

モバイルアプリ開発  
ネイティブですか？ クロスプラットフォームで  
するか？

2018/2/19(月)  
カサレアル様共催セミナー

エクセルソフト株式会社  
ソフトウェア事業部  
新規事業開発室室長  
田淵義人

Twitter: [@ytabuchi](https://twitter.com/ytabuchi)  
facebook: [ytabuchi.xlsoft](https://facebook.com/ytabuchi.xlsoft)



田淵義人@エクセルソフト

営業(セールスエンジニア) 兼 新規事業開発室 室長

Xamarin コミュニティエバンジェリスト

Microsoft MVP Visual Studio and Development Technologies

Xamarin MVP

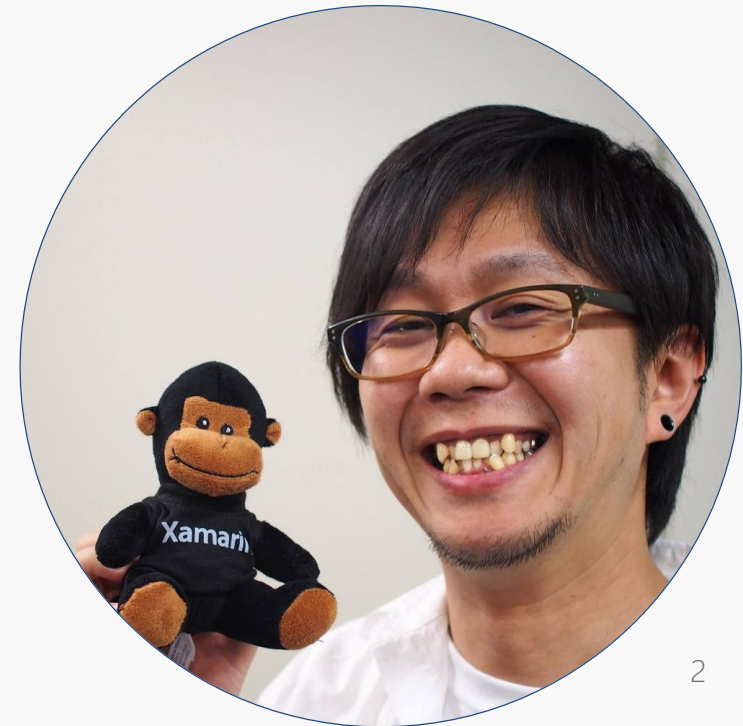
コミュニティ

Japan Xamarin User Group 主宰

Twitter: @ytabuchi

facebook: ytabuchi.xlsoft

Blog: Xamarin 日本語情報

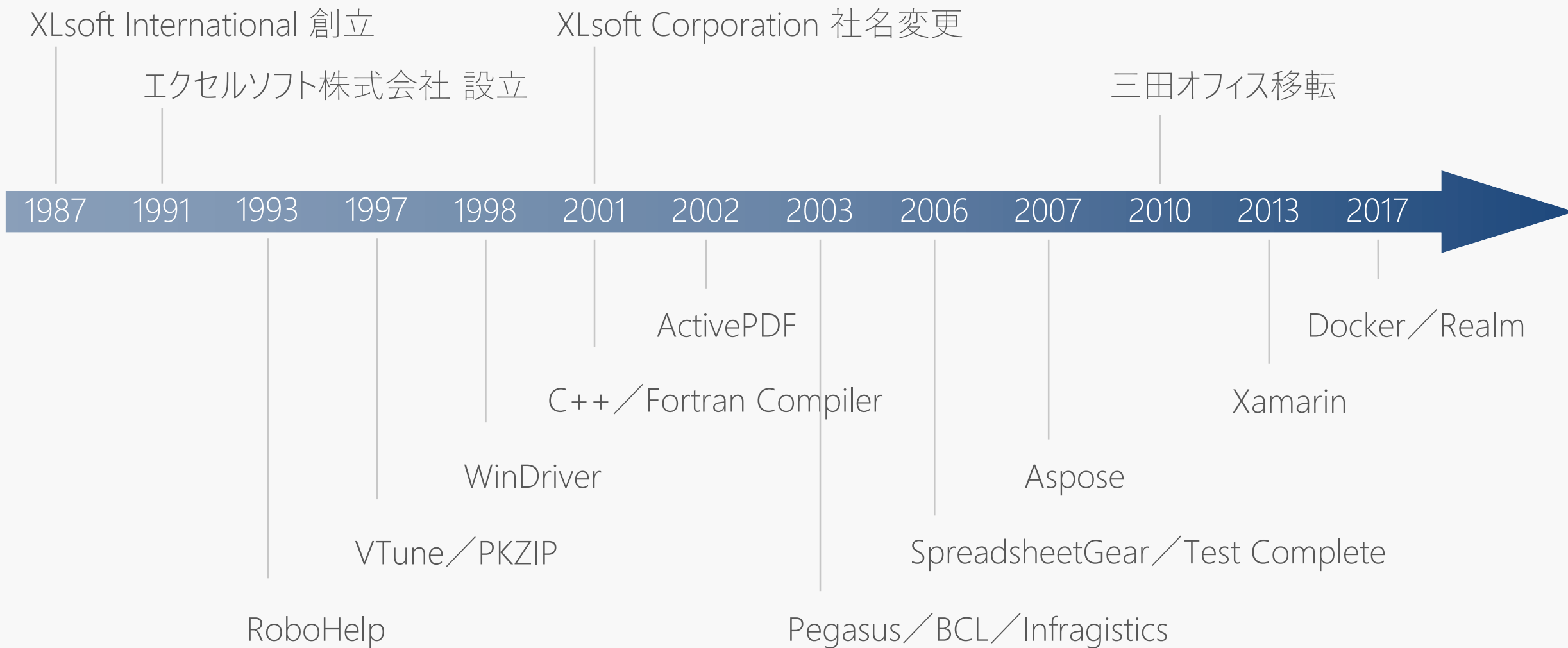


# エクセルソフトについて

開発者向けソフトウェア、ライブラリの販売／サポート  
ソフトウェア、ドキュメントのローカライズ  
海外製品の輸入・販売

開発者に特化した取り扱い製品群  
75,000名のメールニュース  
45名収容可能なセミナールーム

# 沿革



# ゴール

クロスプラットフォーム開発の特徴を知る

Xamarin のイメージを掴む

Xamarin に興味を持っていただく

# トピック

- Xamarinの利点／欠点は
- Xamarin.Forms 使える？
- 性能比較（ネイティブ／Xamarin）
- Web（HTML+Javascript）系との違い

# Xamarin とは

# Xamarin

C#／.NET／Visual Studio

フル “ネイティブ” アプリ

API 100% 移植

iOS は当日アップデート、Android は 1 ヶ月～ 3 ヶ月でアップデート

コード共通化



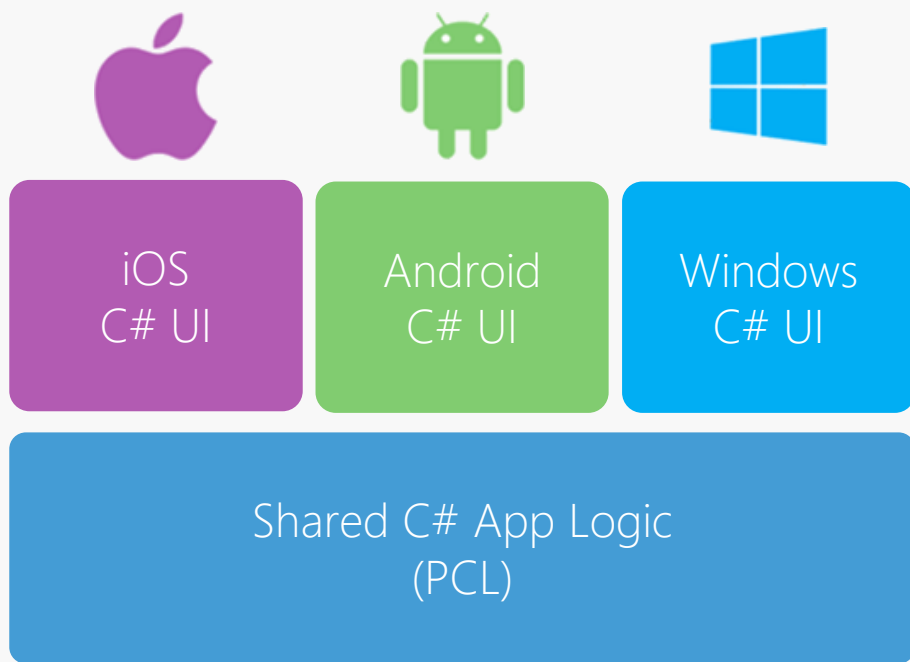
# C#

```
button.Click += async (sender, e) =>
{
    var client = new HttpClient();
    using (var reader = new StreamReader(await client.GetStreamAsync("xxx")))
    {
        var deserializer = new XmlSerializer(typeof(Rss));
        var latest = deserializer.Deserialize(reader) as Rss;
        var feed = latest.Channel.Items
            .Where(x => x.Link.Contains("xamarin"))
            .Select(x => x.Title).ToList();
    }
};
```

# 2つの開発手法

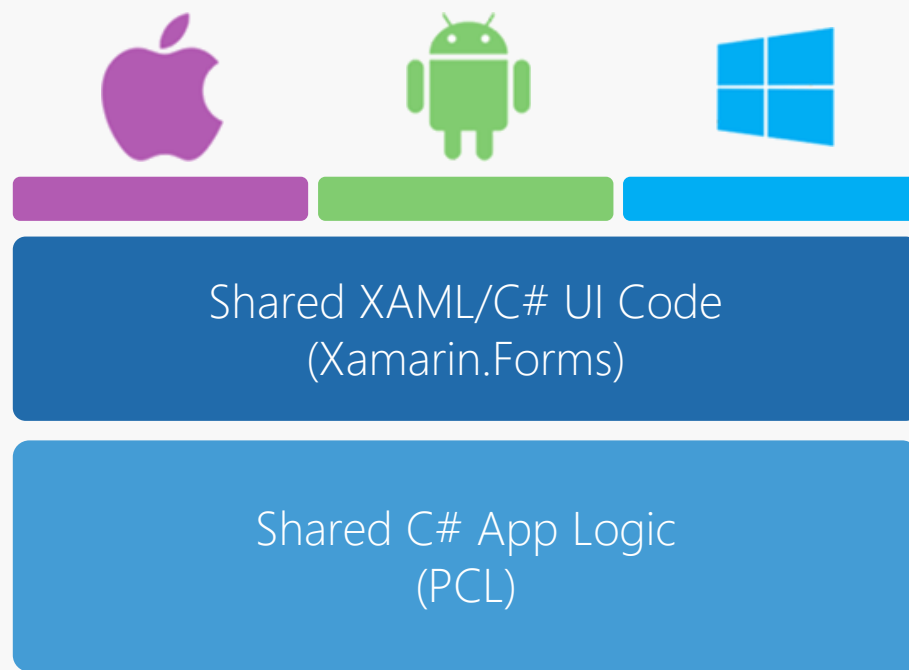
## Xamarin ネイティブ

ロジックのみ共通化  
UIはネイティブで個別に作りこむ



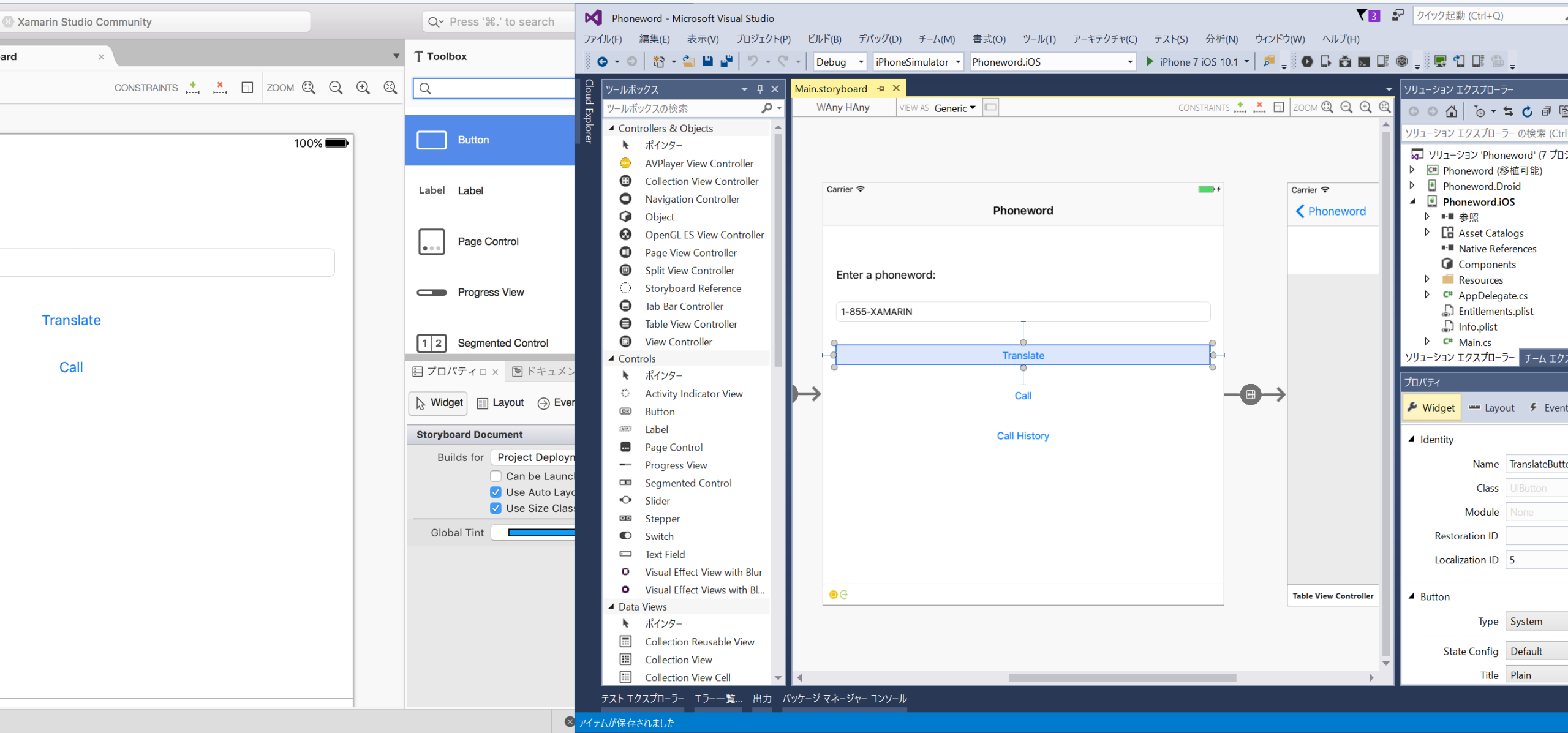
## Xamarin.Forms

ロジックとUIを共通化  
UIは各プラットフォームの  
同じ役割のUIが自動マッピング



# Xamarin.iOS

# Storyboard



# Outlet

ViewController.designer.cs

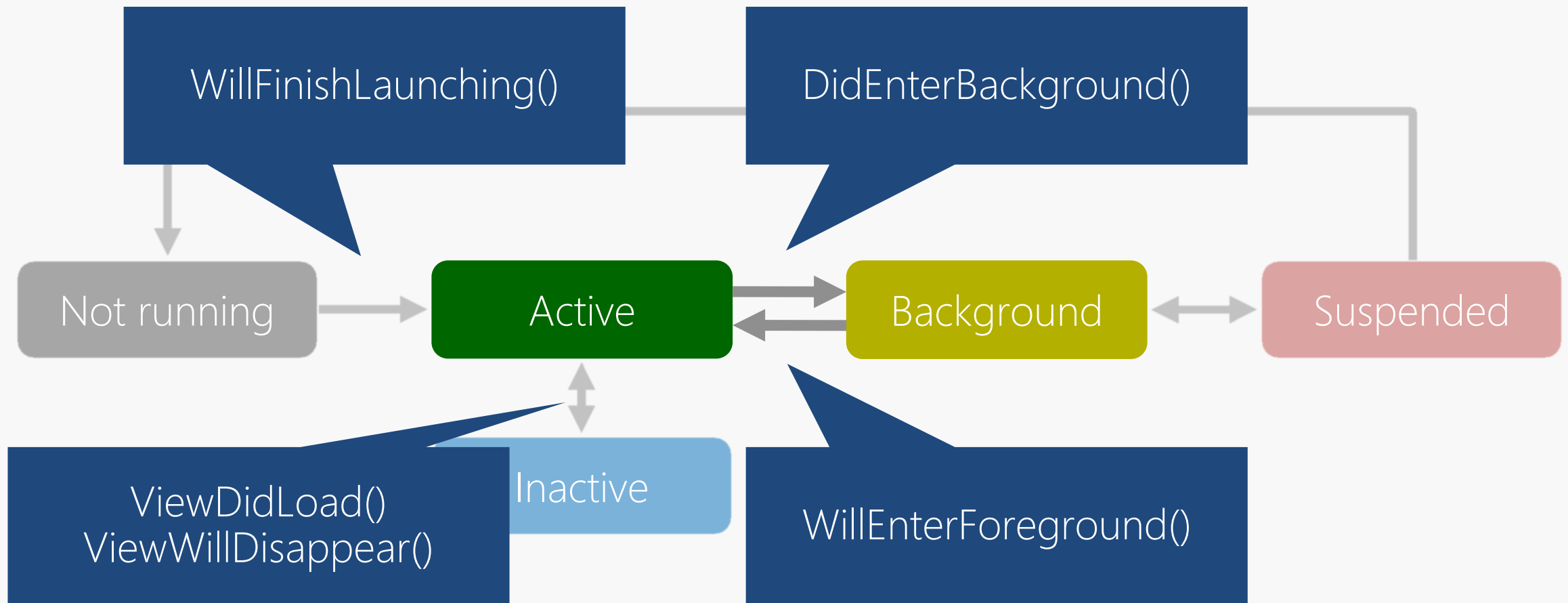
```
[Register ("TodoListViewController")]
partial class TodoListViewController
{
    [Outlet]
    [GeneratedCode ("iOS Designer", "1.0")]
    UIButton LoginButton { get; set; }

    ...
}
```

ViewController.cs

```
LoginButton.TouchUpInside += (object sender,
EventArgs e) => {
    // TODO: add logic here
}
```

# Application Lifecycle



Xamarin.Android

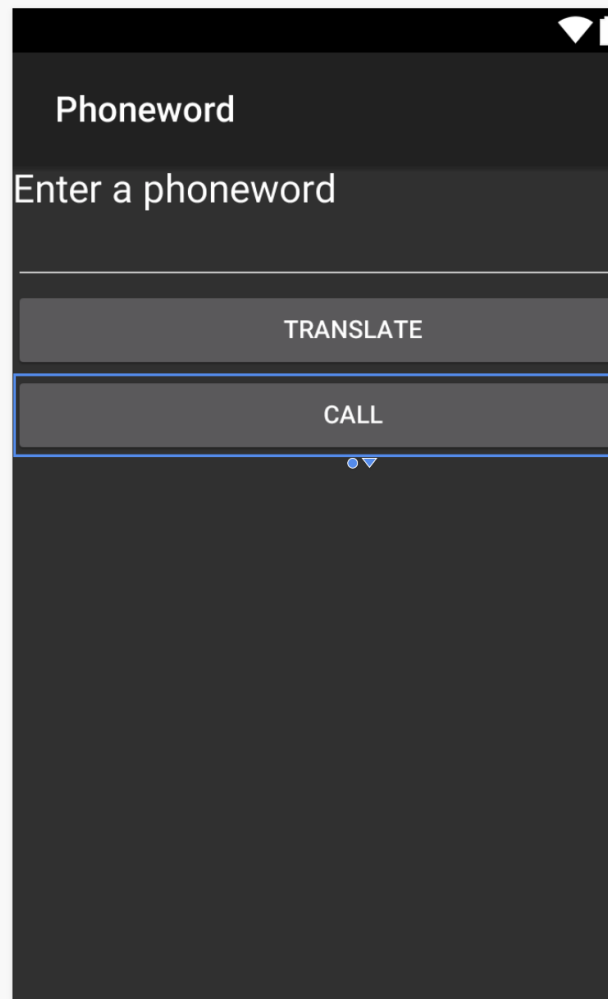
# Layout

iPhone 7 iOS 10.1

Xamarin Studio Community

PhoneTranslator.cs Main.xml

Version: Android 6.0 (v23) Theme: Default Theme



App1 - Microsoft Visual Studio

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) アーキテクチャ(C) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)

Debug Any CPU App1.Droid

x86-Marshmallow (Android 6.0 - API 23)

MainActivity.cs Main.xml\*

Device: Nexus 4 Version: Android 6.0 (v23) Theme: Default Theme

ツールボックス

ツールボックスの検索

- ポインター
- DialerFilter
- Fragment
- GestureOverlayView
- NumberPicker
- SurfaceView
- TextureView
- ToolBar
- TwoLineListItem
- View
- ViewStub
- ZoomButton
- ZoomControls
- Form Widgets
  - ポインター
  - Button
  - CheckBox
  - CheckedTextView
  - Progress Bar (Horizontal)
  - Progress Bar (Large)
  - Progress Bar (Normal)
  - Progress Bar (Small)
  - QuickContactBadge
  - RadioButton
  - RadioGroup
  - RatingBar
  - SeekBar

App1.Droid

Enter a phoneword

TRANSLATE

CALL

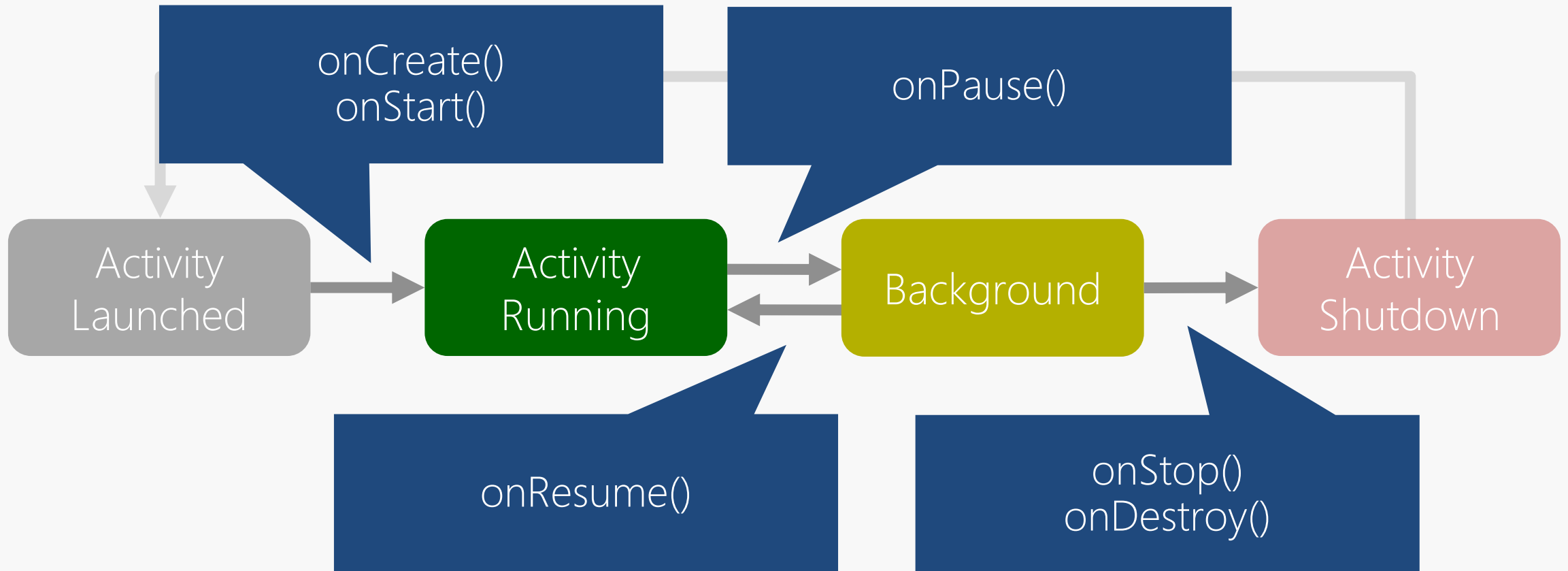


# Intent

```
public class MainActivity : Activity
{
    ...
    void OnClick(object sender, EventArgs e)
    {
        var intent = new Intent(this, typeof(Activity2));

        base.StartActivity(intent);
    }
}
```

# Application Lifecycle



# Xamarin.iOS／Xamarin.Android まとめ

# Xamarin ネイティブ

## iOS

```
namespace XamarinNative.iOS
{
    2 個の参照
    public partial class ViewController : UIViewController
    {
        int count = 1;

        0 個の参照
        public ViewController(IntPtr handle) : base(handle)
        {
        }

        1 個の参照
        public override void ViewDidLoad()
        {
            base.ViewDidLoad();

            Button.TouchUpInside += (sender, e) =>
            {
                var title = string.Format("{0} clicks!",
                    Button.SetTitle(title, UIControlState.Normal);
            }
        }
    }
}
```

## Android

```
namespace XamarinNative.Droid
{
    [Activity(Label = "XamarinNative.Droid", MainLauncher = true)]
    0 個の参照
    public class MainActivity : Activity
    {
        int count = 1;

        1 個の参照
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Main);

            var button = FindViewById<Button>(Resource.Id.button1);
            button.Click += (sender, e) =>
            {
                button.Text = string.Format("{0} clicks!", count);
                count++;
            }
        }
    }
}
```

# Xamarin ネイティブ

UIは個別

ネイティブAPIは個別

PCL／.NET Standard or  
Shared

計算処理

ネットワーク処理

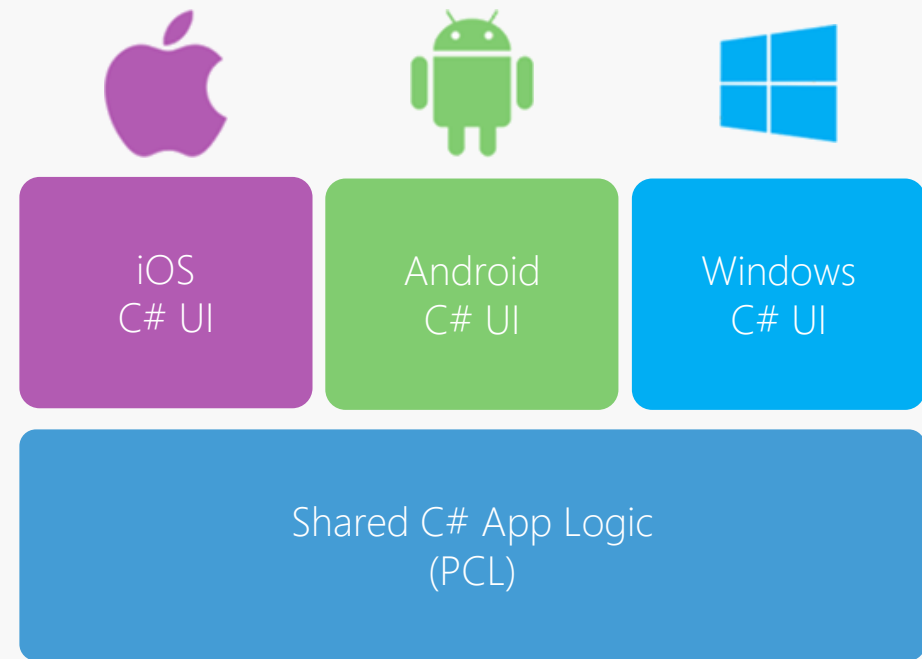
Json, XML などの処理

永続化の処理

## Xamarin Native

ロジックのみ共通化

UIはネイティブで個別に作りこむ



# Xamarin.Forms

# Xamarin.Forms

抽象化UIライブラリ

最大公約数

ワンソース・ネイティブUI/UX

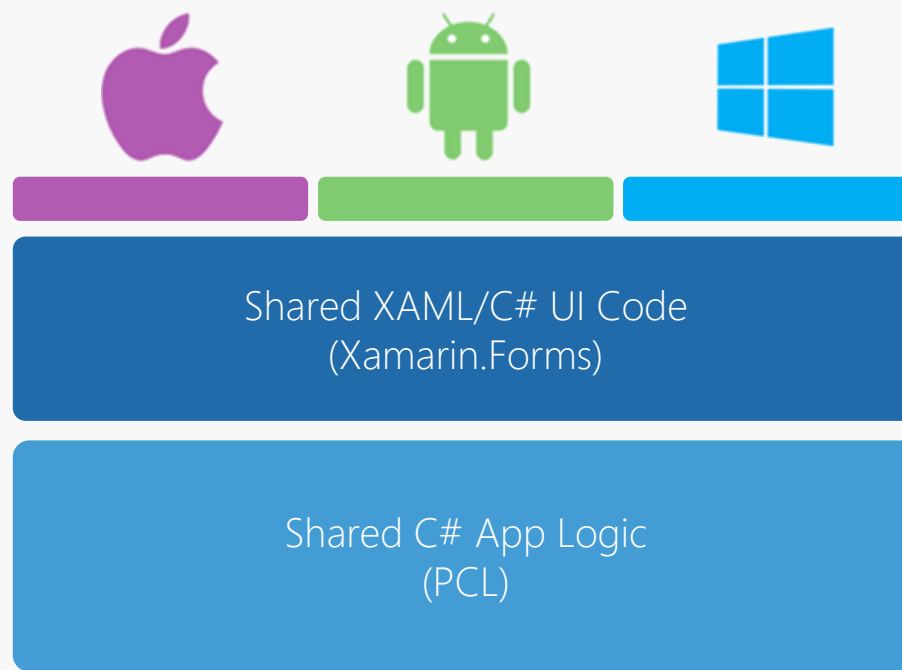
XAML / MVVM

拡張可能

## Xamarin.Forms

ロジックとUIを共通化

UIは各プラットフォームの  
同じ役割のUIが自動マッピング



# Controls

ActivityIndicator

BoxView

Button

DatePicker

Editor

Entry

Image

Label

ListView

Map

OpenGLView

Picker

ProgressBar

SearchBar

Slider

Stepper

Switch

TableView

TimePicker

WebView

EntryCell

ImageCell

SwitchCell

TextCell

ViewCell



# Xamarin.Forms

ワンソース  
ネイティブの  
UI/UX



# XAML

## Xamarin.Forms XAML

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/fo
3             xmlns:x="http://schemas.microsoft.com/win
4             x:Class="XFApp3.XFMainWindow"
5             Title="MainPage">
6     <StackLayout Margin="10"
7                 Spacing="5">
8         <Label Text="{Binding Value}" />
9         <Entry Text="{Binding Value}"
10            Placeholder="input name"/>
11         <StackLayout Orientation="Horizontal"
12                 Spacing="10">
13             <Switch IsToggled="{Binding Toggled}"/>
14             <Label Text="Upper case"
15                 VerticalTextAlignment="Center"/>
16         </StackLayout>
17         <Button Text="Click me!"
18             Command="{Binding GoToCommand}"/>
19     </StackLayout>
20 </ContentPage>
```

## UWP/WPF XAML

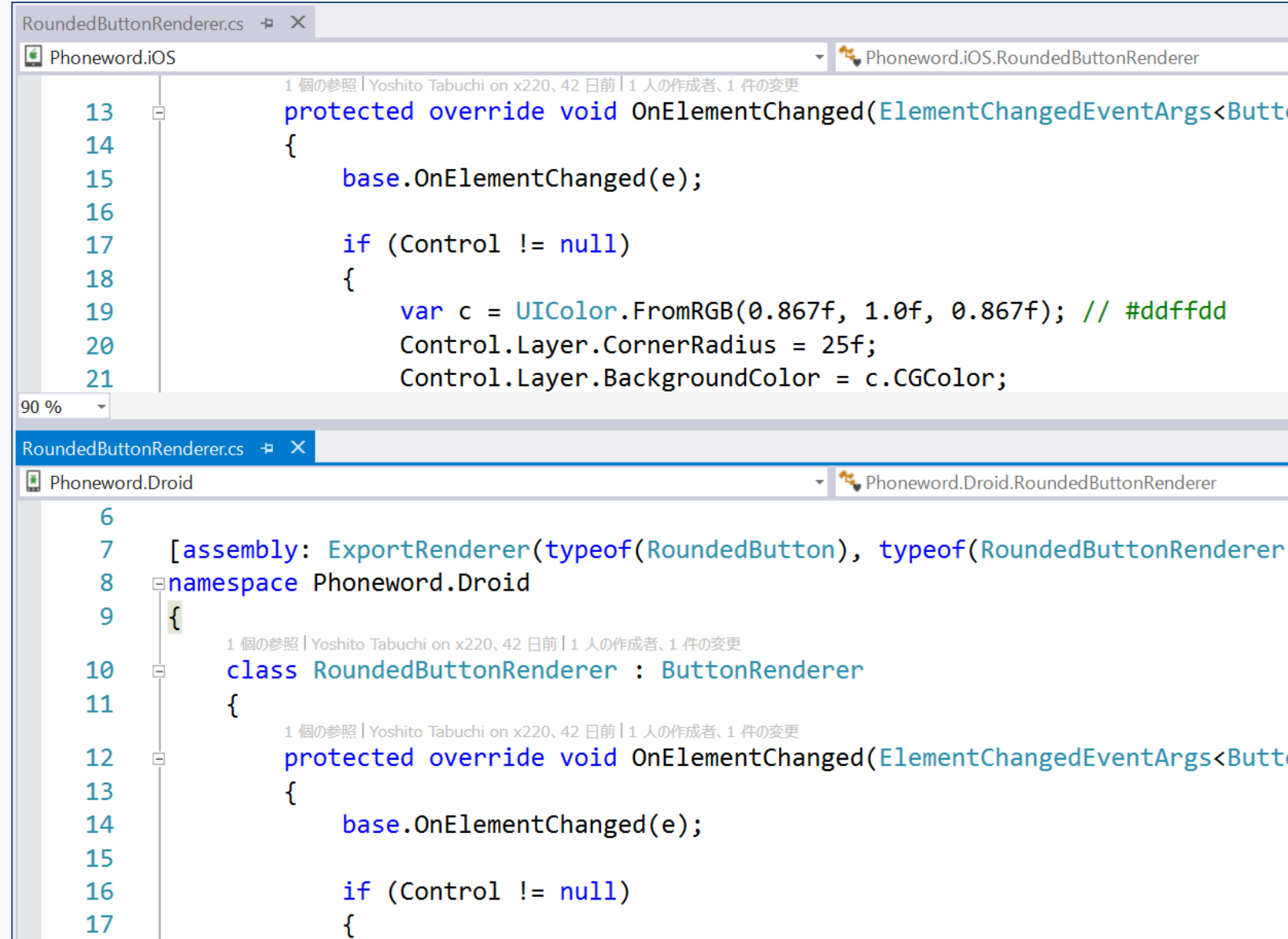
```
8 x:Class= XFApp3.WPF.WPFMainWindow
9 mc:Ignorable="d"
10 Title="MainWindow" Height="350" Width="525">
11
12 <StackPanel Margin="10" >
13     <TextBlock Text="{Binding Value}"
14                 Margin="0, 0, 0, 5"/>
15     <TextBox Text="{Binding Value,
16                 Mode=TwoWay,
17                 UpdateSourceTrigger=PropertyCh
18                 Height="Auto"
19                 Margin="0, 0, 0, 5"/>
20     <CheckBox Content="Upper case"
21                 IsChecked="{Binding Toggled}"
22                 VerticalContentAlignment="Center"
23                 Margin="0, 0, 0, 5"/>
24     <Button Content="Click me!"
25                 Command="{Binding GoToCommand}"
26                 Margin="0, 0, 0, 5"/>
27 </StackPanel>
28
```

# ネイティブコントロール (UI)

Custom Renderer

Effects

Native Embedded



The image shows a screenshot of a Visual Studio code editor with two open files. The top file is 'RoundedButtonRenderer.cs' for the 'Phoneword.iOS' project. It shows a C# class with a 'protected override void OnElementChanged(ElementChangedEventArgs<Button> e)' method. The method calls 'base.OnElementChanged(e);' and then sets the 'Control.Layer.CornerRadius' to 25f and the 'Control.Layer.BackgroundColor' to a color derived from 'UIColor.FromRGB(0.867f, 1.0f, 0.867f); // #ddffdd'. The bottom file is 'RoundedButtonRenderer.cs' for the 'Phoneword.Droid' project. It shows a C# class with a '[assembly: ExportRenderer(typeof(RoundedButton), typeof(RoundedButtonRenderer))]' attribute, a 'namespace Phoneword.Droid' declaration, and a 'class RoundedButtonRenderer : ButtonRenderer' declaration. The 'OnElementChanged' method is also shown, starting with 'protected override void OnElementChanged(ElementChangedEventArgs<Button> e)' and 'base.OnElementChanged(e);'. The code is written in C# and uses standard syntax for Xamarin.iOS and Xamarin.Droid projects.

```
RoundedButtonRenderer.cs
Phoneword.iOS
1 個の参照 | Yoshito Tabuchi on x220, 42 日前 | 1 人の作成者、1 件の変更
13 protected override void OnElementChanged(ElementChangedEventArgs<Button> e)
14 {
15     base.OnElementChanged(e);
16
17     if (Control != null)
18     {
19         var c = UIColor.FromRGB(0.867f, 1.0f, 0.867f); // #ddffdd
20         Control.Layer.CornerRadius = 25f;
21         Control.Layer.BackgroundColor = c.CGColor;
90 %

RoundedButtonRenderer.cs
Phoneword.Droid
1 個の参照 | Yoshito Tabuchi on x220, 42 日前 | 1 人の作成者、1 件の変更
6
7 [assembly: ExportRenderer(typeof(RoundedButton), typeof(RoundedButtonRenderer))]
8 namespace Phoneword.Droid
9 {
10     1 個の参照 | Yoshito Tabuchi on x220, 42 日前 | 1 人の作成者、1 件の変更
11     class RoundedButtonRenderer : ButtonRenderer
12     {
13         1 個の参照 | Yoshito Tabuchi on x220, 42 日前 | 1 人の作成者、1 件の変更
14         protected override void OnElementChanged(ElementChangedEventArgs<Button> e)
15         {
16             base.OnElementChanged(e);
17
18             if (Control != null)
19             {
```

# ネイティブAPI

## Dependency Service Plugin

PhoneDialer.cs

Phoneword.iOS

Phoneword.iOS.PhoneDia

Dial(string number)

```
1 using Foundation;
2 using Phoneword.iOS;
3 using UIKit;
4 using Xamarin.Forms;
5
6 [assembly: Dependency(typeof(PhoneDialer))]
7
8 namespace Phoneword.iOS
9 {
10     1 個の参照 | Yoshito Tabuchi on x220, 56 日前 | 1 人の作成者、1 件の変更
11     public class PhoneDialer : IDialer
12     {
13         0 個の参照 | Yoshito Tabuchi on x220, 56 日前 | 1 人の作成者、1 件の変更
14         public bool Dial(string number)
15         {
16             return UIApplication.SharedApp!
17                 .OpenUrl(new NSURL("tel:" + number));
18         }
19     }
20 }
```

PhoneDialer.cs

Phoneword.Droid

Phoneword.Droid

```
1 using Android.Content;
2 using Android.Telephony;
3 using Phoneword.Droid;
4 using System.Linq;
5 using Xamarin.Forms;
6
7 using Uri = Android.Net.Uri;
8
9 [assembly: Dependency(typeof(PhoneDialer))]
10
11 namespace Phoneword.Droid
12 {
13     1 個の参照 | Yoshito Tabuchi on x220, 56 日前 | 1 人の作成者、1 件の変更
14     public class PhoneDialer : IDialer
15     {
16         0 個の参照 | Yoshito Tabuchi on x220, 56 日前 | 1 人の作成者、1 件の変更
17         public bool Dial(string number)
18         {
19             var context = Android.App.Context;
20             if (context == null)
21                 return false;
22
23             var intent = new Intent(context, typeof(IntentDialer));
24             intent.SetData(Uri.Parse("tel:" + number));
25             if (Intent.Resolve(context, intent) != null)
26             {
27                 context.StartActivity(intent);
28                 return true;
29             }
30             return false;
31         }
32     }
33 }
```

# Xamarin 特徴まとめ

# 必要な知識

	API	UI toolkit	言語	統合開発環境
プラットフォーム 個別	iOS API		Objective-C, Swift	Xcode
	Android API		Java	Android Studio
	Windows API		C#	Visual Studio
Xamarin Native	iOS API		Objective-C, Swift	Xcode
	Android API		Java	Android Studio
	Windows API		C#	Visual Studio
Xamarin.Forms	iOS API		Objective-C, Swift	Xcode
	Android API		Java	Android Studio
	Windows API	Xamarin.Forms	C#	Visual Studio

# Xamarin.Forms vs Xamarin ネイティブ

C#er/XAMLer/WPF/UWP → Forms

iOS/Android ネイティブ経験者 → ネイティブ

素早く簡単に作る → Forms

きれいに細かく作る → ネイティブ

社内プロジェクト → Forms

受託開発 → ??

HTML/JS → Cordova? React Native?

# トピック

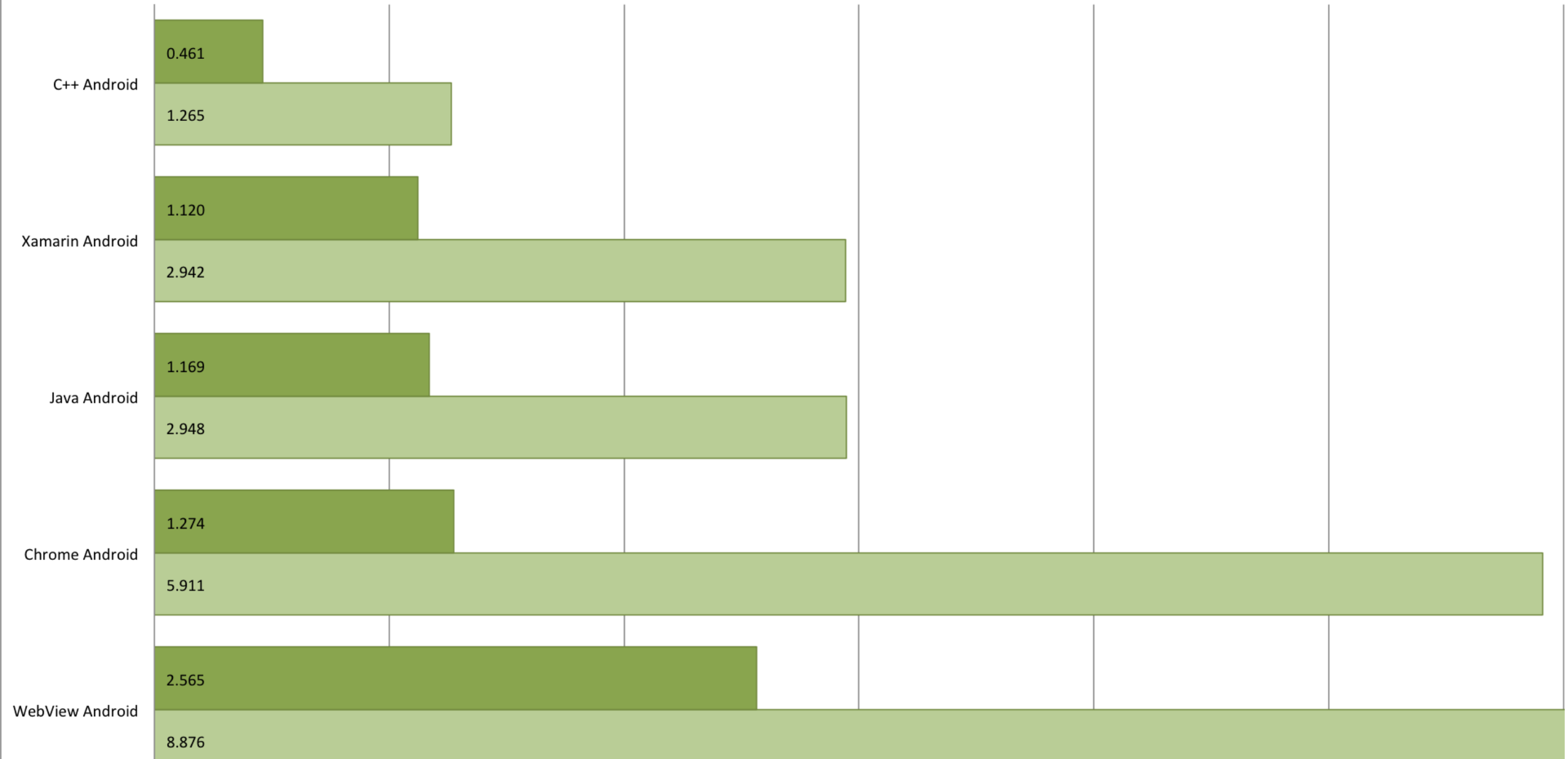
- Xamarinの利点／欠点は
  - 利点：API 100%、Xamarin.Forms で C# の資産（知識）を活用、Xamarin.Forms での素早い開発、Xamarin ネイティブでの詳細な開発
  - 欠点：習得コストの高さ、メモリ管理、アプリサイズ、まったく同じ画面を用意する
- 性能比較（ネイティブ／Xamarin）
  - [Mobile App Performance Redux – Harry Cheung – Medium](#)
  - [Xamarin Performance vs iOS and Android Native Apps Compared](#)
- Web（HTML+Javascript）系との違い
  - 開発者層、アプリの速さ、API の提供率
- Xamarin.Forms 使える？
  - 利用用途を考えれば使える



# Android App Performance

(in seconds - smaller is better)

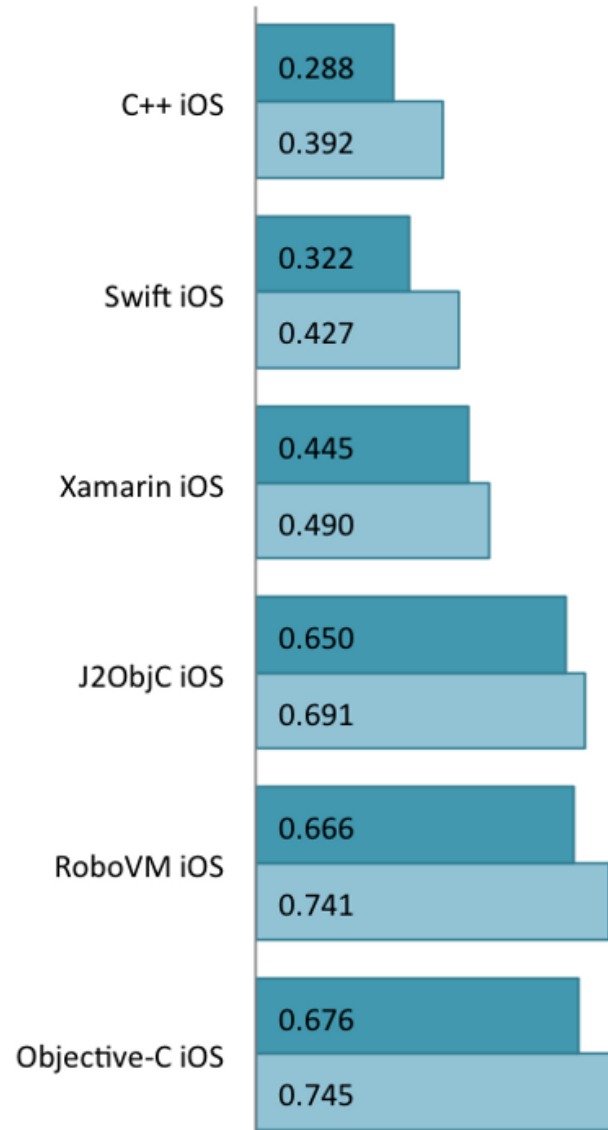
■ HTC Nexus 9 ■ Moto X (2014)



# iOS App Performance

(in seconds - lower is better)

■ iPad Air 2   ■ iPhone 6



事例

# NHK 紅白

## フェンリル株式会社 様

[http://biz.fenrir-inc.com/application\\_development/casestudy\\_app/nhk\\_kouhaku.html](http://biz.fenrir-inc.com/application_development/casestudy_app/nhk_kouhaku.html)

Xamarin ネイティブ製

Reactive Property / Reactive Extensions / MVVMLight Toolkit / PCL Storage / Json.NET / FFImageLoaging Plugin / MBProgressHUD など多数の OSS



# スピードアンサー15

## 株式会社三井住友銀行様

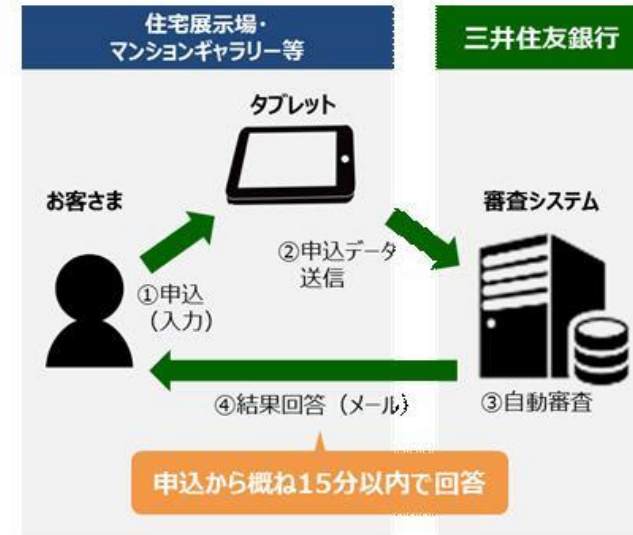
[http://ytabuchi.hatenablog.com/entry/casestudy\\_smbc](http://ytabuchi.hatenablog.com/entry/casestudy_smbc)

Xamarin ネイティブ製

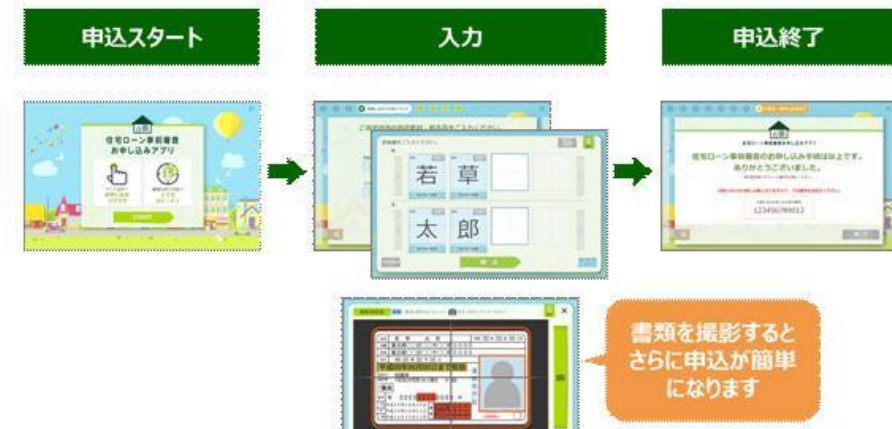
Windowsタブレット端末、iPad、PC  
手書き入力／身分証明書のOCR／  
印鑑の代わりにサイン（Signature  
Pad）

<ご参考>

1. 「スピードアンサー15」申込フロー イメージ図

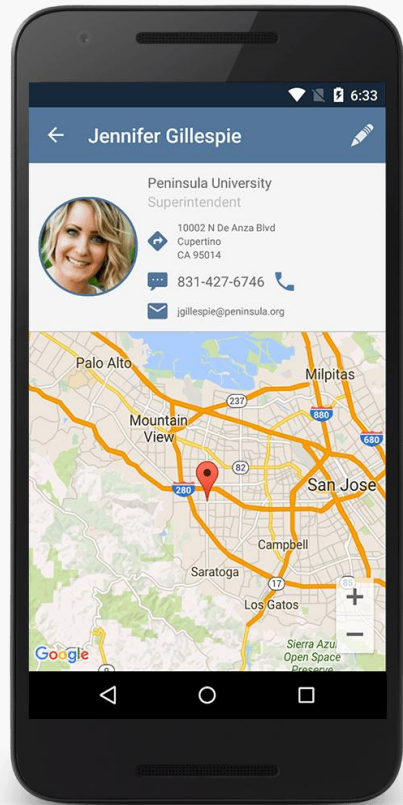


2. 画面展開 イメージ図



# Prebuilt サンプル

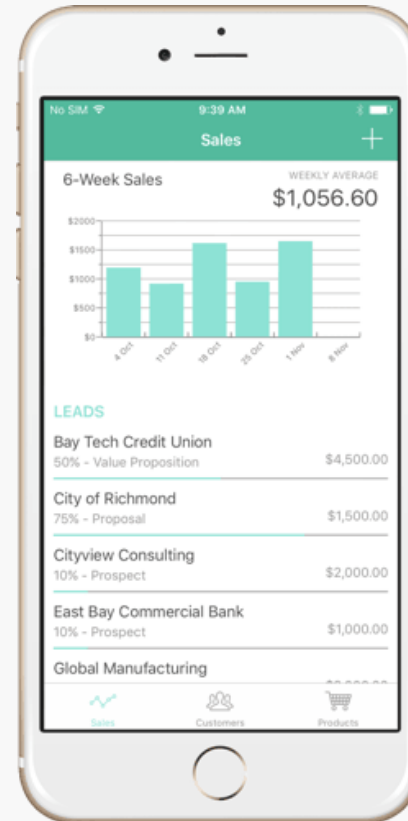
<https://www.xamarin.com/prebuilt>



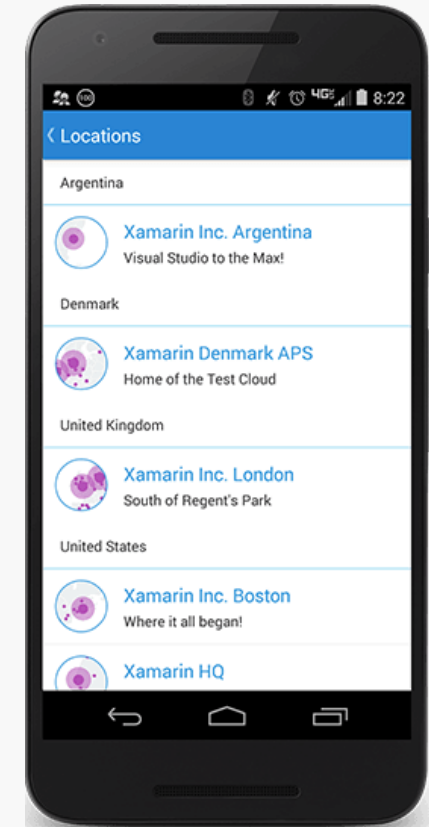
Acquaint



Sport



Xamarin CRM



My Shoppe

まとめ

# Xamarin

C#／.NET／Visual Studio

フル “ネイティブ” アプリ

API 100% 移植

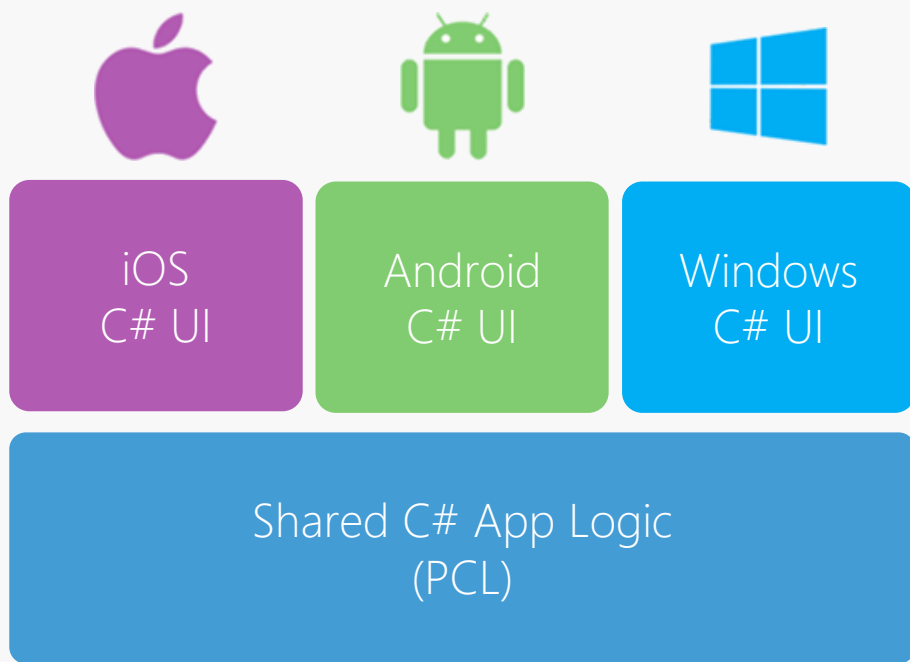
コード共通化



# 2つの開発手法

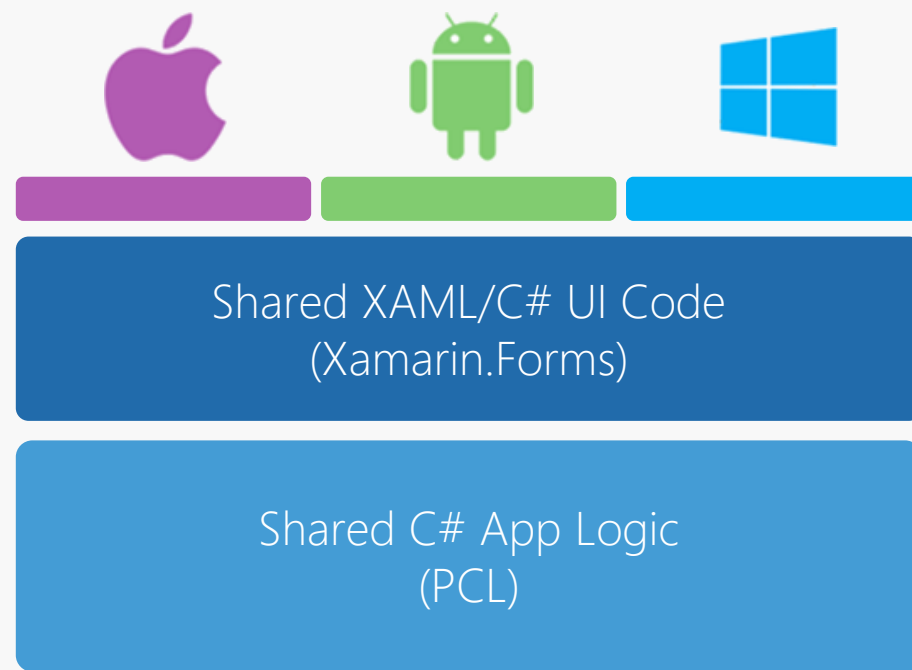
## Xamarin Native

ロジックのみ共通化  
UIはネイティブで個別に作りこむ



## Xamarin.Forms

ロジックとUIを共通化  
UIは各プラットフォームの  
同じ役割のUIが自動マッピング



# 付録：Xamarin をはじめよう

# 丁寧に環境構築

Visual Studio 2017 なら、インストールするだけ！

<http://ytabuchi.hatenablog.com/entry/visualstudio2017>

Android SDK をちゃんとインストール

Intel HAXM の x86 Emulator を使用する

# 日本語情報

[Xamarin 逆引き Tips - Build Insider](#)

[Xamarin に関する投稿 - Qiita](#)

[Xamarin Advent Calendar](#)

[YouTube の JXUG チャンネル](#)

[Insider.NET > .NET TIPS - @IT](#)

[JXUG : 関連ページ、ブロッガー一覧](#)

[Xamarin 日本語ドキュメントの紹介 : XLsoft エクセルソフト](#)

# 書籍

プログラミング Xamarin

Xamarin ネイティブによるモバイルアプリ開発

Essential Xamarin

Xamarinプログラミング入門 C#によるiOS、Androidアプリケーション開発の基本

基礎から学ぶ Xamarinプログラミング

かずきのXamarin.Forms入門

# 手を動かす

Xamarin ハンズオン (初級)

Xamarin.Android ListView ハンズオン (初級)

Xamarin Dev Days Tokyo ハンズオン (中級)

Xamarin & Microsoft Cognitive Services ハンズオン (中級)

Xamarin.Forms CustomRenderer ハンズオン (中級)

Swift を Xamarin.iOS に移植ハンズオン (中級)

Java を Xamarin.Android に移植ハンズオン (中級)

Xamarin.Forms & Prism.Forms、Moq ハンズオン (上級)

JXUG で主催しているハンズオンやもくもく会に参加

<http://jxug.connpass.com>

聞く

Teratail

Facebook の JXUG グループ

Twitter (#Xamarin #JXUG タグで呟く)

# 英語情報

読む・見る・聞く・調べる

[Guides - Xamarin](#) (ドキュメント)

[Recipes - Xamarin](#) (逆引き辞典)

[Xamarin Blog](#)

[Xamarin channel - Youtube](#) (セッション動画)

[Xamarin Forums](#)

[Stackoverflow](#)

Prebuilt アプリを読む





# 有償トレーニング

## 有償トレーニング | Xamarin : XLsoft エクセルソフト

カスタマイズも可能です。

Xamarin を使用した iOS、Android クロスプラットフォーム対応のモバイルアプリの開発トレーニング（初級 2日コース）

Xamarin.Forms による iOS/Android/UWP アプリ構築トレーニング（初級 1日コース）

Xamarin.iOS による iOS アプリ構築トレーニング（初級 1日コース）

Xamarin.Android による Android アプリ構築トレーニング（初級 1日コース）

Xamarin.Forms Custom Renderer 活用トレーニング（中級 1日コース）

ありがとうございます

エクセルソフト株式会社  
ソフトウェア事業部  
新規事業開発室室長  
田淵義人

Twitter: [@ytabuchi](https://twitter.com/ytabuchi)  
facebook: [ytabuchi.xlsoft](https://facebook.com/ytabuchi.xlsoft)

