーム/APIをしてアクセスをすることもできます。

なNuget https://www.nuget.org/packages/Plugin.Permissions ここにリンクのをします **XAML**

XAML

```
 <StackLayout Padding="30" Spacing="10">
      <Button Text="Get Location" Clicked="Button_OnClicked"></Button>
      <Label x:Name="LabelGeolocation"></Label>
      <Button Text="Calendar" StyleId="Calendar"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Camera" StyleId="Camera" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Contacts" StyleId="Contacts"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Microphone" StyleId="Microphone"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Phone" StyleId="Phone" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Photos" StyleId="Photos" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Reminders" StyleId="Reminders"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sensors" StyleId="Sensors" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sms" StyleId="Sms" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Storage" StyleId="Storage" Clicked="ButtonPermission_OnClicked"></Button>
      <Label Text=""/>

    </StackLayout>
```

コード

```
bool busy;
        async void ButtonPermission_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            var status = PermissionStatus.Unknown;
            switch (((Button)sender).StyleId)
            {
                case "Calendar":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Calendar);
                    break;
                case "Camera":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Camera);
                    break;
                case "Contacts":
```

```
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Contacts);
                    break;
                case "Microphone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Microphone);
                    break;
                case "Phone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Phone);
                    break;
                case "Photos":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Photos);
                    break;
                case "Reminders":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Reminders);
                    break;
                case "Sensors":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sensors);
                    break;
                case "Sms":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sms);
                    break;
                case "Storage":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Storage);
                    break;
            }

            await DisplayAlert("Results", status.ToString(), "OK");

            if (status != PermissionStatus.Granted)
            {
                switch (((Button)sender).StyleId)
                {
                    case "Calendar":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Calendar))[Permission.Calendar];
                        break;
                    case "Camera":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Camera))[Permission.Camera];
                        break;
                    case "Contacts":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Contacts))[Permission.Contacts];
                        break;
                    case "Microphone":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Microphone))[Permission.Microphone];

                        break;
                    case "Phone":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Phone))[Permission.Phone];
                        break;
                    case "Photos":
```

```
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Photos))[Permission.Photos];
                        break;
                    case "Reminders":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Reminders))[Permission.Reminders];
                        break;
                    case "Sensors":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sensors))[Permission.Sensors];
                        break;
                    case "Sms":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sms))[Permission.Sms];
                        break;
                    case "Storage":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Storage))[Permission.Storage];
                        break;
                }

                await DisplayAlert("Results", status.ToString(), "OK");

            }

            busy = false;
            ((Button)sender).IsEnabled = true;
        }

        async void Button_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            try
            {
                var status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);
                if (status != PermissionStatus.Granted)
                {
                    if (await
CrossPermissions.Current.ShouldShowRequestPermissionRationaleAsync(Permission.Location))
                    {
                        await DisplayAlert("Need location", "Gunna need that location", "OK");
                    }

                    var results = await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Location);
                    status = results[Permission.Location];
                }

                if (status == PermissionStatus.Granted)
                {
                    var results = await CrossGeolocator.Current.GetPositionAsync(10000);
                    LabelGeolocation.Text = "Lat: " + results.Latitude + " Long: " +
results.Longitude;
                }
                else if (status != PermissionStatus.Unknown)
```

```
                {
                    await DisplayAlert("Location Denied", "Can not continue, try again.",
"OK");
                }
            }
            catch (Exception ex)
            {

                LabelGeolocation.Text = "Error: " + ex;
            }

            ((Button)sender).IsEnabled = true;
            busy = false;
        }
```

オンラインでXamarin Pluginをむ https://riptutorial.com/ja/xamarin-forms/topic/7017/xamarin-plugin

8: Xamarin.Formsセル

Examples

エントリーセル

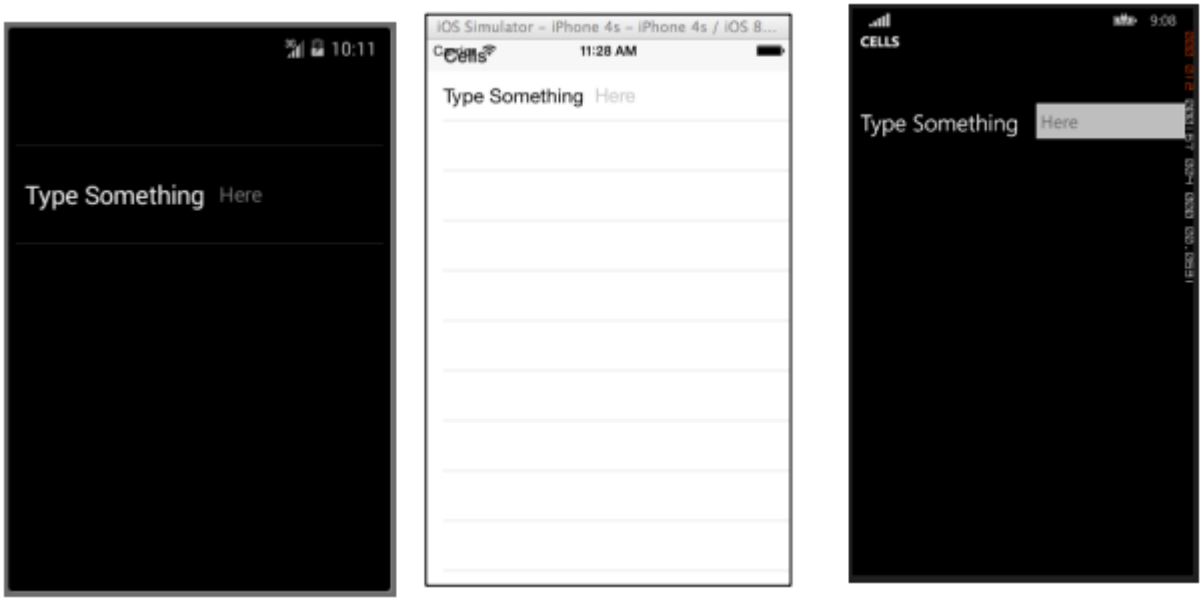
EntryCellは、LabelとEntryのをみわせたCellです。 EntryCellは、アプリケーションでユーザーからデータをするためのをするにちます。それらはにTableViewにすることができ、なフォームとしてうことができます。

XAML

```
<EntryCell Label="Type Something"
Placeholder="Here"/>
```

コード

```
var entryCell = new EntryCell {
Label = "Type Something",
Placeholder = "Here"
};
```



SwitchCell

SwitchCellは、LabelとOn-OffスイッチのをみわせたCellです。 SwitchCellは、やユーザーのやオプションのオン/オフにです。

XAML

```
<SwitchCell Text="Switch It Up!" />
```

コード

```
var switchCell = new SwitchCell {
Text = "Switch It Up!"
};
```

TextCell

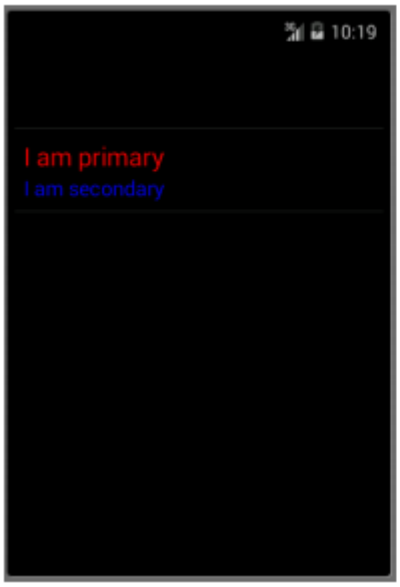
TextCellは、データをするための2つの々のテキストをつセルです。 TextCellは、、、TableViewコントロールとListViewコントロールのでされます。 2つのテキストは、セルのスペースをにするようににされます。このタイプのセルは、データをするためにもよくされるため、ユーザーがこのセルをタップするとのページにします。

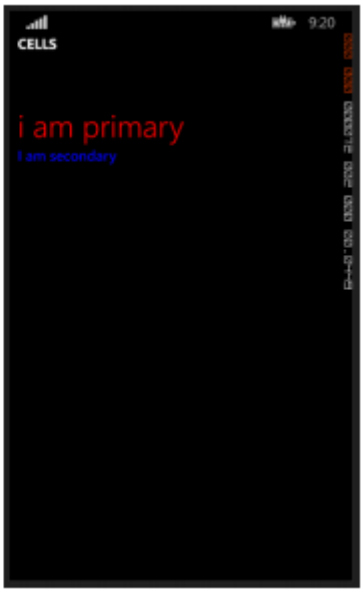
XAML

```
<TextCell Text="I am primary"
TextColor="Red"
Detail="I am secondary"
DetailColor="Blue"/>
```

コード

```
var textCell = new TextCell {
Text = "I am primary",
TextColor = Color.Red,
Detail = "I am secondary",
DetailColor = Color.Blue
};
```





ImageCell

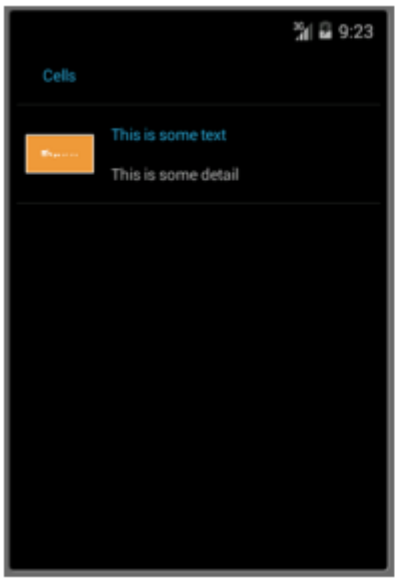
ImageCellはまさにそのようなものです。だけをむシンプルなセルです。このコントロールはのImageコントロールとによくていますが、やははるかにないです。

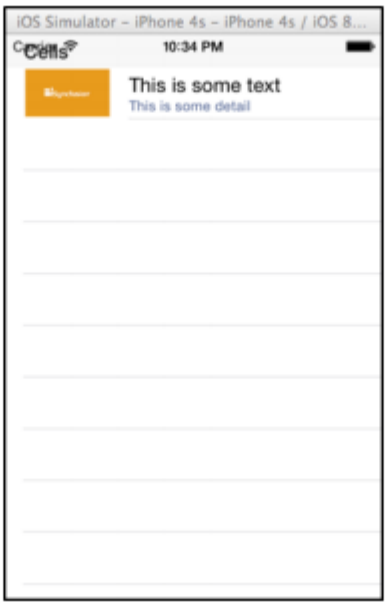
XAML

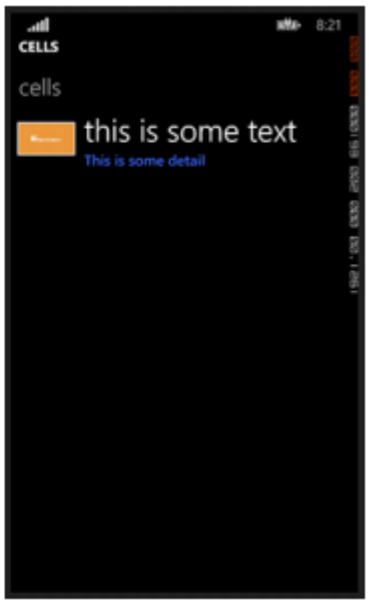
```
<ImageCell ImageSource="http://d2g29cya9iq7ip.cloudfront.net/content/imag
es/company/aboutus-video-bg.png?v=25072014072745")),
 Text="This is some text"
 Detail="This is some detail" />
```

コード

```
var imageCell = new ImageCell {
ImageSource = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.clou
109
dfront.net/content/images/company/aboutus-videobg.png?v=25072014072745")),
 Text = "This is some text",
 Detail = "This is some detail"
};
```







ViewCell

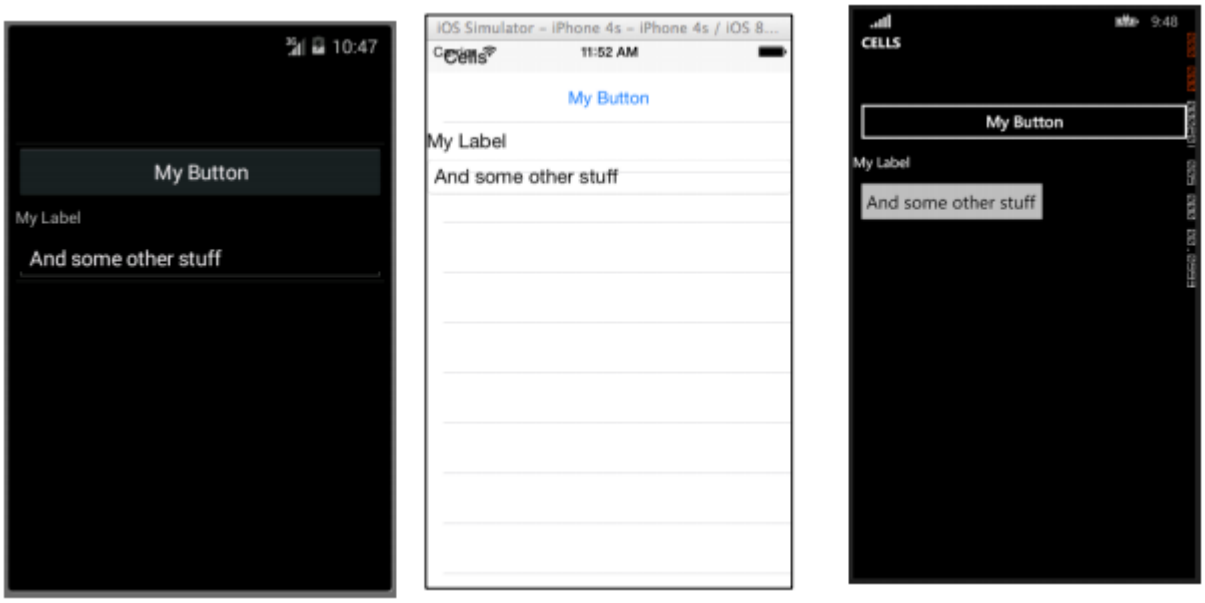
ViewCellはのスレートとえることができます。あなたがむようににえるCellをするのはあなたのなキャンバスです。レイアウトコントロールとにかれたののViewオブジェクトのインスタンスですることもできます。あなたはによってされます。そしておそらくのサイズ。

XAML

```
<ViewCell>
<ViewCell.View>
<StackLayout>
<Button Text="My Button"/>

<Label Text="My Label"/>
<Entry Text="And some other stuff"/>
</StackLayout>
</ViewCell.View>
</ViewCell>
```

コード

```
var button = new Button { Text = "My Button" };
var label = new Label { Text = "My Label" };
var entry = new Entry { Text ="And some other stuff" };
var viewCell = new ViewCell {
View = new StackLayout {
Children = { button, label, entry }
}
};
```



オンラインでXamarin.Formsセルをむ https://riptutorial.com/ja/xamarin-forms/topic/7370/xamarin-formsセル

9: Xamarin.FormsでのBDDユニットテスト

- このライブラリでにするDIコンテナ/リゾルバはAutofacです。
- テストフレームワークはNUnit 3xです。
- Xamarin.Formsフレームワークでこのライブラリをできるはずです
- ソースとサンプルプロジェクトはこちらからできます

Examples

NUnit Test Runnerをしてコマンドとナビゲーションをテストするためのな **Specflow**

なぜたちはこれがなのですか

Xamarin.Formsでユニットテストをうのは、プラットフォームランナーであるため、テストはiOS、Android、Windows、またはMacのUIでするがあります。IDEでテストをする

XamarinはXamarin.TestCloudのらしいUIテストもしますが、テストランナーやローカルビルドサーバーでにしているに、BDDのをし、ViewModelsとCommandをテストするをえている、みみの。

は、Xamarin.FormsでSpecflowをして、アプリケーションにされるMVVMフレームワーク XLabs、 MVVMCross、 Prismなど とはに 、ScenariosからViewModelまでのをにできるライブラリをしました。

BDDをめておいのは、 [Specflow out]をオンにします。

- まだインストールしていないは、ここからまたはVisual Studio IDEからspecflow Visual Studioをインストールしてください https://visualstudiogallery.msdn.microsoft.com/c74211e7-cb6e-4dfa-855d-df0ad4a37dd6
- クラスライブラリをXamarin.Formsプロジェクトにします。それがあなたのテストプロジェクトです。
- SpecFlow.Xamarin.Formsパッケージをしますnugetテストプロジェクトに。
- 'TestApp'をするテストプロジェクトにクラスをし、ビュー/ビューモデルのペアをするとともに、のようにDIをします。

```
public class DemoAppTest : TestApp
{
    protected override void SetViewModelMapping()
```

```
        {
            TestViewFactory.EnableCache = false;

            // register your views / viewmodels below
            RegisterView<MainPage, MainViewModel>();
        }

        protected override void InitialiseContainer()
        {
            // add any di registration here
            // Resolver.Instance.Register<TInterface, TType>();
            base.InitialiseContainer();
        }
    }
```

- Specflowフックをするために、SetupHookクラスをテストプロジェクトにします。でしたクラスと、あなたのアプリケーションののviewmodelをして、のようにテストアプリケーションをブートストラップするがあります

```
    [Binding]
    public class SetupHooks : TestSetupHooks
    {
        /// <summary>
        ///     The before scenario.
        /// </summary>
        [BeforeScenario]
        public void BeforeScenario()
        {
            // bootstrap test app with your test app and your starting viewmodel
            new TestAppBootstrap().RunApplication<DemoAppTest, MainViewModel>();
        }
    }
```

- xamarin.formsビューのコードビヘイビアにcatchブロックをして、xamarin.formsフレームワークをしてアプリをするがありますもしたくない

```
        public YourView()
        {
            try
            {
                InitializeComponent();
            }
            catch (InvalidOperationException soe)
            {
                if (!soe.Message.Contains("MUST"))
                    throw;
            }
        }
```

- プロジェクトにspecflowをするvs specflowにのvs specflowテンプレートを
- TestStepBaseをするステップ·クラスを/し、scenarioContextパラメータをベースにします。
- ナビゲーションサービスやヘルパーをして、ナビゲーション、コマンドの、ビューモデルの

テストをいます。

```
  [Binding]
    public class GeneralSteps : TestStepBase
    {
        public GeneralSteps(ScenarioContext scenarioContext)
            : base(scenarioContext)
        {
            // you need to instantiate your steps by passing the scenarioContext to the base
        }

        [Given(@"I am on the main view")]
        public void GivenIAmOnTheMainView()
        {
            Resolver.Instance.Resolve<INavigationService>().PushAsync<MainViewModel>();


Resolver.Instance.Resolve<INavigationService>().CurrentViewModelType.ShouldEqualType<MainViewModel>();

        }

        [When(@"I click on the button")]
        public void WhenIClickOnTheButton()
        {
            GetCurrentViewModel<MainViewModel>().GetTextCommand.Execute(null);
        }

        [Then(@"I can see a Label with text ""(.*)""")]
        public void ThenICanSeeALabelWithText(string text)
        {
            GetCurrentViewModel<MainViewModel>().Text.ShouldEqual(text);
        }
    }
```

MVVMのない

のをするには、アプリケーションでするナビゲーションステートメントをテストするために、ViewModelにナビゲーションへのフックをするがあります。これをするには

- パッケージSpecFlow.Xamarin.Forms.IViewModelをnugetあなたのPCL Xamarin.Formsプロジェクトに
- ViewModelでIViewModelインターフェイスをします。これにより、にXamarin.Forms INavigationプロパティがされます。
- `public class MainViewModel : INotifyPropertyChanged, IViewModel.IViewModel { public INavigation Navigation { get; set; }`
- テストフレームワークはこれをし、ナビゲーションをします
- XLabs 、 MVVMCross 、 Prismなど、MVVMフレームワークをアプリケーションにできますIViewModelインターフェイスがViewModelにされているり、フレームワークはそれをします。

オンラインでXamarin.FormsでのBDDユニットテストをむ https://riptutorial.com/ja/xamarin-forms/topic/6172/xamarin-formsでのbddユニットテスト

10: Xamarin.Formsでのナビゲーション

Examples

NavigationPageフロー

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public class App : Application
    {
        public App()
        {
            MainPage = new NavigationPage(new FirstPage());
        }
    }

    public class FirstPage : ContentPage
    {
        Label FirstPageLabel { get; set; } = new Label();

        Button FirstPageButton { get; set; } = new Button();

        public FirstPage()
        {
            Title = "First page";

            FirstPageLabel.Text = "This is the first page";
            FirstPageButton.Text = "Navigate to the second page";
            FirstPageButton.Clicked += OnFirstPageButtonClicked;

            var content = new StackLayout();
            content.Children.Add(FirstPageLabel);
            content.Children.Add(FirstPageButton);

            Content = content;
        }

        async void OnFirstPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }
    }

    public class SecondPage : ContentPage
    {
        Label SecondPageLabel { get; set; } = new Label();

        public SecondPage()
        {
            Title = "Second page";

            SecondPageLabel.Text = "This is the second page";
```

```
            Content = SecondPageLabel;
        }
    }
}
```

XAMLをしたNavigationPageフロー

App.xaml.csファイルApp.xamlファイルはデフォルトでスキップされます

```
using Xamrin.Forms

namespace NavigationApp
{
    public partial class App : Application
    {
        public static INavigation GlobalNavigation { get; private set; }

        public App()
        {
            InitializeComponent();
            var rootPage = new NavigationPage(new FirstPage());

            GlobalNavigation = rootPage.Navigation;

            MainPage = rootPage;
        }
    }
}
```

FirstPage.xamlファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="NavigationApp.FirstPage"
    Title="First page">
    <ContentPage.Content>
        <StackLayout>
            <Label
                Text="This is the first page" />
            <Button
                Text="Click to navigate to a new page"
                Clicked="GoToSecondPageButtonClicked"/>
            <Button
                Text="Click to open the new page as modal"
                Clicked="OpenGlobalModalPageButtonClicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

によっては、のナビゲーションではなくグローバルなページでしいページをくがあります。たとえば、のページにボトムメニューがあるは、のナビゲーションでしいページをプッシュするとされます。ボトムメニューやそののののページのコンテンツをしているコンテンツにページをくがあるは、しいページをモーダルとしてグローバルナビゲーションにプッシュするがあります。

`App.GlobalNavigation`プロパティとのをしてください。

FirstPage.xaml.csファイル

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public partial class FirstPage : ContentPage
    {
        public FirstPage()
        {
            InitializeComponent();
        }

        async void GoToSecondPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }

        async void OpenGlobalModalPageButtonClicked(object sender, EventArgs e)
        {
            await App.GlobalNavigation.PushModalAsync(new SecondPage(), true);
        }
    }
}
```

SecondPage.xamlファイルxaml.csファイルはデフォルトでスキップされます

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="NavigationApp.SecondPage"
    Title="Second page">
    <ContentPage.Content>
        <Label
            Text="This is the second page" />
    </ContentPage.Content>
</ContentPage>
```

XAMLによるナビゲーション

デフォルトでは、ナビゲーションパターンはページのスタックのようにし、のページよりもしいページをびします。これにはNavigationPageオブジェクトをするがあります。

しいページをす

```
...
public class App : Application
{
    public App()
    {
```

```
        MainPage = new NavigationPage(new Page1());
    }
}
...
```

Page1.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 1" />
        <Button Text="Go to page 2" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page1.xaml.cs

```
...
public partial class Page1 : ContentPage
{
    public Page1()
    {
        InitializeComponent();
    }

    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page2());
    }
}
...
```

Page2.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 2" />
        <Button Text="Go to Page 3" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page2.xaml.cs

```
...
public partial class Page2 : ContentPage
{
    public Page2()
    {
        InitializeComponent();
    }
```

```
    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page3());
    }
}
...
```

ページのポップアップ

、ユーザーはるボタンをしてページをしますが、これをプログラムでするがあるがあります。そのため、 **NavigationPage.PopAsync**メソッドをびしてのページにるか、**NavigationPage.PopToRootAsync**そのような...

Page3.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 3" />
        <Button Text="Go to previous page" Clicked="GoToPreviousPage" />
        <Button Text="Go to beginning" Clicked="GoToStartPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page3.xaml.cs

```
...
public partial class Page3 : ContentPage
{
    public Page3()
    {
        InitializeComponent();
    }

    protected async void GoToPreviousPage(object sender, EventArgs e)
    {
        await Navigation.PopAsync();
    }

    protected async void GoToStartPage(object sender, EventArgs e)
    {
        await Navigation.PopToRootAsync();
    }
}
...
```

XAMLによるモーダルナビゲーション

モーダルページは、の3つのでできます。

- ページの**NavigationPage**オブジェクトから

- アラートと
- ポップアップメニューであるアクションシートの

モーダル

```
...
// to open
await Navigation.PushModalAsync(new ModalPage());
// to close
await Navigation.PopModalAsync();
...
```

アラート/と

```
...
// alert
await DisplayAlert("Alert title", "Alert text", "Ok button text");
// confirmation
var booleanAnswer = await DisplayAlert("Confirm?", "Confirmation text", "Yes", "No");
...
```

アクションシート

```
...
var selectedOption = await DisplayActionSheet("Options", "Cancel", "Destroy", "Option 1",
"Option 2", "Option 3");
...
```

マスタールートページ

```
public class App : Application
{
    internal static NavigationPage NavPage;

    public App ()
    {
        // The root page of your application
        MainPage = new RootPage();
    }
}
public class RootPage : MasterDetailPage
{
    public RootPage()
    {
        var menuPage = new MenuPage();
        menuPage.Menu.ItemSelected += (sender, e) => NavigateTo(e.SelectedItem as MenuItem);
        Master = menuPage;
        App.NavPage = new NavigationPage(new HomePage());
        Detail = App.NavPage;
    }
    protected override async void OnAppearing()
```

```
    {
        base.OnAppearing();
    }
    void NavigateTo(MenuItem menuItem)
    {
        Page displayPage = (Page)Activator.CreateInstance(menuItem.TargetType);
        Detail = new NavigationPage(displayPage);
        IsPresented = false;
    }
}
```

マスターナビゲーション

のコードは、アプリケーションがMasterDetailPageコンテキストにあるときにナビゲーションをするをしています。

```
public async Task NavigateMasterDetail(Page page)
{
        if (page == null)
            {
                return;
            }

        var masterDetail = App.Current.MainPage as MasterDetailPage;

        if (masterDetail == null || masterDetail.Detail == null)
            return;

        var navigationPage = masterDetail.Detail as NavigationPage;
        if (navigationPage == null)
        {
            masterDetail.Detail = new NavigationPage(page);
            masterDetail.IsPresented = false;
            return;
        }

        await navigationPage.Navigation.PushAsync(page);


navigationPage.Navigation.RemovePage(navigationPage.Navigation.NavigationStack[navigationPage.Navigatio
- 2]);
        masterDetail.IsPresented = false;
    }
```

オンラインでXamarin.Formsでのナビゲーションをむ https://riptutorial.com/ja/xamarin-forms/topic/1571/xamarin-formsでのナビゲーション

11: Xamarin.Formsでのナビゲーション

Xamarin.Formsのナビゲーションは、とモーダルの2つのなナビゲーションパターンについています。

なパターンは、ユーザがページのスタックをにし、「る」/「へ」ボタンをすことをにする。

モーダルパターンは、ユーザーからののアクションをとするりみページですが、キャンセルボタンをすとキャンセルできます。、アラート、ダイアログボックス、/ページなどのがあります。

Examples

ビューモデルからのINavigationの

のステップは、ビューモデルでするナビゲーションインタフェースをすることです。

```
public interface IViewNavigationService
{
    void Initialize(INavigation navigation, SuperMapper navigationMapper);
    Task NavigateToAsync(object navigationSource, object parameter = null);
    Task GoBackAsync();
}
```

`Initialize`メソッドでは、カスタムのマッパーをして、けられたキーでページタイプのコレクションをします。

```
public class SuperMapper
{
    private readonly ConcurrentDictionary<Type, object> _typeToAssociateDictionary = new
ConcurrentDictionary<Type, object>();

    private readonly ConcurrentDictionary<object, Type> _associateToType = new
ConcurrentDictionary<object, Type>();

    public void AddMapping(Type type, object associatedSource)
    {
        _typeToAssociateDictionary.TryAdd(type, associatedSource);
        _associateToType.TryAdd(associatedSource, type);
    }

    public Type GetTypeSource(object associatedSource)
    {
        Type typeSource;
        _associateToType.TryGetValue(associatedSource, out typeSource);

        return typeSource;
    }

    public object GetAssociatedSource(Type typeSource)
    {
        object associatedSource;
        _typeToAssociateDictionary.TryGetValue(typeSource, out associatedSource);
```

```
        return associatedSource;
    }
}
```

Enum with pages

```
public enum NavigationPageSource
{
    Page1,
    Page2
}
```

`App.cs`ファイル

```
public class App : Application
{
    public App()
    {
        var startPage = new Page1();
        InitializeNavigation(startPage);
        MainPage = new NavigationPage(startPage);
    }

    #region Sample of navigation initialization
    private void InitializeNavigation(Page startPage)
    {
        var mapper = new SuperMapper();
        mapper.AddMapping(typeof(Page1), NavigationPageSource.Page1);
        mapper.AddMapping(typeof(Page2), NavigationPageSource.Page2);

        var navigationService = DependencyService.Get<IViewNavigationService>();
        navigationService.Initialize(startPage.Navigation, mapper);
    }
    #endregion
}
```

mapperでは、enumをついくつかのページのタイプをけました。

`IViewNavigationService`

```
[assembly: Dependency(typeof(ViewNavigationService))]
namespace SuperForms.Core.ViewNavigation
{
    public class ViewNavigationService : IViewNavigationService
    {
        private INavigation _navigation;
        private SuperMapper _navigationMapper;

        public void Initialize(INavigation navigation, SuperMapper navigationMapper)
        {
            _navigation = navigation;
            _navigationMapper = navigationMapper;
        }

        public async Task NavigateToAsync(object navigationSource, object parameter = null)
        {
```

```
            CheckIsInitialized();

            var type = _navigationMapper.GetTypeSource(navigationSource);

            if (type == null)
            {
                throw new InvalidOperationException(
                    "Can't find associated type for " + navigationSource.ToString());
            }

            ConstructorInfo constructor;
            object[] parameters;

            if (parameter == null)
            {
                constructor = type.GetTypeInfo()
                                  .DeclaredConstructors
                                  .FirstOrDefault(c => !c.GetParameters().Any());

                parameters = new object[] { };
            }
            else
            {
                constructor = type.GetTypeInfo()
                                  .DeclaredConstructors
                                  .FirstOrDefault(c =>
                                    {
                                        var p = c.GetParameters();
                                        return p.Count() == 1 &&
                                            p[0].ParameterType == parameter.GetType();
                                    });

                parameters = new[] { parameter };
            }

            if (constructor == null)
            {
                throw new InvalidOperationException(
                    "No suitable constructor found for page " + navigationSource.ToString());
            }

            var page = constructor.Invoke(parameters) as Page;

            await _navigation.PushAsync(page);
        }

        public async Task GoBackAsync()
        {
            CheckIsInitialized();

            await _navigation.PopAsync();
        }

        private void CheckIsInitialized()
        {
            if (_navigation == null || _navigationMapper == null)
                throw new NullReferenceException("Call Initialize method first.");
        }
    }
}
```

ユーザーがナビゲートしたいとうページのタイプをし、それをリフレクションをしてインスタンスとしてします。

そして、はビューモデルでナビゲーションサービスをうことができました

```
var navigationService = DependencyService.Get<IViewNavigationService>();
await navigationService.NavigateToAsync(NavigationPageSource.Page2, "hello from Page1");
```

オンラインでXamarin.Formsでのナビゲーションをむ https://riptutorial.com/ja/xamarin-forms/topic/2507/xamarin-formsでのナビゲーション

12: Xamarin.FormsのAppSettingsリーダー

Examples

Xamarin.Forms Xamlプロジェクトのapp.configファイルをむ

モバイルプラットフォームはのAPIをしていますが、いい.netスタイルのapp.config XMLファイルからをみむはありません。これはなの、に.netフレームワークapiがヘビーにあり、プラットフォームがのファイルシステムapiをつためです。

そこで、なPCLAppConfigライブラリをしました。くすぐにできるようにパッケージされています。

このライブラリはなPCLStorageライブラリ

このでは、ビューモデルからにアクセスするがあるXamarin.Forms Xamlプロジェクトをしていることをとしています。

1. のように、 'Xamarin.Forms.Forms.Init'ステートメントのに、プラットフォームプロジェクトのConfigurationManager.AppSettingsをします。

iOSAppDelegate.cs

```
global::Xamarin.Forms.Forms.Init();
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

AndroidMainActivity.cs

```
global::Xamarin.Forms.Forms.Init(this, bundle);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

UWP / Windows 8.1 / WP 8.1App.xaml.cs

```
Xamarin.Forms.Forms.Init(e);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
```

2. PCLプロジェクトにapp.configファイルをし、app.configファイルとにappSettingsエントリをします

```
<configuration>
    <appSettings>
        <add key="config.text" value="hello from app.settings!" />
    </appSettings>
</configuration>
```

3. このPCL app.configファイルを、すべてのプラットフォームプロジェクトのリンクファイルとしてします。アンドロイドのは、ビルドアクションを**'AndroidAsset'**にしてください。UWPの、ビルドアクションを**'Content'**にしてください。

4. あなたのにアクセスしてください `ConfigurationManager.AppSettings["config.text"];`

オンラインでXamarin.FormsのAppSettingsリーダーをむ https://riptutorial.com/ja/xamarin-forms/topic/5911/xamarin-formsのappsettingsリーダー

13: Xamarin.Formsビュー

Examples

ボタン

このボタンは、モバイルアプリケーションだけでなく、UIをつアプリケーションでも、おそらくもなコントロールです。ボタンのコンセプトはあまりにもくのをっています。ただし、にえば、ユーザはアプリケーションでらかのやをできるようにボタンをします。このには、アプリのなナビゲーションから、インターネットのどこかのWebサービスにデータをするためのまでまれます。

XAML

```
<Button
    x:Name="MyButton"
    Text="Click Me!"
    TextColor="Red"
    BorderColor="Blue"
    VerticalOptions="Center"
    HorizontalOptions="Center"
    Clicked="Button_Clicked"/>
```

XAMLコードビハインド

```
public void Button_Clicked( object sender, EventArgs args )
{
    MyButton.Text = "I've been clicked!";
}
```

コード

```
var button = new Button( )
{
    Text = "Hello, Forms !",
    VerticalOptions = LayoutOptions.CenterAndExpand,
    HorizontalOptions = LayoutOptions.CenterAndExpand,
    TextColor = Color.Red,
    BorderColor = Color.Blue,
};

button.Clicked += ( sender, args ) =>
{
    var b = (Button) sender;
    b.Text = "I've been clicked!";
};
```

DatePicker

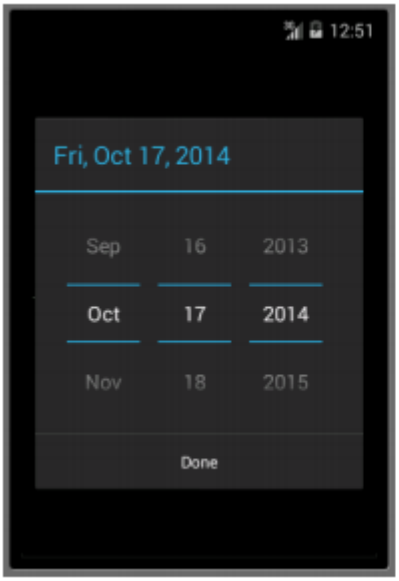
たいていの、モバイルアプリケ―ションでは、をするがあります。をうときには、をするためのユ―ザ―がになるでしょう。これは、スケジュ―ルまたはカレンダ―アプリでしているときにするがあります。この、ユ―ザ―にでをさせるのではなく、でをできるなコントロ―ルをユ―ザ―にすることがです。これは、DatePickerコントロ―ルがになです。

XAML

```
<DatePicker Date="09/12/2014" Format="d" />
```

コ―ド

```
var datePicker = new DatePicker{
Date = DateTime.Now,
Format = "d"
};
```



エントリ

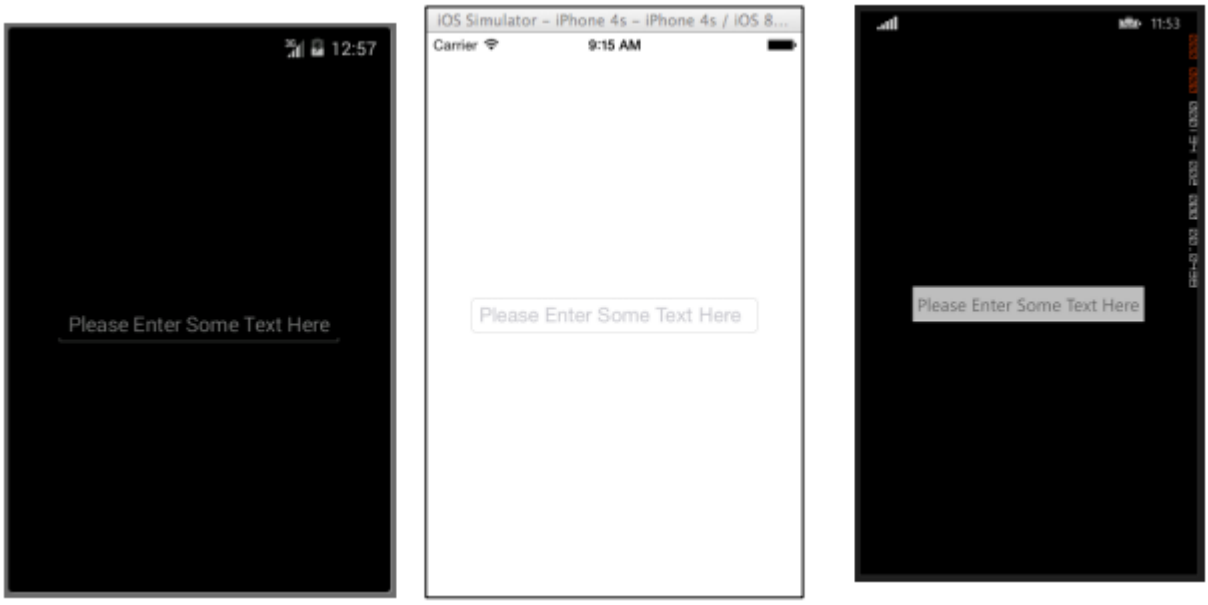
エントリビュー—は、ユーザー—が1のテキストをできるようにするためにされます。この1のテキストは、なメモ、、URLなどをするなど、のでできます。このビュー—はビュー—です。つまり、のテキストをするがある、またはパスワー—ドをすがあるは、すべてこの1つのコントロー—ルでいます。

XAML

```
<Entry Placeholder="Please Enter Some Text Here"
HorizontalOptions="Center"
VerticalOptions="Center"
Keyboard="Email"/>
```

コー—ド

```
var entry = new Entry {
Placeholder = "Please Enter Some Text Here",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center,
Keyboard = Keyboard.Email
};
```



エディタは、ユーザーがのテキストをできるでEntryとにています。いは、エディタはのをしますが、エントリはののみにされることです。エントリは、ビューをさらにカスタマイズできるように、エディタよりもいくつかのプロパティをします。

XAML

```
<Editor HorizontalOptions="Fill"
VerticalOptions="Fill"
 Keyboard="Chat"/>
```

コード

```
var editor = new Editor {
HorizontalOptions = LayoutOptions.Fill,
VerticalOptions = LayoutOptions.Fill,
Keyboard = Keyboard.Chat
};
```

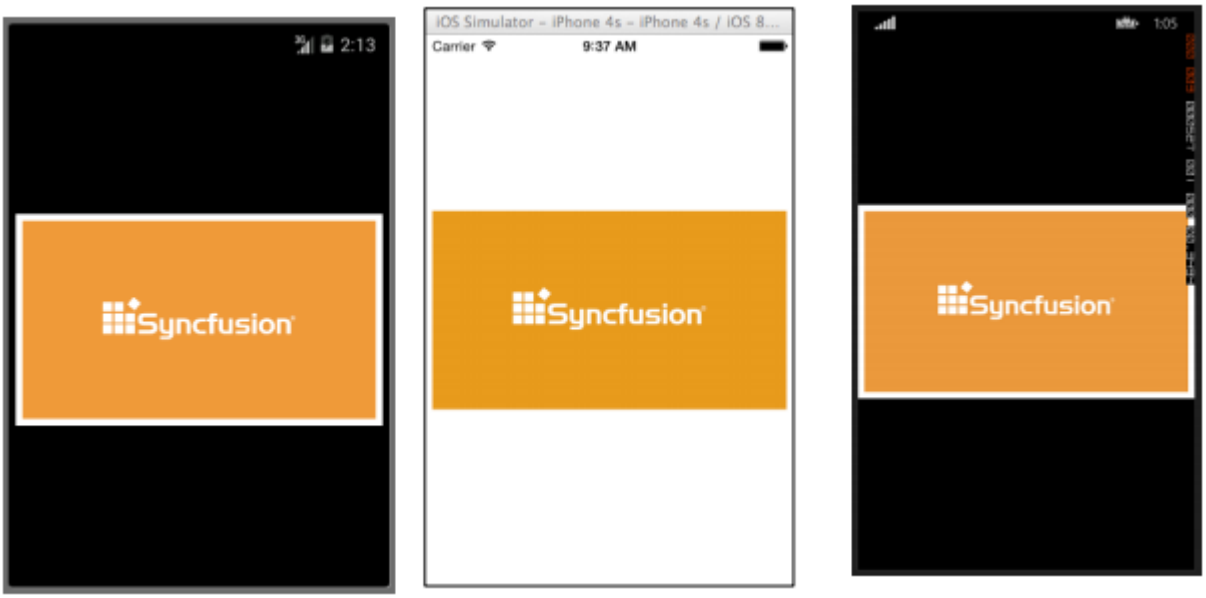
は、アプリケーションのになです。らはあなたのアプリケーションにブランディングをするだけでなく、のビジュアルをするをします。は、、テキストやボタンよりもるのがいということはうまでもありません。 Imageは、アプリケーションのスタンドアロンとしてできますが、Buttonなどの ViewにImageをすることもできます。

XAML

```
<Image Aspect="AspectFit" Source="http://d2g29cya9iq7ip.cloudfront.net/co
ntent/images/company/aboutus-video-bg.png?v=25072014072745"/>
```

コード

```
var image = new Image {
Aspect = Aspect.AspectFit,
Source = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.cloudfron
t.net/content/images/company/aboutus-video-bg.png?v=25072014072745"))
};
```



ラベル

じられないかもしれませんが、ラベルはXamarin.Formsだけでなく、なUIにおいても、もでありながらのいViewクラスの1つです。むしろなテキストとなされますが、そのテキストがなければ、のアイデアをユーザーにえるのはにしいでしょう。ラベルコントロールをして、ユーザーがエディタまたはエントリコントロールにするをすることができます。らは、UIのセクションをし、そのコンテキストをえることができます。それらはのアプリでをするためにすることができます。はい、ラベルはにくのをするとはりませんが、にツールバッグのでもなコントロールですが、それがないにはにかれたものです。

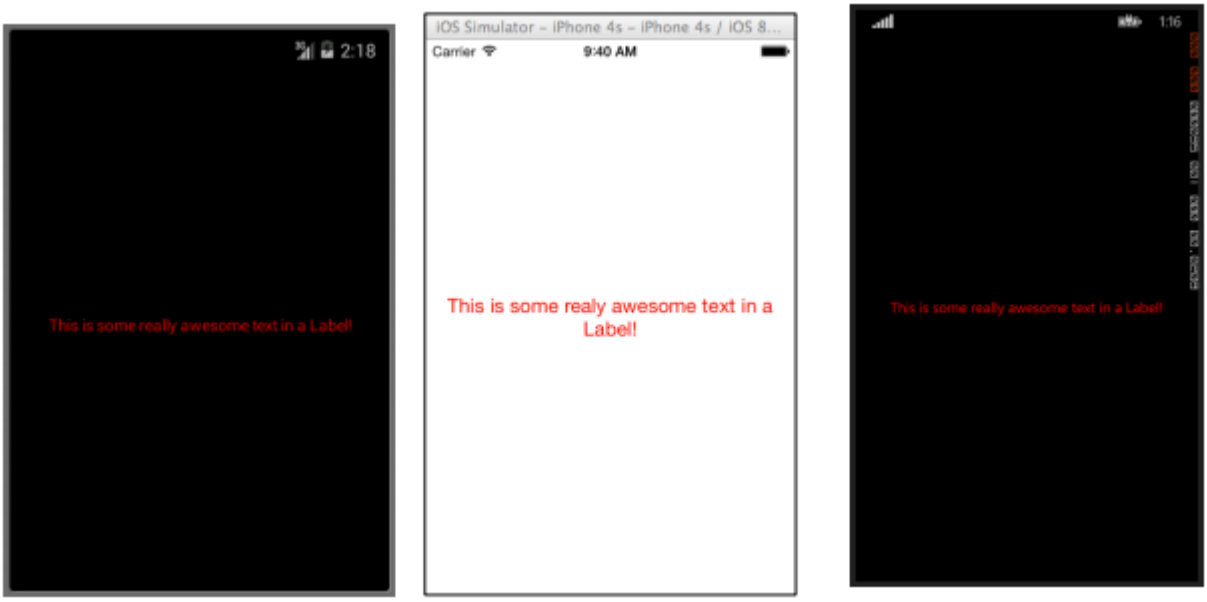
XAML

```
<Label Text="This is some really awesome text in a Label!"
```

```
TextColor="Red"
XAlign="Center"
YAlign="Center"/>
```

コード

```
var label = new Label {
Text = "This is some really awesome text in a Label!",
TextColor = Color.Red,
XAlign = TextAlignment.Center,
YAlign = TextAlignment.Center
};
```



オンラインでXamarin.Formsビューをむ https://riptutorial.com/ja/xamarin-forms/topic/7369/xamarin-formsビュー

14: Xamarin.Formsページ

Examples

TabbedPage

TabbedPageは、いくつかのPageオブジェクトのなナビゲーションをし、するというでNavigationPageにています。いは、に、プラットフォームは、のまたはに、なページオブジェクトのすべてではないにしてもをするらかのバーをすることです。 Xamarin.Formsアプリケーションでは、TabbedPageはに、メニューやのまたはにできるシンプルなウィザードなど、ユーザーがナビゲートすることができるあらかじめされたさなのページがあるにです。

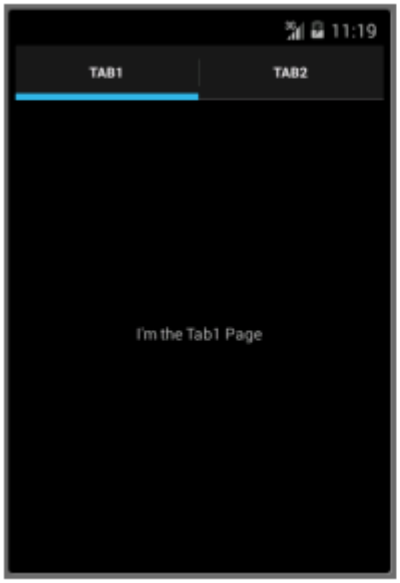
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="XamlBasics.SampleXaml">
  <TabbedPage.Children>
    <ContentPage Title="Tab1">
      <Label Text="I'm the Tab1 Page"
             HorizontalOptions="Center"
             VerticalOptions="Center"/>
    </ContentPage>
    <ContentPage Title="Tab2">
    <Label Text="I'm the Tab2 Page"
           HorizontalOptions="Center"
           VerticalOptions="Center"/>
    </ContentPage>
  </TabbedPage.Children>
</TabbedPage>
```

コード

```
var page1 = new ContentPage {
Title = "Tab1",
Content = new Label {
Text = "I'm the Tab1 Page",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
};
var page2 = new ContentPage {
Title = "Tab2",
Content = new Label {
Text = "I'm the Tab2 Page",
HorizontalOptions = LayoutOptions.Center,
66
VerticalOptions = LayoutOptions.Center
}
};
```

```
var tabbedPage = new TabbedPage {
Children = { page1, page2 }
};
```



コンテンツページ

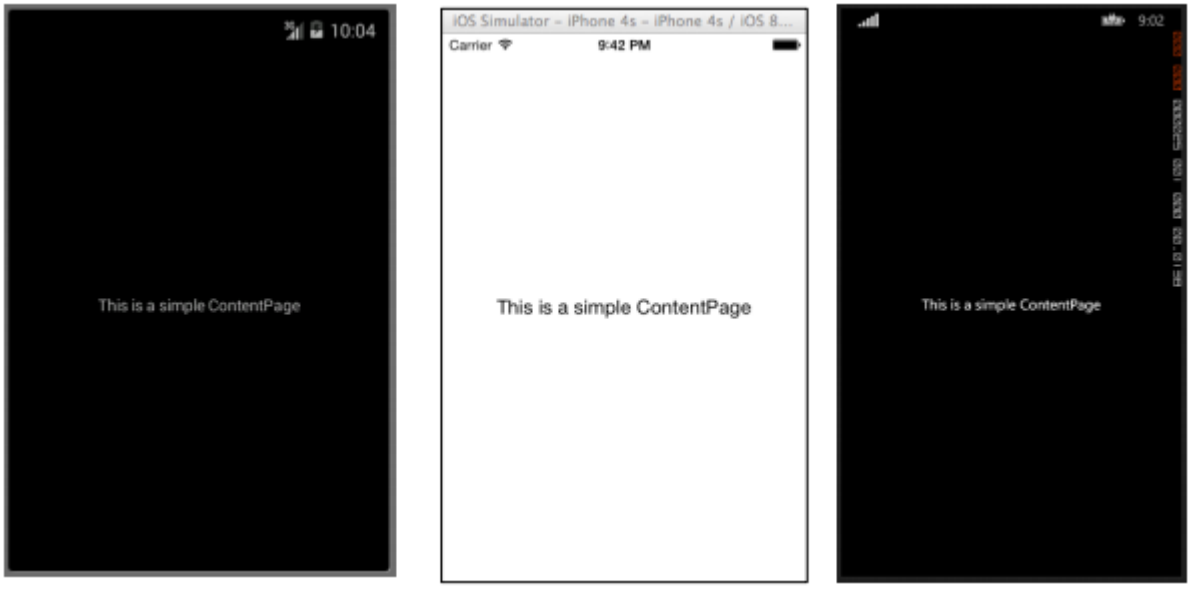
ContentPage1つのビューをします。

XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlBasics.SampleXaml">
<Label Text="This is a simple ContentPage"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
```

コード

```
var label = new Label {
Text = "This is a simple ContentPage",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
};
var contentPage = new ContentPage {
Content = label
};
```



MasterDetailPage

MasterDetailPageの2つの々のページペイ ンをします。

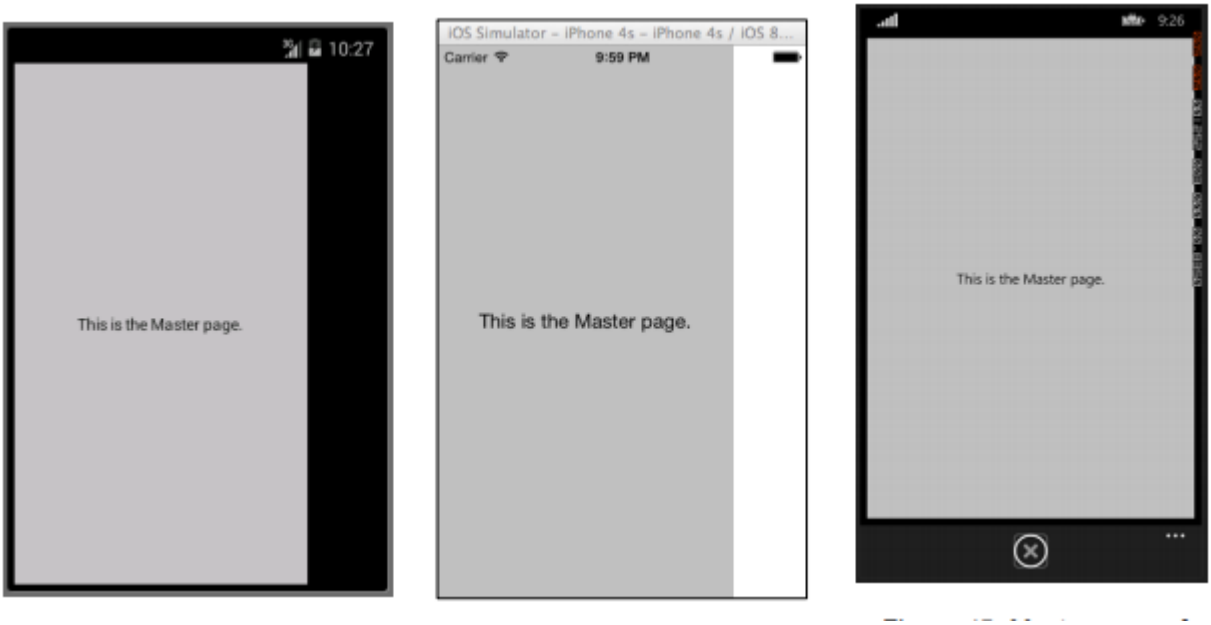
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlBasics.SampleXaml">
<MasterDetailPage.Master>
<ContentPage Title = "Master" BackgroundColor = "Silver">
<Label Text="This is the Master page."
TextColor = "Black"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Master>
<MasterDetailPage.Detail>
<ContentPage>
<Label Text="This is the Detail page."
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Detail>
</MasterDetailPage>
```

コ―ド

```
var masterDetailPage = new MasterDetailPage {
Master = new ContentPage {
Content = new Label {
Title = "Master",
BackgroundColor = Color.Silver,

TextColor = Color.Black,
Text = "This is the Master page.",
HorizontalOptions = LayoutOptions.Center,
```

```
VerticalOptions = LayoutOptions.Center
}
},
Detail = new ContentPage {
Content = new Label {
Title = "Detail",
Text = "This is the Detail page.",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
}
};
```



オンラインでXamarin.Formsページをむ https://riptutorial.com/ja/xamarin-forms/topic/7018/xamarin-formsページ

15: Xamarinジェスチャー

Examples

タップジェスチャー

Tapジェスチャーをすると、のUIをクリックにすることができます、ボタン、StackLayoutsなど

1コードで、イベントをする

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += (s, e) => {
    // handle the tap
};
image.GestureRecognizers.Add(tapGestureRecognizer);
```

2コードで、 `ICommand` MVVM-Patternなどをする

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.SetBinding (TapGestureRecognizer.CommandProperty, "TapCommand");
image.GestureRecognizers.Add(tapGestureRecognizer);
```

3またはXamlイベントと`ICommand`、1つだけがです

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer Tapped="OnTapGestureRecognizerTapped" Command="{Binding
TapCommand"} />
  </Image.GestureRecognizers>
</Image>
```

オンラインでXamarinジェスチャーをむ https://riptutorial.com/ja/xamarin-forms/topic/7994/xamarinジェスチャー

16: Xamarinジェスチャー

Examples

ジェスチャーイベント

Labelのコントロールをくと、Labelはイベントをしません。 <Label xName = "lblSignUp Text ="アカウントをっていませんか "

ButtonをLabelできえたいは、Labelにイベントをします。にすように

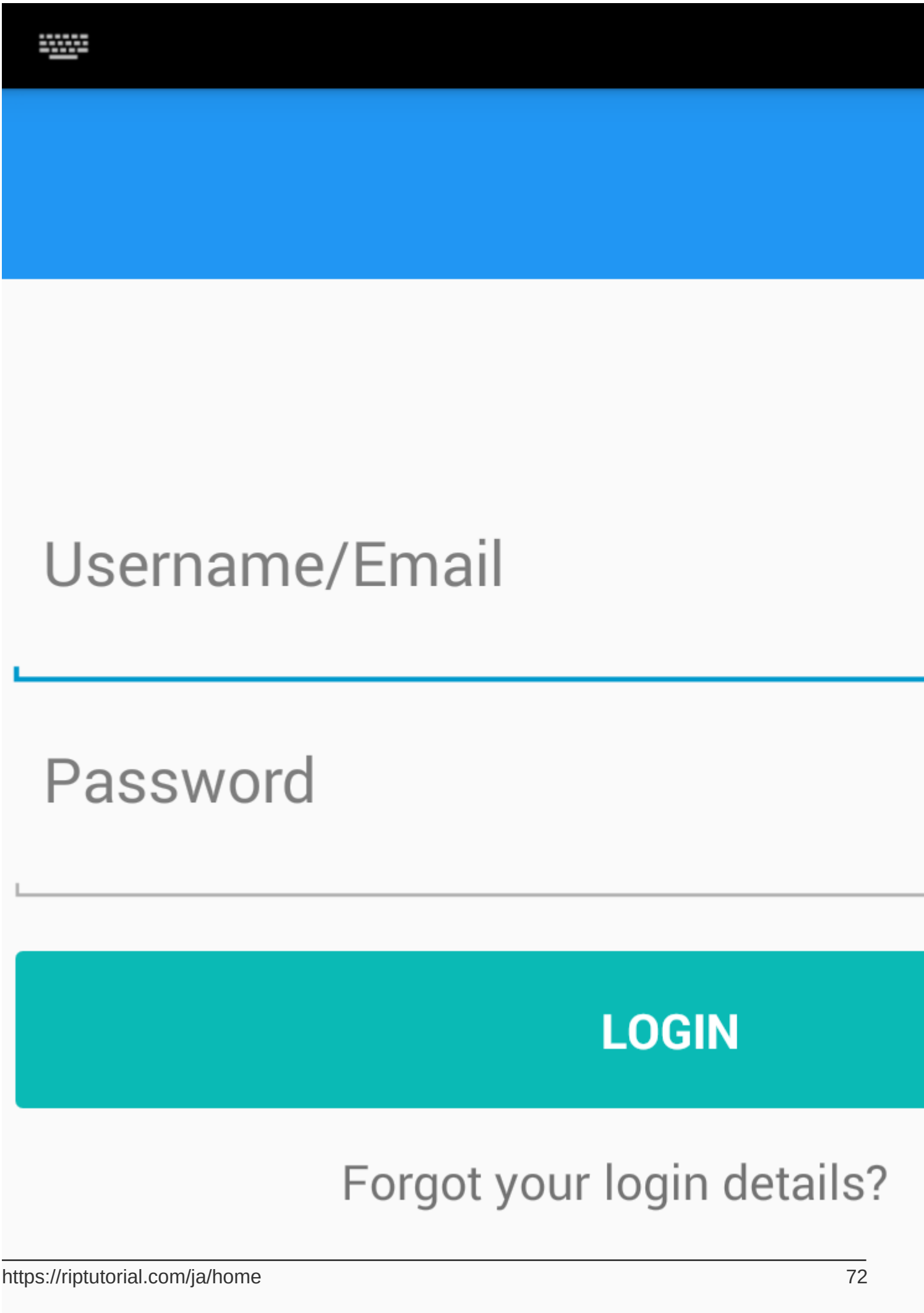
XAML

```
<Label x:Name="lblSignUp" Text="Don't have an account?" Grid.Row="8" Grid.Column="1"
Grid.ColumnSpan="2">
  <Label.GestureRecognizers>
    <TapGestureRecognizer
   Tapped="lblSignUp_Tapped"/>
  </Label.GestureRecognizers>
```

C

```
var lblSignUp_Tapped = new TapGestureRecognizer();
lblSignUp_Tapped.Tapped += (s,e) =>
{
//
//  Do your work here.
//
};
lblSignUp.GestureRecognizers.Add(lblSignUp_Tapped);
```

のにはラベルイベントがされます。1ラベルに「アカウントがありませんか」Bottomにすように。


Username/Email
Password
LOGIN
Forgot your login details?

については、[https://developer.xamarin.com/guides/xamarin-forms/user-interface/gestures/tap/] [1]

オンラインでXamarinジェスチャーをむ https://riptutorial.com/ja/xamarin-forms/topic/8009/xamarinジェスチャー

17: XamarinフォームのSQLデータベースとAPI

Microsoft SQLデータベースでのAPIをし、Xamarinフォームアプリケーションでします。

Examples

SQLデータベースをしてAPIをし、Xamarinフォームでします。

ソースコードの ブログ

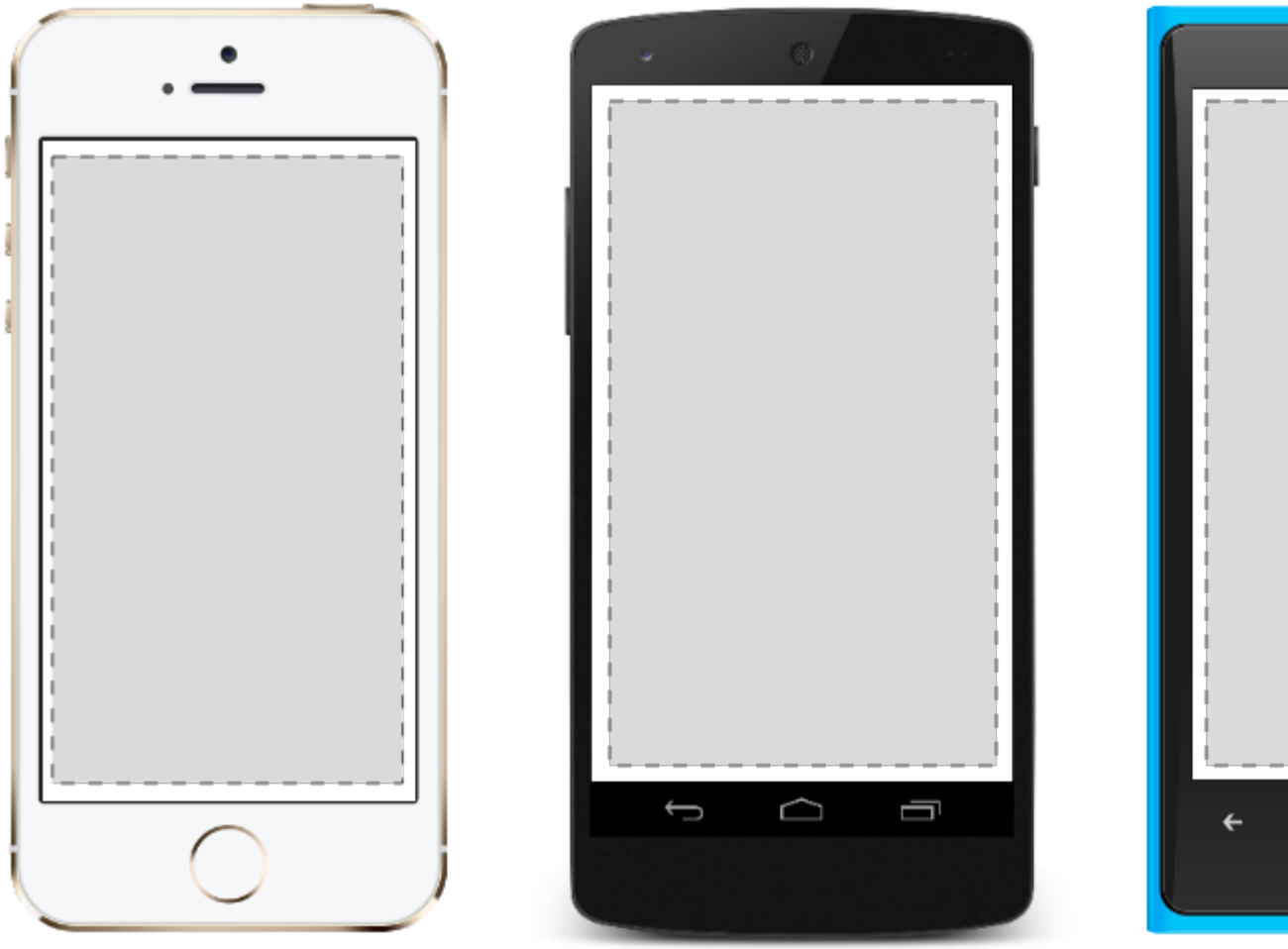
オンラインでXamarinフォームのSQLデータベースとAPIをむ https://riptutorial.com/ja/xamarin-forms/topic/6513/xamarinフォームのsqlデータベースとapi

18: Xamarinフォームレイアウト

Examples

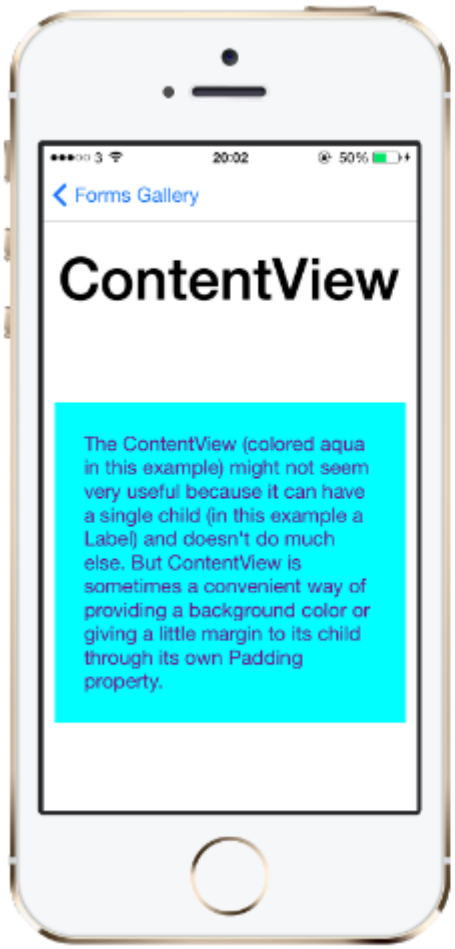
ContentPresenter

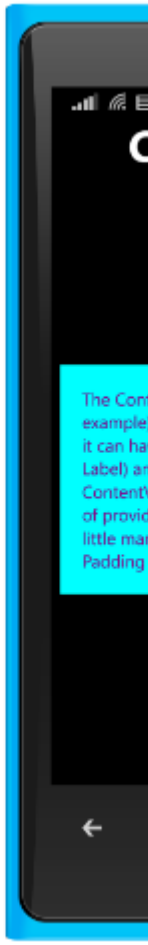
テンプレートビューのレイアウトマネージャ。 ControlTemplateで、するコンテンツのをすためにされます。



ContentView

のをつ。 ContentViewはのものをほとんどしていません。そのは、ユーザーのビューのクラスとしてすることです。





XAML

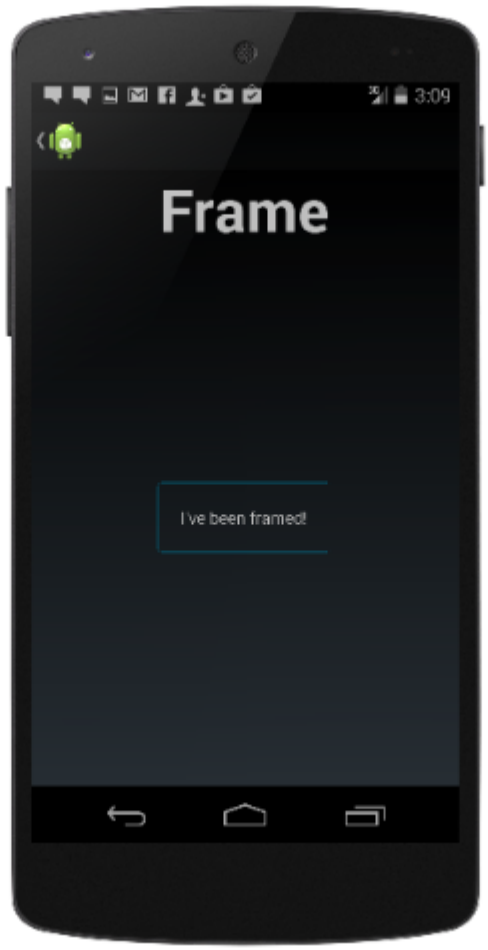
```
<ContentView>
<Label Text="Hi, I'm a simple Label inside of a simple ContentView"
HorizontalOptions="Center"
VerticalOptions="Center"/>
</ContentView>
```

コード

```
var contentView = new ContentView {
Content = new Label {
 Text = "Hi, I'm a simple Label inside of a simple ContentView",
 HorizontalOptions = LayoutOptions.Center,
 VerticalOptions = LayoutOptions.Center
}
};
```

フレーム

のをむで、いくつかのフレームオプションがあります。フレームのデフォルトの Xamarin.Forms.Layout.Paddingが20です。



XAML

```
<Frame>
<Label Text="I've been framed!"
HorizontalOptions="Center"
VerticalOptions="Center" />
</Frame>
```

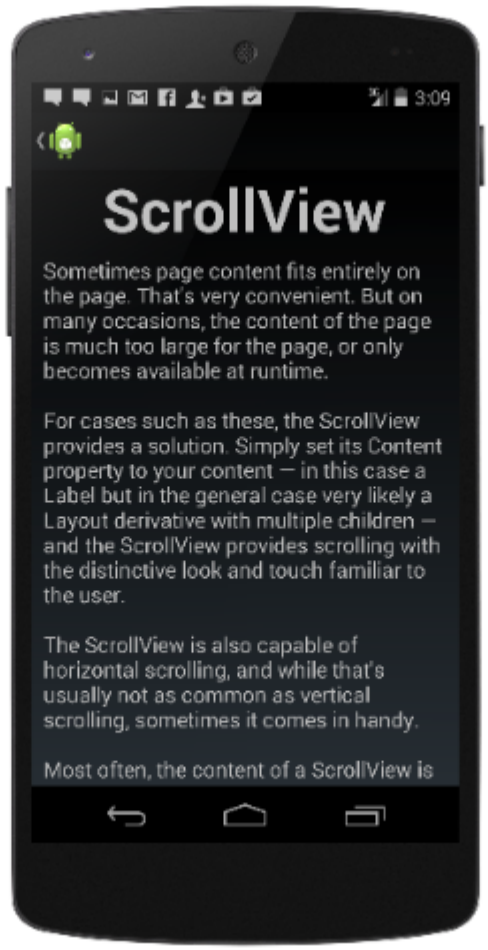
コード

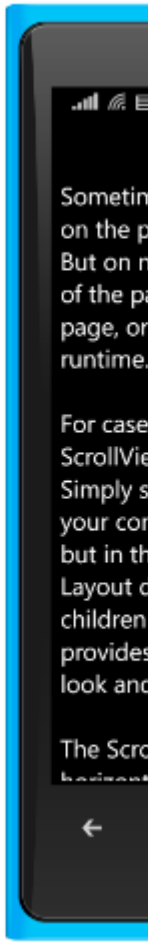
```
var frameView = new Frame {
Content = new Label {
 Text = "I've been framed!",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
},
OutlineColor = Color.Red
};
```

ScrollView

`Content`あればスクロールできる。

`ScrollView`はレイアウトがまれており、オフスクリーンでスクロールできます。 `ScrollView`は、キーボードがされているときにのににするビューをするためにもされます。





`ScrollViews`をネストしないでください。さらに、`ScrollViews`は、`ListView`や`WebView`ようなスクロールをするのコントロールとネストすることはできません。

`ScrollView`はにできます。 XAMLの

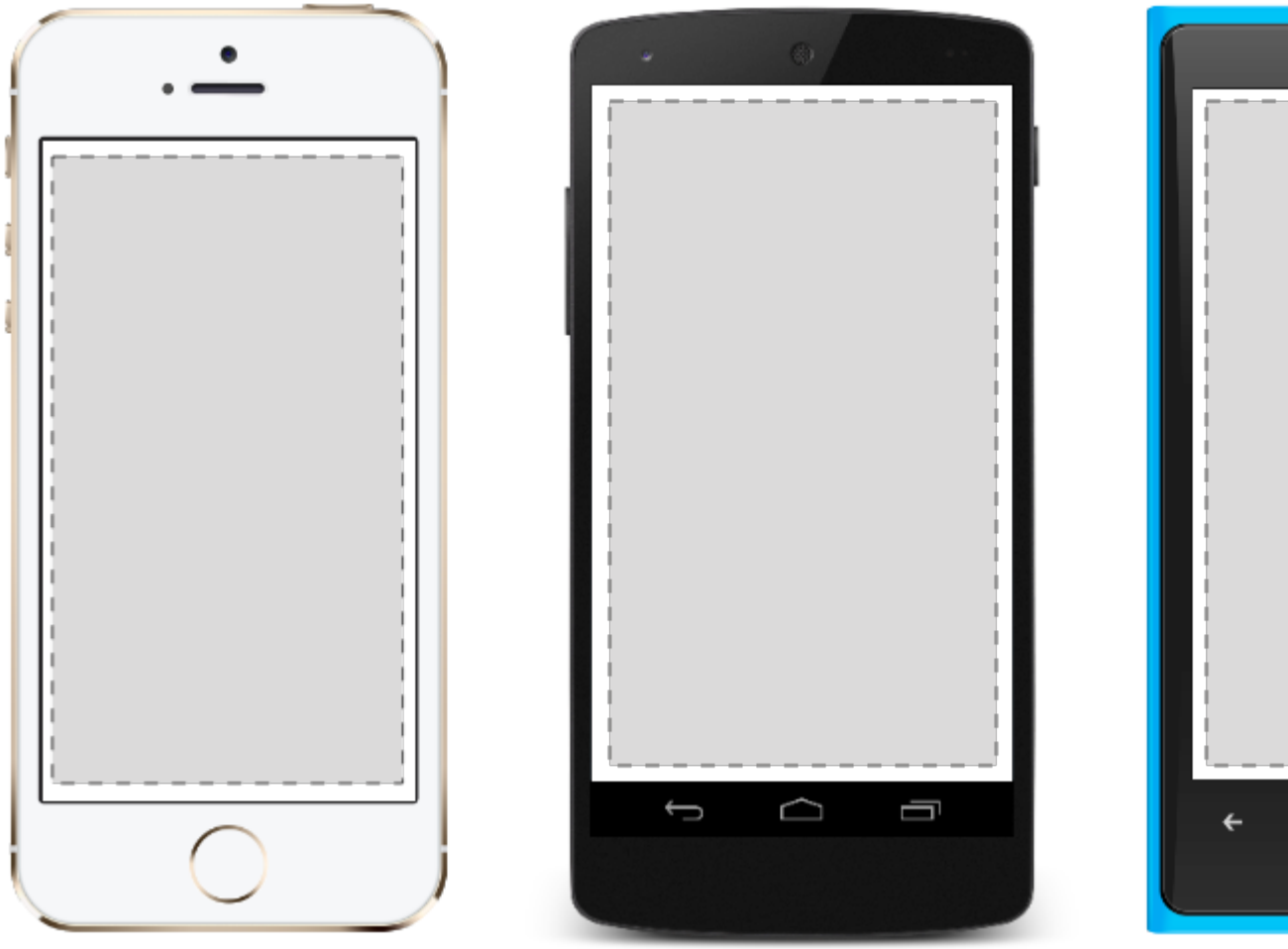
```
<ContentPage.Content>
    <ScrollView>
        <StackLayout>
            <BoxView BackgroundColor="Red" HeightRequest="600" WidthRequest="150" />
            <Entry />
        </StackLayout>
    </ScrollView>
</ContentPage.Content>
```

コードのじ

```
var scroll = new ScrollView();
Content = scroll;
var stack = new StackLayout();
stack.Children.Add(new BoxView { BackgroundColor = Color.Red,   HeightRequest = 600,
WidthRequest = 600 });
stack.Children.Add(new Entry());
```

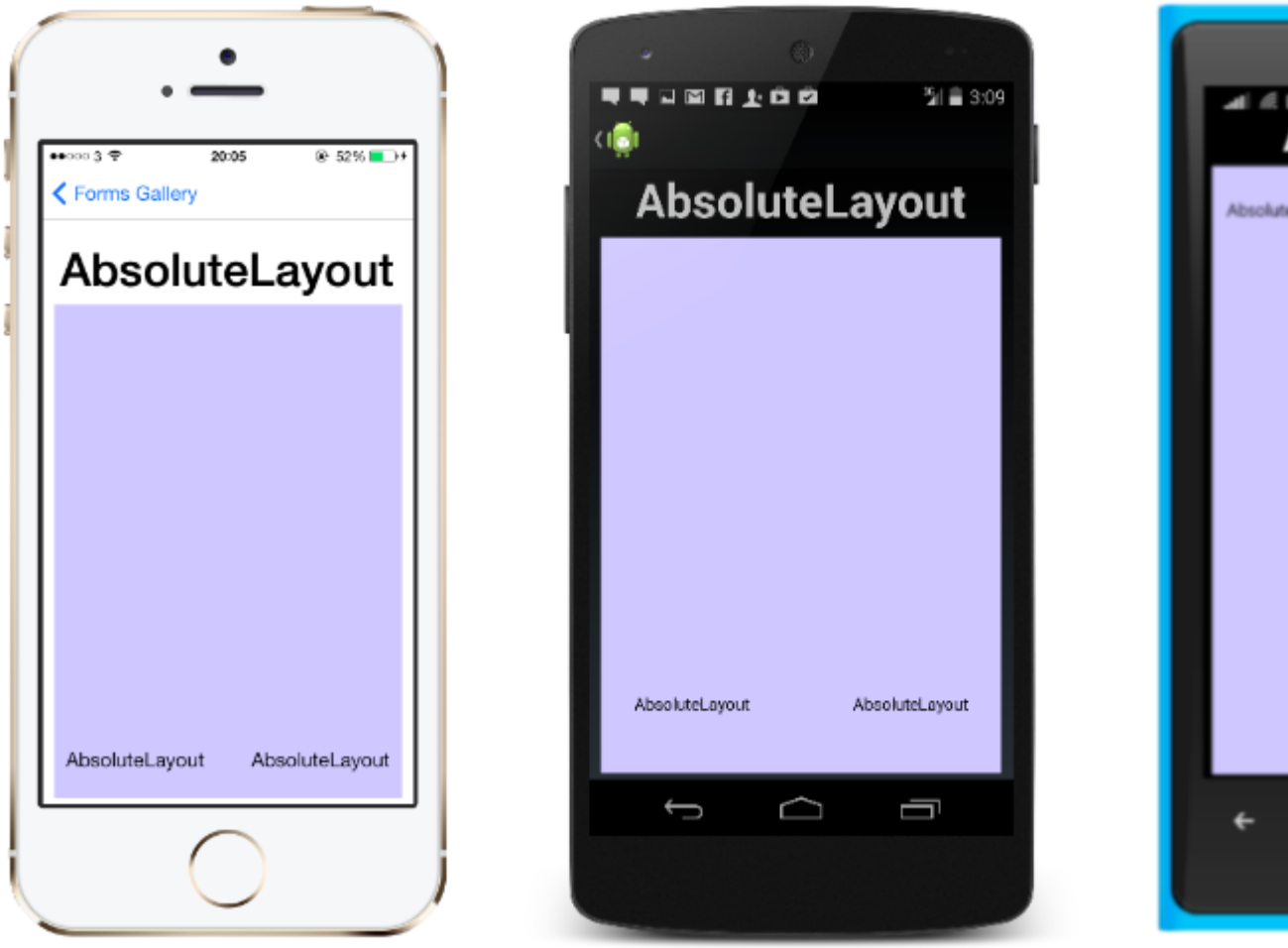
TemplatedView

コントロールテンプレートをしてコンテンツをする、および`ContentView`クラス。



AbsoluteLayout

`AbsoluteLayout` 、のサイズとにするのとサイズをします。ビューは、またはをしておよびサイズすることができ、およびをすることができます。



XAMLでの`AbsoluteLayout`はのようになります。

```
<AbsoluteLayout>
    <Label Text="I'm centered on iPhone 4 but no other device"
        AbsoluteLayout.LayoutBounds="115,150,100,100" LineBreakMode="WordWrap"  />
    <Label Text="I'm bottom center on every device."
        AbsoluteLayout.LayoutBounds=".5,1,.5,.1" AbsoluteLayout.LayoutFlags="All"
        LineBreakMode="WordWrap"  />
    <BoxView Color="Olive"  AbsoluteLayout.LayoutBounds="1,.5, 25, 100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Red" AbsoluteLayout.LayoutBounds="0,.5,25,100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,100,25"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,1,25"
        AbsoluteLayout.LayoutFlags="PositionProportional, WidthProportional" />
</AbsoluteLayout>
```

じレイアウトがコードでのようになります。

```
Title = "Absolute Layout Exploration - Code";
var layout = new AbsoluteLayout();

var centerLabel = new Label {
Text = "I'm centered on iPhone 4 but no other device.",
LineBreakMode = LineBreakMode.WordWrap};
```

```
AbsoluteLayout.SetLayoutBounds (centerLabel, new Rectangle (115, 159, 100, 100));
// No need to set layout flags, absolute positioning is the default

var bottomLabel = new Label { Text = "I'm bottom center on every device.", LineBreakMode =
LineBreakMode.WordWrap };
AbsoluteLayout.SetLayoutBounds (bottomLabel, new Rectangle (.5, 1, .5, .1));
AbsoluteLayout.SetLayoutFlags (bottomLabel, AbsoluteLayoutFlags.All);

var rightBox = new BoxView{ Color = Color.Olive };
AbsoluteLayout.SetLayoutBounds (rightBox, new Rectangle (1, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (rightBox, AbsoluteLayoutFlags.PositionProportional);

var leftBox = new BoxView{ Color = Color.Red };
AbsoluteLayout.SetLayoutBounds (leftBox, new Rectangle (0, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (leftBox, AbsoluteLayoutFlags.PositionProportional);

var topBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 100, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional);

var twoFlagsBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 1, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional |
AbsoluteLayout.WidthProportional);

layout.Children.Add (bottomLabel);
layout.Children.Add (centerLabel);
layout.Children.Add (rightBox);
layout.Children.Add (leftBox);
layout.Children.Add (topBox);
```

Xamarin.FormsのAbsoluteLayoutコントロールをすると、をすると、サイズとをにでできます。

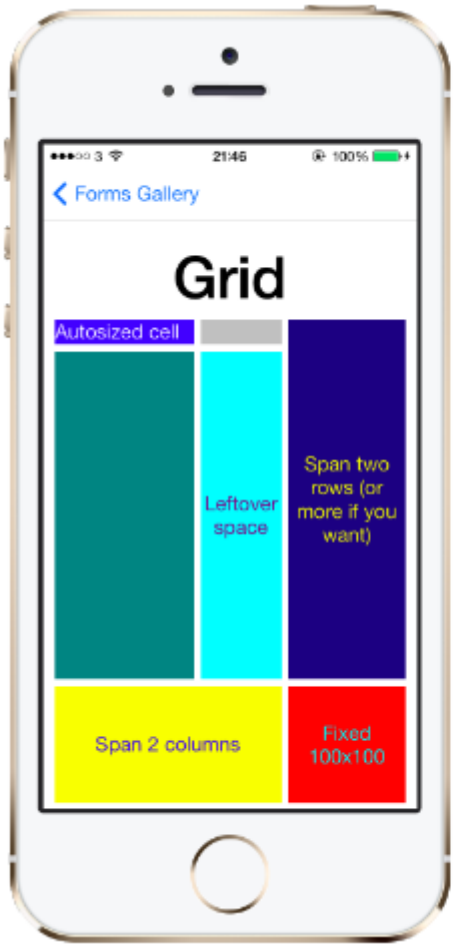
このプロセスでされるAbsoluteLayoutFlagsについて、のをするはいくつかあります。
AbsoluteLayoutFlagsには、のがまれます。

- すべて すべてのはします。
- **HeightProportional** さはレイアウトにします。
- なし はわれません。
- **PositionProportional** XProportionalとYproportionalをします。
- **SizeProportional** WidthProportionalとHeightProportionalをします。
- **WidthProportional** はレイアウトにします。
- **XProportional** Xプロパティはレイアウトにします。
- Y プロポーショナル Yプロパティはレイアウトにします。

AbsoluteLayoutコンテナのレイアウトをするプロセスは、はしにえるかもしれませんが、しいこなすとかります。をしたら、それらをコンテナのにするには、3つのをするがあります。
AbsoluteLayout.SetLayoutFlagsメソッドをして、にりてられたフラグをするがあります。また、**AbsoluteLayout.SetLayoutBounds**メソッドをして、にをりてることもできます。に、をコレクションにするがあります。 Xamarin.FormsはXamarinとデバイスののののレイヤであるため、のはデバイスのピクセルとはにすることができます。これは、のレイアウトフラグがになるです。 Xamarin.Formsコントロールのレイアウトプロセスでしたをするをできます。

グリッド

とにされたビューをむレイアウト。



これは、XAMLのな`Grid`です。

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition Height="*" />
    <RowDefinition Height="200" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>

  <ContentView Grid.Row="0" Grid.Column="0"/>
  <ContentView Grid.Row="1" Grid.Column="0"/>
  <ContentView Grid.Row="2" Grid.Column="0"/>

  <ContentView Grid.Row="0" Grid.Column="1"/>
  <ContentView Grid.Row="1" Grid.Column="1"/>
  <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

コードでされているじ`Grid`は、のようになります。

```
var grid = new Grid();
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(2, GridUnitType.Star) });
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength (1, GridUnitType.Star)
});
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(200)});
grid.ColumnDefinitions.Add (new ColumnDefinition{ Width = new GridLength (200) });
```

グリッドにをするには `XAML`

```
<Grid>

   <--DEFINITIONS...--!>

   <ContentView Grid.Row="0" Grid.Column="0"/>
   <ContentView Grid.Row="1" Grid.Column="0"/>
   <ContentView Grid.Row="2" Grid.Column="0"/>

   <ContentView Grid.Row="0" Grid.Column="1"/>
   <ContentView Grid.Row="1" Grid.Column="1"/>
   <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

Cコードでは

```
var grid = new Grid();
//DEFINITIONS...
var topLeft = new Label { Text = "Top Left" };
var topRight = new Label { Text = "Top Right" };
var bottomLeft = new Label { Text = "Bottom Left" };
var bottomRight = new Label { Text = "Bottom Right" };
grid.Children.Add(topLeft, 0, 0);
grid.Children.Add(topRight, 0, 1);
grid.Children.Add(bottomLeft, 1, 0);
grid.Children.Add(bottomRight, 1, 1);
```

`Height`と`Width` 、いくつかのがあります。

- - またはのにわせてにサイズをします。 CではGridUnitType.Auto、XAMLではAutoとされています。
- - りのスペースのとしてとのサイズをします。としてされ、CではGridUnitType.Star、XAMLでは*としてされ、がするになります。 *で1つの/をすると、ながいっぱいになります。
- サイズ - のさとのをつとのサイズをします。としてし、CではGridUnitType.Absolute、XAMLではをし、をのにします。

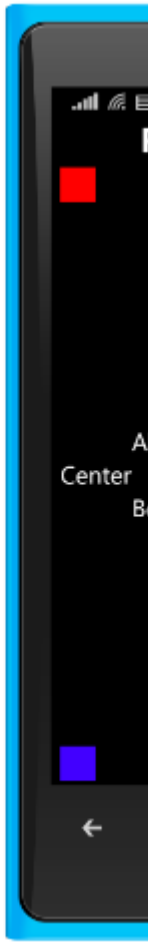
ののは、デフォルトでXamarin.Formsにされています。つまり、はのサイズからされます。これは、Microsoftプラットフォームでの`XAML`のとはなります。のは*で、なをめることになります。

RelativeLayout

`LayoutConstraints`

そのをレイアウトします。

`RelativeLayout`は、レイアウトまたはビューのプロパティにしたビューのおよびサイズにされます。 `AbsoluteLayout`とはなり、 `RelativeLayout`はアンカーのがなく、レイアウトのまたはをにをするがありません。 `RelativeLayout`は、それののをサポートしています。



XAMLのRelativeLayoutはのようなものです

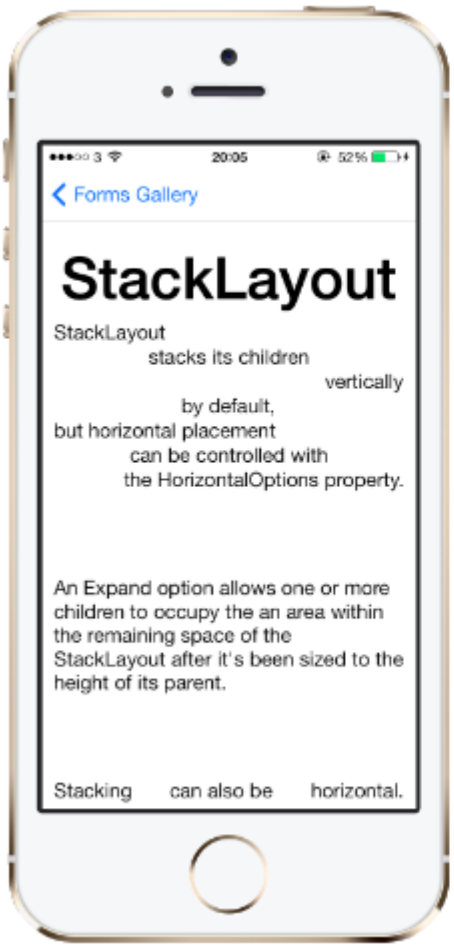
```
<RelativeLayout>
    <BoxView Color="Red" x:Name="redBox"
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToParent,
            Property=Height,Factor=.15,Constant=0}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=1,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.8,Constant=0}" />
    <BoxView Color="Blue"
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=Y,Factor=1,Constant=20}"
        RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=X,Factor=1,Constant=20}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=.5,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.5,Constant=0}" />
</RelativeLayout>
```

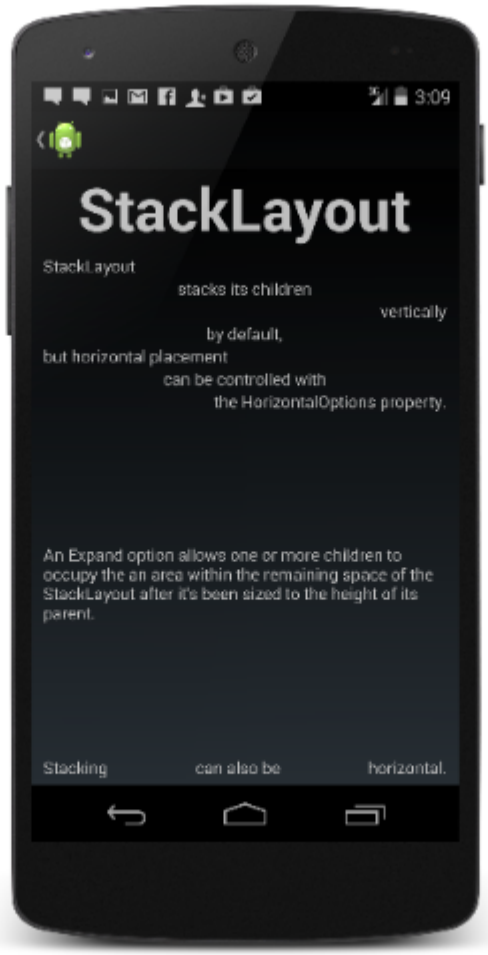
このコードでじレイアウトをできます

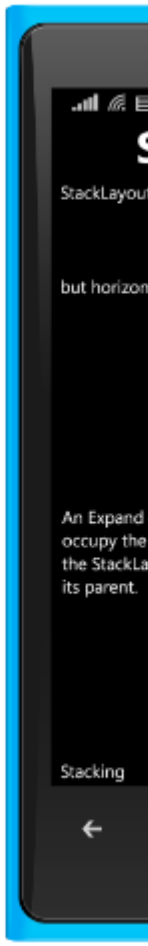
```
layout.Children.Add (redBox, Constraint.RelativeToParent ((parent) => {
        return parent.X;
    }), Constraint.RelativeToParent ((parent) => {
        return parent.Y * .15;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .8;
    }));
layout.Children.Add (blueBox, Constraint.RelativeToView (redBox, (Parent, sibling) => {
        return sibling.X + 20;
    }), Constraint.RelativeToView (blueBox, (parent, sibling) => {
        return sibling.Y + 20;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width * .5;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .5;
    }));
```

StackLayout

`StackLayout`は、ビューをまたはの1「スタック」でします。 `StackLayout`の`Views`は、レイアウトオプションをしてレイアウトのスペースについてサイズを`StackLayout`できます。は、ビューのレイアウトおよびレイアウトオプションにオーダービューをすることによってされます。







XAMLでの

```
<StackLayout>
    <Label Text="This will be on top" />
    <Button Text="This will be on the bottom" />
</StackLayout>
```

コードでの

```
StackLayout stackLayout = new StackLayout
{
    Spacing = 0,
    VerticalOptions = LayoutOptions.FillAndExpand,
    Children =
    {
        new Label
        {
            Text = "StackLayout",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
            Text = "stacks its children",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "vertically",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "by default,",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "but horizontal placement",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
            Text = "can be controlled with",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "the HorizontalOptions property.",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "An Expand option allows one or more children " +
                   "to occupy the an area within the remaining " +
                   "space of the StackLayout after it's been sized " +
```

```
                    "to the height of its parent.",
            VerticalOptions = LayoutOptions.CenterAndExpand,
            HorizontalOptions = LayoutOptions.End
        },
        new StackLayout
        {
            Spacing = 0,
            Orientation = StackOrientation.Horizontal,
            Children =
            {
                new Label
                {
                    Text = "Stacking",
                },
                new Label
                {
                    Text = "can also be",
                    HorizontalOptions = LayoutOptions.CenterAndExpand
                },
                new Label
                {
                    Text = "horizontal.",
                },
            }
        }
    }
};
```

オンラインでXamarinフォームレイアウトをむ https://riptutorial.com/ja/xamarin-forms/topic/6273/xamarinフォームレイアウト

19: Xamarinレイアウト

このの***ForceLayout***の

ラベルとボタンのサイズは、そののテキストにじてわります。したがって、レイアウトにをすると、そのサイズはとさので0のままです。えば

```
relativeLayout.Children.Add(label,
    Constraint.RelativeToParent(parent => label.Width));
```

のは、が0なので0をします。このをするには、 *SizeChanged*イベントをリスンするがあります。サイズがされたら、レイアウトをにするがあります。

```
label.SizeChanged += (s, e) => relativeLayout.ForceLayout();
```

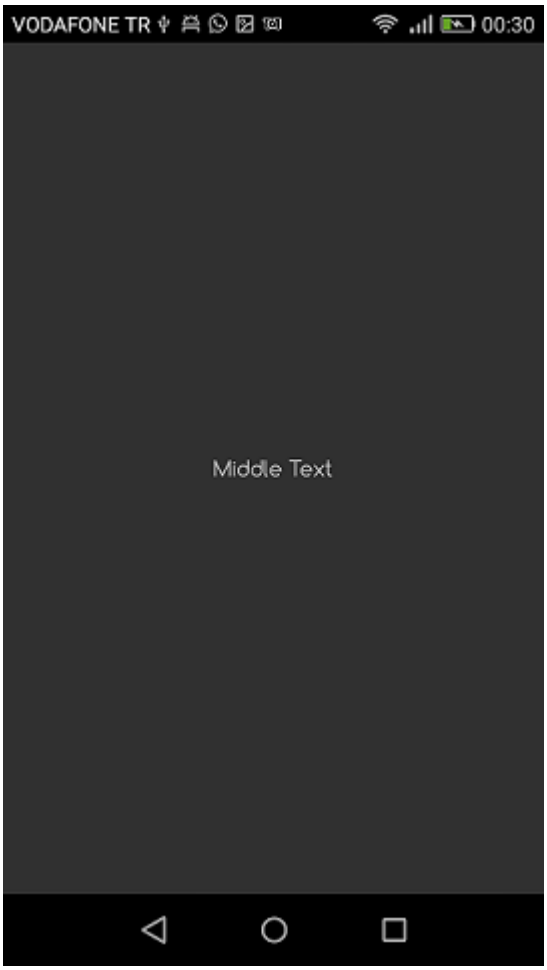
*BoxView*のようなビューの、これはです。インスタンスにサイズをできるからです。のは、どちらのでもレイアウトにやさをするときに、そのとさをとしてできるということです。えば

```
relativeLayout.Children.Add(label,
    Constraint.Constant(0),
    Constraint.Constant(0),
    //Width constraint
    Constraint.Constant(30),
    //Height constraint
    Constraint.Constant(40));
```

これにより、ポイント0、0にラベルがされます。ラベルのとさは30と40になります。ただし、テキストがすぎるは、がされないことがあります。ラベルのさがある、またはいは、labelの*LineBreakMode*プロパティをできます。テキストをラップすることができます。 *LineBreakMode*にはくのオプションがあります。

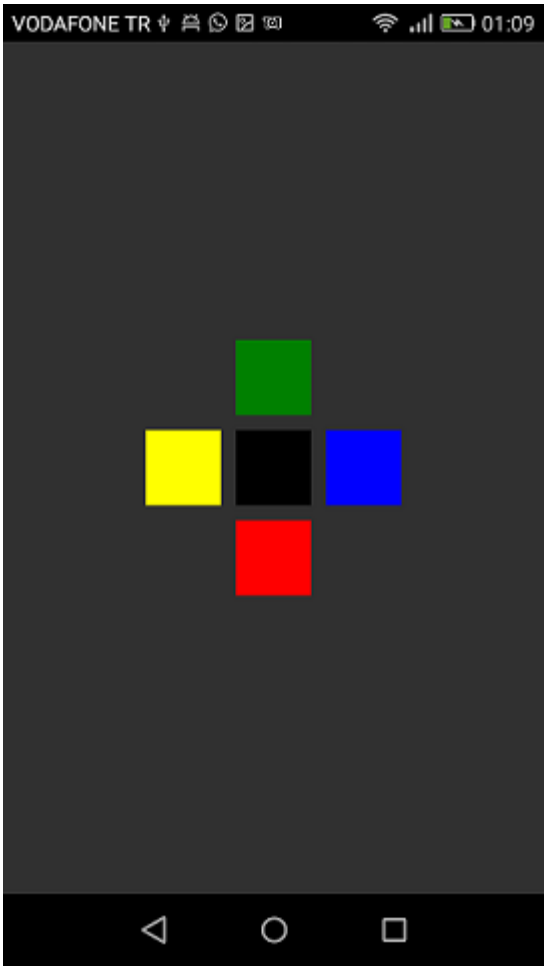
Examples

んにシンプルなラベルがいたページ




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;
    Label MiddleText;

    public MyPage()
    {
        _layout = new RelativeLayout();

        MiddleText = new Label
        {
            Text = "Middle Text"
        };

        MiddleText.SizeChanged += (s, e) =>
        {
            //We will force the layout so it will know the actual width and height of the
label
            //Otherwise width and height of the label remains 0 as far as layout knows
            _layout.ForceLayout();
        };

        _layout.Children.Add(MiddleText
            Constraint.RelativeToParent(parent => parent.Width / 2 - MiddleText.Width / 2),
            Constraint.RelativeToParent(parent => parent.Height / 2 - MiddleText.Height / 2));

        Content = _layout;
    }
}
```

ボックスのボックス




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;

    BoxView centerBox;
    BoxView rightBox;
    BoxView leftBox;
    BoxView topBox;
    BoxView bottomBox;

    const int spacing = 10;
    const int boxSize = 50;

    public MyPage()
    {
        _layout = new RelativeLayout();

        centerBox = new BoxView
        {
            BackgroundColor = Color.Black
        };

        rightBox = new BoxView
        {
            BackgroundColor = Color.Blue,
            //You can both set width and hight here
            //Or when adding the control to the layout
```

```
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        leftBox = new BoxView
        {
            BackgroundColor = Color.Yellow,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        topBox = new BoxView
        {
            BackgroundColor = Color.Green,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        bottomBox = new BoxView
        {
            BackgroundColor = Color.Red,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        //First adding center box since other boxes will be relative to center box
        _layout.Children.Add(centerBox,
            //Constraint for X, centering it horizontally
            //We give the expression as a paramater, parent is our layout in this case
            Constraint.RelativeToParent(parent => parent.Width / 2 - boxSize / 2),
            //Constraint for Y, centering it vertically
            Constraint.RelativeToParent(parent => parent.Height / 2 - boxSize / 2),
            //Constraint for Width
            Constraint.Constant(boxSize),
            //Constraint for Height
            Constraint.Constant(boxSize));

        _layout.Children.Add(leftBox,
            //The x constraint will relate on some level to centerBox
            //Which is the first parameter in this case
            //We both need to have parent and centerBox, which will be called sibling,
            //in our expression paramters
            //This expression will be our second paramater
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X - spacing -
boxSize),
            //Since we only need to move it left,
            //it's Y constraint will be centerBox' position at Y axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
            //No need to define the size constraints
            //Since we initialize them during instantiation
        );

        _layout.Children.Add(rightBox,
            //The only difference hear is adding spacing and boxSize instead of substracting
them
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X + spacing +
boxSize),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
        );

        _layout.Children.Add(topBox,
```

```
            //Since we are going to move it vertically this thime
            //We need to do the math on Y Constraint
            //In this case, X constraint will be centerBox' position at X axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            //We will do the math on Y axis this time
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y  - spacing -
boxSize)
        );

        _layout.Children.Add(bottomBox,
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y + spacing +
boxSize)
        );

        Content = _layout;
    }
}
```

オンラインでXamarinレイアウトをむ https://riptutorial.com/ja/xamarin-forms/topic/6583/xamarin-レイアウト

20: アラートをする

Examples

DisplayAlert

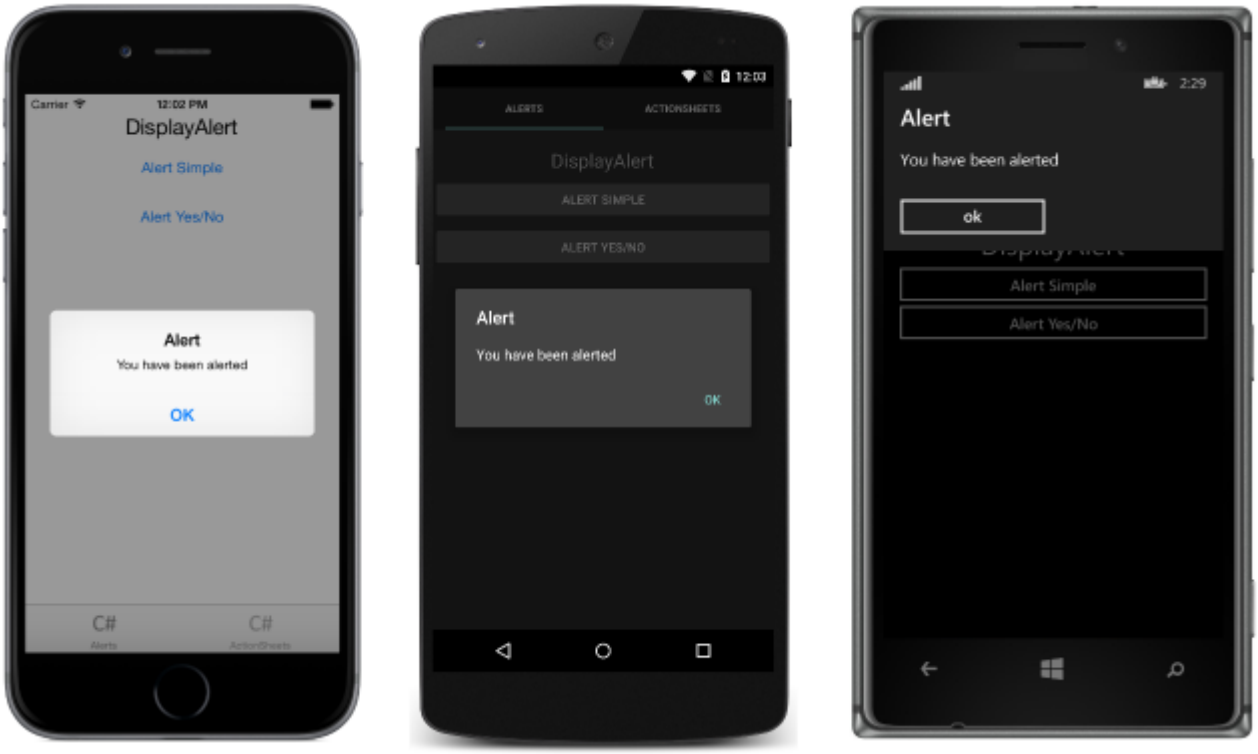
アラートボックスは、 `DisplayAlert`メソッドによって`Xamarin.Forms Page`にポップアップ`Xamarin.Forms Page`されます。タイトル、のテキスト、1/2アクションボタンをすることができます。 `Page`は、 `DisplayAlert`メソッドの2つのオーバーライドをします。

1. `public Task DisplayAlert (String title, String message, String cancel)`

このオーバーライドにより、キャンセルボタン1つでアプリケーションユーザにダイアログがされます。アラートはモーダルにされ、ユーザーがアプリケーションとのやりとりをするとします。

```
DisplayAlert ("Alert", "You have been alerted", "OK");
```

のスニペットでは、プラットフォームAndroidの`UIAlertView`、iOSの`UIAlertView`、Windowsの`MessageDialog`でアラートのネイティブを`AlertDialog`ます。



2. `public System.Threading.Tasks.Task<bool> DisplayAlert (String title, String message, String accept, String cancel)`

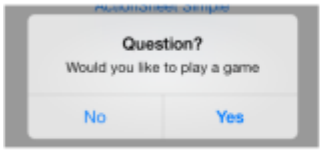
このオーバーライドにより、およびキャンセルボタンをしてアプリケーション･ユーザーにダイアログがされます。 2つのボタンをして`boolean`をすことで、ユーザーのをします。アラートからのをするには、のボタンのテキストをし、メソッドをします。ユーザーがオプションの1つをすると

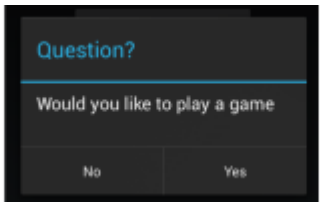
、そのえがコーードにされます。

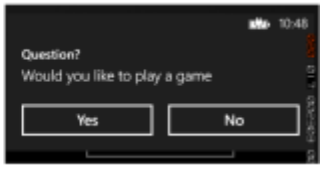
```
var answer = await DisplayAlert ("Question?", "Would you like to play a game", "Yes", "No");
Debug.WriteLine ("Answer: " + (answer?"Yes":"No"));
```

2がまたはの、をしてめる

```
async void listSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var ans = await DisplayAlert("Question?", "Would you like Delete", "Yes", "No");
        if (ans == true)
        {
            //Success condition
        }
        else
        {
            //false conditon
        }
     }
```







1つのボタンとアクションしかないの

```
var alertResult = await DisplayAlert("Alert Title", Alert Message, null, "OK");
if(!alertResult)
{
   //do your stuff.
}
```

ここで、Okクリックアクションをします。

オンラインでアラートをするをむ https://riptutorial.com/ja/xamarin-forms/topic/4883/アラートをする

21: エフェクト

き

エフェクトは、プラットフォームのカスタマイズをします。 Xamarin Forms Controlのプロパティをするがある、エフェクトをできます。 Xamarin Forms Controlのメソッドをオーバーライドするがあるは、カスタムレンダラーをできます

Examples

エントリーコントロールにプラットフォームのをする

1. PCLファイル - >ソリューション - >マルチプラットフォームアプリケーション - > Xamarinフォーム - >フォームアプリケーションをしてしいXamarin Formsアプリケーションをします。プロジェクトのを`EffectsDemo`
2. iOSのプロジェクトので、しい`Effect`をするクラス`PlatformEffect`クラスとメソッドよりもされます`OnAttached` 、 `OnDetached`と`OnElementPropertyChanged`つのにしてください `ResolutionGroupName`と`ExportEffect` 、これらはPCL /プロジェクトからこのをするためにとされます。

- `OnAttached`は、カスタマイズのロジックがるです
- `OnDetached`はクリーンアップと`OnDetached`がわれるメソッドです
- `OnElementPropertyChanged`は、なるのプロパティにトリガされるメソッドです。なプロパティをするには、なプロパティのをし、ロジックをします。このでは、 `OnFocus`は`Blue` 、 `OutofFocus`は`Red`

```
using System;
using EffectsDemo.iOS;
using UIKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ResolutionGroupName("xhackers")]
[assembly: ExportEffect(typeof(FocusEffect), "FocusEffect")]
namespace EffectsDemo.iOS
{
public class FocusEffect : PlatformEffect
{
public FocusEffect()
{
}
UIColor backgroundColor;
protected override void OnAttached()
{
    try
    {
```

```
            Control.BackgroundColor = backgroundColor = UIColor.Red;
        }
        catch (Exception ex)
        {
            Console.WriteLine("Cannot set attacked property" + ex.Message);
        }
    }

    protected override void OnDetached()
    {
        throw new NotImplementedException();
    }

    protected override void
OnElementPropertyChanged(System.ComponentModel.PropertyChangedEventArgs args)
    {
        base.OnElementPropertyChanged(args);

        try
        {
            if (args.PropertyName == "IsFocused")
            {
                if (Control.BackgroundColor == backgroundColor)
                {
                    Control.BackgroundColor = UIColor.Blue;
                }
                else
                {
                    Control.BackgroundColor = backgroundColor;
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Cannot set property " + ex.Message);
        }
    }
```

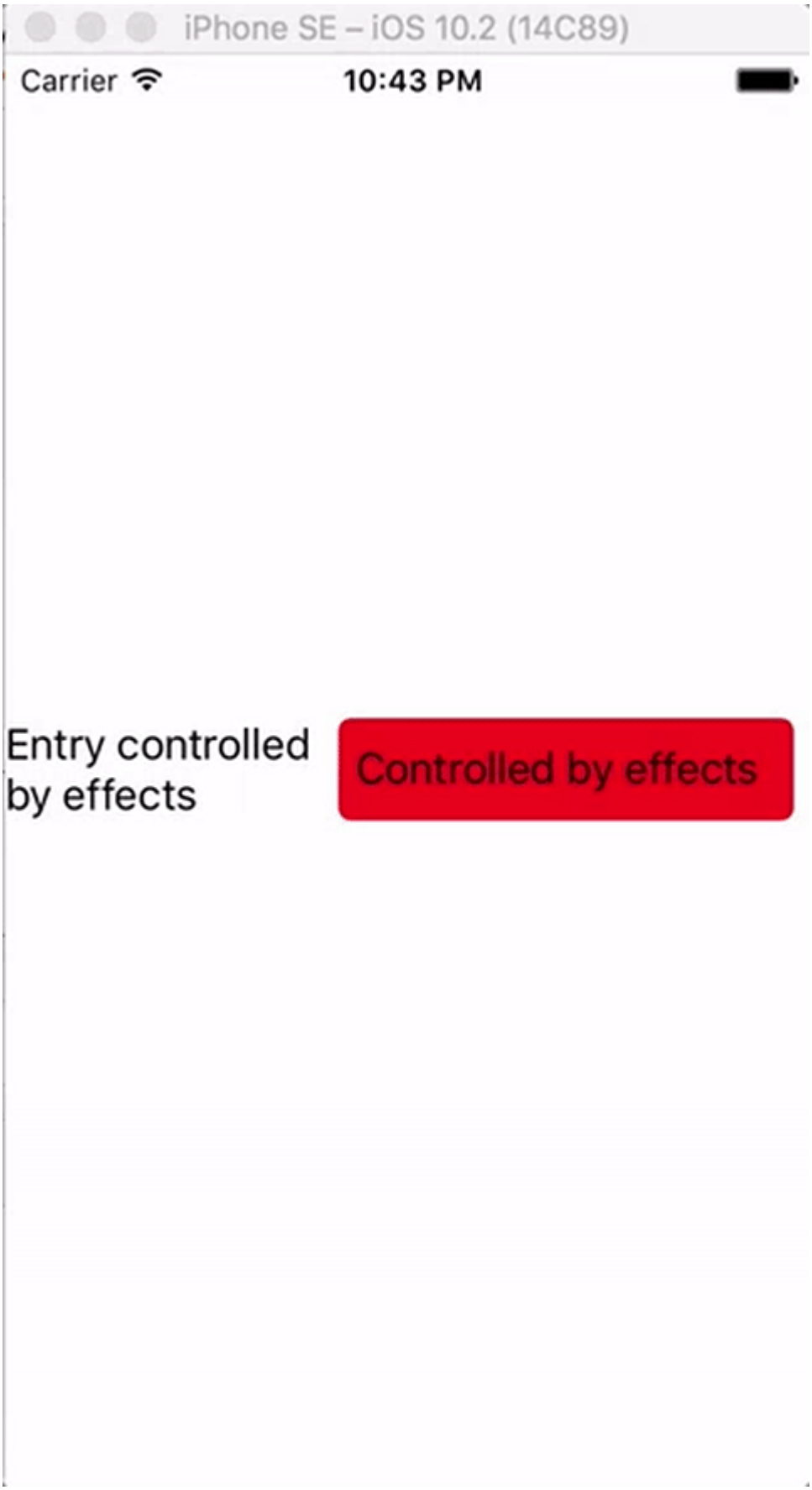
}}

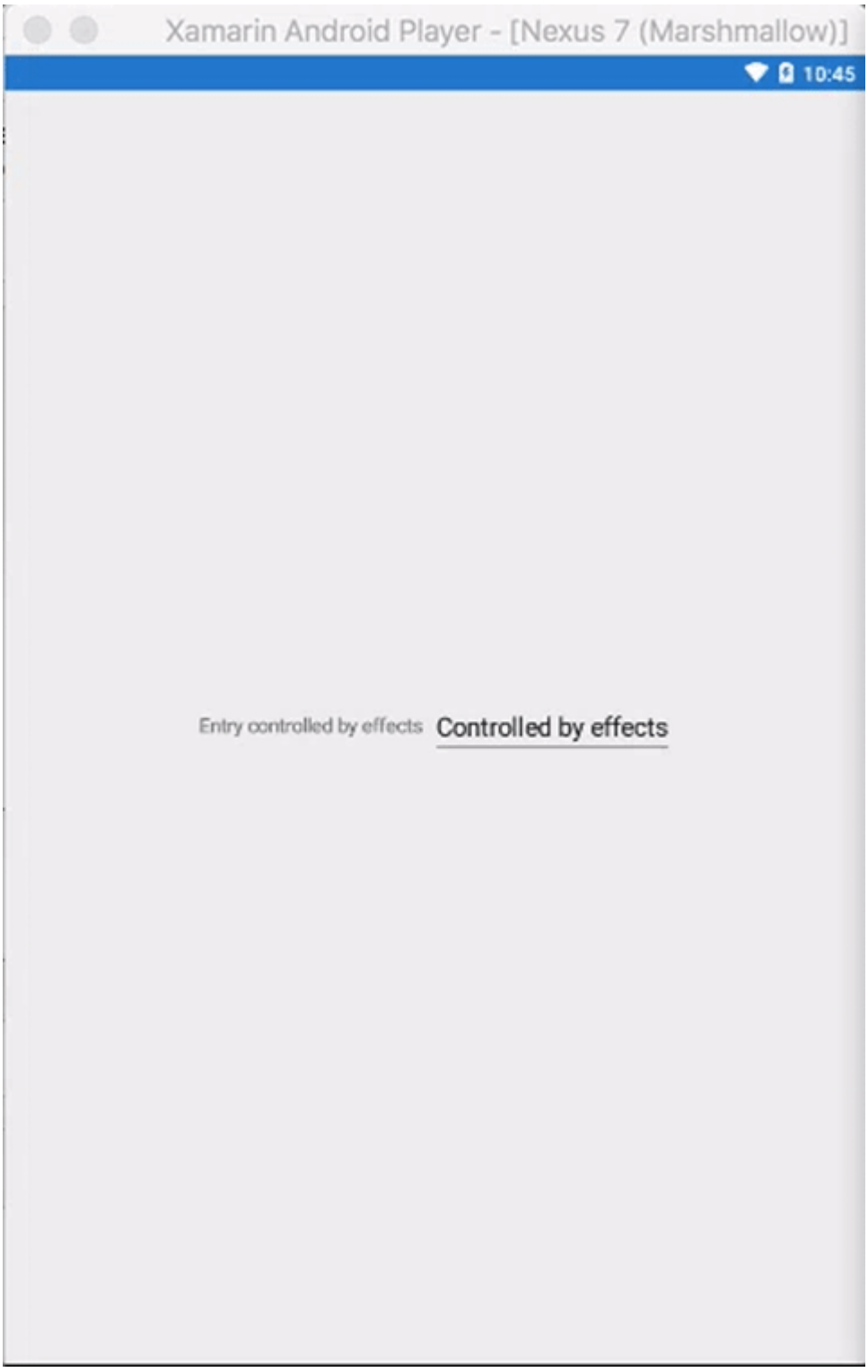
3. このエフェクトをアプリケーションでするには、 `PCL`プロジェクトで、 `FocusEffect`からする`RoutingEffect`というのしいクラスをします。これは、PCLがプラットフォームのエフェクトのインプリメンテーションをインスタンスするためにです。のサンプルコード

```
using Xamarin.Forms;
namespace EffectsDemo
{
    public class FocusEffect : RoutingEffect
    {
        public FocusEffect() : base("xhackers.FocusEffect")
        {
        }
    }
}
```

4. XAMLの`Entry`コントロールにをする

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:EffectsDemo" x:Class="EffectsDemo.EffectsDemoPage">
<StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
<Label Text="Effects Demo" HorizontalOptions="StartAndExpand" VerticalOptions="Center"
></Label>
<Entry Text="Controlled by effects" HorizontalOptions="FillAndExpand"
VerticalOptions="Center">
    <Entry.Effects>
        <local:FocusEffect>
        </local:FocusEffect>
    </Entry.Effects>
</Entry>
</StackLayout>
</ContentPage>
```


iPhone SE – iOS 10.2 (14C89)
Carrier
10:43 PM
Entry controlled by effects
Controlled by effects



エフェクトはiOSバージョンでのみされていたため、 `Entry`がされても`iOS Simulator`され、 `Droid`プロジェクトで`Effect`がされなかったため、 `Android Emulator`もこりません

オンラインでエフェクトをむ https://riptutorial.com/ja/xamarin-forms/topic/9252/エフェクト

22: カスタムコントロールの

Examples

Xamarin Formsカスタムコントロールをするネイティブはありません

は、なXamarinフォームカスタムコントロールのです。これにしてカスタムレンダリングはされていませんが、にはのコードでにすることができます。 `Label`と`Entry`にしてカスタムレンダラーとじコントロールをします。

カスタムコントロールは、 `ContentView`と`Label` 、 `Entry` 、および`BoxView` 2してのにされ、そのに、`StackLayout`。また、のバインドなプロパティと`TextChanged`イベントをします。

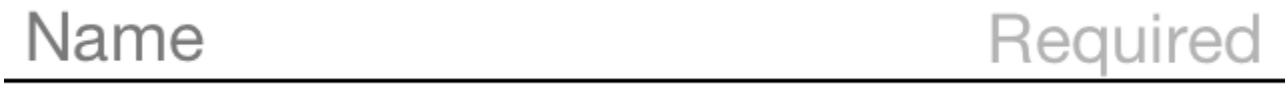
カスタムのバインドなプロパティは、のようにされ、コントロールのこのは`Label`と`Entry` がカスタムのバインドなプロパティにバインドされることによってします。
`BindingPropertyChangedDelegate`をするがあります。

```
public class InputFieldContentView : ContentView {

    #region Properties

    /// <summary>
    /// Attached to the <c>InputFieldContentView</c>'s <c>ExtendedEntryOnTextChanged()</c>
event, but returns the <c>sender</c> as <c>InputFieldContentView</c>.
    /// </summary>
    public event System.EventHandler<TextChangedEventArgs> OnContentViewTextChangedEvent; //In
OnContentViewTextChangedEvent() we return our custom InputFieldContentView control as the
sender but we could have returned the Entry itself as the sender if we wanted to do that
instead.

    public static readonly BindableProperty LabelTextProperty =
BindableProperty.Create("LabelText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string LabelText {
        get { return (string)GetValue(LabelTextProperty); }
        set { SetValue(LabelTextProperty, value); }
    }

    public static readonly BindableProperty LabelColorProperty =
BindableProperty.Create("LabelColor", typeof(Color), typeof(InputFieldContentView),
Color.Default);

    public Color LabelColor {
        get { return (Color)GetValue(LabelColorProperty); }
        set { SetValue(LabelColorProperty, value); }
    }

    public static readonly BindableProperty EntryTextProperty =
BindableProperty.Create("EntryText", typeof(string), typeof(InputFieldContentView),
string.Empty, BindingMode.TwoWay, null, OnEntryTextChanged);

    public string EntryText {
```

```
        get { return (string)GetValue(EntryTextProperty); }
        set { SetValue(EntryTextProperty, value); }
    }

    public static readonly BindableProperty PlaceholderTextProperty =
BindableProperty.Create("PlaceholderText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string PlaceholderText {
        get { return (string)GetValue(PlaceholderTextProperty); }
        set { SetValue(PlaceholderTextProperty, value); }
    }

    public static readonly BindableProperty UnderlineColorProperty =
BindableProperty.Create("UnderlineColor", typeof(Color), typeof(InputFieldContentView),
Color.Black, BindingMode.TwoWay, null, UnderlineColorChanged);

    public Color UnderlineColor {
        get { return (Color)GetValue(UnderlineColorProperty); }
        set { SetValue(UnderlineColorProperty, value); }
    }

    private BoxView _underline;

    #endregion

    public InputFieldContentView() {

        BackgroundColor   = Color.Transparent;
        HorizontalOptions = LayoutOptions.FillAndExpand;

        Label label = new Label {
            BindingContext    = this,
            HorizontalOptions = LayoutOptions.StartAndExpand,
            VerticalOptions   = LayoutOptions.Center,
            TextColor         = Color.Black
        };

        label.SetBinding(Label.TextProperty, (InputFieldContentView view) => view.LabelText,
BindingMode.TwoWay);
        label.SetBinding(Label.TextColorProperty, (InputFieldContentView view) =>
view.LabelColor, BindingMode.TwoWay);

        Entry entry = new Entry {
            BindingContext          = this,
            HorizontalOptions       = LayoutOptions.End,
            TextColor               = Color.Black,
            HorizontalTextAlignment = TextAlignment.End
        };

        entry.SetBinding(Entry.PlaceholderProperty, (InputFieldContentView view) =>
view.PlaceholderText, BindingMode.TwoWay);
        entry.SetBinding(Entry.TextProperty, (InputFieldContentView view) => view.EntryText,
BindingMode.TwoWay);

        entry.TextChanged += OnTextChangedEvent;

        _underline = new BoxView {
            BackgroundColor   = Color.Black,
            HeightRequest     = 1,
            HorizontalOptions = LayoutOptions.FillAndExpand
```

```
        };

        Content = new StackLayout {
            Spacing           = 0,
            HorizontalOptions = LayoutOptions.FillAndExpand,
            Children          = {
                new StackLayout {
                    Padding           = new Thickness(5, 0),
                    Spacing           = 0,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    Orientation       = StackOrientation.Horizontal,
                    Children          = { label, entry }
                }, _underline
            }
        };

        SizeChanged += (sender, args) => entry.WidthRequest = Width * 0.5 - 10;
    }

    private static void OnEntryTextChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView = (InputFieldContentView)bindable;
        contentView.EntryText             = (string)newValue;
    }

    private void OnTextChangedEvent(object sender, TextChangedEventArgs args) {
        if(OnContentViewTextChangedEvent != null) { OnContentViewTextChangedEvent(this, new
TextChangedEventArgs(args.OldTextValue, args.NewTextValue)); } //Here is where we pass in
'this' (which is the InputFieldContentView) instead of 'sender' (which is the Entry control)
    }

    private static void UnderlineColorChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView      = (InputFieldContentView)bindable;
        contentView._underline.BackgroundColor = (Color)newValue;
    }
}
```

そして、ここにiOSののがありますには、iOSでボーダーをしてののカスタムフォントをするためにされているLabelとEntryカスタムレンダラをしたときのがされます。



UnderlineColorされたときにBoxView.BackgroundColorがされるというBoxView.BackgroundColorしました。 BoxViewのBackgroundColorプロパティをバインドしたでも、 UnderlineColorChangedデリゲートをするまではされません。

バインドなスパンのコレクションでラベルけする

FormattedTextプロパティのりにラッパーをつカスタムラベルをしました

```
public class MultiComponentLabel : Label
{
    public IList<TextComponent> Components { get; set; }
```

```
    public MultiComponentLabel()
    {
        var components = new ObservableCollection<TextComponent>();
        components.CollectionChanged += OnComponentsChanged;
        Components = components;
    }

    private void OnComponentsChanged(object sender, NotifyCollectionChangedEventArgs e)
    {
        BuildText();
    }

    private void OnComponentPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        BuildText();
    }

    private void BuildText()
    {
        var formattedString = new FormattedString();
        foreach (var component in Components)
        {
            formattedString.Spans.Add(new Span { Text = component.Text });
            component.PropertyChanged -= OnComponentPropertyChanged;
            component.PropertyChanged += OnComponentPropertyChanged;
        }

        FormattedText = formattedString;
    }
}
```

カスタム`TextComponent`のコレクションをしました

```
public class TextComponent : BindableObject
{
    public static readonly BindableProperty TextProperty =
        BindableProperty.Create(nameof(Text),
                                typeof(string),
                                typeof(TextComponent),
                                default(string));

    public string Text
    {
        get { return (string)GetValue(TextProperty); }
        set { SetValue(TextProperty, value); }
    }
}
```

テキストコンポーネントのコレクションがされたり、のコンポーネントの`Text`プロパティがされたりすると、ベース`Label FormattedText`プロパティが`FormattedText`ます。

`XAML`どのようにしたのですか

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
```

```
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.MultiComponentLabelPage">
<controls:MultiComponentLabel Margin="0,20,0,0">
    <controls:MultiComponentLabel.Components>
      <controls:TextComponent Text="Time"/>
      <controls:TextComponent Text=": "/>
      <controls:TextComponent Text="{Binding CurrentTime, Source={x:Reference Page}}"/>
    </controls:MultiComponentLabel.Components>
  </controls:MultiComponentLabel>
</ContentPage>
```

ページのコードビハインド

```
public partial class MultiComponentLabelPage : ContentPage
{
    private string _currentTime;

    public string CurrentTime
    {
        get { return _currentTime; }
        set
        {
            _currentTime = value;
            OnPropertyChanged();
        }
    }

    public MultiComponentLabelPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();

        Device.StartTimer(TimeSpan.FromSeconds(1), () =>
        {
            CurrentTime = DateTime.Now.ToString("hh : mm : ss");
            return true;
        });
    }
}
```

MaxLengthプロパティをしてカスタムEntryコントロールをする

Xamarinフォームエントリコントロールには、 MaxLengthプロパティはありません。これをするには、Bindable MaxLengthプロパティをして、のようにEntryをできます。に、 EntryのTextChangedイベントをし、これがびされたときにTextのさをするだけです。

```
class CustomEntry : Entry
{
    public CustomEntry()
    {
        base.TextChanged += Validate;
```

```
    }

    public static readonly BindableProperty MaxLengthProperty =
BindableProperty.Create(nameof(MaxLength), typeof(int), typeof(CustomEntry), 0);

    public int MaxLength
    {
        get { return (int)GetValue(MaxLengthProperty); }
        set { SetValue(MaxLengthProperty, value); }
    }

    public void Validate(object sender, TextChangedEventArgs args)
    {
        var e = sender as Entry;
        var val = e?.Text;

        if (string.IsNullOrEmpty(val))
            return;

        if (MaxLength > 0 && val.Length > MaxLength)
            val = val.Remove(val.Length - 1);

        e.Text = val;
    }
}
```

XAMLでの

```
   <ContentView xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:customControls="clr-namespace:CustomControls;assembly=CustomControls"
         x:Class="Views.TestView">
<ContentView.Content>
    <customControls:CustomEntry MaxLength="10" />
</ContentView.Content>
```

オンラインでカスタムコントロールのをむ https://riptutorial.com/ja/xamarin-forms/topic/3913/カスタムコントロールの

23: カスタムコントロールの

き

すべてのXamarin.Formsビューには、ネイティブコントロールのインスタンスをするプラットフォームのレンダラーがしています。ビューがのプラットフォームでレンダリングされると、ViewRendererクラスがインスタンスされます。

これをうプロセスはのとおりです。

Xamarin.Formsカスタムコントロールをします。

Xamarin.Formsからカスタムコントロールをします。

プラットフォームのコントロールのカスタムレンダラーをします。

Examples

CheckBoxコントロールの

このでは、AndroidとiOSのカスタムチェックボックスをします。

カスタムコントロールの

```
namespace CheckBoxCustomRendererExample
{
    public class Checkbox : View
    {
        public static readonly BindableProperty IsCheckedProperty =
BindableProperty.Create<Checkbox, bool>(p => p.IsChecked, true, propertyChanged: (s, o, n) =>
{ (s as Checkbox).OnChecked(new EventArgs()); });
        public static readonly BindableProperty ColorProperty =
BindableProperty.Create<Checkbox, Color>(p => p.Color, Color.Default);

        public bool IsChecked
        {
            get
            {
                return (bool)GetValue(IsCheckedProperty);
            }
            set
            {
                SetValue(IsCheckedProperty, value);
            }
        }

        public Color Color
        {
            get
```

```
            {
                return (Color)GetValue(ColorProperty);
            }
            set
            {
                SetValue(ColorProperty, value);
            }
        }

        public event EventHandler Checked;

        protected virtual void OnChecked(EventArgs e)
        {
            if (Checked != null)
                Checked(this, e);
        }
    }
}
```

まず、Androidのカスタムレンダラーから、このソリューションの`Android`にしいクラス `CheckboxCustomRenderer` をします。

すべきいくつかのな

- レンダラーが`Xamarin.Forms`されるように、クラスのにExportRendererをするがあります。このようにして、 `Xamarin.Forms`は`Android`で`Checkbox`オブジェクトをしようとするときに、このレンダラーをします。
- `OnElementChanged`メソッドでは、ほとんどのをいます。ここでは、ネイティブコントロールをインスタンスしてします。

カスタムコントロールをする

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:CheckBoxCustomRendererExample"
x:Class="CheckBoxCustomRendererExample.CheckBoxCustomRendererExamplePage">
    <StackLayout Padding="20">
        <local:Checkbox Color="Aqua" />
    </StackLayout>
</ContentPage>
```

プラットフォームでのカスタムレンダラーの

カスタムレンダラークラスをするプロセスはのとおりです。

1. カスタムコントロールをレンダリングする`ViewRenderer<T1,T2>`クラスのサブクラスをします。ののはレンダラーのためのカスタムコントロール、このは`CheckBox`なければなりません。2のは、カスタムコントロールをするネイティブコントロールです。
2. カスタムコントロールをする`OnElementChanged`メソッドをオーバーライドし、カスタムコントロールをカスタマイズするロジックをします。このメソッドは、する`Xamarin.Forms`コント

ロールがされたときにびされます。
3. カスタムレンダラークラスに`ExportRenderer`をして、 `Xamarin.Forms`カスタムコントロールのレンダリングにすることをします。このは、 `Xamarin.Forms`カスタムレンダラーをするためにされます。

Androidカスタムレンダラーの

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.Droid
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBox>
    {
        private CheckBox checkBox;

        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);
            var model = e.NewElement;
            checkBox = new CheckBox(Context);
            checkBox.Tag = this;
            CheckboxPropertyChanged(model, null);
            checkBox.SetOnClickListener(new ClickListener(model));
            SetNativeControl(checkBox);
        }
        private void CheckboxPropertyChanged(Checkbox model, String propertyName)
        {
            if (propertyName == null || Checkbox.IsCheckedProperty.PropertyName ==
propertyName)
            {
                checkBox.Checked = model.IsChecked;
            }

            if (propertyName == null || Checkbox.ColorProperty.PropertyName == propertyName)
            {
                int[][] states = {
                    new int[] { Android.Resource.Attribute.StateEnabled}, // enabled
                    new int[] {Android.Resource.Attribute.StateEnabled}, // disabled
                    new int[] {Android.Resource.Attribute.StateChecked}, // unchecked
                    new int[] { Android.Resource.Attribute.StatePressed}  // pressed
                };
                var checkBoxColor = (int)model.Color.ToAndroid();
                int[] colors = {
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor
                };
                var myList = new Android.Content.Res.ColorStateList(states, colors);
                checkBox.ButtonTintList = myList;

            }
        }

        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            if (checkBox != null)
            {
```

```
                base.OnElementPropertyChanged(sender, e);

                CheckboxPropertyChanged((Checkbox)sender, e.PropertyName);
            }
        }

        public class ClickListener : Java.Lang.Object, IOnClickListener
        {
            private Checkbox _myCheckbox;
            public ClickListener(Checkbox myCheckbox)
            {
                this._myCheckbox = myCheckbox;
            }
            public void OnClick(global::Android.Views.View v)
            {
                _myCheckbox.IsChecked = !_myCheckbox.IsChecked;
            }
        }
    }
}
```

iOSけカスタムレンダラーの

iOSではチェックボックスにみまれていないので、に`CheckBoxView`をし、にXamarin.Formsチェックボックスのレンダラーをします。

`CheckBoxView`はchecked_checkbox.pngとunchecked_checkbox.pngという2つのにづいているため、 `Color`プロパティはされます。

CheckBoxビュー

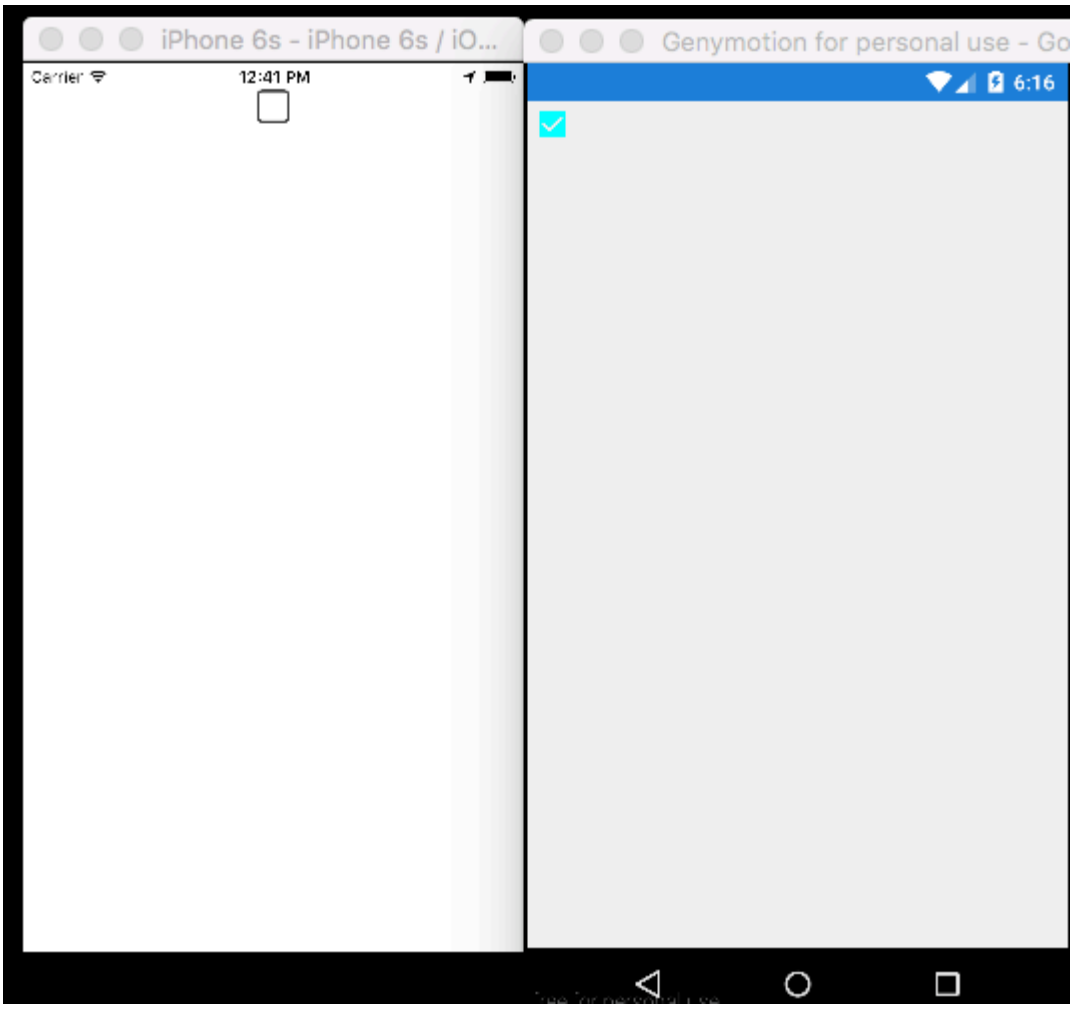
```
namespace CheckBoxCustomRendererExample.iOS
{
    [Register("CheckBoxView")]
    public class CheckBoxView : UIButton
    {
        public CheckBoxView()
        {
            Initialize();
        }

        public CheckBoxView(CGRect bounds)
            : base(bounds)
        {
            Initialize();
        }

        public string CheckedTitle
        {
            set
            {
                SetTitle(value, UIControlState.Selected);
            }
        }

        public string UncheckedTitle
```

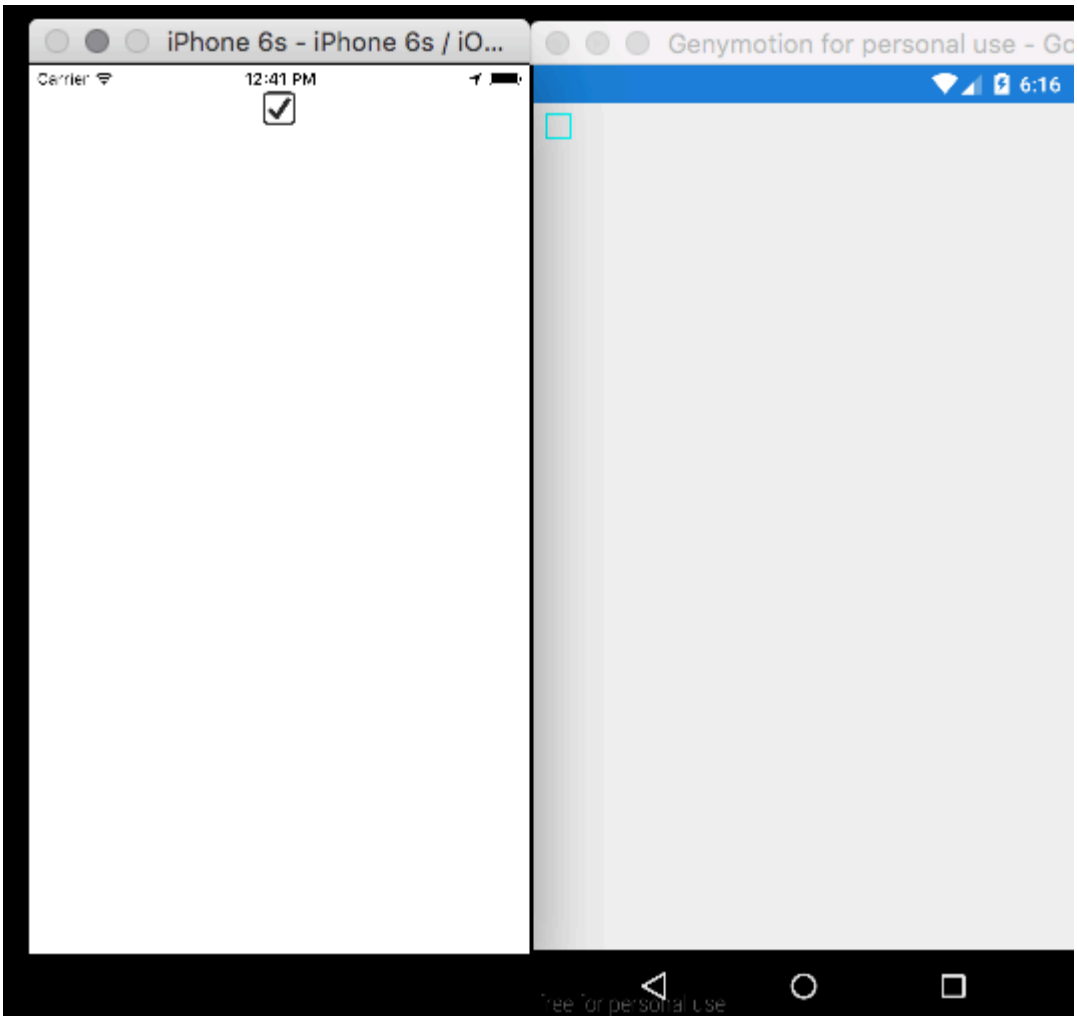
```
        {
            set
            {
                SetTitle(value, UIControlState.Normal);
            }
        }

        public bool Checked
        {
            set { Selected = value; }
            get { return Selected; }
        }

        void Initialize()
        {
            ApplyStyle();

            TouchUpInside += (sender, args) => Selected = !Selected;
            // set default color, because type is not UIButtonType.System
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Normal);
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Selected);
        }

        void ApplyStyle()
        {
            SetImage(UIImage.FromBundle("Images/checked_checkbox.png"),
UIControlState.Selected);
            SetImage(UIImage.FromBundle("Images/unchecked_checkbox.png"),
UIControlState.Normal);
        }
    }
}
```

CheckBoxカスタムレンダラー

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.iOS
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBoxView>
    {

        /// <summary>
        /// Handles the Element Changed event
        /// </summary>
        /// <param name="e">The e.</param>
        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);

            if (Element == null)
                return;

            BackgroundColor = Element.BackgroundColor.ToUIColor();
            if (e.NewElement != null)
            {
                if (Control == null)
                {
                    var checkBox = new CheckBoxView(Bounds);
                    checkBox.TouchUpInside += (s, args) => Element.IsChecked =
Control.Checked;
```

```
                    SetNativeControl(checkBox);
                }
                Control.Checked = e.NewElement.IsChecked;
            }

            Control.Frame = Frame;
            Control.Bounds = Bounds;

        }

        /// <summary>
        /// Handles the <see cref="E:ElementPropertyChanged" /> event.
        /// </summary>
        /// <param name="sender">The sender.</param>
        /// <param name="e">The <see cref="PropertyChangedEventArgs"/> instance containing the
event data.</param>
        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName.Equals("Checked"))
            {
                Control.Checked = Element.IsChecked;
            }
        }
    }
}
```





オンラインでカスタムコントロールのをむ https://riptutorial.com/ja/xamarin-forms/topic/5975/カスタムコントロールの

24: カスタムコントロールの

Examples

カスタムボタンの

```
/// <summary>
/// Button with some additional options
/// </summary>
public class TurboButton : Button
{
    public static readonly BindableProperty StringDataProperty = BindableProperty.Create(
      propertyName: "StringData",
      returnType: typeof(string),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(string));

    public static readonly BindableProperty IntDataProperty = BindableProperty.Create(
      propertyName: "IntData",
      returnType: typeof(int),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(int));

    /// <summary>
    /// You can put here some string data
    /// </summary>
    public string StringData
    {
        get { return (string)GetValue(StringDataProperty); }
        set { SetValue(StringDataProperty, value); }
    }

    /// <summary>
    /// You can put here some int data
    /// </summary>
    public int IntData
    {
        get { return (int)GetValue(IntDataProperty); }
        set { SetValue(IntDataProperty, value); }
    }

    public TurboButton()
    {
        PropertyChanged += CheckIfPropertyLoaded;
    }

    /// <summary>
    /// Called when one of properties is changed
    /// </summary>
    private void CheckIfPropertyLoaded(object sender, PropertyChangedEventArgs e)
    {
        //example of using PropertyChanged
        if(e.PropertyName == "IntData")
        {
            //IntData is now changed, you can operate on updated value
        }
```

```
    }
}
```

XAMLでの

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="SomeApp.Pages.SomeFolder.Example"
    xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp">
    <StackLayout>
        <customControls:TurboButton x:Name="exampleControl" IntData="2" StringData="Test" />
    </StackLayout>
</ContentPage>
```

さて、あなたのプロパティをCでうことができます

```
exampleControl.IntData
```

TurboButtonクラスがプロジェクトにされるは、でするがあります。はこのでそれをやった

```
xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp"
```

「customControls」をのににすることができます。どのようにびすかはあなたです。

オンラインでカスタムコントロールのをむ https://riptutorial.com/ja/xamarin-forms/topic/6592/カスタムコントロールの

25: カスタムレンダラー

Examples

ListViewのカスタムレンダラー

カスタムレンダラーをすると、はプラットフォームのXamarin.Formsコントロールのとをカスタマイズできます。はネイティブコントロールのをできます。

たとえば、 `ListView`スクロールをにするがあります。 iOSですべてのアイテムがにされていても`ListView`をスクロールすることはできません。 `Xamarin.Forms.ListView`はそのようなをしません。この、レンダラーがけにています。

まず、なバインドなプロパティをするカスタムコントロールをPCLプロジェクトでするがあります。

```
public class SuperListView : ListView
{
    public static readonly BindableProperty IsScrollingEnableProperty =
        BindableProperty.Create(nameof(IsScrollingEnable),
                                typeof(bool),
                                typeof(SuperListView),
                                true);

    public bool IsScrollingEnable
    {
        get { return (bool)GetValue(IsScrollingEnableProperty); }
        set { SetValue(IsScrollingEnableProperty, value); }
    }
}
```

のステップでは、プラットフォームのレンダラーをします。

iOS

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.iOS.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;


            Control.ScrollEnabled = superListView.IsScrollingEnable;
        }
```

```
    }
}
```

アンドロイドAndroidのリストには、すべてのアイテムがにされているはスクロールしないので、スクロールをにすることはできませんが、ネイティブプロパティをすることはできます。

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.Droid.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void
OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;
        }
    }
}
```

レンダラーの`Element`プロパティは、PCLプロジェクトの`SuperListView`コントロールです。

レンダラーの`Control`プロパティはネイティブコントロールです。 `Android.Widget.ListView` Android とのため`UIKit.UITableView` iOS。

`XAML`どのようにするのでしょうか

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.SuperListViewPage">


  <controls:SuperListView ItemsSource="{Binding Items, Source={x:Reference Page}}"
                          IsScrollingEnable="false"
                          Margin="20">
    <controls:SuperListView.ItemTemplate>
      <DataTemplate>
        <ViewCell>
          <Label Text="{Binding .}"/>
        </ViewCell>
      </DataTemplate>
    </controls:SuperListView.ItemTemplate>
  </controls:SuperListView>
</ContentPage>
```

ページの`.cs`ファイル

```
public partial class SuperListViewPage : ContentPage
{
    private ObservableCollection<string> _items;
```

```
    public ObservableCollection<string> Items
    {
        get { return _items; }
        set
        {
            _items = value;
            OnPropertyChanged();
        }
    }

    public SuperListViewPage()
    {
        var list = new SuperListView();

        InitializeComponent();

        var items = new List<string>(10);
        for (int i = 1; i <= 10; i++)
        {
            items.Add($"Item {i}");
        }

        Items = new ObservableCollection<string>(items);
    }
}
```

BoxViewのカスタムレンダラー

カスタムレンダラのヘルプをすると、しいプロパティをして、ネイティブプラットフォームでのでレンダリングすることができます。ネイティブプラットフォームでは、コードをすることはできません。このでは、とをボックスビューにします。

まず、なバインドなプロパティをするカスタムコントロールをPCLプロジェクトでするがあります。

```
namespace Mobile.Controls
{
    public class ExtendedBoxView : BoxView
    {
        /// <summary>
        /// Respresents the background color of the button.
        /// </summary>
        public static readonly BindableProperty BorderRadiusProperty =
BindableProperty.Create<ExtendedBoxView, double>(p => p.BorderRadius, 0);

        public double BorderRadius
        {
            get { return (double)GetValue(BorderRadiusProperty); }
            set { SetValue(BorderRadiusProperty, value); }
        }

        public static readonly BindableProperty StrokeProperty =
            BindableProperty.Create<ExtendedBoxView, Color>(p => p.Stroke, Color.Transparent);

        public Color Stroke
        {
```

```
            get { return (Color)GetValue(StrokeProperty); }
            set { SetValue(StrokeProperty, value); }
        }

        public static readonly BindableProperty StrokeThicknessProperty =
            BindableProperty.Create<ExtendedBoxView, double>(p => p.StrokeThickness, 0);

        public double StrokeThickness
        {
            get { return (double)GetValue(StrokeThicknessProperty); }
            set { SetValue(StrokeThicknessProperty, value); }
        }
    }
}
```

のステップでは、プラットフォームのレンダラーをします。

iOS

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.iOS.Renderers
{
 public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
 {
    public ExtendedBoxViewRenderer()
    {
    }

    protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
    {
        base.OnElementChanged(e);
        if (Element == null)
            return;

        Layer.MasksToBounds = true;
        Layer.CornerRadius = (float)((ExtendedBoxView)this.Element).BorderRadius / 2.0f;
    }


    protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        base.OnElementPropertyChanged(sender, e);
        if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
        {
            SetNeedsDisplay();
        }
    }

    public override void Draw(CGRect rect)
    {
        ExtendedBoxView roundedBoxView = (ExtendedBoxView)this.Element;
        using (var context = UIGraphics.GetCurrentContext())
        {
            context.SetFillColor(roundedBoxView.Color.ToCGColor());
            context.SetStrokeColor(roundedBoxView.Stroke.ToCGColor());
            context.SetLineWidth((float)roundedBoxView.StrokeThickness);

            var rCorner = this.Bounds.Inset((int)roundedBoxView.StrokeThickness / 2,
(int)roundedBoxView.StrokeThickness / 2);
```

```
            nfloat radius = (nfloat)roundedBoxView.BorderRadius;
            radius = (nfloat)Math.Max(0, Math.Min(radius, Math.Max(rCorner.Height / 2,
rCorner.Width / 2)));

            var path = CGPath.FromRoundedRect(rCorner, radius, radius);
            context.AddPath(path);
            context.DrawPath(CGPathDrawingMode.FillStroke);
        }
    }

 }
}
```

りしますが、メソッドでにじてカスタマイズすることもできます。

アンドロイドでもじです

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.Droid
{
    /// <summary>
    ///
    /// </summary>
    public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        /// <summary>
        ///
        /// </summary>
        public ExtendedBoxViewRenderer()
        {
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="e"></param>
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            SetWillNotDraw(false);

            Invalidate();
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
            {
                Invalidate();
```

```
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="canvas"></param>
        public override void Draw(Canvas canvas)
        {
            var box = Element as ExtendedBoxView;
            base.Draw(canvas);
            Paint myPaint = new Paint();

            myPaint.SetStyle(Paint.Style.Stroke);
            myPaint.StrokeWidth = (float)box.StrokeThickness;
            myPaint.SetARGB(convertTo255ScaleColor(box.Color.A),
convertTo255ScaleColor(box.Color.R), convertTo255ScaleColor(box.Color.G),
convertTo255ScaleColor(box.Color.B));
            myPaint.SetShadowLayer(20, 0, 5, Android.Graphics.Color.Argb(100, 0, 0, 0));

            SetLayerType(Android.Views.LayerType.Software, myPaint);

            var number = (float)box.StrokeThickness / 2;
            RectF rectF = new RectF(
                        number, // left
                        number, // top
                        canvas.Width - number, // right
                        canvas.Height - number // bottom
                );


            var radius = (float)box.BorderRadius;
            canvas.DrawRoundRect(rectF, radius, radius, myPaint);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="color"></param>
        /// <returns></returns>
        private int convertTo255ScaleColor(double color)
        {
            return (int) Math.Ceiling(color * 255);
        }
}
```

}

XAML

にしたで、たちのコントロールをします。

```
xmlns:Controls="clr-namespace:Mobile.Controls"
```

に、コントロールをのようにし、にされたプロパティをします。

```
<Controls:ExtendedBoxView
    x:Name="search_boxview"
```

```
    Color="#444"
    BorderRadius="5"
    HorizontalOptions="CenterAndExpand"
    />
```

ネイティブプロジェクトからレンダラーにアクセスする

```
var renderer = Platform.GetRenderer(visualElement);

if (renderer == null)
{
    renderer = Platform.CreateRenderer(visualElement);
    Platform.SetRenderer(visualElement, renderer);
}

DoSomeThingWithRender(render); // now you can do whatever you want with render
```

FramePCLiOSパーツのカスタムレンダラーをえたいラベル

のステップPCLパート

```
using Xamarin.Forms;

namespace ProjectNamespace
{
    public class ExtendedFrame : Frame
    {
        /// <summary>
        /// The corner radius property.
        /// </summary>
        public static readonly BindableProperty CornerRadiusProperty =
            BindableProperty.Create("CornerRadius", typeof(double), typeof(ExtendedFrame),
0.0);

        /// <summary>
        /// Gets or sets the corner radius.
        /// </summary>
        public double CornerRadius
        {
            get { return (double)GetValue(CornerRadiusProperty); }
            set { SetValue(CornerRadiusProperty, value); }
        }
    }
}
```

2ステップiOSパート

```
using ProjectNamespace;
using ProjectNamespace.iOS;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ExportRenderer(typeof(ExtendedFrame), typeof(ExtendedFrameRenderer))]
namespace ProjectNamespace.iOS
{
```

```
    public class ExtendedFrameRenderer : FrameRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<Frame> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.MasksToBounds = true;
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedFrame.CornerRadiusProperty.PropertyName)
            {
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }
    }
}
```

3のExtendedFrameをびすXAMLコード

XAMLのでそれをしたい、これをくことをれないでください

```
xmlns:controls="clr-namespace:ProjectNamespace;assembly:ProjectNamespace"
```

```
xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
```

これで、ExtendedFrameをのようにすることができます

```
<controls:ExtendedFrame
    VerticalOptions="FillAndExpand"
    HorizontalOptions="FillAndExpand"
    BackgroundColor="Gray"
    CornerRadius="35.0">
    <Frame.Content>
        <Label
            Text="MyText"
            TextColor="Blue"/>
    </Frame.Content>
</controls:ExtendedFrame>
```

なをつみをびたBoxView

のステップPCLパート

```
public class RoundedBoxView : BoxView
```

```
{
    public static readonly BindableProperty CornerRadiusProperty =
        BindableProperty.Create("CornerRadius", typeof(double), typeof(RoundedEntry),
default(double));

    public double CornerRadius
    {
        get
        {
            return (double)GetValue(CornerRadiusProperty);
        }
        set
        {
            SetValue(CornerRadiusProperty, value);
        }
    }

    public static readonly BindableProperty FillColorProperty =
        BindableProperty.Create("FillColor", typeof(string), typeof(RoundedEntry),
default(string));

    public string FillColor
    {
        get
        {
            return (string) GetValue(FillColorProperty);
        }
        set
        {
            SetValue(FillColorProperty, value);
        }
    }
}
```

ドロイド

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.Droid
{
    public class RoundedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);
            SetWillNotDraw(false);
            Invalidate();
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);
            SetWillNotDraw(false);
            Invalidate();
        }

        public override void Draw(Canvas canvas)
        {
            var box = Element as RoundedBoxView;
            var rect = new Rect();
```

```
            var paint = new Paint
            {
                Color = Xamarin.Forms.Color.FromHex(box.FillColor).ToAndroid(),
                AntiAlias = true,
            };

            GetDrawingRect(rect);

            var radius = (float)(rect.Width() / box.Width * box.CornerRadius);

            canvas.DrawRoundRect(new RectF(rect), radius, radius, paint);
        }
    }
}
```

3iOSの

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.iOS
{
    public class RoundedBoxViewRenderer : BoxRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = Color.FromHex((Element as
RoundedBoxView).FillColor).ToCGColor();
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = (Element as RoundedBoxView).FillColor.ToCGColor();
            }
        }
    }
}
```

オンラインでカスタムレンダラーをむ https://riptutorial.com/ja/xamarin-forms/topic/2949/カスタムレンダラー

26: キャッシング

Examples

Akavacheをしたキャッシュ

Akavacheについて

Akavacheは、データをキャッシュするリーチをするになライブラリです。 Akavacheはキーバリューストレージインターフェイスをし、SQLite3のでします。にNo-SQLソリューションとしてスキーマをさせておくはありません。に、データをすることなくアプリをにするがあるは、ほとんどのモバイルアプリケーションにです。

Xamarinの

Akavacheはかに、Xamarinアプリケーションになキャッシングライブラリです。なデータ、バイナリデータ、またはのデータでするがないにります。の、Akavacheをしてください。

- ののデータをキャッシュするにはアプリがですされるエンティティのをできます。
- あなたのアプリをオフラインでさせたい。
- データのスキーマをしてフリーズするのはしいです。たとえば、さまざまなきオブジェクトをむリストがあります。
- データへのなキーアクセスでです。なクエリをするはありません。

Akavacheはデータストレージにとって「の」ではありませんので、のようなには2えてください。

- あなたのデータエンティティはおいにくのをっています。
- あなたはオフラインでするためにあなたのアプリをにとしません。
- あなたは、なのデータをローカルにするがあります。
- データをバージョンでするがあります。
- あなたは、グループ、などのようなSQLのななクエリをするがあります。

  には、されたフィールドをみきするだけで、でデータをすることができます。

な

Akavacheとのがとばれるオブジェクトをしてわれます`BlobCache`。

Akavacheのメソッドのほとんどはなをしますが、メソッドのおかげでそれらもつことができます

。

```
using System.Reactive.Linq;   // IMPORTANT - this makes await work!

// Make sure you set the application name before doing any inserts or gets
BlobCache.ApplicationName = "AkavacheExperiment";

var myToaster = new Toaster();
await BlobCache.UserAccount.InsertObject("toaster", myToaster);

//
// ...later, in another part of town...
//

// Using async/await
var toaster = await BlobCache.UserAccount.GetObject<Toaster>("toaster");

// or without async/await
Toaster toaster;

BlobCache.UserAccount.GetObject<Toaster>("toaster")
    .Subscribe(x => toaster = x, ex => Console.WriteLine("No Key!"));
```

エラー

```
Toaster toaster;

try {
    toaster = await BlobCache.UserAccount.GetObjectAsync("toaster");
} catch (KeyNotFoundException ex) {
    toaster = new Toaster();
}

// Or without async/await:
toaster = await BlobCache.UserAccount.GetObjectAsync<Toaster>("toaster")
    .Catch(Observable.Return(new Toaster()));
```

オンラインでキャッシングをむ https://riptutorial.com/ja/xamarin-forms/topic/3644/キャッシング

27: ジェスチャー

Examples

TapGestureRecognizerをしてをタップにする

Xamarin.Formsにはいくつかのデフォルトがあり、そのうちの1つは`TapGestureRecognizer`です。

ほとんどすべてのビジュアルにできます。 `Image`バインドするなをてみましょう。ここでコードでそれをうです。

```
var tappedCommand = new Command(() =>
{
    //handle the tap
});

var tapGestureRecognizer = new TapGestureRecognizer { Command = tappedCommand };
image.GestureRecognizers.Add(tapGestureRecognizer);
```

またはXAML

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer
                Command="{Binding TappedCommand}"
                NumberOfTapsRequired="2" />
  </Image.GestureRecognizers>
</Image>
```

ここでは、コマンドはデータバインディングをしてされます。ごのように、 `NumberOfTapsRequired`をして、よりくのタップをにしてからアクションをすることもできます。デフォルトは1タップです。

のジェスチャーはピンチとパンです。

ピンチジェスチャーでをズームする

`Image` またはのをズームにするには、 `PinchGestureRecognizer`をする`PinchGestureRecognizer`があります。コードでこれをうはのとおりです。

```
var pinchGesture = new PinchGestureRecognizer();
pinchGesture.PinchUpdated += (s, e) => {
// Handle the pinch
};

image.GestureRecognizers.Add(pinchGesture);
```

しかし、XAMLからもうことができます。

```
<Image Source="waterfront.jpg">
  <Image.GestureRecognizers>
    <PinchGestureRecognizer PinchUpdated="OnPinchUpdated" />
  </Image.GestureRecognizers>
</Image>
```

ィベントハンドラでは、をズ―ムするためのコ―ドをするがあります。もちろん、のもすることができます。

```
void OnPinchUpdated (object sender, PinchGestureUpdatedEventArgs e)
{
    // ... code here
}
```

のジェスチャ―はタップとパンです。

PanGestureRecognizerでズ―ムされたコンテンツをすべてする

あなたがズ―ムしていたとき`Image`またはのコンテンツをあなたはりにドラッグすることもできます`Image`のでズ―ムで、そののすべてをします。

これはPanGestureRecognizerをすることでできます。コ―ドから、これはのようになります

```
var panGesture = new PanGestureRecognizer();
panGesture.PanUpdated += (s, e) => {
  // Handle the pan
};

image.GestureRecognizers.Add(panGesture);
```

これは、XAMLからもうことができます。

```
<Image Source="MonoMonkey.jpg">
  <Image.GestureRecognizers>
    <PanGestureRecognizer PanUpdated="OnPanUpdated" />
  </Image.GestureRecognizers>
</Image>
```

コ―ドビハインドイベントでは、これにじてパンをできるようになりました。このメソッドのシグネチャをしてします。

```
void OnPanUpdated (object sender, PanUpdatedEventArgs e)
{
  // Handle the pan
}
```

ユ―ザ―がMR.Gesturesでにれたにピンをする

ジェスチャ―でされたXamarinsは、になタッチしかしません。えば、しているのをるはありません。 MR.Gesturesは、14のタッチイベントをするコンポ―ネントです。タッチするのは、すべて

のMR.Gesturesイベントにされる`EventArgs`です。

のどこにでもピンをしたいは、 `Tapping`イベントをする`MR.Gestures.AbsoluteLayout`をするのがもなです。

```
<mr:AbsoluteLayout x:Name="MainLayout" Tapping="OnTapping">
    ...
</mr:AbsoluteLayout>
```

ごのように、 `Tapping="OnTapping"`は、ネストされた`GestureRecognizers`したXamarinsよりも.NETにています。そのはiOSからコピーされ、.NETにとってはちょっといがする。

あなたのコードでは、 `OnTapping`ように`OnTapping`ハンドラをすることができます

```
private void OnTapping(object sender, MR.Gestures.TapEventArgs e)
{
    if (e.Touches?.Length > 0)
    {
        Point touch = e.Touches[0];
        var image = new Image() { Source = "pin" };
        MainLayout.Children.Add(image, touch);
    }
}
```

`Tapping`イベントのわりに、 `TappingCommand`をしてViewModelにバインドすることもできますが、このなではになります。

MR.Gesturesのサンプルは、GitHubのGestureSampleアプリとMR.GesturesのWebサイトにあります。これらはまた、イベントハンドラ、コマンド、MVVMなどののすべてのタッチイベントをするをしています...

オンラインでジェスチャーをむ https://riptutorial.com/ja/xamarin-forms/topic/3914/ジェスチャー

28: スタイルのカスタムフォント

るリソース

- Xamarinスタイル
- iOSとAndroidのカスタムフォントをXamarin.Formsでする
- カスタムレンダラー
- リソース
- プロパティ

Examples

Sylesでのカスタムフォントへのアクセス

Xamarin.Formsは、グローバルスタイルでクロスプラットフォームアプリケーションをスタイリングするためのれたメカニズムをします。

モバイルのでは、アプリケーションはきれいで、のアプリケーションかられているがあります。このの1つは、アプリケーションでされるカスタムフォントです。

Xamarin.FormsのXAML Stylingのパワーサポートにより、あなたのカスタムフォントですべてのラベルのベーススタイルがされました。

iOSとAndroidプロジェクトにカスタムフォントをめるには、GeraldによってかれたiosとAndroidでカスタムフォントをするXamarin.Formsポストのガイドにってください。

App.xamlファイルリソースセクションでスタイルをします。これにより、すべてのスタイルがグローバルにされます。

のGeraldから、StyleIdプロパティをするがありますが、これはバインドなプロパティではないため、Style Setterでこのプロパティをするには、Attachableプロパティをするがあります。

```
public static class FontHelper
{
    public static readonly BindableProperty StyleIdProperty =
        BindableProperty.CreateAttached(
            propertyName: nameof(Label.StyleId),
            returnType: typeof(String),
            declaringType: typeof(FontHelper),
            defaultValue: default(String),
            propertyChanged: OnItemTappedChanged);

    public static String GetStyleId(BindableObject bindable) =>
(String)bindable.GetValue(StyleIdProperty);

    public static void SetStyleId(BindableObject bindable, String value) =>
bindable.SetValue(StyleIdProperty, value);
```

```
    public static void OnItemTappedChanged(BindableObject bindable, object oldValue, object
newValue)
    {
        var control = bindable as Element;
        if (control != null)
        {
            control.StyleId = GetStyleId(control);
        }
    }
}
```

に、App.xamlリソースにスタイルをします。

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:h="clr-namespace:My.Helpers"
             x:Class="My.App">

  <Application.Resources>

    <ResourceDictionary>
        <Style x:Key="LabelStyle" TargetType="Label">
            <Setter Property="FontFamily" Value="Metric Bold" />
            <Setter Property="h:FontHelper.StyleId" Value="Metric-Bold" />
        </Style>
    </ResourceDictionary>

  </Application.Resources>

</Application>
```

のによると、AndroidプラットフォームのLabelRendererからするLabelのカスタムレンダラーをするがあります。

```
internal class LabelExRenderer : LabelRenderer
{
    protected override void OnElementChanged(ElementChangedEventArgs<Label> e)
    {
        base.OnElementChanged(e);
        if (!String.IsNullOrEmpty(e.NewElement?.StyleId))
        {
            var font = Typeface.CreateFromAsset(Forms.Context.ApplicationContext.Assets,
e.NewElement.StyleId + ".ttf");
            Control.Typeface = font;
        }
    }
}
```

iOSプラットフォームの、カスタムレンダラはありません。

これで、あなたのページのマークアップでスタイルをることができます

のラベルについて

```
<Label Text="Some text" Style={StaticResource LabelStyle} />
```

または、スタイルをLabesStyleについてして、ページのすべてのラベルにスタイルをします

```
<!-- language: xaml -->

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="My.MainPage">

  <ContentPage.Resources>

    <ResourceDictionary>
        <Style TargetType="Label" BasedOn={StaticResource LabelStyle}>
        </Style>
    </ResourceDictionary>

  </ContentPage.Resources>

  <Label Text="Some text" />

</ContentPage>
```

オンラインでスタイルのカスタムフォントをむ https://riptutorial.com/ja/xamarin-forms/topic/4854/スタイルのカスタムフォント

29: データバインディング

えられる

System.ArrayTypeMismatchExceptionとのないとしてにアクセスしようとしました。

このは、XAMLのコンパイルがなに、バインドなプロパティにコレクションをバインドしようとしたときにするがあります。なは、 `Picker.Items`にバインドしようとしてい`Picker.Items`。。

System.ArgumentException 'Xamarin.Forms.Binding'のオブジェクトを 'System.String'にすることはできません。

このは、XAMLのコンパイルがなに、バインドなプロパティにコレクションをバインドしようとしたときにするがあります。なは、 `Picker.Items`にバインドしようとしてい`Picker.Items`。。

`Picker.Items`プロパティはバインドできません。

このコードはエラーをきこします

```
<!-- BAD CODE: will cause an error -->
<Picker Items="{Binding MyViewModelItems}" SelectedIndex="0" />
```

はのいずれかです。

> System.ArrayTypeMismatchExceptionとのないとしてにアクセスしようとしました。

または

> System.ArgumentException 'Xamarin.Forms.Binding'のオブジェクトを 'System.String' にすることはできません。

には、 `Items`プロパティはバインドできません。ソリューションは、のような、のカスタムコントロールをしたり、サードパーティのコントロールをすることをむ`BindablePicker`から FreshEssentials 。プロジェクトにFreshEssentials NuGetパッケージをインストールすると、バインドな`ItemsSource`プロパティをつパッケージの`BindablePicker`コントロールをできます。

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:fe="clr-namespace:FreshEssentials;assembly=FreshEssentials"
```

```
            xmlns:my="clr-namespace:MyAssembly;assembly=MyAssembly"
            x:Class="MyNamespace.MyPage">
  <ContentPage.BindingContext>
    <my:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <fe:BindablePicker ItemsSource="{Binding MyViewModelItems}" SelectedIndex="0" />
  </ContentPage.Content>
</ContentPage>
```

Examples

ViewModelへのバインディング

EntryPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:vm="clr-namespace:MyAssembly.ViewModel;assembly=MyAssembly"
             x:Class="MyAssembly.EntryPage">
  <ContentPage.BindingContext>
    <vm:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <StackLayout VerticalOptions="FillAndExpand"
                 HorizontalOptions="FillAndExpand"
                 Orientation="Vertical"
                 Spacing="15">
      <Label Text="Name:" />
      <Entry Text="{Binding Name}" />
      <Label Text="Phone:" />
      <Entry Text="{Binding Phone}" />
      <Button Text="Save" Command="{Binding SaveCommand}" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

MyViewModel.cs

```
using System;
using System.ComponentModel;

namespace MyAssembly.ViewModel
{
    public class MyViewModel : INotifyPropertyChanged
    {
        private string _name = String.Empty;
        private string _phone = String.Empty;

        public string Name
        {
            get { return _name; }
            set
            {
                if (_name != value)
```

```
                {
                    _name = value;
                    OnPropertyChanged(nameof(Name));
                }
            }
        }

        public string Phone
        {
            get { return _phone; }
            set
            {
                if (_phone != value)
                {
                    _phone = value;
                    OnPropertyChanged(nameof(Phone));
                }
            }
        }

        public ICommand SaveCommand { get; private set; }

        public MyViewModel()
        {
            SaveCommand = new Command(SaveCommandExecute);
        }

        private void SaveCommandExecute()
        {

        }

        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

オンラインでデータバインディングをむ https://riptutorial.com/ja/xamarin-forms/topic/3915/データバインディング

30: トリガーとビヘイビア

Examples

Xamarinフォームのトリガの

`Trigger`は、あなたのアプリケーションにいくつかのUXをするなです。これをうなの1つは、する`Entry`にテキストがされているかどうかについて`Label`の`TextColor`をする`Trigger`をすることです。

このために`Trigger`をすると、 `Label.TextColor`をテキストがされていないからユーザーがテキストをするとすぐににできます。

コンバータコンバータには、クラスのしいインスタンスがされるたびにされないように、バインディングでされる`Instance`がえられます

```
/// <summary>
/// Used in a XAML trigger to return <c>true</c> or <c>false</c> based on the length of
<c>value</c>.
/// </summary>
public class LengthTriggerConverter : Xamarin.Forms.IValueConverter {

    /// <summary>
    /// Used so that a new instance is not created every time this converter is used in the
XAML code.
    /// </summary>
    public static LengthTriggerConverter Instance = new LengthTriggerConverter();

    /// <summary>
    /// If a `ConverterParameter` is passed in, a check to see if <c>value</c> is greater than
<c>parameter</c> is made. Otherwise, a check to see if <c>value</c> is over 0 is made.
    /// </summary>
    /// <param name="value">The length of the text from an Entry/Label/etc.</param>
    /// <param name="targetType">The Type of object/control that the text/value is coming
from.</param>
    /// <param name="parameter">Optional, specify what length to test against (example: for 3
Letter Name, we would choose 2, since the 3 Letter Name Entry needs to be over 2 characters),
if not specified, defaults to 0.</param>
    /// <param name="culture">The current culture set in the device.</param>
    /// <returns><c>object</c>, which is a <c>bool</c> (<c>true</c> if <c>value</c> is greater
than 0 (or is greater than the parameter), <c>false</c> if not).</returns>
    public object Convert(object value, System.Type targetType, object parameter, CultureInfo
culture)     { return DoWork(value, parameter); }
    public object ConvertBack(object value, System.Type targetType, object parameter,
CultureInfo culture) { return DoWork(value, parameter); }

    private static object DoWork(object value, object parameter) {
        int parameterInt = 0;

        if(parameter != null) { //If param was specified, convert and use it, otherwise, 0 is
used

            string parameterString = (string)parameter;

            if(!string.IsNullOrEmpty(parameterString)) { int.TryParse(parameterString, out
```

```
parameterInt); }
        }

        return (int)value > parameterInt;
    }
}
```

XAMLXAMLコードは、 `Entry x:Name`をして、 `Entry.Text`プロパティでするさが3をえています。

```
<StackLayout>
    <Label Text="3 Letter Name">
        <Label.Triggers>
            <DataTrigger TargetType="Label"
                         Binding="{Binding Source={x:Reference NameEntry},
                                           Path=Text.Length,
                                           Converter={x:Static
helpers:LengthTriggerConverter.Instance},
                                           ConverterParameter=2}"
                          Value="False">
                <Setter Property="TextColor"
                        Value="Gray"/>
            </DataTrigger>
        </Label.Triggers>
    </Label>
    <Entry x:Name="NameEntry"
           Text="{Binding MealAmount}"
           HorizontalOptions="StartAndExpand"/>
</StackLayout>
```

マルチトリガ

MultiTriggerはにはありませんが、にながいくつかあります。 MultiTriggerは、TriggerまたはDataTriggerとにしますが、のがあります。 Settersがするには、すべてのがたされていなければなりません。ここにながあります

```
<!-- Text field needs to be initialized in order for the trigger to work at start -->
<Entry x:Name="email" Placeholder="Email" Text="" />
<Entry x:Name="phone" Placeholder="Phone" Text="" />
<Button Text="Submit">
    <Button.Triggers>
        <MultiTrigger TargetType="Button">
            <MultiTrigger.Conditions>
                <BindingCondition Binding="{Binding Source={x:Reference email},
Path=Text.Length}" Value="0" />
                <BindingCondition Binding="{Binding Source={x:Reference phone},
Path=Text.Length}" Value="0" />
            </MultiTrigger.Conditions>
            <Setter Property="IsEnabled" Value="False" />
        </MultiTrigger>
    </Button.Triggers>
</Button>
```

このでは、とメールの2つのなるエントリがあり、そのうちの1つはするがあります。
MultiTriggerは、のフィールドがのにボタンをにします。

オンラインでトリガーとビヘイビアをむ https://riptutorial.com/ja/xamarin-forms/topic/6012/トリガーとビヘイビア

31: なぜXamarinフォームとXamarinフォームをするのか

Xamarin Formsのをして、さらにしくべることができます。

https://www.xamarin.com/forms

Examples

なぜXamarinフォームとXamarinフォームをするのか

Xamarinはますますがまっています.Xamarin.FormsとXamarin.PlatformXamarin.iOSとXamarin.Androidをするタイミングをめるのはしいです。

まず、Xamarin.Formsをできるアプリケーションのをっておくがあります。

1. プロトタイプ - さまざまなデバイスでのアプリケーションのをします。
2. プラットフォームのAPIなどをとしないアプリケーションですが、Xamarinはできるだけくのクロスプラットフォームをするためにしくいています。
3. コードがなアプリケーション - UIよりです。
4. されるデータがなよりもなアプリケーション

にもくのがあります。

1. がアプリケーションをしますかチームがなモバイルでされている、Xamarin.Formsをにできます。しかし、プラットフォームごとに1つのネイティブがあれば、フォームはよりきなになるがあります。
2. また、Xamarin.Formsではまだいくつかのがすることがあります.Xamarin.Formsプラットフォームはされています。
3. フォームのをするとコストをするために、ながになながあります。
4. なをたないエンタープライズアプリケーションをするは、Xamarin.Formsをするがいでしょう。モバイルエリアではなく、にイベントではなくモードコードをすることができます。コードのは、くのプラットフォームでできます。

のに**Xamarin.Forms**をしないでください。

1. カスタムをし、プラットフォームのAPIにアクセスするがあります
2. モバイルアプリケーションのカスタムUIをするがあります

3. のがXamarin.Formsのができていないモバイルデバイスののののように
4. あなたのチームは、プラットフォームのモバイルJavaおよび/またはSwift / Objective Cのモバイル

オンラインでなぜXamarinフォームとXamarinフォームをするのかをむ
https://riptutorial.com/ja/xamarin-forms/topic/6869/なぜxamarinフォームとxamarinフォームをするのか

32: プッシュ

はプラットフォームのやイベントにきくするため、Xamarin Formsでプッシュをするされたはありません。そのため、プラットフォームのコードがににになります。

ただし、 `DependencyService`をすることにより、できるだけくのコードをすることができます。また、のGitHubにあるrdelrosarioのプラグインがあります 。

コードとスクリーンショットは、プロセスをよりしくするGerald Versluisのブログシリーズからりげられています。

Examples

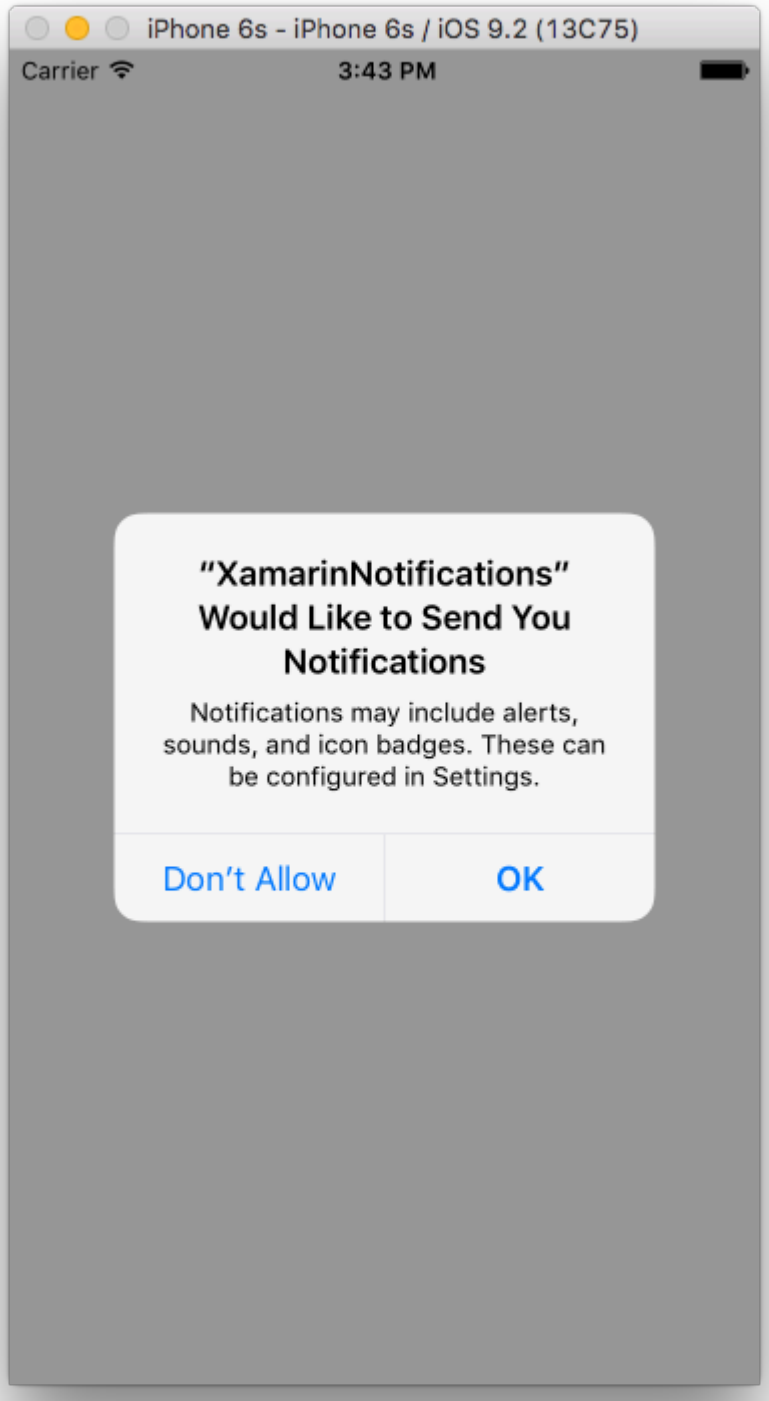
iOSとAzureのプッシュ

プッシュのをするには、のコードをするがあります。

```
// registers for push
var settings = UIUserNotificationSettings.GetSettingsForTypes(
    UIUserNotificationType.Alert
    | UIUserNotificationType.Badge
    | UIUserNotificationType.Sound,
    new NSSet());

UIApplication.SharedApplication.RegisterUserNotificationSettings(settings);
UIApplication.SharedApplication.RegisterForRemoteNotifications();
```

このコードは、 `AppDelegate.cs`ファイルの`FinishedLaunching`でアプリケーションがするときにすることもできます。または、ユーザーがプッシュをにすることをしたときはいつでもできます。

このコードをすると、アプリがをできることをけれるかどうかをユーザーにするがされます。したがって、ユーザーがそれをするシナリオもしてください

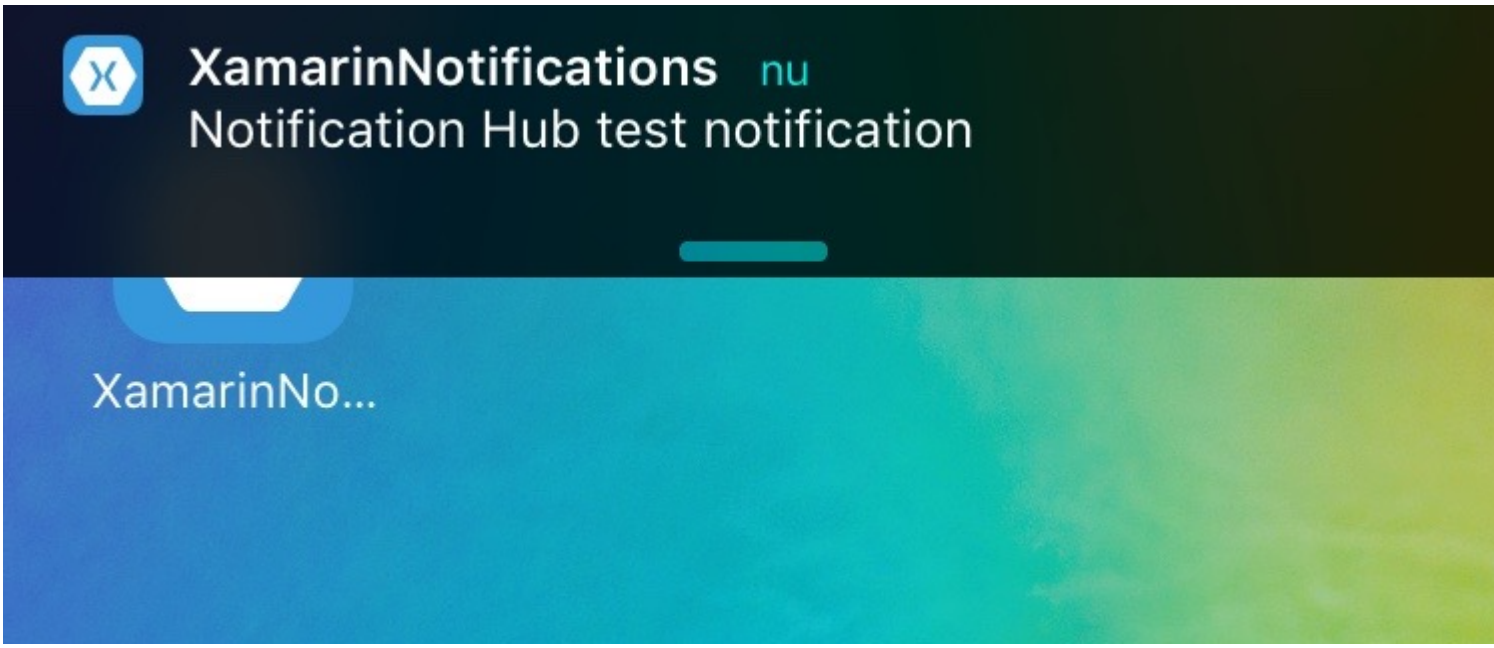


これらは、iOSでプッシュをするためのがなイベントです。 `AppDelegate.cs`ファイルでそれらをつけることができます。

```
// We've successfully registered with the Apple notification service, or in our case Azure
public override void RegisteredForRemoteNotifications(UIApplication application, NSData
deviceToken)
{
   // Modify device token for compatibility Azure
    var token = deviceToken.Description;
```

```
    token = token.Trim('<', '>').Replace(" ", "");

    // You need the Settings plugin for this!
    Settings.DeviceToken = token;

    var hub = new SBNotificationHub("Endpoint=sb://xamarinnotifications-
ns.servicebus.windows.net/;SharedAccessKeyName=DefaultListenSharedAccessSignature;SharedAccessKey=<your
own key>", "xamarinnotifications");

    NSSet tags = null; // create tags if you want, not covered for now
    hub.RegisterNativeAsync(deviceToken, tags, (errorCallback) =>
    {
        if (errorCallback != null)
        {
            var alert = new UIAlertView("ERROR!", errorCallback.ToString(), null, "OK", null);
            alert.Show();
        }
    });
}

// We've received a notification, yay!
public override void ReceivedRemoteNotification(UIApplication application, NSDictionary
userInfo)
{
    NSObject inAppMessage;

    var success = userInfo.TryGetValue(new NSString("inAppMessage"), out inAppMessage);

    if (success)
    {
        var alert = new UIAlertView("Notification!", inAppMessage.ToString(), null, "OK",
null);
        alert.Show();
    }
}

// Something went wrong while registering!
public override void FailedToRegisterForRemoteNotifications(UIApplication application, NSError
error)
{
   var alert = new UIAlertView("Computer says no", "Notification registration failed! Try
again!", null, "OK", null);

   alert.Show();
}
```

がされると、これはそのようにえます。



AzureでAndroidプッシュ

Androidでのはもうしであり、の`Service`をするがあります。

まず、がプッシュをできるかどうかをし、プッシュがあればGoogleにします。これは`MainActivity.cs`ファイルのこのコードでうことができます。

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

    global::Xamarin.Forms.Forms.Init(this, bundle);

    // Check to ensure everything's setup right for push
    GcmClient.CheckDevice(this);
    GcmClient.CheckManifest(this);
    GcmClient.Register(this, NotificationsBroadcastReceiver.SenderIDs);

    LoadApplication(new App());
}
```

SenderIDはコードのにされ、プッシュメッセージをできるようにGoogleのダッシュボードからしたプロジェクトです。

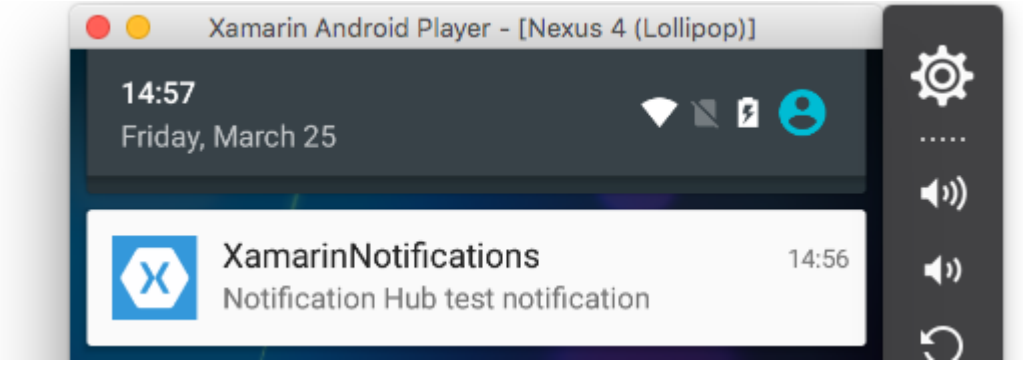
```
using Android.App;
using Android.Content;
using Gcm.Client;
using Java.Lang;
using System;
using WindowsAzure.Messaging;
using XamarinNotifications.Helpers;

// These attributes are to register the right permissions for our app concerning push messages
[assembly: Permission(Name = "com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
[assembly: UsesPermission(Name =
"com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
```

```
[assembly: UsesPermission(Name = "com.google.android.c2dm.permission.RECEIVE")]

//GET_ACCOUNTS is only needed for android versions 4.0.3 and below
[assembly: UsesPermission(Name = "android.permission.GET_ACCOUNTS")]
[assembly: UsesPermission(Name = "android.permission.INTERNET")]
[assembly: UsesPermission(Name = "android.permission.WAKE_LOCK")]

namespace XamarinNotifications.Droid.PlatformSpecifics
{
    // These attributes belong to the BroadcastReceiver, they register for the right intents
    [BroadcastReceiver(Permission = Constants.PERMISSION_GCM_INTENTS)]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_MESSAGE },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_REGISTRATION_CALLBACK },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_LIBRARY_RETRY },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]

    // This is the bradcast reciever
    public class NotificationsBroadcastReceiver : GcmBroadcastReceiverBase<PushHandlerService>
    {
        // TODO add your project number here
        public static string[] SenderIDs = { "96688------" };
    }

    [Service] // Don't forget this one! This tells Xamarin that this class is a Android
Service
    public class PushHandlerService : GcmServiceBase
    {
        // TODO add your own access key
        private string _connectionString =
ConnectionString.CreateUsingSharedAccessKeyWithListenAccess(
            new Java.Net.URI("sb://xamarinnotifications-ns.servicebus.windows.net/"), "<your
key here>");

        // TODO add your own hub name
        private string _hubName = "xamarinnotifications";

        public static string RegistrationID { get; private set; }

        public PushHandlerService() : base(NotificationsBroadcastReceiver.SenderIDs)
        {
        }

        // This is the entry point for when a notification is received
        protected override void OnMessage(Context context, Intent intent)
        {
            var title = "XamarinNotifications";

            if (intent.Extras.ContainsKey("title"))
                title = intent.Extras.GetString("title");

            var messageText = intent.Extras.GetString("message");

            if (!string.IsNullOrEmpty(messageText))
                CreateNotification(title, messageText);
        }

        // The method we use to compose our notification
        private void CreateNotification(string title, string desc)
        {
```

```
        // First we make sure our app will start when the notification is pressed
        const int pendingIntentId = 0;
        const int notificationId = 0;

        var startupIntent = new Intent(this, typeof(MainActivity));
        var stackBuilder = TaskStackBuilder.Create(this);

        stackBuilder.AddParentStack(Class.FromType(typeof(MainActivity)));
        stackBuilder.AddNextIntent(startupIntent);

        var pendingIntent =
            stackBuilder.GetPendingIntent(pendingIntentId, PendingIntentFlags.OneShot);

        // Here we start building our actual notification, this has some more
        // interesting customization options!
        var builder = new Notification.Builder(this)
            .SetContentIntent(pendingIntent)
            .SetContentTitle(title)
            .SetContentText(desc)
            .SetSmallIcon(Resource.Drawable.icon);

        // Build the notification
        var notification = builder.Build();
        notification.Flags = NotificationFlags.AutoCancel;

        // Get the notification manager
        var notificationManager =
            GetSystemService(NotificationService) as NotificationManager;

        // Publish the notification to the notification manager
        notificationManager.Notify(notificationId, notification);
    }

    // Whenever an error occurs in regard to push registering, this fires
    protected override void OnError(Context context, string errorId)
    {
        Console.Out.WriteLine(errorId);
    }

    // This handles the successful registration of our device to Google
    // We need to register with Azure here ourselves
    protected override void OnRegistered(Context context, string registrationId)
    {
        var hub = new NotificationHub(_hubName, _connectionString, context);

        Settings.DeviceToken = registrationId;

        // TODO set some tags here if you want and supply them to the Register method
        var tags = new string[] { };

        hub.Register(registrationId, tags);
    }

    // This handles when our device unregisters at Google
    // We need to unregister with Azure
    protected override void OnUnRegistered(Context context, string registrationId)
    {
        var hub = new NotificationHub(_hubName, _connectionString, context);

        hub.UnregisterAll(registrationId);
    }
```

```
    }
}
```

Androidのサンプルはのようになります。

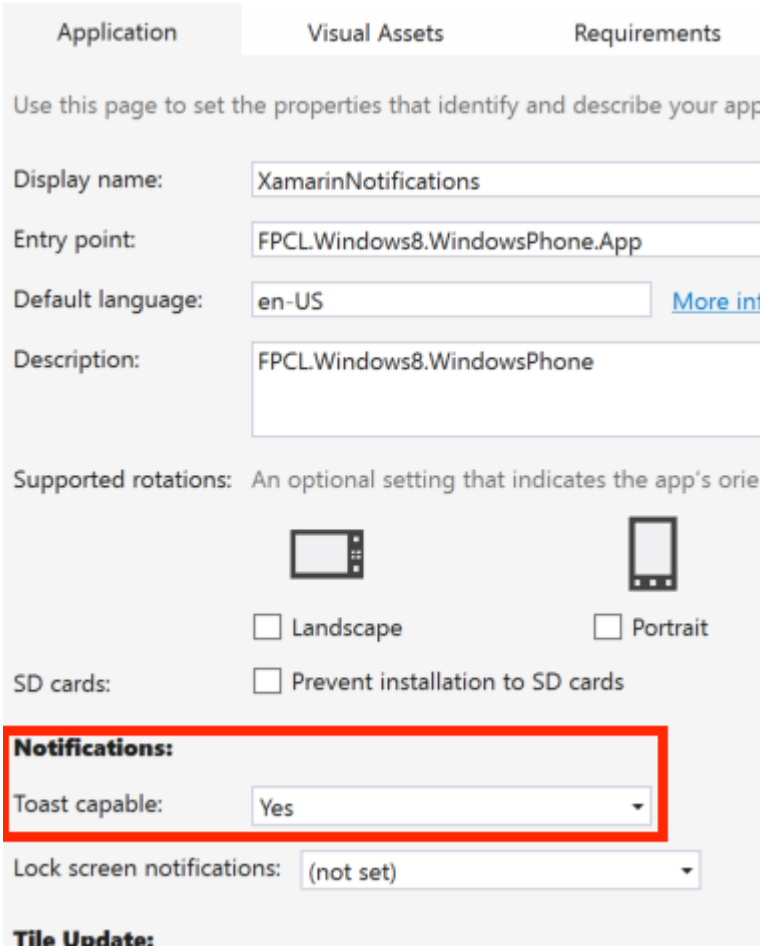


AzureをしたWindows Phoneのプッシュ

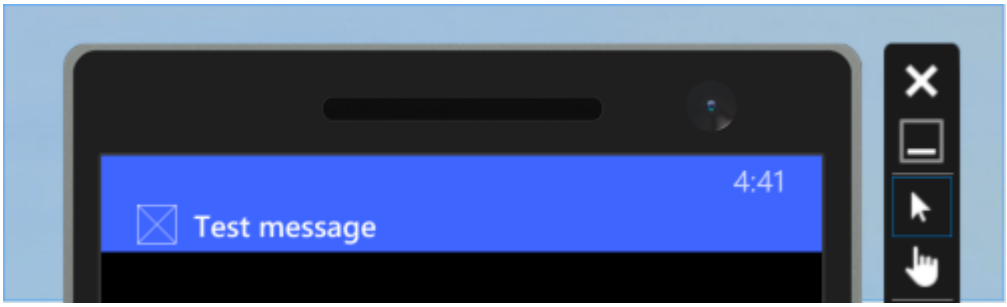
Windows Phoneでは、プッシュでをするには、コードのにあるコードをするがあります。これは `App.xaml.cs`ファイルにあります。

```
protected async override void OnLaunched(LaunchActivatedEventArgs e)
{
    var channel = await
PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();

    // TODO add connection string here
    var hub = new NotificationHub("XamarinNotifications", "<connection string with listen
access>");
    var result = await hub.RegisterNativeAsync(channel.Uri);

    // Displays the registration ID so you know it was successful
    if (result.RegistrationId != null)
    {
        Settings.DeviceToken = result.RegistrationId;
    }

    // The rest of the default code is here
}
```

また、 `Package.appxmanifest`ファイルのをにすることをれないでください。



プッシュのサンプルはのようになります。



オンラインでプッシュをむ https://riptutorial.com/ja/xamarin-forms/topic/5042/プッシュ

33: プッシュ

AWSサービスLingo

エンドポイント - エンドポイントは、メールアドレスなど、AWS SNSがでヒットできるもの

トピック - に、すべてのエンドポイントをむグループ

- をけるためにあなたの/クライアントにサインアップします

なプッシュLingo

APNS - Appleプッシュサービス。プッシュをできるのはAppleだけです。これがたちがなをアプリにするです。 AWS SNSには、AppleがAPNSにわってAPNSにをすることをSNSにするためのがされています。

GCM - Google Cloud MessagingはAPNSとによくています。プッシュをできるのはGoogleだけです。まず、GCMでアプリケーションをし、トークンをAWS SNSにきします。 SNSは、GCMをしデータをするなをすべてします。

Examples

iOSの

1. がです
2. Appleデベロッパーアカウントにアクセスし、プッシュをにしてプロビジョニングプロファイルをする
3. あなたのにするためのらかのがですAWS、Azure..etc ここで**AWS**をします

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();

   //after typical Xamarin.Forms Init Stuff

   //variable to set-up the style of notifications you want, iOS supports 3 types

   var pushSettings = UIUserNotificationSettings.GetSettingsForTypes(
              UIUserNotificationType.Alert |
              UIUserNotificationType.Badge |
              UIUserNotificationType.Sound,
              null );


        //both of these methods are in iOS, we have to override them and set them up
        //to allow push notifications
```

```
        app.RegisterUserNotificationSettings(pushSettings);  //pass the supported push
notifications settings to register app in settings page


}

public override async void RegisteredForRemoteNotifications(UIApplication application, NSData
token)
    {
        AmazonSimpleNotificationServiceClient snsClient = new
AmazonSimpleNotificationServiceClient("your AWS credentials here");

        // This contains the registered push notification token stored on the phone.
        var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
"");

        if (!string.IsNullOrEmpty(deviceToken))
        {
            //register with SNS to create an endpoint ARN, this means AWS can message your
phone
            var response = await snsClient.CreatePlatformEndpointAsync(
            new CreatePlatformEndpointRequest
            {
                Token = deviceToken,
                PlatformApplicationArn = "yourARNwouldgohere" /* insert your platform
application ARN here */
            });

            var endpoint = response.EndpointArn;

            //AWS lets you create topics, so use subscribe your app to a topic, so you can
easily send out one push notification to all of your users
            var subscribeResponse = await snsClient.SubscribeAsync(new SubscribeRequest
            {
                TopicArn = "YourTopicARN here",
                Endpoint = endpoint,
                Protocol = "application"

            });

        }

    }
```

オンラインでプッシュをむ https://riptutorial.com/ja/xamarin-forms/topic/5998/プッシュ

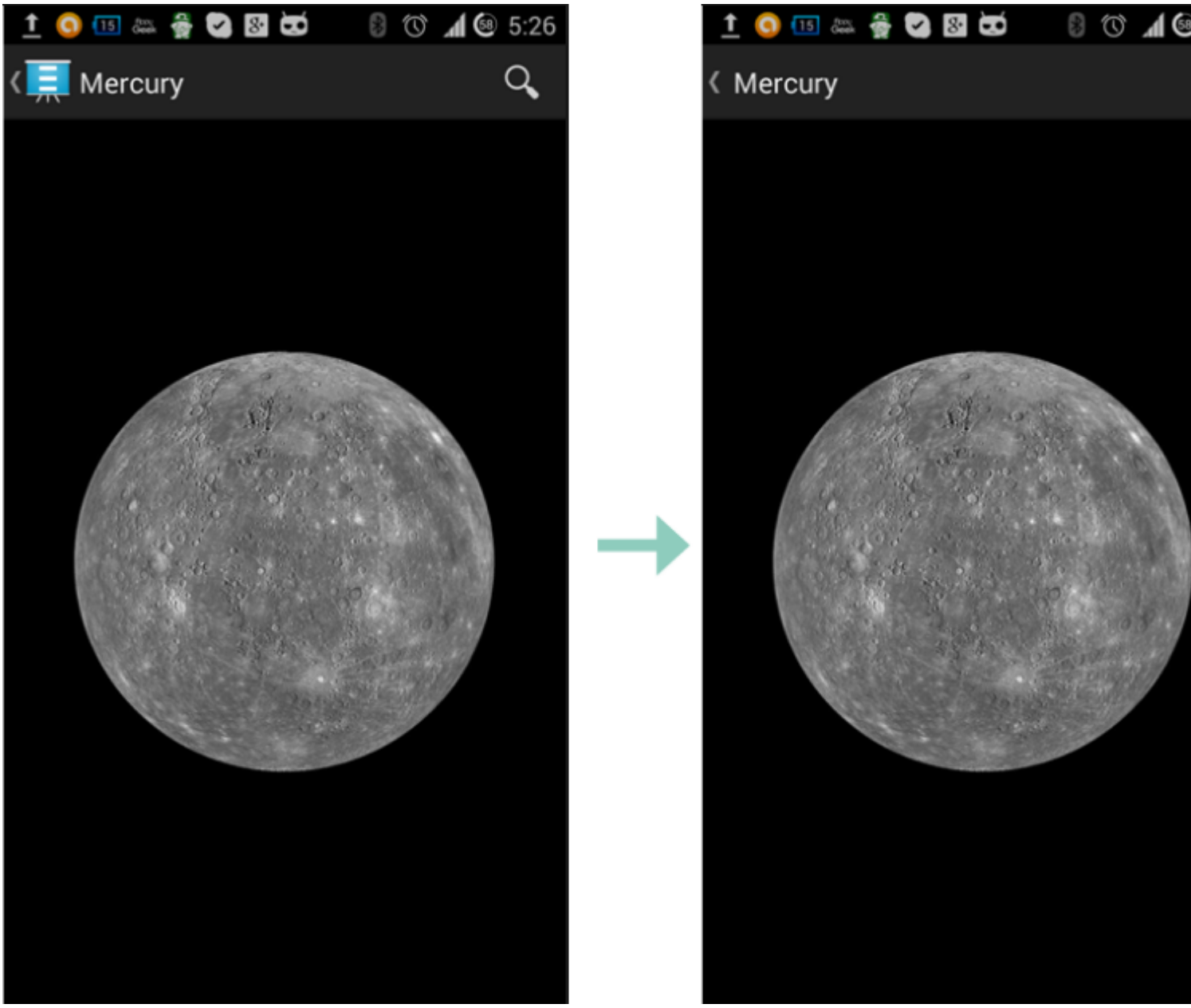
34: プラットフォームの

ターゲットプラットフォーム

```
if(Device.OS == TargetPlatform.Android)
{

}
else if (Device.OS == TargetPlatform.iOS)
{

}
else if (Device.OS == TargetPlatform.WinPhone)
{

}
else if (Device.OS == TargetPlatform.Windows)
{

}
else if (Device.OS == TargetPlatform.Other)
{

}
```

Examples

Anroidのナビゲーションヘッダーのアイコンをする



*empty.png*というさななをする

```
public class MyPage : ContentPage
{
    public Page()
    {
        if (Device.OS == TargetPlatform.Android)
            NavigationPage.SetTitleIcon(this, "empty.png");
    }
}
```

iOSでラベルのフォントサイズをさくする

```
Label label = new Label
{
    Text = "text"
};
if(Device.OS == TargetPlatform.iOS)
{
```

```
    label.FontSize = label.FontSize - 2;
}
```

オンラインでプラットフォームのをむ https://riptutorial.com/ja/xamarin-forms/topic/6636/プラットフォームの

35: プラットフォームの

Examples

イディオムの

たとえば、ビューがされているかどうか、またはやタブレットのどちらのレイアウトのきをするかなど、Cコードからイディオムのをうことができます。

```
if (Device.Idiom == TargetIdiom.Phone)
{
    this.panel.Orientation = StackOrientation.Vertical;
}
else
{
    this.panel.Orientation = StackOrientation.Horizontal;
}
```

これらのは、XAMLコードからすることもできます。

```
<StackLayout x:Name="panel">
  <StackLayout.Orientation>
    <OnIdiom x:TypeArguments="StackOrientation">
      <OnIdiom.Phone>Vertical</OnIdiom.Phone>
      <OnIdiom.Tablet>Horizontal</OnIdiom.Tablet>
    </OnIdiom>
  </StackLayout.Orientation>
</StackLayout>
```

プラットフォームの

Cコードからのプラットフォームのをうことができます。たとえば、すべてのプラットフォームのパディングをすることができます。

```
if (Device.OS == TargetPlatform.iOS)
{
    panel.Padding = new Thickness (10);
}
else
{
    panel.Padding = new Thickness (20);
}
```

ヘルパーメソッドは、されたCでもできます。

```
panel.Padding = new Thickness (Device.OnPlatform(10,20,0));
```

これらのは、XAMLコードからすることもできます。

```
<StackLayout x:Name="panel">
  <StackLayout.Padding>
    <OnPlatform x:TypeArguments="Thickness"
      iOS="10"
      Android="20" />
  </StackLayout.Padding>
</StackLayout>
```

スタイルの

XAMLをする、`Style`すると、のスタイルきビューを1かからできます。すべてのイディオムとプラットフォームのをスタイルにすることもできます。

```
<Style TargetType="StackLayout">
  <Setter Property="Padding">
    <Setter.Value>
      <OnPlatform x:TypeArguments="Thickness"
                  iOS="10"
                  Android="20"/>
    </Setter.Value>
  </Setter>
</Style>
```

カスタムビュー

それらのツールのおかげで、あなたのページにできるカスタムビューをすることができます。

`File > New > File... > Forms > Forms ContentView (Xaml)`とのレイアウトのためのビューを`TabletHome.xaml`と`PhoneHome.xaml`。

に、「`File > New > File... > Forms > Forms ContentPage`をし、のをむ`HomePage.cs`をし`HomePage.cs`。

```
using Xamarin.Forms;

public class HomePage : ContentPage
{
    public HomePage()
    {
        if (Device.Idiom == TargetIdiom.Phone)
        {
            Content = new PhoneHome();
        }
        else
        {
            Content = new TabletHome();
        }

    }
}
```

これで、`Phone`と`Tablet`イディオムのビューがなる`HomePage`がされました。

オンラインでプラットフォームのをむ https://riptutorial.com/ja/xamarin-forms/topic/5012/プラットフォームの

36: マップの

プロジェクトをのコンピュータでするは、SHA-1のフィンガープリントがなるビルドマシンではしないため、しいAPIキーをするがあります。

あなたは*Xamarin.Forms*のマップをするでしたプロジェクト、することができますここに

Examples

Xamarin.Formsでマップをする Xamarin Studio

Xamarin Formsをすると、プラットフォームでネイティブマップAPIをできます。なのは、Nugetから*Xamarin.Forms.Maps*パッケージをダウンロードし、それをプロジェクトPCLプロジェクトをむにインストールすることだけです。

マップの

まず、このコードをプラットフォームのプロジェクトにするがあります。これをうには、ののように、 `Xamarin.FormsMaps.Init` メソッドびしをするがあります。

---

iOSプロジェクト

*AppDelegate.cs*ファイル

```
[Register("AppDelegate")]
public partial class AppDelegate : Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
{
    public override bool FinishedLaunching(UIApplication app, NSDictionary options)
    {
        Xamarin.Forms.Forms.Init();
        Xamarin.FormsMaps.Init();

        LoadApplication(new App());

        return base.FinishedLaunching(app, options);
    }
}
```

Androidプロジェクト

*MainActivity.cs*ファイル

```
[Activity(Label = "MapExample.Droid", Icon = "@drawable/icon", Theme = "@style/MyTheme",
```

```
MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
public class MainActivity : Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        Xamarin.Forms.Forms.Init(this, bundle);
        Xamarin.FormsMaps.Init(this, bundle);

        LoadApplication(new App());
    }
}
```

プラットフォーム

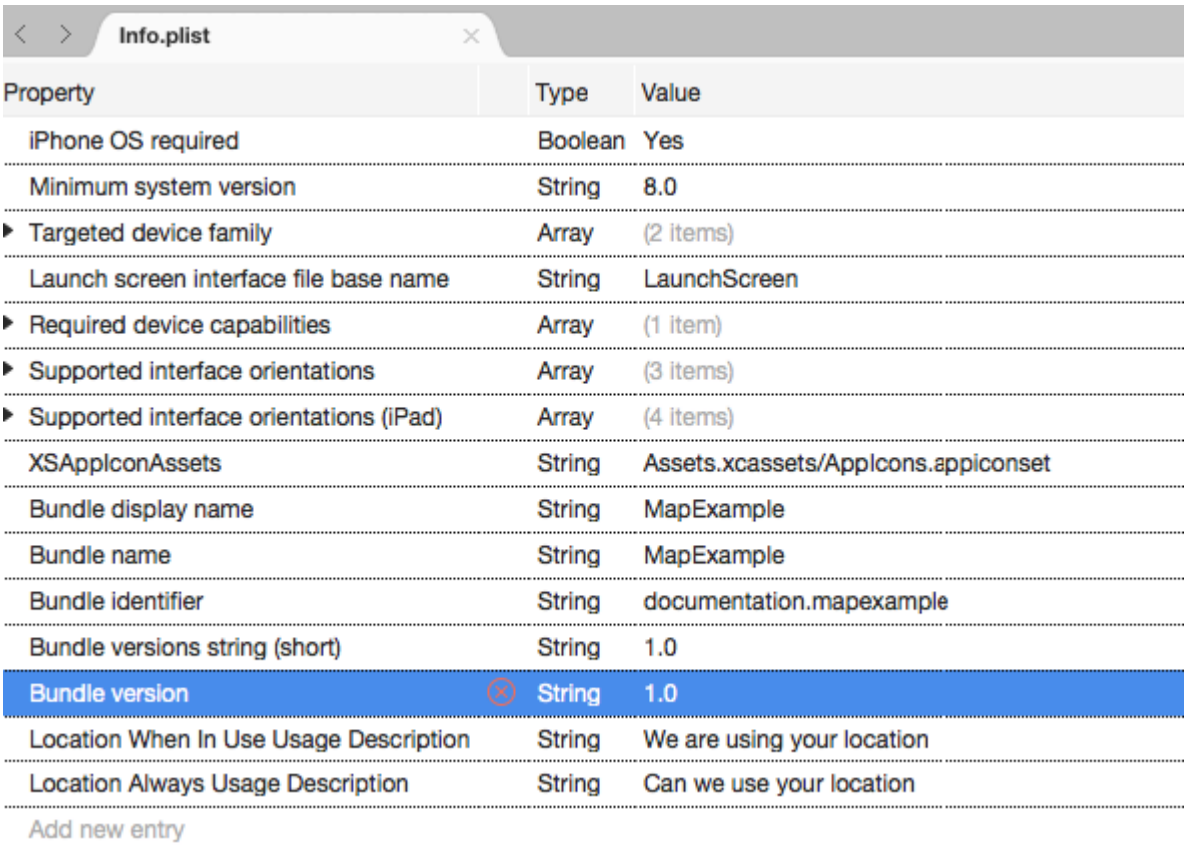
マップがされるに、のプラットフォームでのがです。

---

iOSプロジェクト

iOSプロジェクトでは*Info.plist*ファイルに2つのエントリをするだけです

- のある`NSLocationWhenInUseUsageDescription We are using your location`
- `NSLocationAlwaysUsageDescription`のある `Can we use your location`



Androidプロジェクト

Googleマップをするには、APIキーをしてプロジェクトにするがあります。このキーをするには、のにってください。

1. オプション `/System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands` ツールのをしますデフォルトは `/System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands`

2. オプションをいてキーツールのにします

   ```
   cd /System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands
   ```

3. のkeytoolコマンドをします。

   ```
   keytool -list -v -keystore "/Users/[USERNAME]/.local/share/Xamarin/Mono for
   Android/debug.keystore" -alias androiddebugkey -storepass android -keypass android
   ```

   [USERNAME]がのユーザーフォルダであることはらかです。あなたはにこれにたかをるべきです

   ```
   Alias name: androiddebugkey
   Creation date: Jun 30, 2016
   Entry type: PrivateKeyEntry
   Certificate chain length: 1
   Certificate[1]:
   Owner: CN=Android Debug, O=Android, C=US
   Issuer: CN=Android Debug, O=Android, C=US
   ```

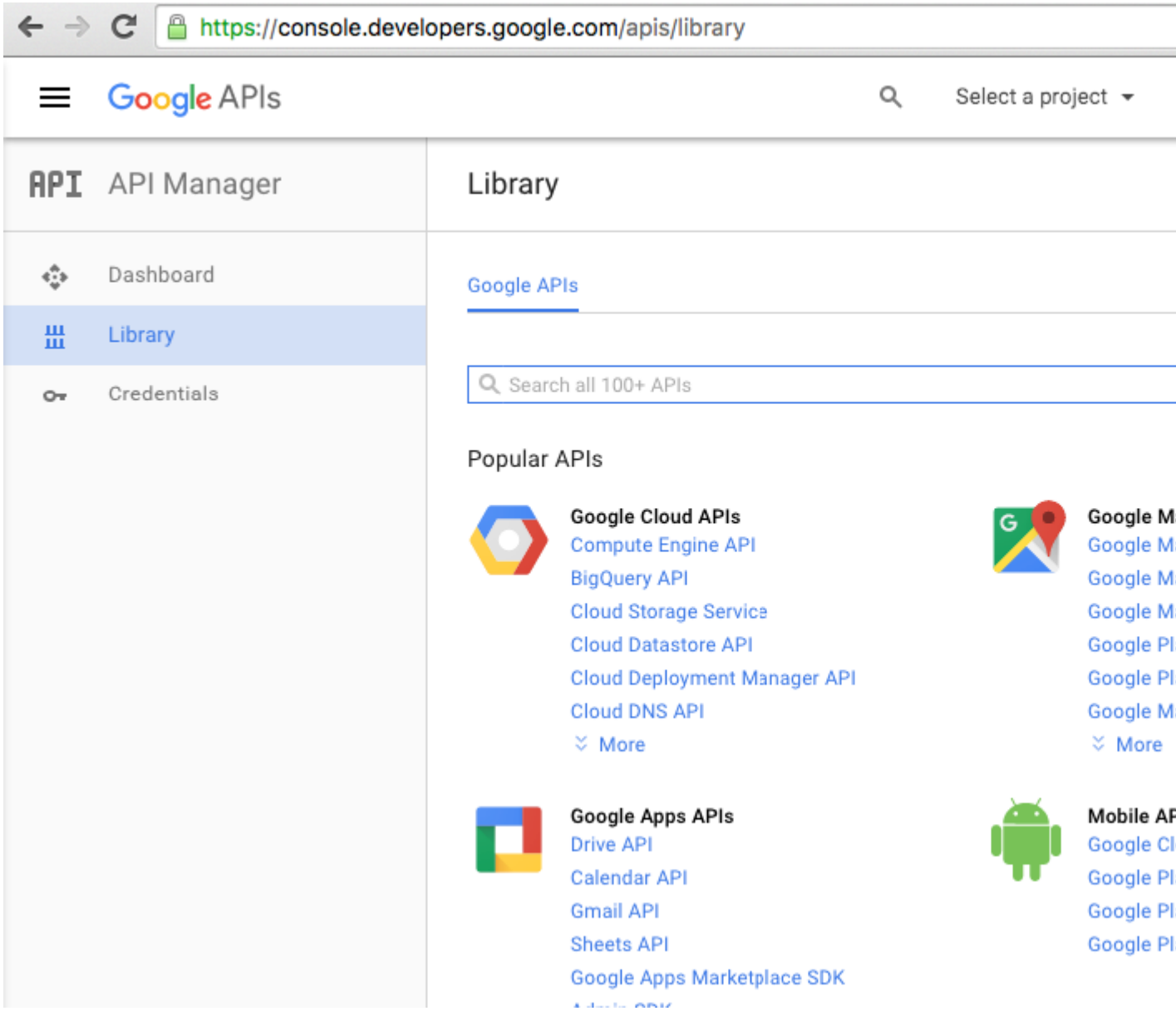
```
Serial number: 4b5ac934
Valid from: Thu Jun 30 10:22:00 EEST 2016 until: Sat Jun 23 10:22:00 EEST 2046
Certificate fingerprints:
         MD5:  4E:49:A7:14:99:D6:AB:9F:AA:C7:07:E2:6A:1A:1D:CA
         SHA1: 57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
         SHA256:
70:E1:F3:5B:95:69:36:4A:82:A9:62:F3:67:B6:73:A4:DD:92:95:51:44:E3:4C:3D:9E:ED:99:03:09:9F:90:3F

         Signature algorithm name: SHA256withRSA
         Version: 3
```

4. このでなのは、SHA1フィンガープリントだけです。たちの、それはこれとじです

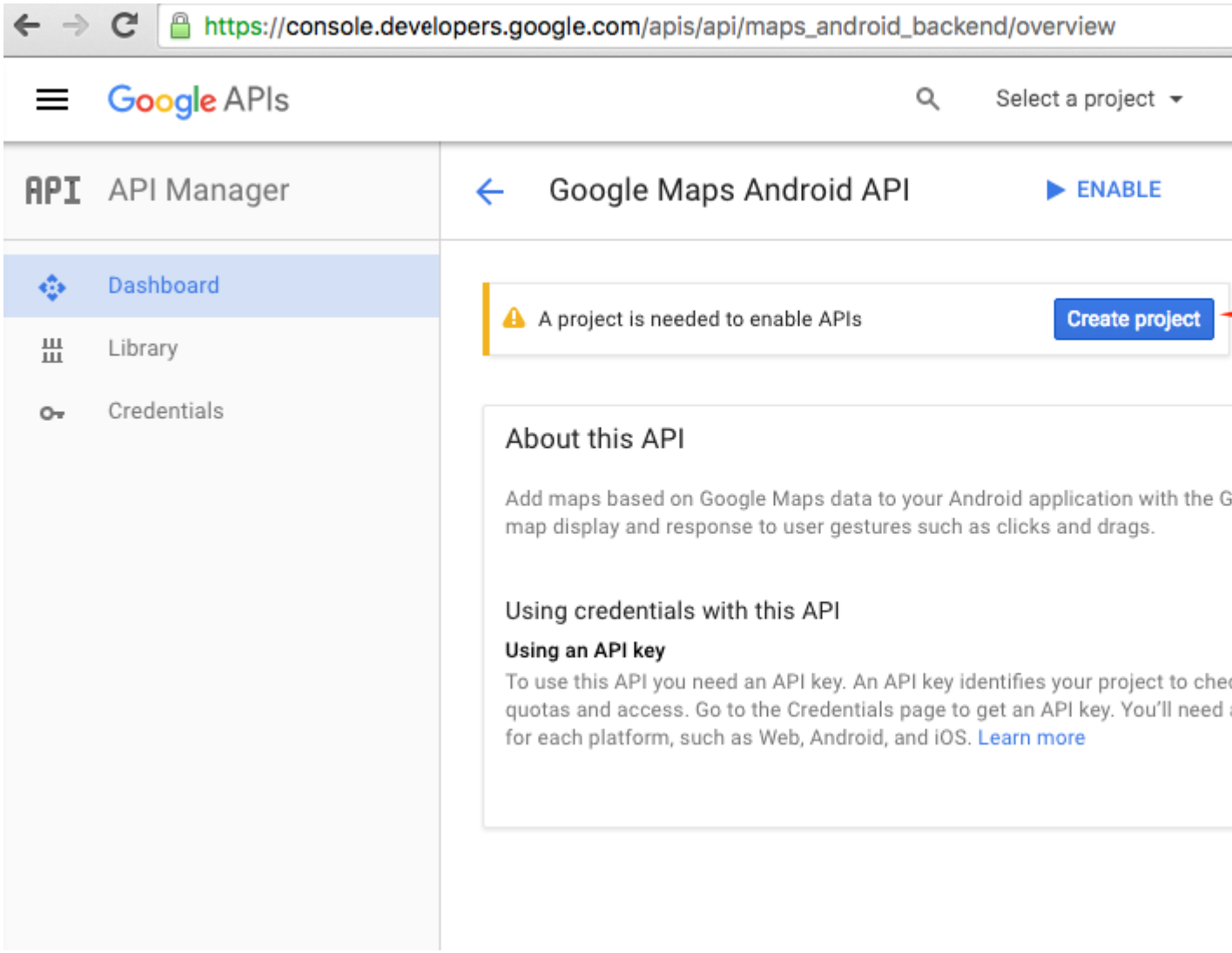
```
57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
```

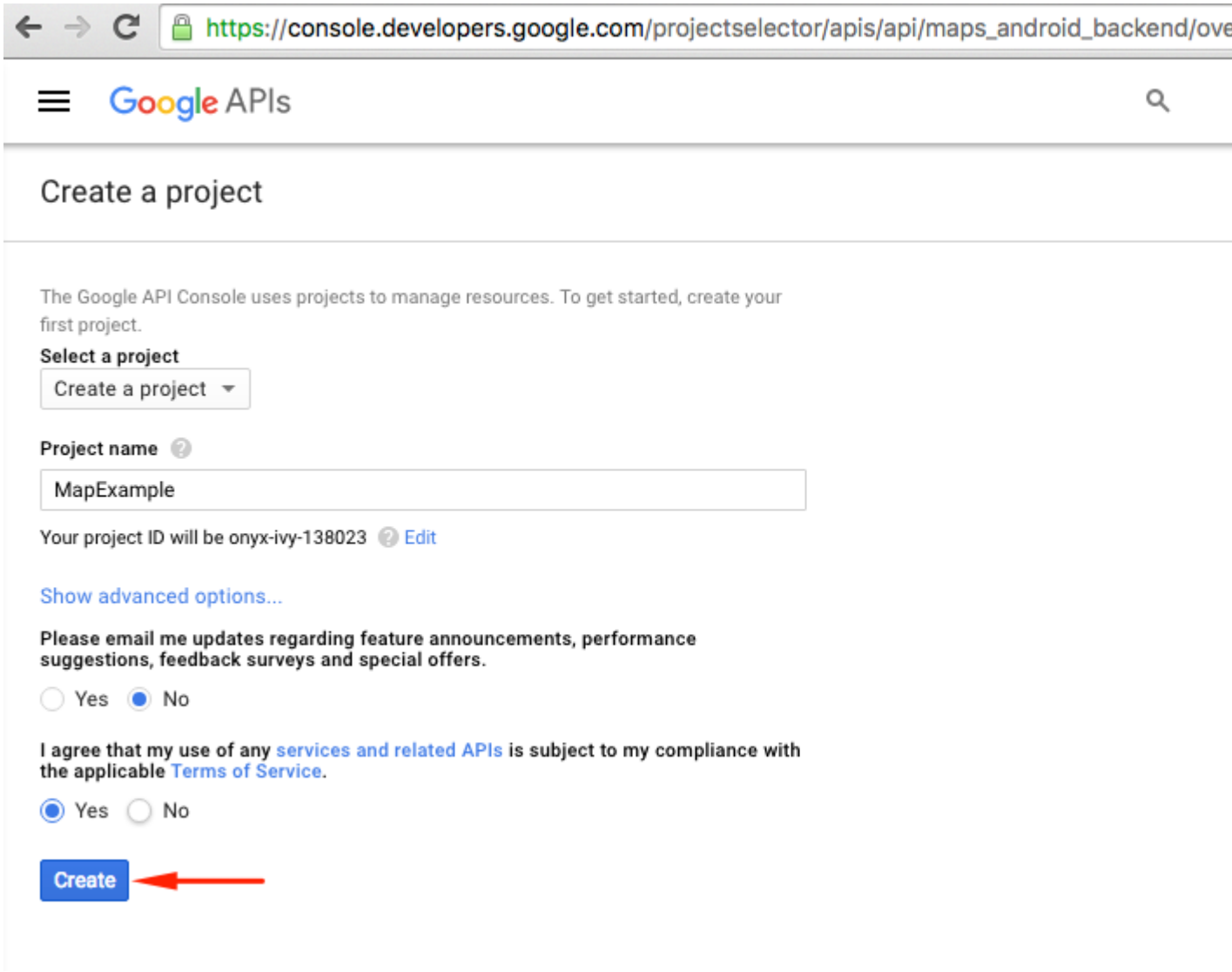
   このキーのどこかをコピーまたはしてください。たちはでそれをとします。

5. Google Developers Consoleにします .Google Maps Android APIをするがあるため、してください

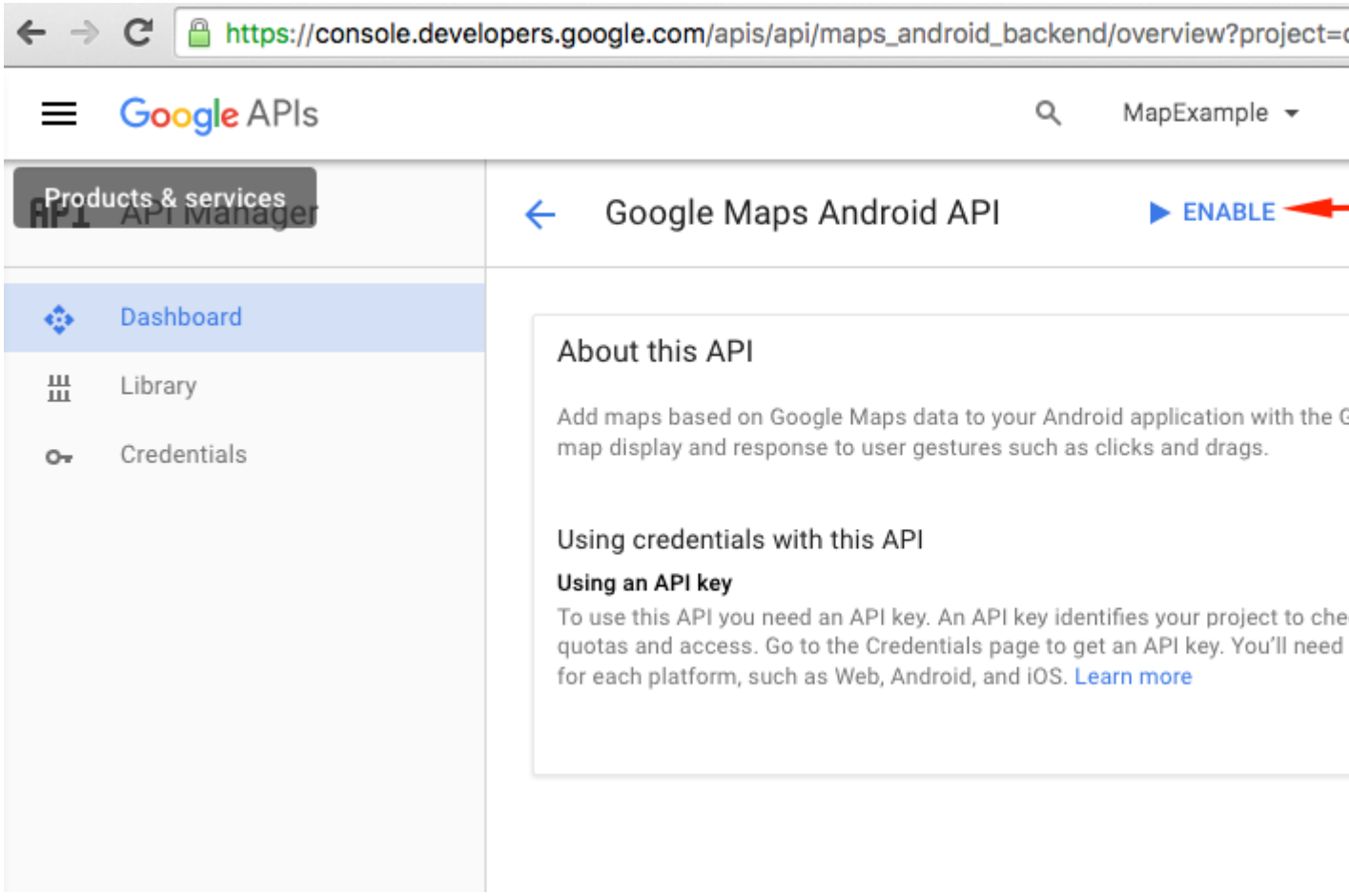


6. Googleでは、APIをにするためのプロジェクトをし、このヒントにってプロジェクトをするようします。

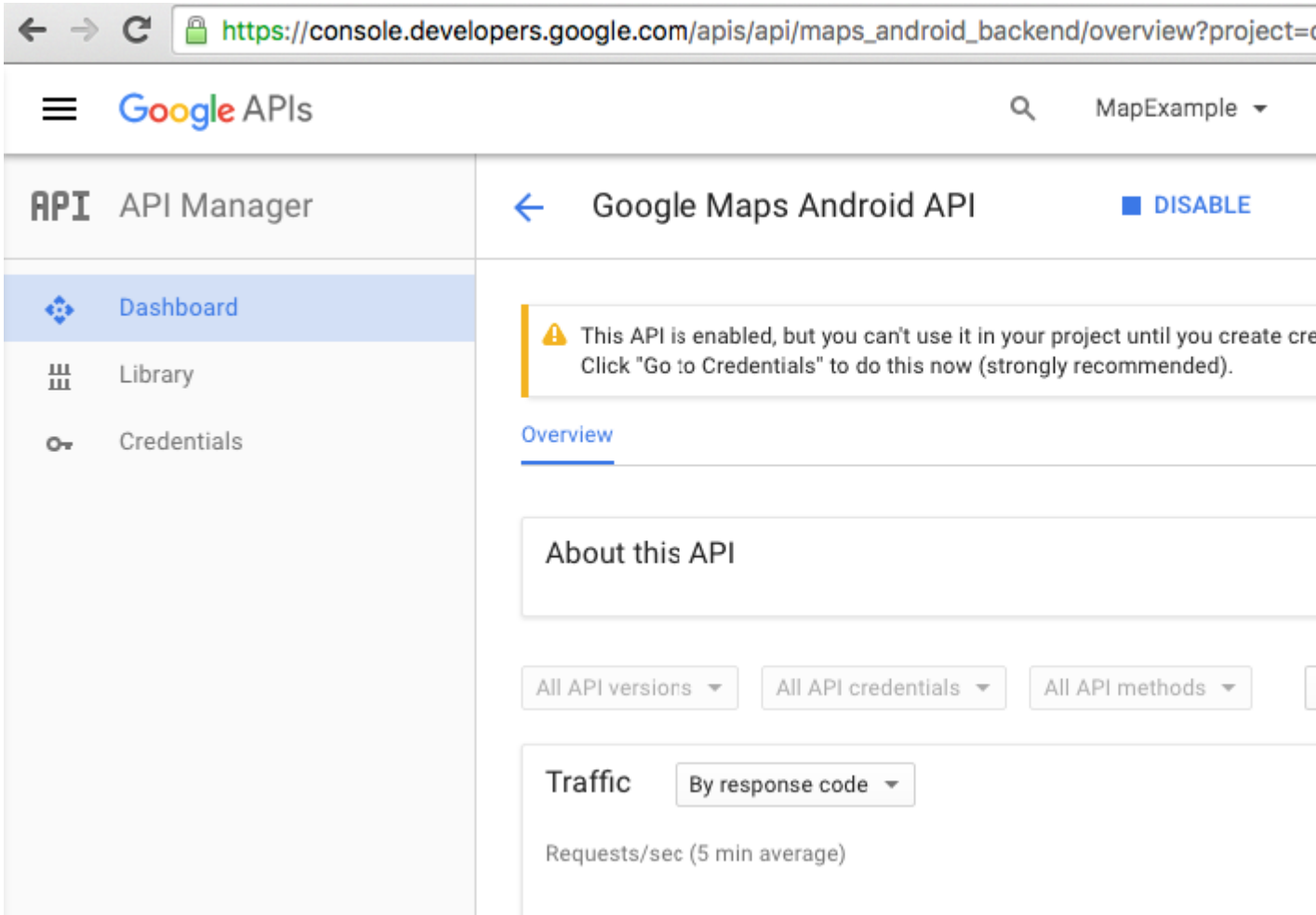




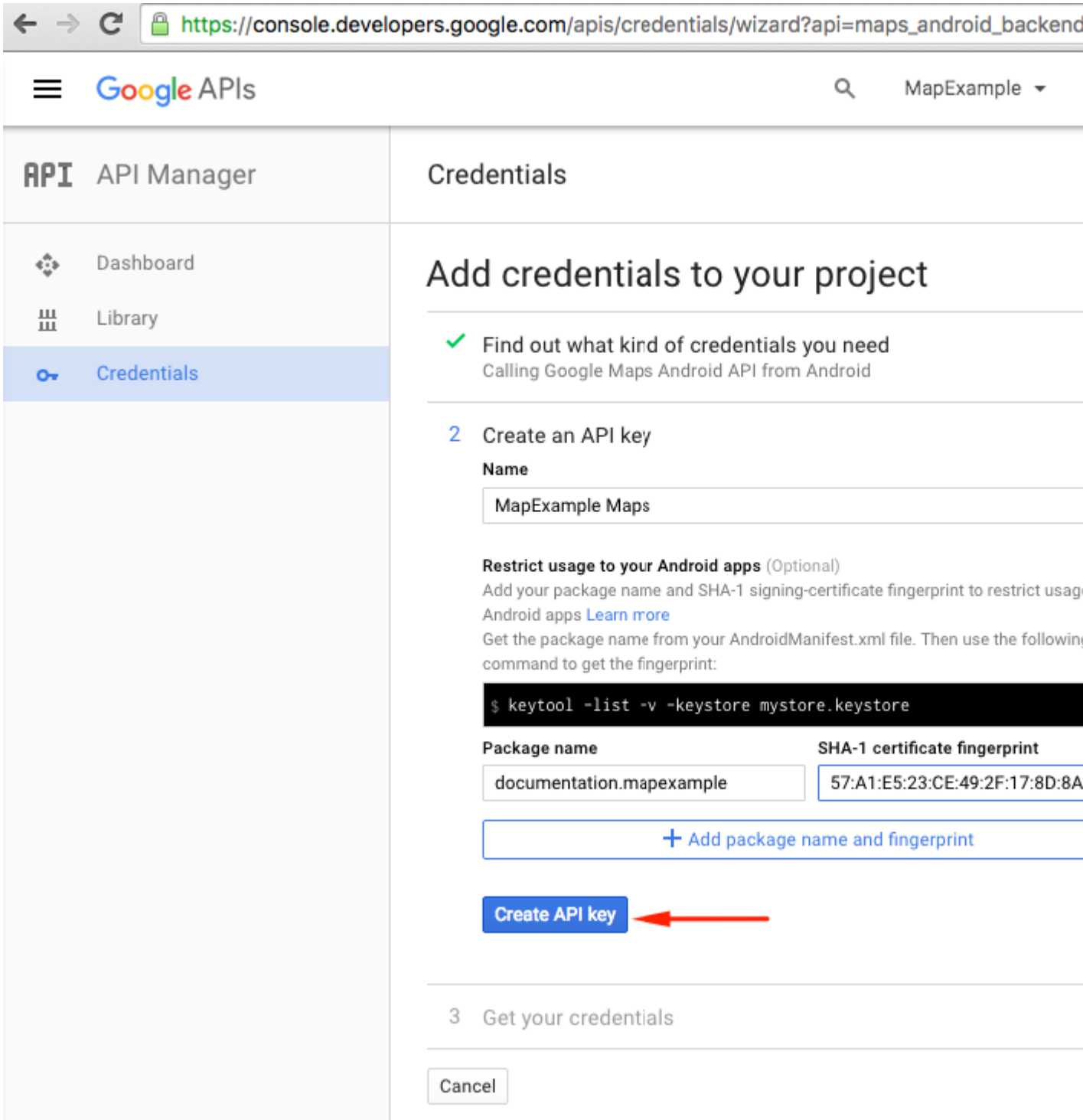
7. プロジェクトにGoogle Maps APIをにする



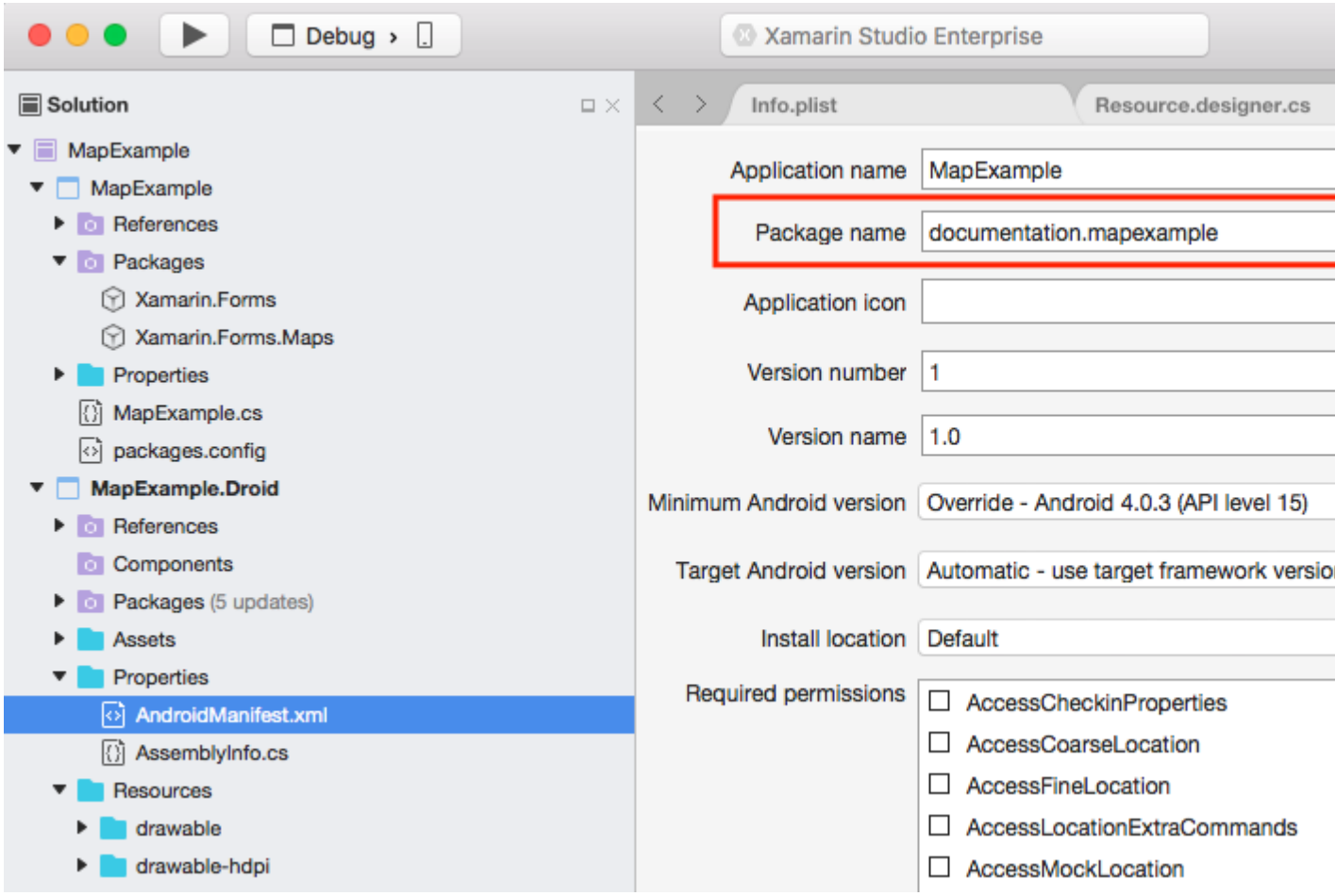
apiをにした、あなたのアプリケーションのをするがあります。このヒントにってください



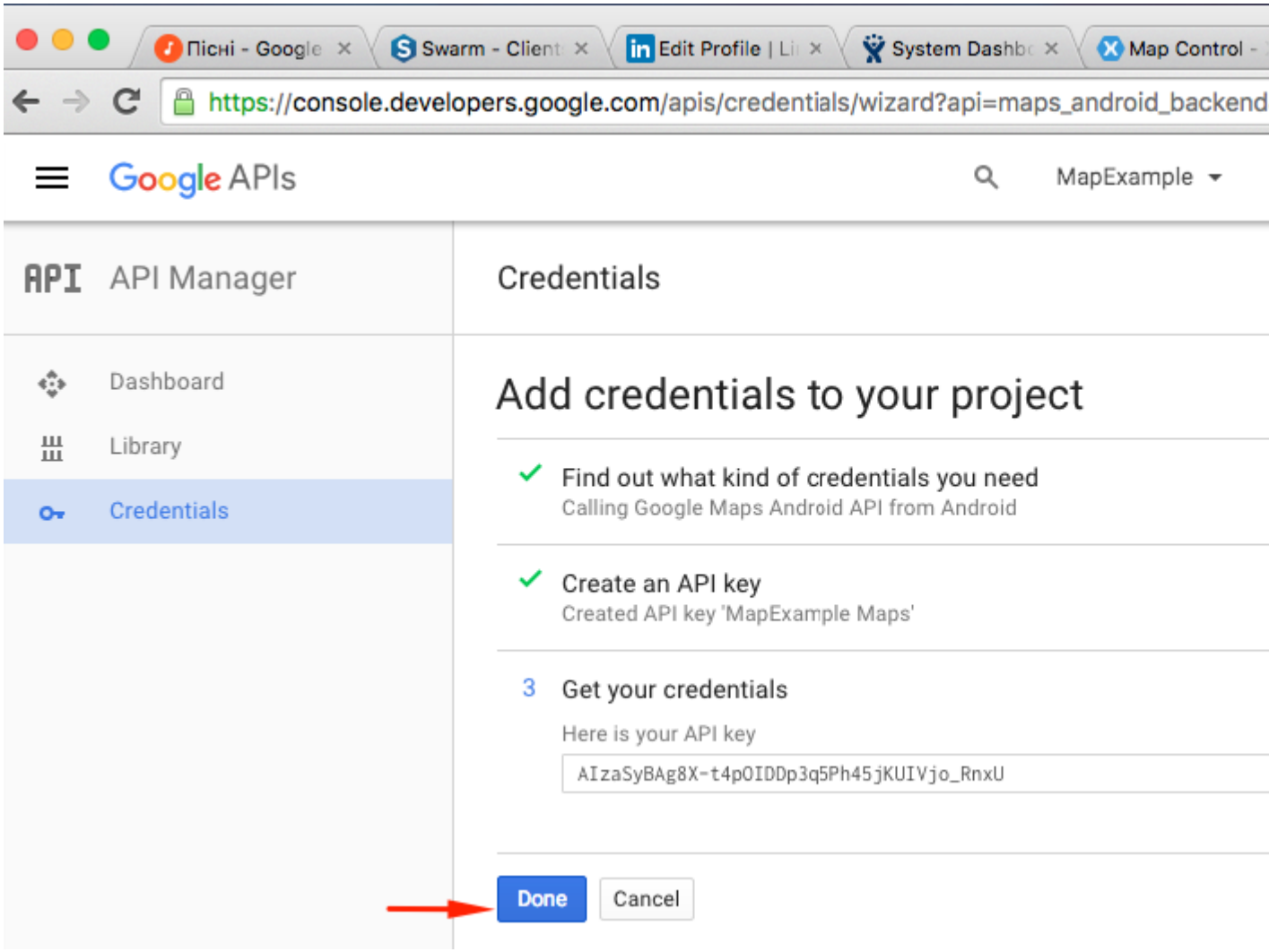
8. のページでAndroidプラットフォームをし、「どのがですか」をタップします。ボタンをし、APIキーのをし、「パッケージとを」をタップし、4でパッケージとSHA1フィンガープリントをし、にAPIキーをします。



Xamarin Studioであなたのパッケージをつけるには、あなたの.Droidソリューション - > AndroidManifest.xml



9. 、しいAPIキーをコピーしてくださいで「」ボタンをすことをれないでください、
`AndroidManifest.xml`ファイルにりけてください



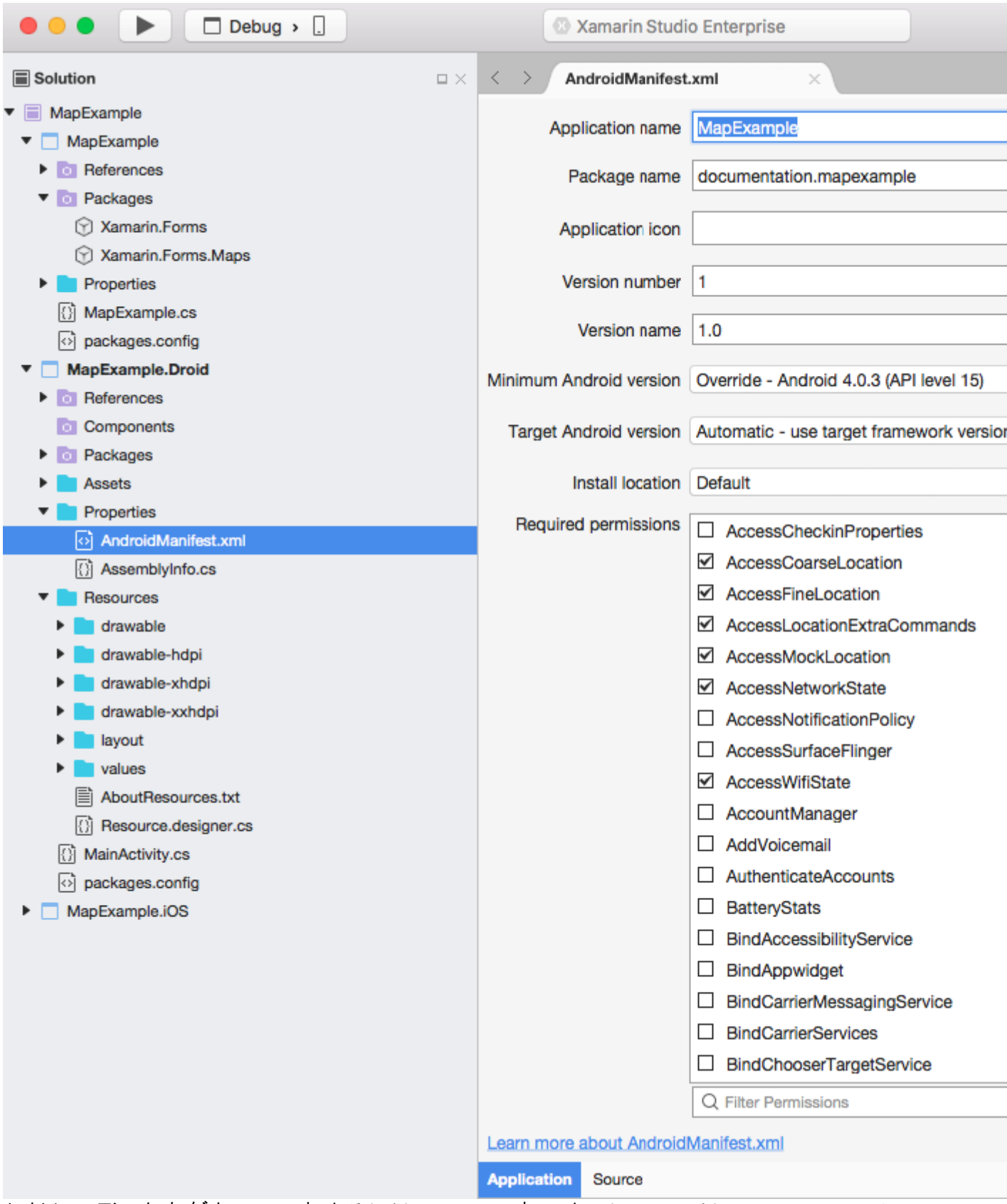
ファイル*AndroidManifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="documentation.mapexample">
    <uses-sdk
        android:minSdkVersion="15" />
    <application
        android:label="MapExample">
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBAg8X-t4pOIDDp3q5Ph45jKUIVjo_RnxU" />
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

また、マニフェストでいくつかのをにして、いくつかのをにするがあります

- いにアクセスする
- ファインロケーションにアクセスする

- アクセスロケーションのコマンド
- アクセスモックの
- アクセスネットワーク
- Wifiにアクセスする
- インターネット



ただし、データをダウンロードするには、の2つのがです。については、 Androidのアクセ

スをごください。それはAndroidののすべてのステップです。

> アンドロイドシミュレータでアプリをするには、Google PlayサービスをAndroid スマートフォンにインストールするがあります。 このチュートリアルにって、Xamarin Android PlayerにPlayサービスをインストールしてください。プレイストアのインストールにGoogle Playサービスのアップデートがつからないは、マップサービスにしているアプリからすることができます

をする

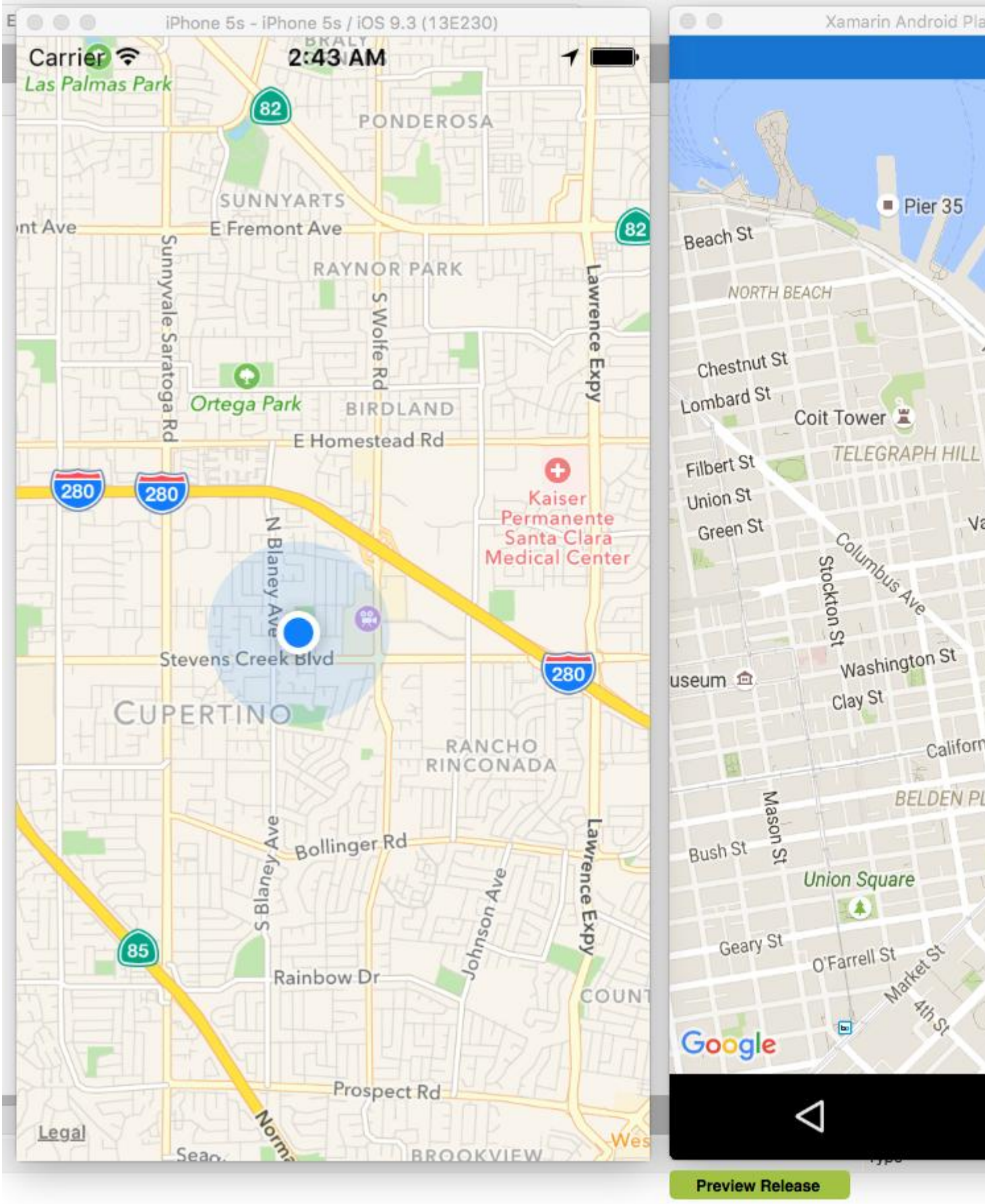
クロスプラットフォームプロジェクトにマップビューをするのはです。ここでは、それをうのをしますXAMLをしないPCLプロジェクトをしています。

PCLプロジェクト

*MapExample.cs*ファイル

```
public class App : Application
{
    public App()
    {
        var map = new Map();
        map.IsShowingUser = true;

        var rootPage = new ContentPage();
        rootPage.Content = map;

        MainPage = rootPage;
    }
}
```

それでです。 iOSやAndroidでアプリをすると、マップビューがされます



オンラインでマップのをむ https://riptutorial.com/ja/xamarin-forms/topic/3917/マップの

37: ユニットテスト

Examples

ビューモデルのテスト

めるに...

アプリケーションにしては、ViewModelはビジネスロジックとルールをすべてんだクラスで、にじてアプリケーションをさせます。 UI、データレイヤー、ネイティブフィーチャー、APIコールなどのをできるだけさせることもです。これらにより、VMはテストになります。
つまり、あなたのViewModel

- UIクラスビュー、ページ、スタイル、イベントにしないでください。
- のクラスのデータをするべきではありません。
- ビジネスロジックをし、なデータをUIにするがあります。
- をしてされるインタフェースをしてのコンポーネントデータベース、HTTP、UIをするがあります。

  ViewModelには、のVMタイプのプロパティもあります。えば、 `ContactsPageViewModel` は、 `ObservableCollection<ContactListItemViewModel>`よう `ObservableCollection<ContactListItemViewModel>`コレクションの `ObservableCollection<ContactListItemViewModel>`

ビジネス

するのがあるとします。

```
As an unauthorized user
I want to log into the app
So that I will access the authorized features
```

ユーザーストーリーをにした、のシナリオをしました。

```
Scenario: trying to log in with valid non-empty creds
  Given the user is on Login screen
   When the user enters 'user' as username
    And the user enters 'pass' as password
    And the user taps the Login button
   Then the app shows the loading indicator
    And the app makes an API call for authentication

Scenario: trying to log in empty username
  Given the user is on Login screen
```

```
   When the user enters '  ' as username
    And the user enters 'pass' as password
    And the user taps the Login button
   Then the app shows an error message saying 'Please, enter correct username and password'
    And the app doesn't make an API call for authentication
```

たちはこれらの2つのシナリオだけにとどまります。もちろん、のコーディングのにすべてをするがありますが、ビューモデルのテストにれていればです。

なTDDアプローチにって、テストされるのクラスをくことからめましょう。に、テストをし、ビジネスをすることでにします。

のクラス

```
public abstract class BaseViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

サービス

あなたのビューモデルがUIとHTTPクラスをしてはならないことをえていますかわりにクラスとしてし、 のにしないようにするがあります。

```
/// <summary>
/// Provides authentication functionality.
/// </summary>
public interface IAuthenticationService
{
    /// <summary>
    /// Tries to authenticate the user with the given credentials.
    /// </summary>
    /// <param name="userName">UserName</param>
    /// <param name="password">User's password</param>
    /// <returns>true if the user has been successfully authenticated</returns>
    Task<bool> Login(string userName, string password);
}

/// <summary>
/// UI-specific service providing abilities to show alert messages.
/// </summary>
public interface IAlertService
{
    /// <summary>
    /// Show an alert message to the user.
    /// </summary>
```

```
    /// <param name="title">Alert message title</param>
    /// <param name="message">Alert message text</param>
    Task ShowAlert(string title, string message);
}
```

ViewModelスタブの

さて、たちはログインのペ―ジクラスをっていますが、まずViewModelからめましょう

```
public class LoginPageViewModel : BaseViewModel
{
    private readonly IAuthenticationService authenticationService;
    private readonly IAlertService alertService;

    private string userName;
    private string password;
    private bool isLoading;

    private ICommand loginCommand;

    public LoginPageViewModel(IAuthenticationService authenticationService, IAlertService
alertService)
    {
        this.authenticationService = authenticationService;
        this.alertService = alertService;
    }

    public string UserName
    {
        get
        {
            return userName;
        }
        set
        {
            if (userName!= value)
            {
                userName= value;
                OnPropertyChanged();
            }
        }
    }

    public string Password
    {
        get
        {
            return password;
        }
        set
        {
            if (password != value)
            {
                password = value;
                OnPropertyChanged();
            }
        }
```

```
    }

    public bool IsLoading
    {
        get
        {
            return isLoading;
        }
        set
        {
            if (isLoading != value)
            {
                isLoading = value;
                OnPropertyChanged();
            }
        }
    }

    public ICommand LoginCommand => loginCommand ?? (loginCommand = new Command(Login));

    private void Login()
    {
        authenticationService.Login(UserName, Password);
    }
}
```

2つの`string`プロパティとUIにバインドするコマンドをしました。このトピックでは、ページクラス、XAMLマークアップをし、それにViewModelをバインドするについてはもではないため、ここではしません。

LoginPageViewModel インスタンスをするには

はあなたがおそらくコンストラクタでVMをしていたといます。ではわかるように、VMは2つのサービスがコンストラクタパラメータとしてされているため、`var viewModel = new LoginPageViewModel()`はできません。あなたがDependency Injectionにしていないなら、それについてふのがのです。このをらずにうことなく、なユニットテストはです。

テスト

のユースケースにしたがっていくつかのテストをいてみましょう。まず、しいアセンブリクライブラリのみをするか、Microsoftテストツールをするはなテストプロジェクトをするがありますをするがあります。`ProjectName.Tests`ようなをけ、のPCLプロジェクトへのをします。

はこのではNUnitとMoqをするつもりですが、あなたのテストライブラリをうことができます。らになものはもないでしょう。

さて、それはテストクラスです

```
[TestFixture]
```

```
public class LoginPageViewModelTest
{
}
```

テストの

の2つのシナリオのテストはのとおりです。される1つにつき1つのテストをち、1つのテストですべてをチェックしないようにしてください。これは、コードでがしたかについてよりなレポートをけるのにちます。

```
[TestFixture]
public class LoginPageViewModelTest
{
    private readonly Mock<IAuthenticationService> authenticationServiceMock =
        new Mock<IAuthenticationService>();
    private readonly Mock<IAlertService> alertServiceMock =
        new Mock<IAlertService>();

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_LoadingIndicatorShown(string userName, string password)
    {
        LoginPageViewModel model = CreateViewModelAndLogin(userName, password);

        Assert.IsTrue(model.IsLoading);
    }

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_AuthenticationRequested(string userName, string password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(userName, password), Times.Once);
    }

    [TestCase("", "pass")]
    [TestCase("   ", "pass")]
    [TestCase(null, "pass")]
    public void LogInWithEmptyuserName_AuthenticationNotRequested(string userName, string
password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(It.IsAny<string>(), It.IsAny<string>()),
Times.Never);
    }

    [TestCase("", "pass", "Please, enter correct username and password")]
    [TestCase("   ", "pass", "Please, enter correct username and password")]
    [TestCase(null, "pass", "Please, enter correct username and password")]
    public void LogInWithEmptyUserName_AlertMessageShown(string userName, string password,
string message)
    {
        CreateViewModelAndLogin(userName, password);

        alertServiceMock.Verify(x => x.ShowAlert(It.IsAny<string>(), message));
    }
```

```
    private LoginPageViewModel CreateViewModelAndLogin(string userName, string password)
    {
        var model = new LoginPageViewModel(
            authenticationServiceMock.Object,
            alertServiceMock.Object);

        model.UserName = userName;
        model.Password = password;

        model.LoginCommand.Execute(null);

        return model;
    }
}
```

さあ、いくぞ

、は、ViewModelの`Login`メソッドのしいをくことです。それだけです。

ビジネスロジックの

```
private async void Login()
{
    if (String.IsNullOrWhiteSpace(UserName) || String.IsNullOrWhiteSpace(Password))
    {
        await alertService.ShowAlert("Warning", "Please, enter correct username and
password");
    }
    else
    {
        IsLoading = true;
        bool isAuthenticated = await authenticationService.Login(UserName, Password);
    }
}
```

テストをした

これで、コードをしいテストでカバーしけることができ、よりした、セーフなテストがになりました。

オンラインでユニットテストをむ https://riptutorial.com/ja/xamarin-forms/topic/3529/ユニットテスト

38: リストビュ—の

き

このドキュメントでは、Xamarin Forms ListViewのさまざまなコンポ—ネントのについてしくしています。

Examples

XAMLとコ—ドビハインドでリフレッシュする

Xamarinの`ListView`でPull to Refreshをにするには、 `PullToRefresh`になっていることをしてから、`ListView`をプルするときにびすコマンドのをするがあります。

```
<ListView x:Name="itemListView" IsPullToRefreshEnabled="True" RefreshCommand="Refresh">
```

じことがコ—ドのですることができます

```
itemListView.IsPullToRefreshEnabled = true;
itemListView.RefreshCommand = Refresh;
```

に、 `Refresh`コマンドがコ—ドでをするかをするがあります。

```
public ICommand Refresh
{
    get
    {
        itemListView.IsRefreshing = true; //This turns on the activity
                                          //Indicator for the ListView
        //Then add your code to execute when the ListView is pulled
        itemListView.IsRefreshing = false;
    }
}
```

オンラインでリストビュ—のをむ https://riptutorial.com/ja/xamarin-forms/topic/9487/リストビュ—の

39: ローカルデータベースの

Examples

プロジェクトでのSQLite.NETの

SQLite.NETは、 `Xamarin.Forms` プロジェクトで `SQLite` バージョン3をしてローカルデータベースのサポートをできるオープンソースライブラリです。

のは、このコンポーネントを `Xamarin.Forms` プロジェクトにめるをしています。

1. バージョンのSQLite.csクラスをダウンロードし、プロジェクトにします。
2. データベースにまれるすべてのテーブルは、プロジェクトのクラスとしてモデルするがあります。テーブルは、クラスになくとも2つのをすることでされます。 `Table` クラスと `PrimaryKey` プロパティです。

このでは、 `Song` というのしいクラスがプロジェクトにされ、のようにされます。

```
using System;
using SQLite;

namespace SongsApp
{
    [Table("Song")]
    public class Song
    {
        [PrimaryKey]
        public string ID { get; set; }
        public string SongName { get; set; }
        public string SingerName { get; set; }
    }
}
```

3. に、 `SQLiteConnection` クラスSQLite.csにまれていますをする `Database` というしいクラスをします。このしいクラスでは、テーブルのデータベースアクセス、テーブル、CRUDのコードがされています。サンプルコードをにします。

```
using System;
using System.Linq;
using System.Collections.Generic;
using SQLite;

namespace SongsApp
{
    public class BaseDatos : SQLiteConnection
    {
        public BaseDatos(string path) : base(path)
        {
            Initialize();
```

```
        }

        void Initialize()
        {
            CreateTable<Song>();
        }

        public List<Song> GetSongs()
        {
            return Table<Song>().ToList();
        }

        public Song GetSong(string id)
        {
            return Table<Song>().Where(t => t.ID == id).First();
        }

        public bool AddSong(Song song)
        {
            Insert(song);
        }

        public bool UpdateSong(Song song)
        {
            Update(song);
        }

        public void DeleteSong(Song song)
        {
            Delete(song);
        }
    }
}
```

4. のでかるように、 `Database`クラスのコンストラクタには、SQLiteデ―タベ―スファイルをするファイルのをす`path`パラメ―タがまれています。な`Database`オブジェクトは`App.cs`でできます。 `path`はプラットフォ―ムです。

```
public class App : Application
{
    public static Database DB;

    public App ()
    {
        string dbFile = "SongsDB.db3";

#if __ANDROID__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        var dbPath = System.IO.Path.Combine(docPath, dbFile);
#else
#if __IOS__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        string libPath = System.IO.Path.Combine(docPath, "..", "Library");
        var dbPath = System.IO.Path.Combine(libPath, dbFile);
#else
        var dbPath = System.IO.Path.Combine(ApplicationData.Current.LocalFolder.Path, dbFile);
#endif
#endif
```

```
        DB = new Database(dbPath);

        // The root page of your application
        MainPage = new SongsPage();
    }
}
```

5. Songsテーブルへの**CRUD**をするがあるときはいつでも、 Appクラスをして**DB**オブジェクトをびしてください。たとえば、ユーザーがボタンをクリックしたにしいSongをするには、のコードをします。

```
void AddNewSongButton_Click(object sender, EventArgs a)
{
    Song s = new Song();
    s.ID = Guid.NewGuid().ToString();
    s.SongName = songNameEntry.Text;
    s.SingerName = singerNameEntry.Text;

    App.DB.AddSong(song);
}
```

Visual Studio 2015でxamarin.formsをしてローカルデータベースをする

SQliteのステップバイステップの

1. のは、このコンポーネントを**Xamarin.Forms**プロジェクトにめるをしています。パッケージをpcl、Andriod、Windows、Iosにするの**Manage Nuget packages**をクリックして、Browseをクリックして**SQLite.Net.Core-**インストールがしたら**PCL** 、 **SQLite Net Extensions**をしてください

2. コードのにクラス**Employee.cs**をするには

```
using SQLite.Net.Attributes;

    namespace DatabaseEmployeeCreation.SqlLite
    {
        public   class Employee
        {
            [PrimaryKey,AutoIncrement]
            public int Eid { get; set; }
            public string Ename { get; set; }
            public string Address { get; set; }
            public string phonenumber { get; set; }
            public string email { get; set; }
        }
    }
```

3. 1つのインターフェイスをするには**ISQLite**

```
using SQLite.Net;
           //using SQLite.Net;
           namespace DatabaseEmployeeCreation.SqlLite.ViewModel
           {
```

```
            public interface ISQLite
            {
                SQLiteConnection GetConnection();
            }
        }
```

4. のコードは、データベースロジックとメソッドのための1つのクラスをします。

SQLite.Netをします。 using System.Collections.Generic; System.Linqをします。 Xamarin.Formsをします。DatabaseEmployeeCreation.SqlLite.ViewModel {publicクラスDatabaseLogic {オブジェクトロッカー=しいオブジェクト; SQLiteConnectionデータベース。

```
public DatabaseLogic()
{
    database = DependencyService.Get<ISQLite>().GetConnection();
    // create the tables
    database.CreateTable<Employee>();
}

public IEnumerable<Employee> GetItems()
{
    lock (locker)
    {
        return (from i in database.Table<Employee>() select i).ToList();
    }
}

public IEnumerable<Employee> GetItemsNotDone()
{
    lock (locker)
    {
        return database.Query<Employee>("SELECT * FROM [Employee]");
    }
}

public Employee GetItem(int id)
{
    lock (locker)
    {
        return database.Table<Employee>().FirstOrDefault(x => x.Eid == id);
    }
}

public int SaveItem(Employee item)
{
    lock (locker)
    {
        if (item.Eid != 0)
        {
            database.Update(item);
            return item.Eid;
        }
        else
        {
            return database.Insert(item);
        }
    }
}
```

```
    public int DeleteItem(int Eid)
    {
        lock (locker)
        {
            return database.Delete<Employee>(Eid);
        }
    }
}
```

}

5. xaml.forms EmployeeRegistration.xamlをする

```
    <?xml version="1.0" encoding="utf-8" ?>
    <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
                 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                 x:Class="DatabaseEmployeeCreation.SqlLite.EmployeeRegistration"
      Title="{Binding Name}" >
      <StackLayout VerticalOptions="StartAndExpand" Padding="20">

        <Label Text="Ename" />
        <Entry x:Name="nameEntry" Text="{Binding Ename}"/>
        <Label Text="Address" />
        <Editor x:Name="AddressEntry" Text="{Binding Address}"/>
        <Label Text="phonenumber" />
        <Entry x:Name="phonenumberEntry" Text="{Binding phonenumber}"/>
        <Label Text="email" />
        <Entry x:Name="emailEntry" Text="{Binding email}"/>

        <Button Text="Add" Clicked="addClicked"/>

       <!-- <Button Text="Delete" Clicked="deleteClicked"/>-->

        <Button Text="Details" Clicked="DetailsClicked"/>

        <!--  <Button Text="Edit" Clicked="speakClicked"/>-->

      </StackLayout>
    </ContentPage>
```

EmployeeRegistration.cs

```
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using DatabaseEmployeeCreation.SqlLite.Views;
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;

    using Xamarin.Forms;

    namespace DatabaseEmployeeCreation.SqlLite
    {
        public partial class EmployeeRegistration : ContentPage
        {
            private int empid;
```

```
        private Employee obj;

        public EmployeeRegistration()
        {
            InitializeComponent();

        }

        public EmployeeRegistration(Employee obj)
        {
            this.obj = obj;
            var eid = obj.Eid;
            Navigation.PushModalAsync(new EmployeeRegistration());
            var Address = obj.Address;
            var email = obj.email;
            var Ename = obj.Ename;
            var phonenumber = obj.phonenumber;
            AddressEntry. = Address;
            emailEntry.Text = email;
            nameEntry.Text = Ename;

            //AddressEntry.Text = obj.Address;
            //emailEntry.Text = obj.email;
            //nameEntry.Text = obj.Ename;
            //phonenumberEntry.Text = obj.phonenumber;

            Employee empupdate = new Employee(); //updateing Values
            empupdate.Address = AddressEntry.Text;
            empupdate.Ename = nameEntry.Text;
            empupdate.email = emailEntry.Text;
            empupdate.Eid = obj.Eid;
            App.Database.SaveItem(empupdate);
            Navigation.PushModalAsync(new EmployeeRegistration());

        }

        public EmployeeRegistration(int empid)
        {
            this.empid = empid;
            Employee lst = App.Database.GetItem(empid);
            //var Address = lst.Address;
            //var email = lst.email;
            //var Ename = lst.Ename;
            //var phonenumber = lst.phonenumber;
            //AddressEntry.Text = Address;
            //emailEntry.Text = email;
            //nameEntry.Text = Ename;
            //phonenumberEntry.Text = phonenumber;

            // to retriva values based on id to
            AddressEntry.Text = lst.Address;
            emailEntry.Text = lst.email;
            nameEntry.Text = lst.Ename;
            phonenumberEntry.Text = lst.phonenumber;

            Employee empupdate = new Employee(); //updateing Values
            empupdate.Address = AddressEntry.Text;
            empupdate.email = emailEntry.Text;
            App.Database.SaveItem(empupdate);
            Navigation.PushModalAsync(new EmployeeRegistration());
        }
```

```
            void addClicked(object sender, EventArgs e)
            {
                //var createEmp = (Employee)BindingContext;
                Employee emp = new Employee();
                emp.Address = AddressEntry.Text;
                emp.email = emailEntry.Text;
                emp.Ename = nameEntry.Text;
                emp.phonenumber = phonenumberEntry.Text;
                App.Database.SaveItem(emp);
                this.Navigation.PushAsync(new EmployeeDetails());

            }
            //void deleteClicked(object sender, EventArgs e)
            //{
            //    var emp = (Employee)BindingContext;
            //    App.Database.DeleteItem(emp.Eid);
            //    this.Navigation.PopAsync();
            //}
            void DetailsClicked(object sender, EventArgs e)
            {
                var empcancel = (Employee)BindingContext;
                this.Navigation.PushAsync(new EmployeeDetails());
            }
            //    void speakClicked(object sender, EventArgs e)
            //    {
            //        var empspek = (Employee)BindingContext;
            //        //DependencyService.Get<ITextSpeak>().Speak(empspek.Address + " " +
empspek.Ename);
            //    }
        }
    }
```

6. コードのにEmployeeDetailsをする

```
using DatabaseEmployeeCreation;
using DatabaseEmployeeCreation.SqlLite;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;

namespace DatabaseEmployeeCreation.SqlLite.Views
{
    public partial class EmployeeDetails : ContentPage
    {
        ListView lv = new ListView();
        IEnumerable<Employee> lst;
        public EmployeeDetails()
        {
            InitializeComponent();
            displayemployee();
        }

        private void displayemployee()
        {
            Button btn = new Button()
```

```
            {

                Text = "Details",
                BackgroundColor = Color.Blue,
            };
            btn.Clicked += Btn_Clicked;
            //IEnumerable<Employee> lst = App.Database.GetItems();
            //IEnumerable<Employee> lst1 = App.Database.GetItemsNotDone();
            //IEnumerable<Employee> lst2 = App.Database.GetItemsNotDone();
            Content = new StackLayout()
            {
                Children = { btn },
            };
        }

        private void Btn_Clicked(object sender, EventArgs e)
        {
            lst = App.Database.GetItems();

            lv.ItemsSource = lst;
            lv.HasUnevenRows = true;
            lv.ItemTemplate = new DataTemplate(typeof(OptionsViewCell));

            Content = new StackLayout()
            {
                Children = { lv },
            };

        }
    }
```

```
public class OptionsViewCell : ViewCell
{

    int empid;
    Button btnEdit;
    public OptionsViewCell()
    {
    }
    protected override void OnBindingContextChanged()
    {
        base.OnBindingContextChanged();

        if (this.BindingContext == null)
            return;

        dynamic obj = BindingContext;
        empid = Convert.ToInt32(obj.Eid);
        var lblname = new Label
        {
            BackgroundColor = Color.Lime,
            Text = obj.Ename,
        };

        var lblAddress = new Label
        {
            BackgroundColor = Color.Yellow,
            Text = obj.Address,
        };
```

```
            var lblphonenumber = new Label
            {
                BackgroundColor = Color.Pink,
                Text = obj.phonenumber,
            };

            var lblemail = new Label
            {
                BackgroundColor = Color.Purple,
                Text = obj.email,
            };

            var lbleid = new Label
            {
                BackgroundColor = Color.Silver,
                Text = (empid).ToString(),
            };

            //var lblname = new Label
            //{
            //    BackgroundColor = Color.Lime,
            //    // HorizontalOptions = LayoutOptions.Start
            //};
            //lblname.SetBinding(Label.TextProperty, "Ename");

            //var lblAddress = new Label
            //{
            //    BackgroundColor = Color.Yellow,
            //    //HorizontalOptions = LayoutOptions.Center,
            //};
            //lblAddress.SetBinding(Label.TextProperty, "Address");

            //var lblphonenumber = new Label
            //{
            //    BackgroundColor = Color.Pink,
            //    //HorizontalOptions = LayoutOptions.CenterAndExpand,
            //};
            //lblphonenumber.SetBinding(Label.TextProperty, "phonenumber");

            //var lblemail = new Label
            //{
            //    BackgroundColor = Color.Purple,
            //    // HorizontalOptions = LayoutOptions.CenterAndExpand
            //};
            //lblemail.SetBinding(Label.TextProperty, "email");
            //var lbleid = new Label
            //{
            //    BackgroundColor = Color.Silver,
            //    // HorizontalOptions = LayoutOptions.CenterAndExpand
            //};
            //lbleid.SetBinding(Label.TextProperty, "Eid");
            Button btnDelete = new Button
            {
                BackgroundColor = Color.Gray,

                Text = "Delete",
                //WidthRequest = 15,
                //HeightRequest = 20,
                TextColor = Color.Red,
                HorizontalOptions = LayoutOptions.EndAndExpand,
            };
```

```
                btnDelete.Clicked += BtnDelete_Clicked;
                //btnDelete.PropertyChanged += BtnDelete_PropertyChanged;

                btnEdit = new Button
                {
                    BackgroundColor = Color.Gray,
                    Text = "Edit",
                    TextColor = Color.Green,
                };
                // lbleid.SetBinding(Label.TextProperty, "Eid");
                btnEdit.Clicked += BtnEdit_Clicked1; ;
                //btnEdit.Clicked += async (s, e) =>{
                //    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration());
                //};

                View = new StackLayout()
                {
                    Orientation = StackOrientation.Horizontal,
                    BackgroundColor = Color.White,
                    Children = { lbleid, lblname, lblAddress, lblemail, lblphonenumber,
btnDelete, btnEdit },
                };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.Center, WidthRequest = 10,
BackgroundColor = Color.Yellow, Children = { lblAddress } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, WidthRequest = 30, BackgroundColor
= Color.Yellow, Children = { lblemail } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, BackgroundColor = Color.Green,
Children = { lblphonenumber } };




                //string Empid =c.eid ;

            }

            private async void BtnEdit_Clicked1(object sender, EventArgs e)
            {
               Employee obj= App.Database.GetItem(empid);
                if (empid > 0)
                {
                    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(obj));
                }
                else {
                await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(empid));
                }
            }



            private void BtnDelete_Clicked(object sender, EventArgs e)
            {
```

```
                // var eid = Convert.ToInt32(empid);
                // var item = (Xamarin.Forms.Button)sender;
                int eid = empid;
                App.Database.DeleteItem(eid);
            }
            //private void BtnDelete_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
            //{
            // var ename=  e.PropertyName;
            //}
        }

        //private void BtnDelete_Clicked(object sender, EventArgs e)
        //{
        //    var eid = 8;
        //    var item = (Xamarin.Forms.Button)sender;

        //    App.Database.DeleteItem(eid);
        //}
    }
```

7. AndroidおよびiOSのGetConnectionメソッドでメソッドをするには

```
    using System;
    using Xamarin.Forms;
    using System.IO;
    using DatabaseEmployeeCreation.Droid;
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using SQLite;
    using SQLite.Net;

    [assembly: Dependency(typeof(SQLiteEmployee_Andriod))]
    namespace DatabaseEmployeeCreation.Droid
    {
        public class SQLiteEmployee_Andriod : ISQLite
        {
            public SQLiteEmployee_Andriod()
            {
            }

            #region ISQLite implementation
            public SQLiteConnection GetConnection()
            {
                //var sqliteFilename = "EmployeeSQLite.db3";
                //string documentsPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal); // Documents
folder
                //var path = Path.Combine(documentsPath, sqliteFilename);

                //// This is where we copy in the prepopulated database
                //Console.WriteLine(path);
                //if (!File.Exists(path))
                //{
                //    var s =
Forms.Context.Resources.OpenRawResource(Resource.Raw.EmployeeSQLite);  // RESOURCE NAME ###

                //    // create a write stream
                //    FileStream writeStream = new FileStream(path, FileMode.OpenOrCreate,
FileAccess.Write);
                //    // write to the stream
```

```
                //    ReadWriteStream(s, writeStream);
                //}

                //var conn = new SQLiteConnection(path);

                //// Return the database connection
                //return conn;
                var filename = "DatabaseEmployeeCreationSQLite.db3";
                var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
                var path = Path.Combine(documentspath, filename);
                var platform = new SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
                var connection = new SQLite.Net.SQLiteConnection(platform, path);
                return connection;
            }

            //public  SQLiteConnection GetConnection()
            //{
            //    var filename = "EmployeeSQLite.db3";
            //    var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            //    var path = Path.Combine(documentspath, filename);

            //    var platform = new
SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
            //    var connection = new SQLite.Net.SQLiteConnection(platform, path);
            //    return connection;
            //}
            #endregion

            /// <summary>
            /// helper method to get the database out of /raw/ and into the user filesystem
            /// </summary>
            void ReadWriteStream(Stream readStream, Stream writeStream)
            {
                int Length = 256;
                Byte[] buffer = new Byte[Length];
                int bytesRead = readStream.Read(buffer, 0, Length);
                // write the required bytes
                while (bytesRead > 0)
                {
                    writeStream.Write(buffer, 0, bytesRead);
                    bytesRead = readStream.Read(buffer, 0, Length);
                }
                readStream.Close();
                writeStream.Close();
            }
        }
    }
```

はのががしたになですっています

オンラインでローカルデータベースのをむ https://riptutorial.com/ja/xamarin-forms/topic/5997/ローカルデータベースの

40: なXamarin.Formsアプリライフサイクルプラットフォームに

Examples

Xamarin.Formsのライフサイクルはのアプリのライフサイクルではなく、クロスプラットフォームのです。

さまざまなプラットフォームのネイティブアプリライフサイクルメソッドをてみましょう。

アンドロイド。

```
//Xamarin.Forms.Platform.Android.FormsApplicationActivity lifecycle methods:
protected override void OnCreate(Bundle savedInstanceState);
protected override void OnDestroy();
protected override void OnPause();
protected override void OnRestart();
protected override void OnResume();
protected override void OnStart();
protected override void OnStop();
```

iOS。

```
//Xamarin.Forms.Platform.iOS.FormsApplicationDelegate lifecycle methods:
public override void DidEnterBackground(UIApplication uiApplication);
public override bool FinishedLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void OnActivated(UIApplication uiApplication);
public override void OnResignActivation(UIApplication uiApplication);
public override void WillEnterForeground(UIApplication uiApplication);
public override bool WillFinishLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void WillTerminate(UIApplication uiApplication);
```

Windows。

```
//Windows.UI.Xaml.Application lifecycle methods:
public event EventHandler<System.Object> Resuming;
public event SuspendingEventHandler Suspending;
protected virtual void OnActivated(IActivatedEventArgs args);
protected virtual void OnFileActivated(FileActivatedEventArgs args);
protected virtual void OnFileOpenPickerActivated(FileOpenPickerActivatedEventArgs args);
protected virtual void OnFileSavePickerActivated(FileSavePickerActivatedEventArgs args);
protected virtual void OnLaunched(LaunchActivatedEventArgs args);
protected virtual void OnSearchActivated(SearchActivatedEventArgs args);
protected virtual void OnShareTargetActivated(ShareTargetActivatedEventArgs args);
protected virtual void OnWindowCreated(WindowCreatedEventArgs args);

//Windows.UI.Xaml.Window lifecycle methods:
public event WindowActivatedEventHandler Activated;
```

```
public event WindowClosedEventHandler Closed;
public event WindowVisibilityChangedEventHandler VisibilityChanged;
```

そして、 **Xamarin.Forms**アプリライフサイクルメソッド

```
//Xamarin.Forms.Application lifecycle methods:
protected virtual void OnResume();
protected virtual void OnSleep();
protected virtual void OnStart();
```

Xamarin.Formsのクロスプラットフォームアプリケーションのライフサイクルのは、リストをするだけでにわかります。これは、あなたのアプリケーションがどのようなにあるのかというながかりをえますが、ほとんどの、プラットフォームのロジックをするがあります。

オンラインでなXamarin.Formsアプリライフサイクルプラットフォームにをむ
https://riptutorial.com/ja/xamarin-forms/topic/8329/なxamarin-formsアプリライフサイクル-プラットフォームに-

41:

Examples

iOSでのについてする1つの

iOSプロジェクトの `Main.cs`ファイルにし、にすように`Main.cs`コードをします。

```
    static void Main(string[] args)
    {
        try
        {
            UIApplication.Main(args, null, "AppDelegate");
        }
        catch (Exception ex)
        {
            Debug.WriteLine("iOS Main Exception: {0}", ex);

            var watson = new LittleWatson();
            watson.SaveReport(ex);
        }
    }
```

ポータブルコードでされる`ILittleWatson`インターフェイスは、のようになります。

```
public interface ILittleWatson
{
    Task<bool> SendReport();

    void SaveReport(Exception ex);
}
```

iOSプロジェクトの

```
[assembly: Xamarin.Forms.Dependency(typeof(LittleWatson))]
namespace SomeNamespace
{
    public class LittleWatson : ILittleWatson
    {
        private const string FileName = "Report.txt";

        private readonly static string DocumentsFolder;
        private readonly static string FilePath;

        private TaskCompletionSource<bool> _sendingTask;

        static LittleWatson()
        {
            DocumentsFolder =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
            FilePath = Path.Combine(DocumentsFolder, FileName);
        }
```

```
        public async Task<bool> SendReport()
        {
            _sendingTask = new TaskCompletionSource<bool>();

            try
            {
                var text = File.ReadAllText(FilePath);
                File.Delete(FilePath);
                if (MFMailComposeViewController.CanSendMail)
                {
                    var email = ""; // Put receiver email here.
                    var mailController = new MFMailComposeViewController();
                    mailController.SetToRecipients(new string[] { email });
                    mailController.SetSubject("iPhone error");
                    mailController.SetMessageBody(text, false);
                    mailController.Finished += (object s, MFComposeResultEventArgs args) =>
                    {
                        args.Controller.DismissViewController(true, null);
                        _sendingTask.TrySetResult(true);
                    };

                    ShowViewController(mailController);
                }
            }
            catch (FileNotFoundException)
            {
                // No errors found.
                _sendingTask.TrySetResult(false);
            }

            return await _sendingTask.Task;
        }

        public void SaveReport(Exception ex)
        {
            var exceptionInfo = $"{ex.Message} - {ex.StackTrace}";
            File.WriteAllText(FilePath, exceptionInfo);
        }

        private static void ShowViewController(UIViewController controller)
        {
            var topController = UIApplication.SharedApplication.KeyWindow.RootViewController;
            while (topController.PresentedViewController != null)
            {
                topController = topController.PresentedViewController;
            }

            topController.PresentViewController(controller, true, null);
        }
    }
}
```

そして、どこかで、どこにアプリがするのかをれます

```
        var watson = DependencyService.Get<ILittleWatson>();
        if (watson != null)
        {
            await watson.SendReport();
        }
```

オンライ ンでをむ https://riptutorial.com/ja/xamarin-forms/topic/6428/

42: サービス

PCLまたはプロジェクトからプラットフォームのAPIにアクセスします。

Examples

カメラとギャラリーにアクセスします。

アクセスコード[https://github.com/vDoers/vDoersCameraAccess]

オンラインでサービスをむ https://riptutorial.com/ja/xamarin-forms/topic/6127/サービス

43: Picker - Xamarinフォーム Android および iOS

Picker XFAndroidとiOS

Examples

contact_picker.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
    public class App : Application
    {
        public App ()
        {
            // The root page of your application
            MainPage = new MyPage();
        }

        protected override void OnStart ()
        {
            // Handle when your app starts
        }

        protected override void OnSleep ()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume ()
        {
            // Handle when your app resumes
        }
    }
}
```

MyPage.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
    public class MyPage : ContentPage
    {
        Button button;
        public MyPage ()
```

```
        {
            button = new Button {
                Text = "choose contact"
            };

            button.Clicked += async (object sender, EventArgs e) => {

                    if (Device.OS == TargetPlatform.iOS) {
                        await Navigation.PushModalAsync (new ChooseContactPage ());
                        }
                    else if (Device.OS == TargetPlatform.Android)
                    {
                        MessagingCenter.Send (this, "android_choose_contact", "number1");
                    }


            };

            Content = new StackLayout {
                Children = {
                    new Label { Text = "Hello ContentPage" },
                    button
                }
            };
        }

        protected override void OnSizeAllocated (double width, double height)
        {
            base.OnSizeAllocated (width, height);

            MessagingCenter.Subscribe<MyPage, string> (this, "num_select", (sender, arg) => {
                DisplayAlert ("contact", arg, "OK");
            });

        }
    }
}
```

ChooseContactPicker.cs

```
using System;
using Xamarin.Forms;

namespace contact_picker
{
    public class ChooseContactPage : ContentPage
    {
        public ChooseContactPage ()
        {

        }
    }
}
```

ChooseContactActivity.cs

```
using Android.App;
```

```
using Android.OS;
using Android.Content;
using Android.Database;
using Xamarin.Forms;

namespace contact_picker.Droid
{

    [Activity (Label = "ChooseContactActivity")]

    public class ChooseContactActivity : Activity
    {
        public string type_number = "";
        protected override void OnCreate (Bundle savedInstanceState)
        {

            base.OnCreate (savedInstanceState);

            Intent intent = new Intent(Intent.ActionPick,
Android.Provider.ContactsContract.CommonDataKinds.Phone.ContentUri);
            StartActivityForResult(intent, 1);
        }

        protected override void OnActivityResult (int requestCode, Result resultCode, Intent
data)
        {
            // TODO Auto-generated method stub

            base.OnActivityResult (requestCode, resultCode, data);
            if (requestCode == 1) {
                if (resultCode == Result.Ok) {

                    Android.Net.Uri contactData = data.Data;
                    ICursor cursor = ContentResolver.Query(contactData, null, null, null,
null);

                    cursor.MoveToFirst();

                    string number =
cursor.GetString(cursor.GetColumnIndexOrThrow(Android.Provider.ContactsContract.CommonDataKinds.Phone.N



                    var twopage_renderer = new MyPage();
                    MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select",
number);
                    Finish ();
                    Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();


                }
                else if (resultCode == Result.Canceled)
                {
                    Finish ();
                }
            }
        }
    }
}
```

MainActivity.cs

```
using System;

using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Xamarin.Forms;

namespace contact_picker.Droid
{
    [Activity (Label = "contact_picker.Droid", Icon = "@drawable/icon", MainLauncher = true,
ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
    {
        protected override void OnCreate (Bundle bundle)
        {
            base.OnCreate (bundle);

            global::Xamarin.Forms.Forms.Init (this, bundle);

            LoadApplication (new App ());


            MessagingCenter.Subscribe<MyPage, string>(this, "android_choose_contact", (sender,
args) => {
                Intent i = new Intent (Android.App.Application.Context,
typeof(ChooseContactActivity));
                i.PutExtra ("number1", args);
                StartActivity (i);
            });
        }
    }
}
```

ChooseContactRenderer.cs

```
using UIKit;
using AddressBookUI;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;
using contact_picker;
using contact_picker.iOS;


[assembly: ExportRenderer (typeof(ChooseContactPage), typeof(ChooseContactRenderer))]

namespace contact_picker.iOS
{
    public partial class ChooseContactRenderer : PageRenderer
    {
        ABPeoplePickerNavigationController _contactController;
```

```
        public string type_number;

        protected override void OnElementChanged (VisualElementChangedEventArgs e)
        {
            base.OnElementChanged (e);

            var page = e.NewElement as ChooseContactPage;

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

            }

        }

        public override void ViewDidLoad ()
        {
            base.ViewDidLoad ();

            _contactController = new ABPeoplePickerNavigationController ();

            this.PresentModalViewController (_contactController, true); //display contact
chooser


            _contactController.Cancelled += delegate {
                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();

                this.DismissModalViewController (true); };

            _contactController.SelectPerson2 += delegate(object sender,
ABPeoplePickerSelectPerson2EventArgs e) {

                var getphones = e.Person.GetPhones();
                string number = "";

                if (getphones == null)
                {
                    number = "Nothing";
                }
                else if (getphones.Count > 1)
                {
                    //il ya plus de 2 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value + "/" + number;
                    }
                }
                else if (getphones.Count == 1)
                {
                    //il ya 1 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value;
                    }
                }


                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();
```

```
                var twopage_renderer = new MyPage();
                MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select", number);
                this.DismissModalViewController (true);



            };
        }

        public override void ViewDidUnload ()
        {
            base.ViewDidUnload ();

            // Clear any references to subviews of the main view in order to
            // allow the Garbage Collector to collect them sooner.
            //
            // e.g. myOutlet.Dispose (); myOutlet = null;

            this.DismissModalViewController (true);
        }

        public override bool ShouldAutorotateToInterfaceOrientation (UIInterfaceOrientation
toInterfaceOrientation)
        {
            // Return true for supported orientations
            return (toInterfaceOrientation != UIInterfaceOrientation.PortraitUpsideDown);
        }
    }
}
```

オンラインでPicker - XamarinフォームAndroidおよびiOSをむ https://riptutorial.com/ja/xamarin-forms/topic/6659/picker-----xamarinフォーム-androidおよびios-

クレジット

S. No		Contributors
1	Xamarin.Formsをいめる	Akshay Kulkarni, chrisntr, Community, Demitrian, hankide, jdstaerk, Manohar, patridge, Sergey Metlov, spaceplane
2	CarouselView - プレリリース	dpserge
3	DependencyService	Steven Thewissen
4	DependencyServiceをしたネイティブへのアクセス	Gerald Versluis, hankide, hvaughan3, Sergey Metlov
5	MessagingCenter	Gerald Versluis
6	OAuth2	Eng Soon Cheah
7	Xamarin Plugin	Eng Soon Cheah
8	Xamarin.Formsセル	Eng Soon Cheah
9	Xamarin.FormsでのBDDユニットテスト	Ben Ishiyama-Levy
10	Xamarin.Formsでのナビゲーション	Fernando Arreguín, jimmgarr, Lucas Moura Veloso, Paul, Sergey Metlov, Taras Shevchuk, Willian D. Andrade
11	Xamarin.FormsのAppSettingsリーダー	Ben Ishiyama-Levy, GvSharma
12	Xamarin.Formsビュー	Aaron Thompson, Eng Soon Cheah
13	Xamarin.Formsページ	Eng Soon Cheah
14	Xamarinジェスチャー	Joehl
15	XamarinフォームのSQLデータベースとAPI	RIYAZ

16	Xamarinフォームレイアウト	Eng Soon Cheah, Gerald Versluis, Lucas Moura Veloso
17	Xamarinレイアウト	Ege Aydın
18	アラートをする	aboozz pallikara, GvSharma, Sreeraj, Yehor Hromadskyi
19	エフェクト	Swaminathan Vetri
20	カスタムコントロールの	hvaughan3, spaceplane, Yehor Hromadskyi
21	カスタムレンダラー	Bonelol, hankide, Nicolas Bodin-Ripert, Nicolas Bodin-Ripert, nishantvodoo, Yehor Hromadskyi, Zverev Eugene
22	キャッシング	Sergey Metlov
23	ジェスチャー	doerig, Gerald Versluis, Michael Rumpler
24	スタイルのカスタムフォント	Roma Rudyak
25	データバインディング	Andrew, Matthew, Yehor Hromadskyi
26	トリガーとビヘイビア	hamalaiv, hvaughan3
27	なぜXamarinフォームとXamarinフォームをするのか	Daniel Krzyczkowski, mike
28	プッシュ	Gerald Versluis, user1568891
29	プラットフォームの	Ege Aydın
30	プラットフォームの	Alois, GalaxiaGuy, Paul
31	マップの	Taras Shevchuk
32	ユニットテスト	jerone, Sergey Metlov
33	リストビューの	cvanbeek
34	ローカルデータベースの	Luis Beltran, Manohar
35	なXamarin.Formsア	Zverev Eugene

	プリライフサイクルプラットフォームに	
36		Yehor Hromadskyi
37	サービス	RIYAZ
38	Picker - XamarinフォームAndroidおよびiOS	Pucho Eric