

Xamarin基礎講座

Xamarinハンズオン



JAPAN
XAMARIN
USER
GROUP

Japan Xamarin User Group

田淵 義人

080-7015-3586

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

facebook: [xlsoft.ytabuchi](https://www.facebook.com/xlsoft.ytabuchi)

Blog: Xamarin日本語情報

自己紹介

田淵義人@エクセルソフト

Xamarin コミュニティエバンジェリスト

2016年4月 Microsoft MVP Visual Studio and Development Technologies 受賞

連載・執筆

Build Insider, マイナビニュース

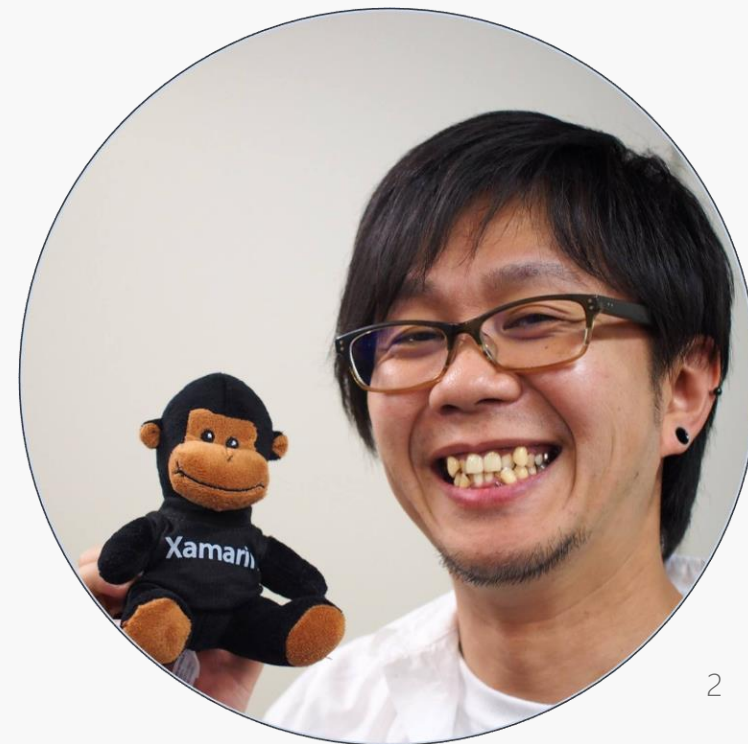
.NET開発テクノロジー入門2016年版 (Xamarinの章)

コミュニティ

Twitter: @ytabuchi

facebook: ytabuchi.xlsoft

Blog: Xamarin 日本語情報



本日のスケジュール

第1部 30分 + 90分

Xamarin 概要

Android, iOS 概要

Xamarin ネイティブ

第2部 30分 + 90分

Xamarin.Forms

まとめ

LT大会 & キャッチアップ 30分

資料

<http://github.com/ytabuchi/XamarinHOL>

Xamarin (ザマリン・企業)

Miguel, Nat

Mono, Ximian

Novell, Attachmate

Microsoft

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

C#

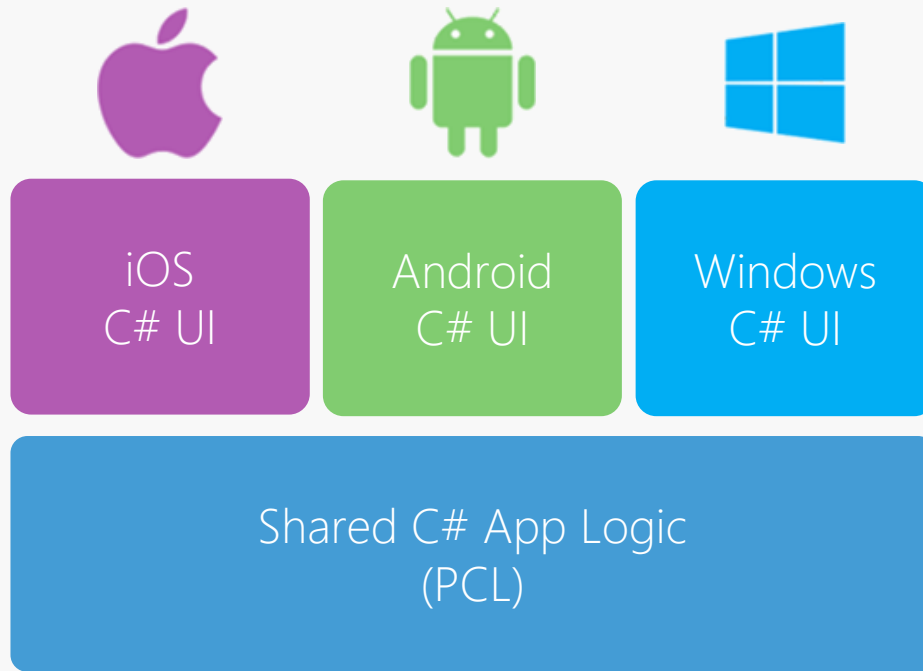
```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Xml.Serialization;

button.Click += async (sender, e) =>
{
    using (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

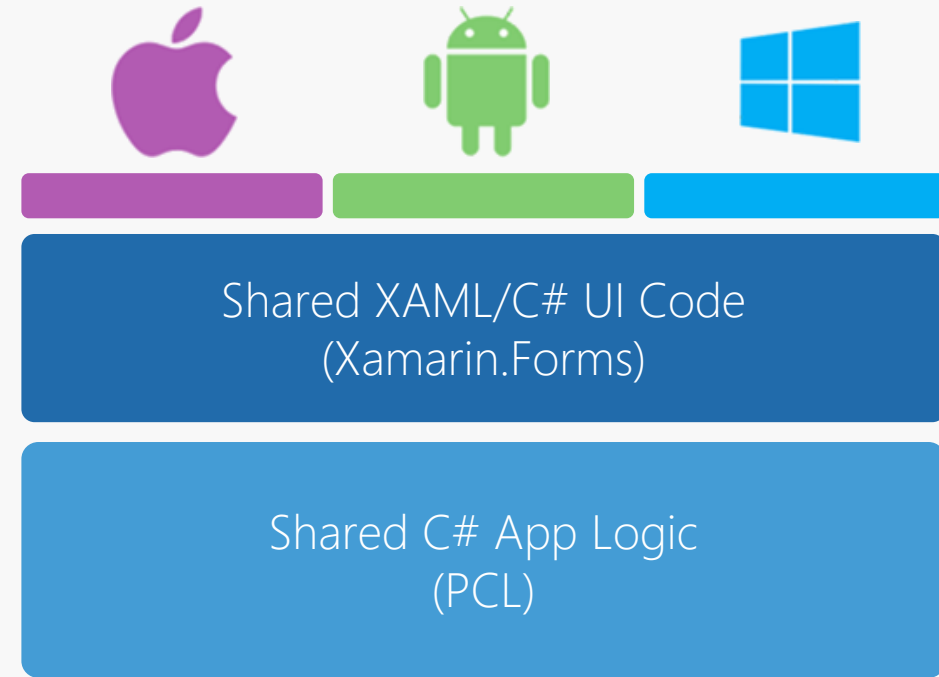
Xamarin Native

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



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

必要なライセンス

Visual Studio Community の利用条件：

<https://www.microsoft.com/ja-jp/dev/products/community.aspx>

Edition	ライセンス形態		iOS	価格	契約単位	年額
Community	-		○	無料	-	-
Professional	スタンドアロン		×	62,383円	買切り	-
	クラウド サブスクリプション	月単位	×	4,590円	月契約	55,080円
		年単位	○	54,978円	1年契約	54,978円
Professional with MSDN	標準 サブスクリプション	MS Store	○	149,877円	1年契約	149,877円
		Open Business	○	150,000円	2年契約	75,000円
		Open Value	○	211,500円	3年契約	70,500円
Enterprise	クラウド サブスクリプション	月単位	×	25,500円	月契約	306,000円
		年単位	○	305,898円	1年契約	305,898円
Enterprise with MSDN	標準 サブスクリプション	MS Store	○	749,876円	1年契約	749,876円
		Open Business	○	1,180,000円	2年契約	590,000円
		Open Value	○	1,390,500円	3年契約	463,500円

<http://nuits.hatenadiary.jp/entry/2016/05/06/174037>

Xamarin.Android

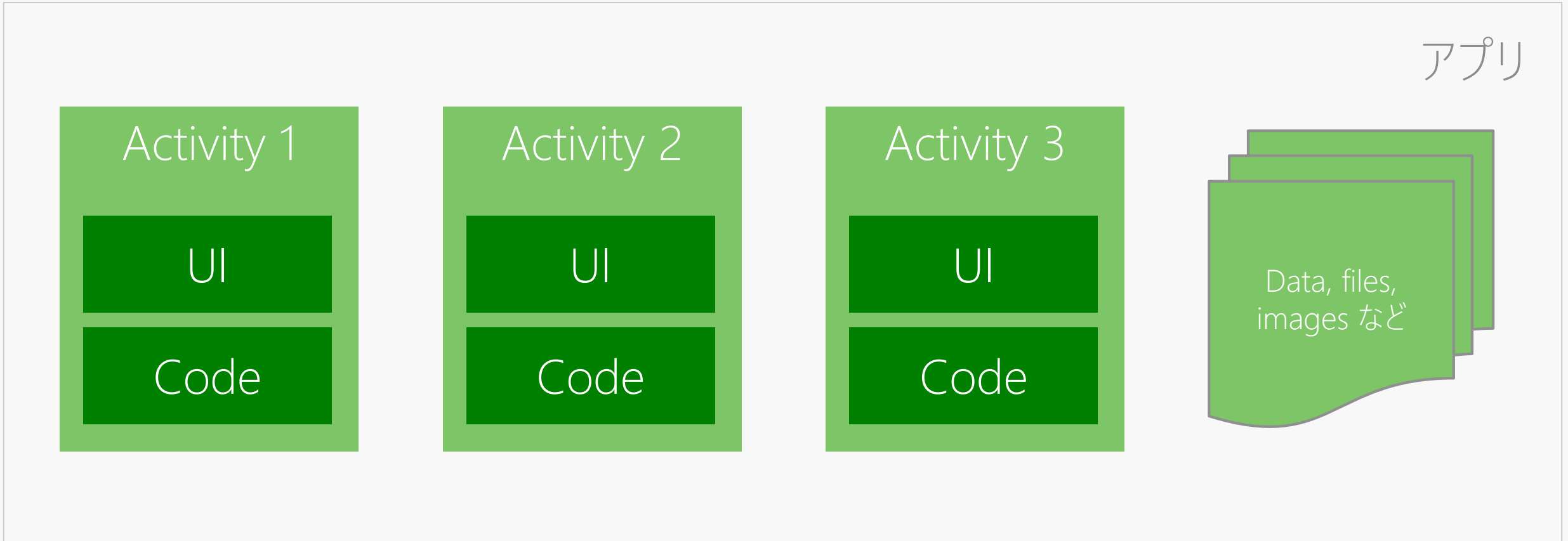
構成

ソースファイル
(C#)

UI 定義
(axml)

メタデータ
(Resources)

Activity

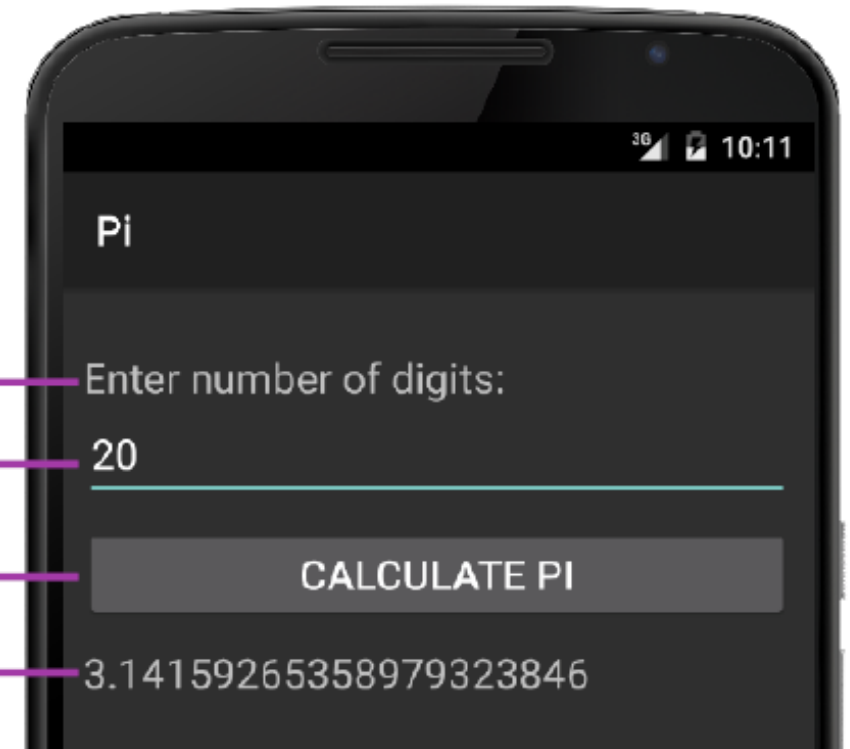


Layout

Child views are
nested inside a
layout panel →

Pi.xml

```
<LinearLayout ... >
  <TextView ... />
  <EditText ... />
  <Button ... />
  <TextView ... />
</LinearLayout ... >
```



Layout

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)

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

MainActivity.cs Main.xml*

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

App1.Droid

Enter a phoneword

TRANSLATE

CALL

ソリューション エクスプローラー

ソリューション エクスプローラーの検索 (Ctrl+)

- layout
 - Main.xml
- values
 - AboutResources.txt
 - Resource.designer.cs
- MainActivity.cs
- App1.iOS
- App1.WinPhone (Windows Phone 8.1)
 - Properties
 - 参照
 - Assets
 - App.xml
 - MainPage.xml
 - Package.appxmanifest

ソリューション エクスプローラー チーム エクスプローラー

プロパティ

button1 android.widget.Button

layout_marginTop	
layout_width	match_parent
Main	
hint	
id	@+id/CallButton
style	
tag	
text	Call
Main - Design-time	
tools:text	

Activity + Layout

Pi.xml

```
<LinearLayout ... >  
  <TextView ... />  
  <EditText ... />  
  <Button ... />  
  <TextView ... />  
</LinearLayout>
```

PiActivity.cs

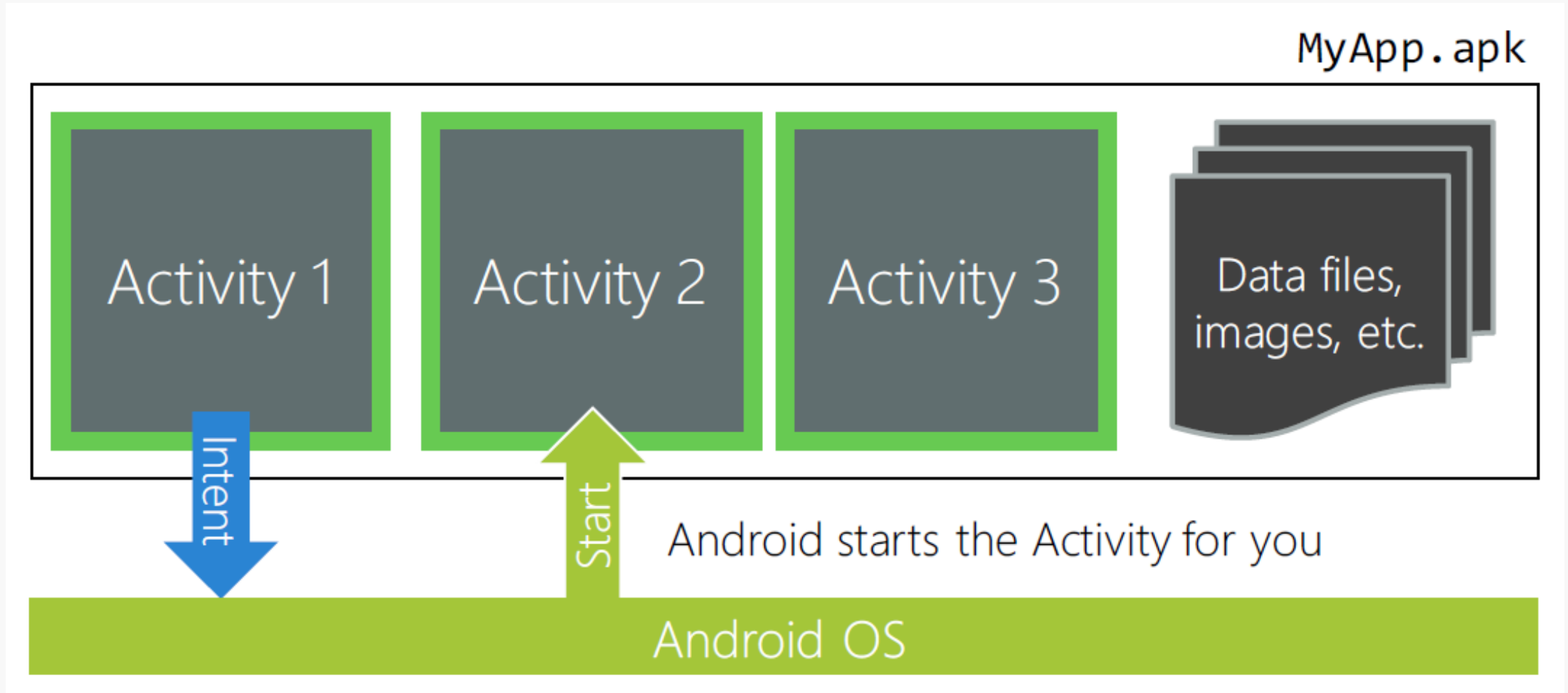
```
[Activity]  
public class PiActivity : Activity  
{  
    ...  
    ...  
}
```

Resource Id

```
[Activity(MainLauncher = true)]
public class MainActivity : Activity
{
    protected override void onCreate(Bundle bundle)
    {
        base.onCreate(bundle);
        SetContentView(Resource.Layout.Main);

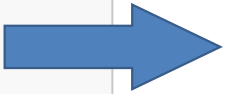
        var et = FindViewById<EditText>(Resource.Id.digitsInput);
        ...
    }
    ...
}
```

Intent



Intent

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



Intent

Explicit
creation



```
var bundle = new Bundle();  
bundle.PutInt("ContactId", 123456789);  
  
var intent = new Intent();  
intent.PutExtras(bundle);
```

Convenience
methods



```
var intent = new Intent();  
intent.PutExtra("ContactId", 123456789);
```

Navigation



Xamarin.iOS

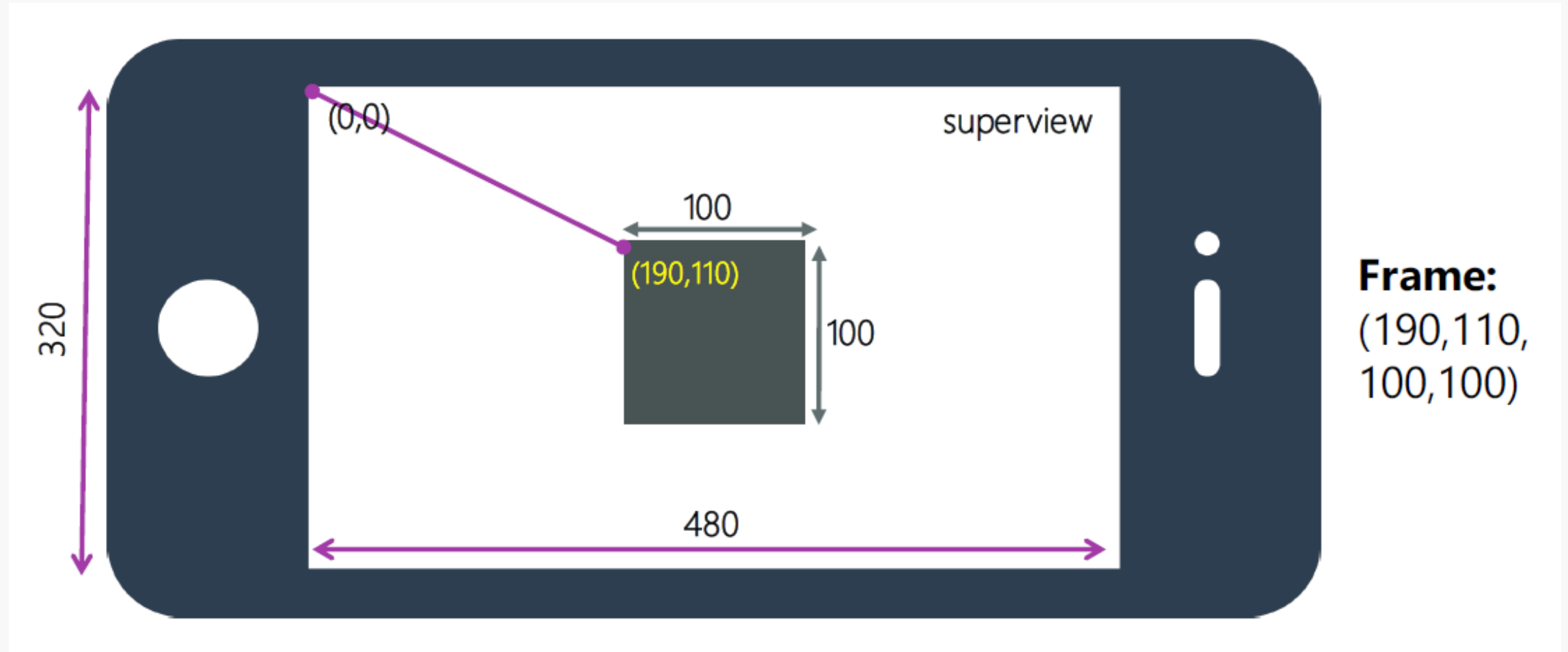
構成

ソースファイル
(C#)

UI 定義
(Storyboard + XiB)

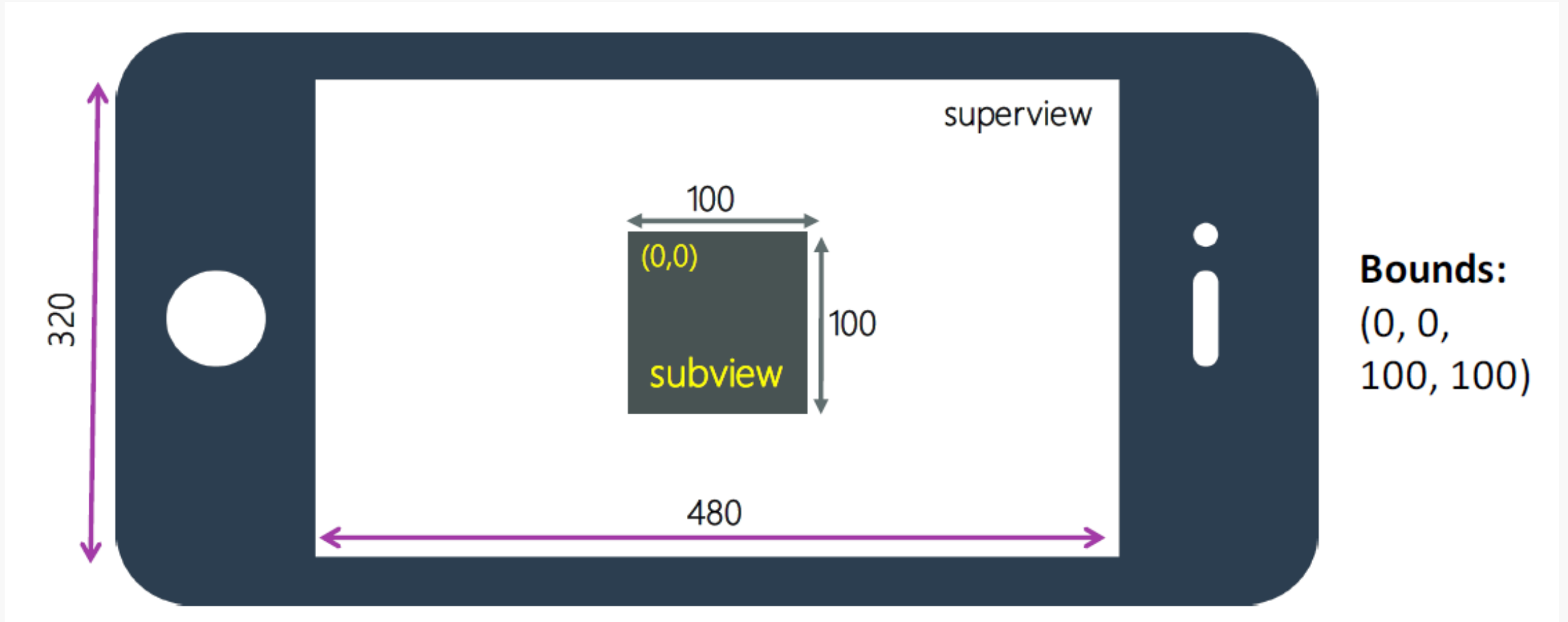
メタデータ
(property lists)

Frame



Frame:
(190,110,
100,100)

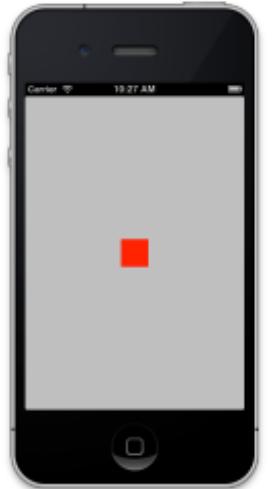
Bounds



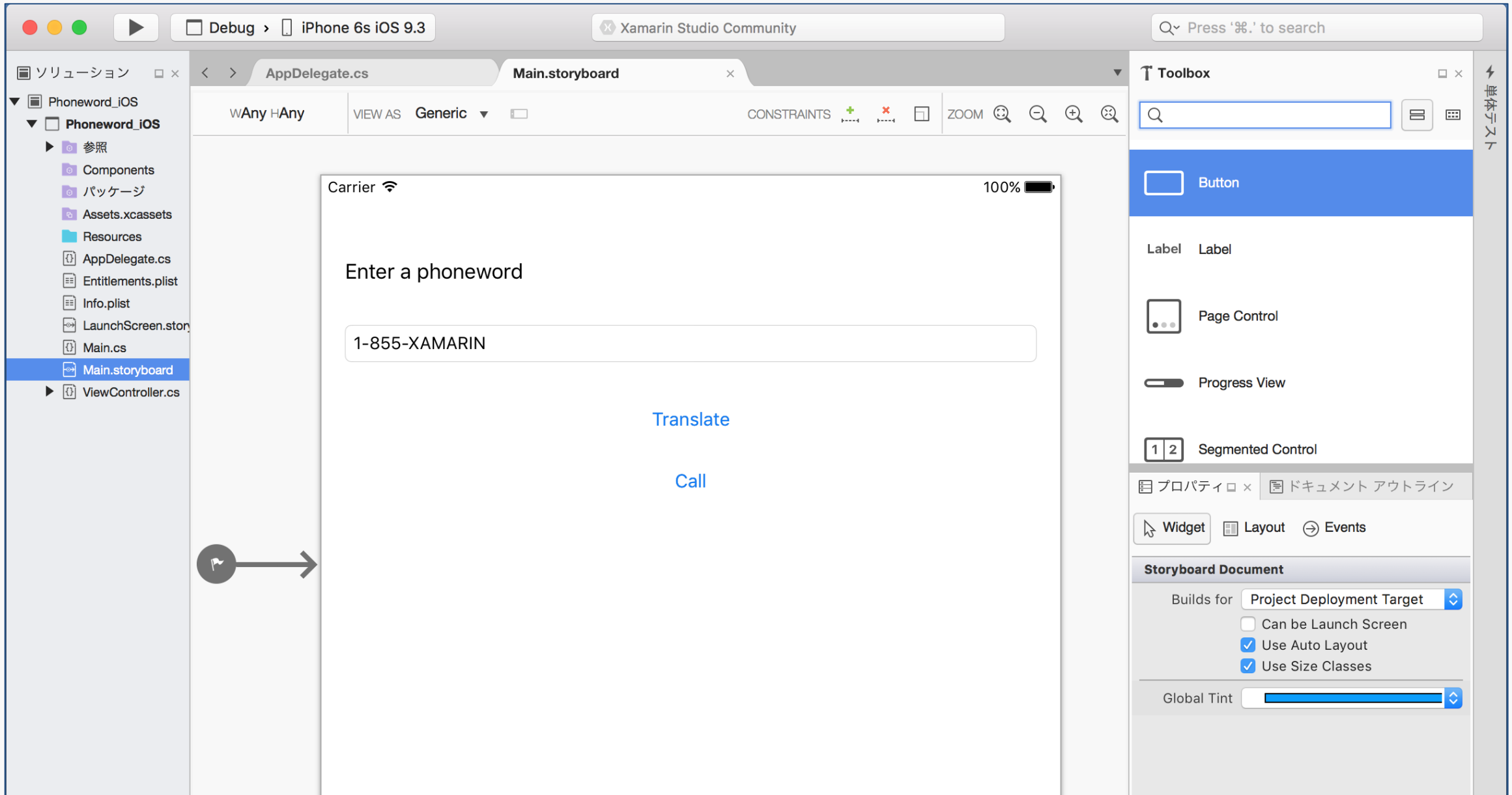
View (コードで)

```
public override void ViewDidLoad()
{
    nfloat height = View.Bounds.Height; // Current view coordinates
    nfloat width = View.Bounds.Width;

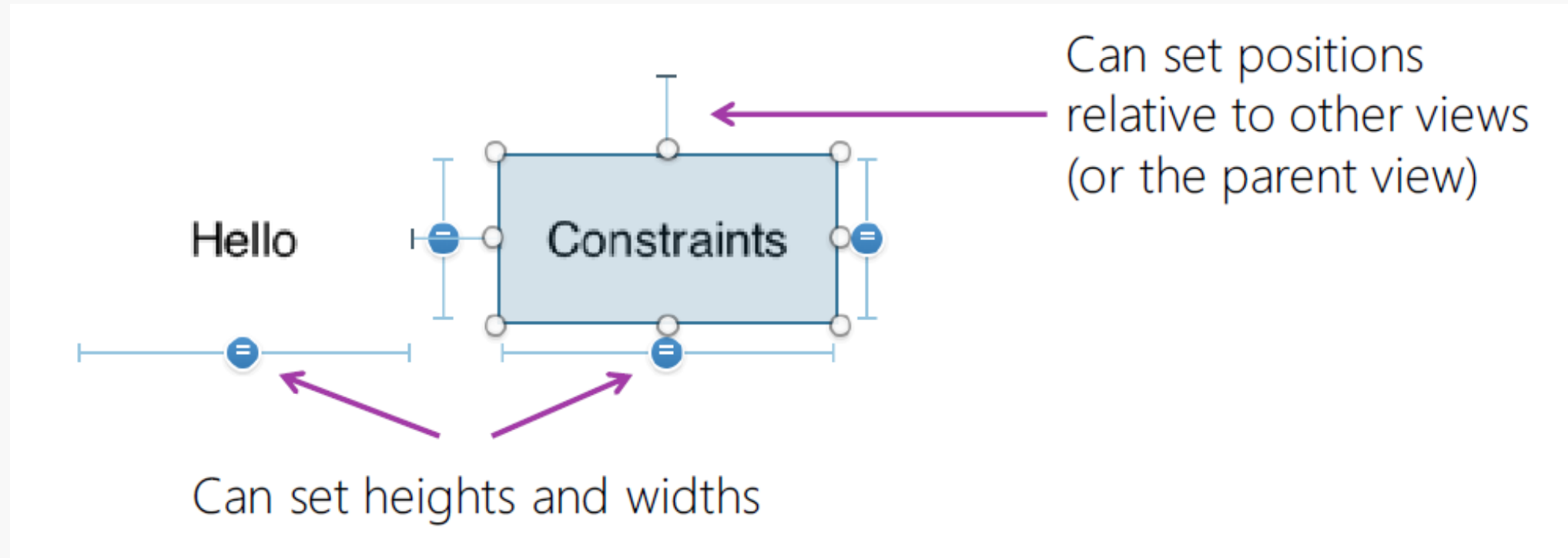
    var subview = new UIView() {
        Frame = new CGRect(width/2-20, height/2-20, 40,40)
    };
    ...
}
```



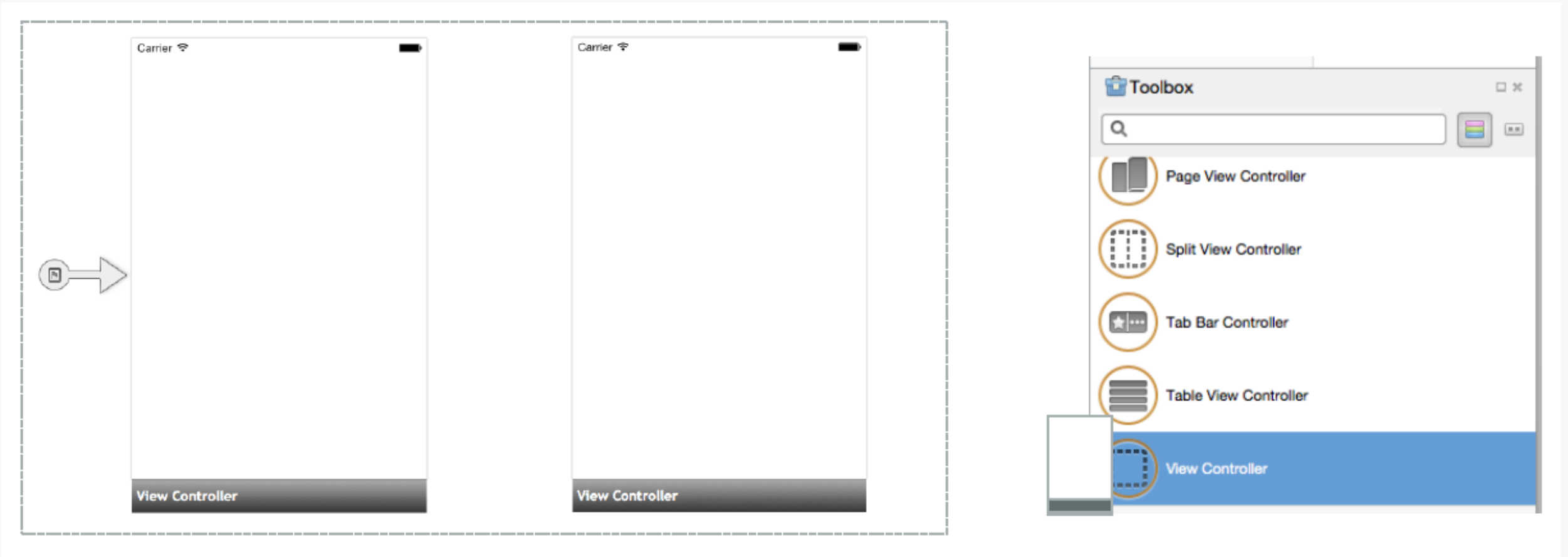
Storyboard



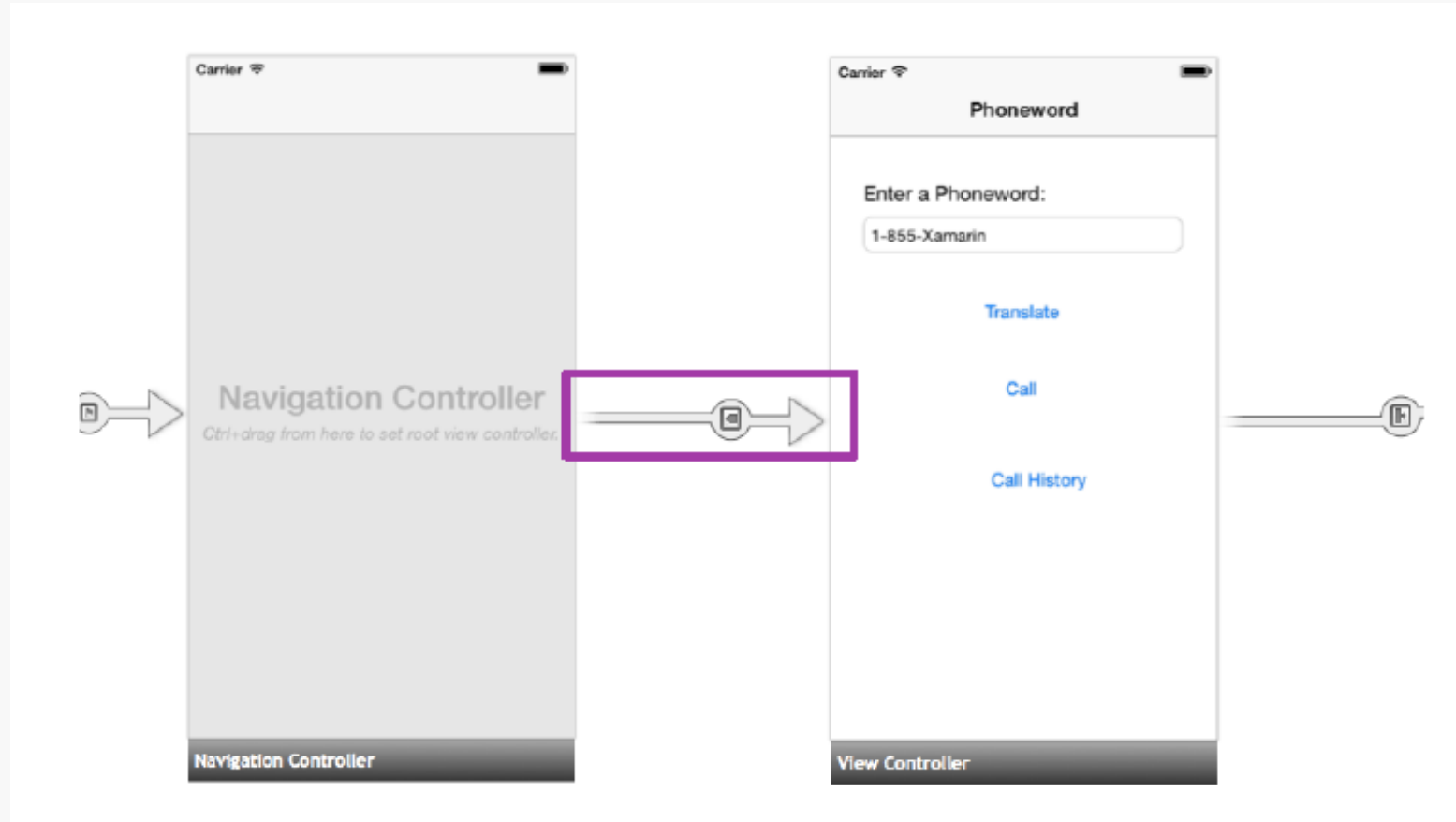
Constraints (制約)



Multi Screen



Segue



Segue

```
partial void ShowAboutPage(UIButton sender)
{
    this.PerformSegue("AboutSegue", this);
}
```

Takes the identifier of the segue

.. And the sender

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 vs Shared

ネットワーク

Json, XML

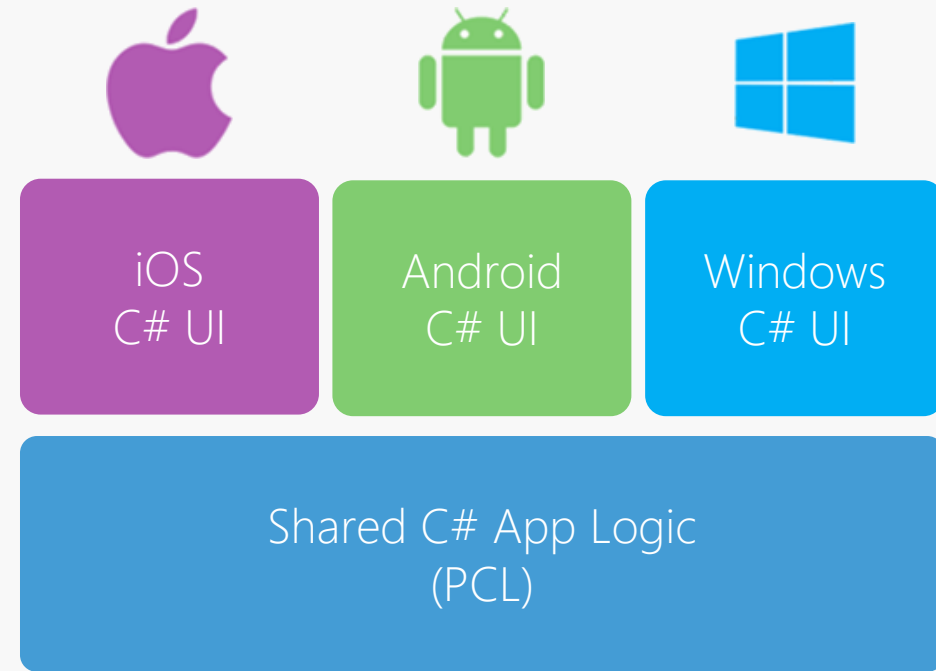
永続化

async/await

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

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

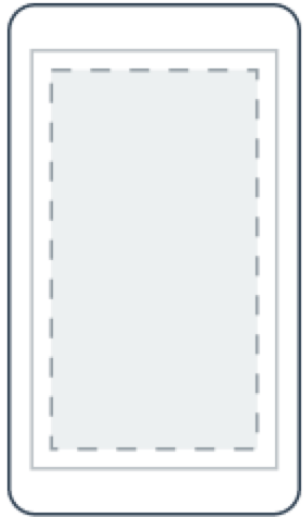
UIは各プラットフォームの
同じ役割のUIが自動マッピング



Shared XAML/C# UI Code
(Xamarin.Forms)

Shared C# App Logic
(PCL)

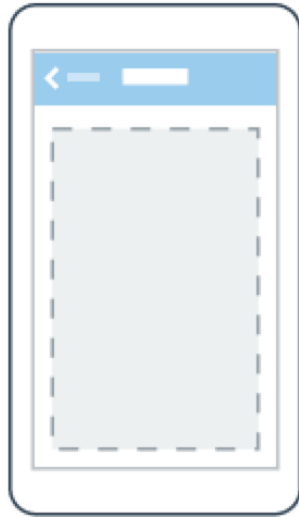
Pages



ContentPage



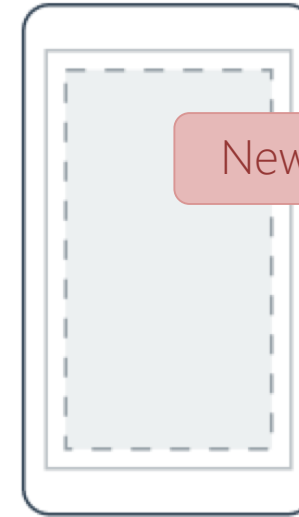
MasterDetailPage



NavigationPage

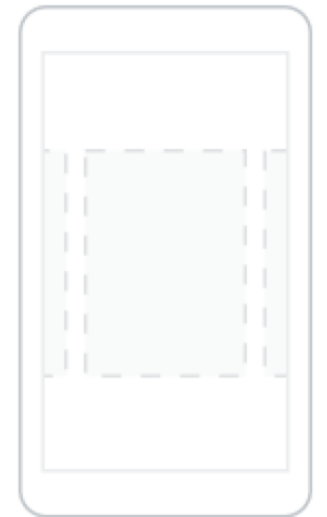


TabbedPage



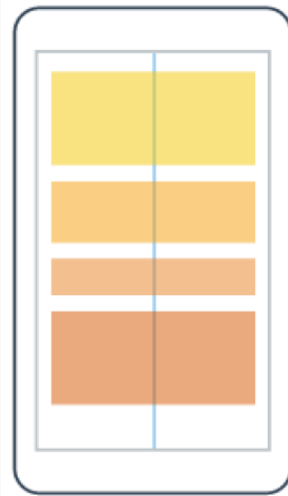
TemplatedPage

New!

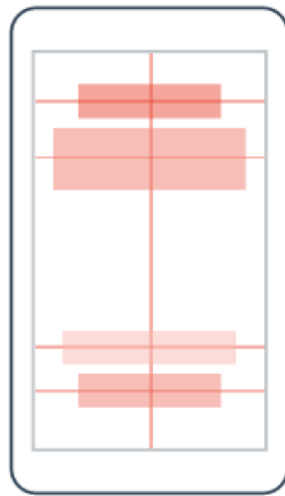


CarouselPage

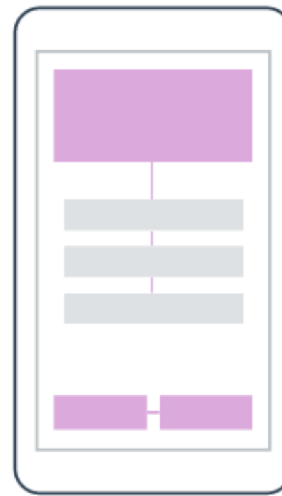
Layouts



StackLayout



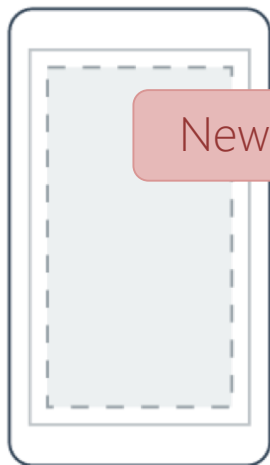
AbsoluteLayout



RelativeLayout



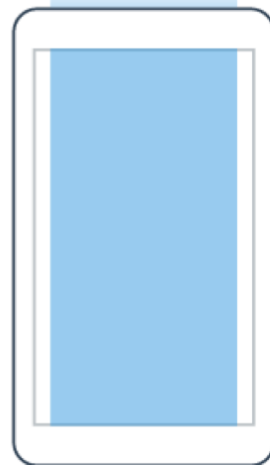
GridLayout



ContentPresenter



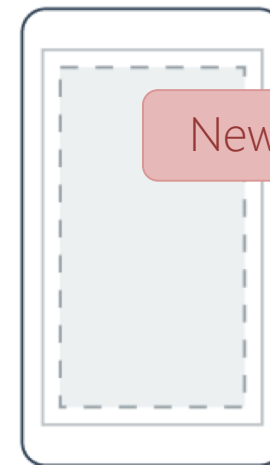
ContentView



ScrollView



Frame



TemplatedView

Controls

ActivityIndicator

BoxView

Button

DatePicker

Editor

Entry

Image

Label

ListView

Map

OpenGLView

Picker

ProgressBar

SearchBar

Slider

Stepper

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
```

XAMLの機能

Feature	Supported in Xamarin.Forms
XAML 2009 compliance	✓
Shapes (Rectangle, Ellipse, Path, etc.)	BoxView
Resources, Styles and Triggers	✓
Data binding	✓ *not all features
Data templates	✓
Control templates	Custom renderers
Render Transforms	✓
Animations	Code-only
Custom XAML behaviors	✓
Custom markup extensions	✓
Value converters	✓

XAMLの機能

Resource

Style

Trigger

Behavior

Value Converter

Data Template







Data Binding

PreviewPage.xaml

```
15
16 <Label Text="DataBinding"
17       Margin="20, 15, 20, 0"
18       HorizontalTextAlignment="Center"
19       FontSize="20"
20       FontAttributes="Bold"/>
21
22 <ListView Grid.Row="1" ItemsSource="{Binding Persons}"
23           HasUnevenRows="true">
24     <ListView.ItemTemplate>
25       <DataTemplate>
26         <ViewCell>
27           <Grid Padding="17, 10" RowSpacing="10" Columns="2">
28             <Grid.RowDefinitions>
29               <RowDefinition Height="Auto"/>
30               <RowDefinition Height="*" />
31             </Grid.RowDefinitions>
32             <Grid.ColumnDefinitions>
33               <ColumnDefinition Width="Auto"/>
34               <ColumnDefinition Width="*" />
35             </Grid.ColumnDefinitions>
36
37             <controls:CircleImage BorderColor="#3498DB"
38                                   BorderThickness="2"
39                                   Source="{Binding Photo}"
40                                   HeightRequest="65"
41                                   WidthRequest="65"
42                                   Grid.RowSpan="2"/>
43             <StackLayout Grid.Row="0"
44                           Grid.Column="1"
45                           Orientation="Horizontal">
46               <Label Text="{Binding Name}"
47                       TextColor="#3498DB"
48                       FontSize="20"
49                       HorizontalOptions="StartAndExpand"
50               <Label Text="{Binding Department}"
51                       FontSize="13"
52                       HorizontalOptions="End"
53                       VerticalTextAlignment="End"/>
54             </StackLayout>
55           </Grid>
56         </ViewCell>
57       </DataTemplate>
58     </ListView.ItemTemplate>
59 </ListView>
```

Device: Phone Tablet Platform: Android iOS

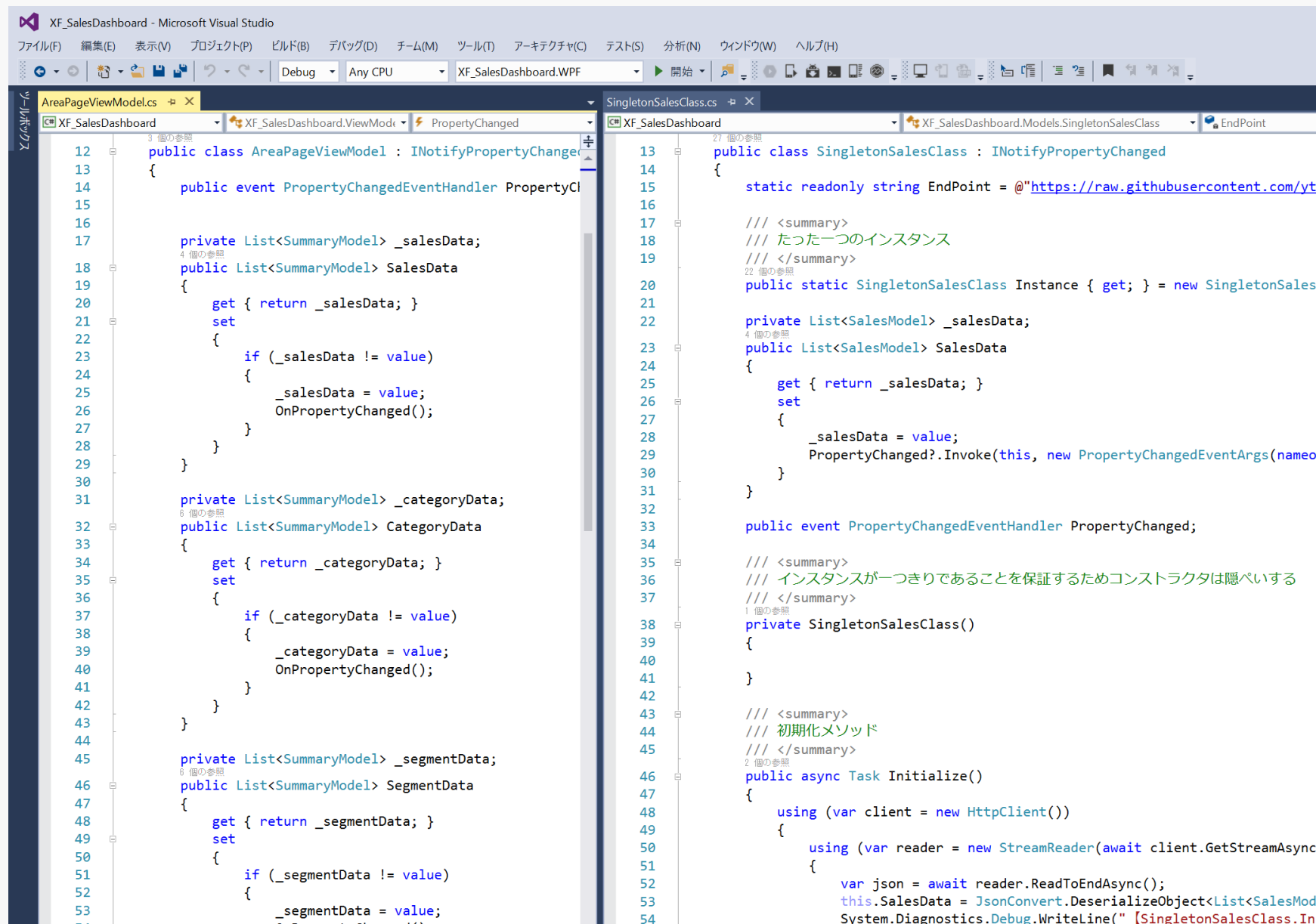
DataBinding

	John Silverstain	Marketi
MELBOURNE	Followers: 243	29
	Pam Tailor	Desi
SIDNEY	Followers: 24	32
	Casy Niman	Accour
HOBART	Followers: 267	58
	Gorge Tach	Desi
NEWCASTLE	Followers: 127	29
	Cristina Maciel	MobileD
HOBART	Followers: 80	32
	Simon Deuva	Med

45

MVVM

Model View ViewModel Data Binding Messaging Center

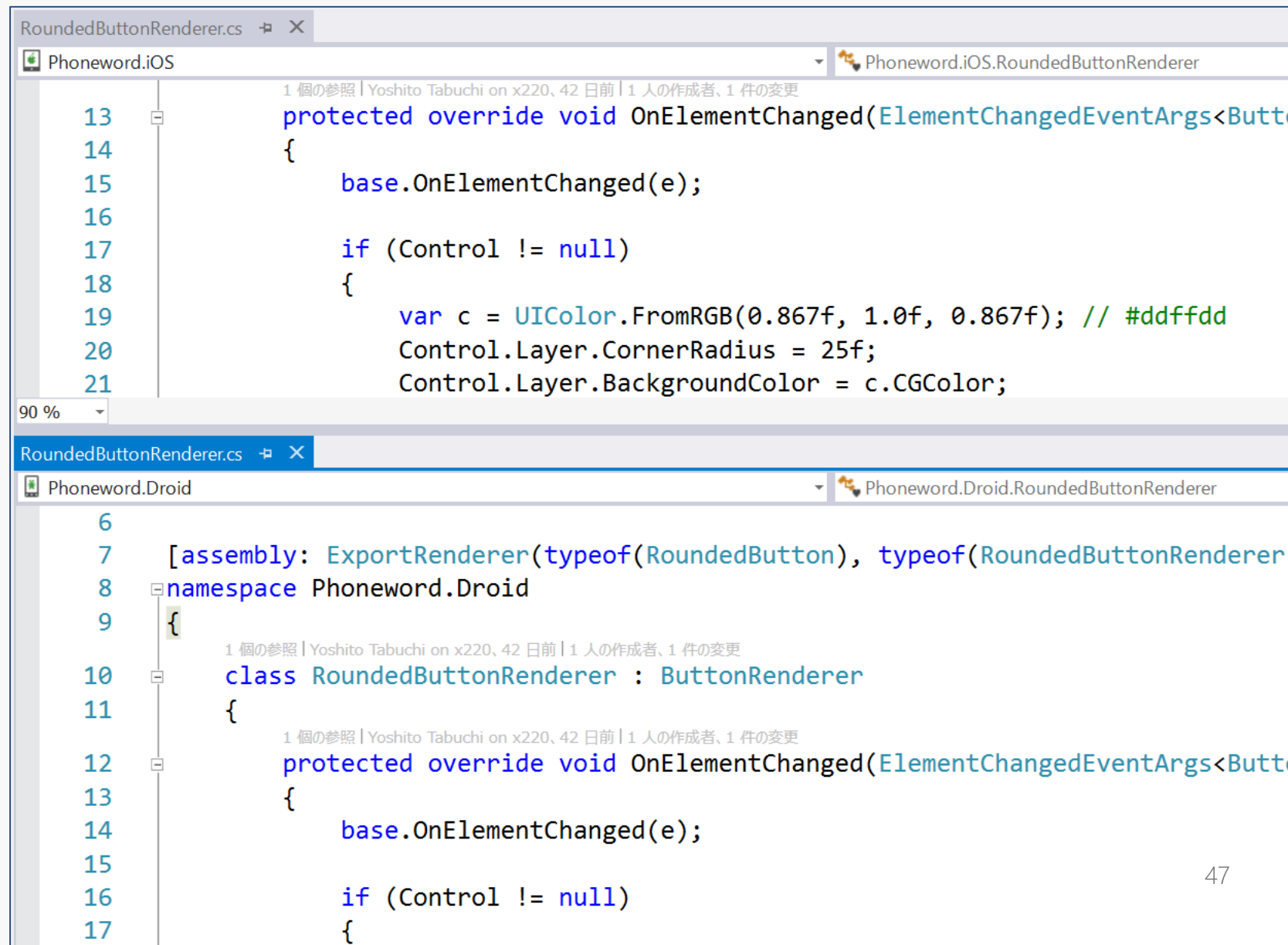


```
XF_SalesDashboard - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) アーキテクチャ(C) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug Any CPU XF_SalesDashboard.WPF 開始

AreaPageViewModel.cs
XF_SalesDashboard
XF_SalesDashboard.ViewModel
PropertyChanged
12 public class AreaPageViewModel : INotifyPropertyChanged
13 {
14     public event PropertyChangedEventHandler PropertyChanged
15
16
17     private List<SummaryModel> _salesData;
18     public List<SummaryModel> SalesData
19     {
20         get { return _salesData; }
21         set
22         {
23             if (_salesData != value)
24             {
25                 _salesData = value;
26                 OnPropertyChanged();
27             }
28         }
29     }
30
31     private List<SummaryModel> _categoryData;
32     public List<SummaryModel> CategoryData
33     {
34         get { return _categoryData; }
35         set
36         {
37             if (_categoryData != value)
38             {
39                 _categoryData = value;
40                 OnPropertyChanged();
41             }
42         }
43     }
44
45     private List<SummaryModel> _segmentData;
46     public List<SummaryModel> SegmentData
47     {
48         get { return _segmentData; }
49         set
50         {
51             if (_segmentData != value)
52             {
53                 _segmentData = value;
54
SingletonSalesClass.cs
XF_SalesDashboard
XF_SalesDashboard.Models.SingletonSalesClass
EndPoint
13 public class SingletonSalesClass : INotifyPropertyChanged
14 {
15     static readonly string EndPoint = @"https://raw.githubusercontent.com/yt
16
17     /// <summary>
18     /// たった一つのインスタンス
19     /// </summary>
20     public static SingletonSalesClass Instance { get; } = new SingletonSales
21
22     private List<SalesModel> _salesData;
23     public List<SalesModel> SalesData
24     {
25         get { return _salesData; }
26         set
27         {
28             _salesData = value;
29             PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameo
30         }
31     }
32
33     public event PropertyChangedEventHandler PropertyChanged;
34
35     /// <summary>
36     /// インスタンスが一つきりであることを保証するためコンストラクタは隠ぺいする
37     /// </summary>
38     private SingletonSalesClass()
39     {
40
41     }
42
43     /// <summary>
44     /// 初期化メソッド
45     /// </summary>
46     public async Task Initialize()
47     {
48         using (var client = new HttpClient())
49         {
50             using (var reader = new StreamReader(await client.GetStreamAsync
51             {
52                 var json = await reader.ReadToEndAsync();
53                 this.SalesData = JsonConvert.DeserializeObject<List<SalesMod
54                 System.Diagnostics.Debug.WriteLine($"[SingletonSalesClass.I
```


ネイティブコントロール (UI)

Custom Renderer Effects



```
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;
22     }
23 }

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             {
20                 var c = UIColor.FromRGB(0.867f, 1.0f, 0.867f); // #ddffdd
21                 Control.Layer.CornerRadius = 25f;
22                 Control.Layer.BackgroundColor = c.CGColor;
23             }
24         }
25     }
26 }
```

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                 new NSURL("tel:" + number))
18         }
19     }
20 }
```

```
PhoneDialer.cs
Phoneword.Droid
Phoneword.Droid
Dial(string number)

1 using Android.Content;
2 using Android.Telephon
3 using Phoneword.Droid
4 using System.Linq;
5 using Xamarin.Forms;
6
7 using Uri = Android.Ne
8
9 [assembly: Dependency
10
11 namespace Phoneword.Dr
12 {
13     1 個の参照 | Yoshito Tabuchi on x22
14     public class Phone
15     {
16         0 個の参照 | Yoshito Tabuchi
17         public bool D
18         {
19             var contex
20             if (contex
21                 return
22
23             var intent
24             intent.Set
25                 48
26             if (IsInte
27         {
```


アジェンダ

Xamarin 概要

Android, iOS 概要

Xamarin ネイティブ, Xamarin.Forms

開発手法

まとめ

Xamarin

C# / .NET / Visual Studio

フル “ネイティブ” アプリ

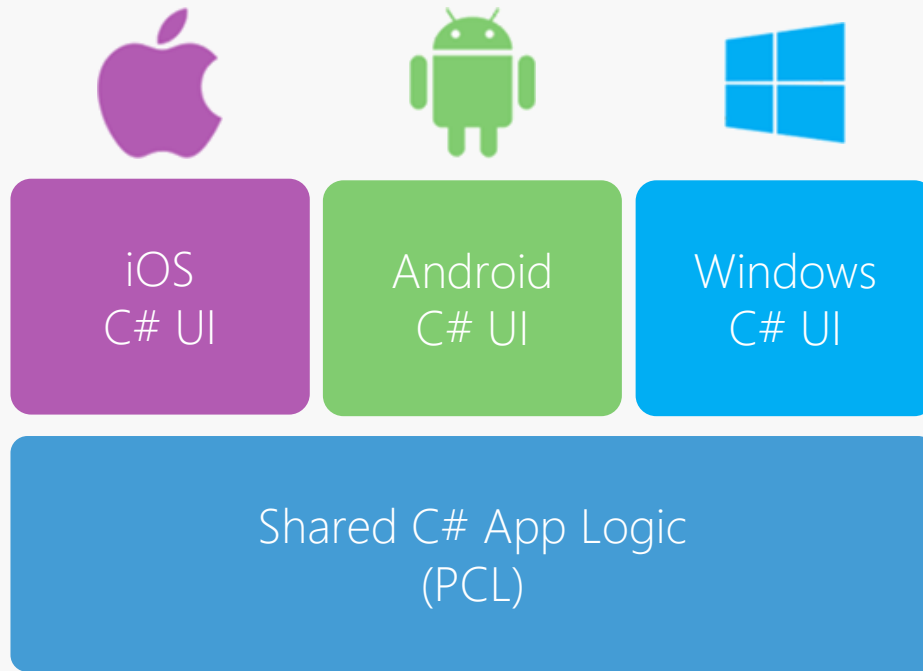
API 100% 移植

コード共通化

2つの開発手法

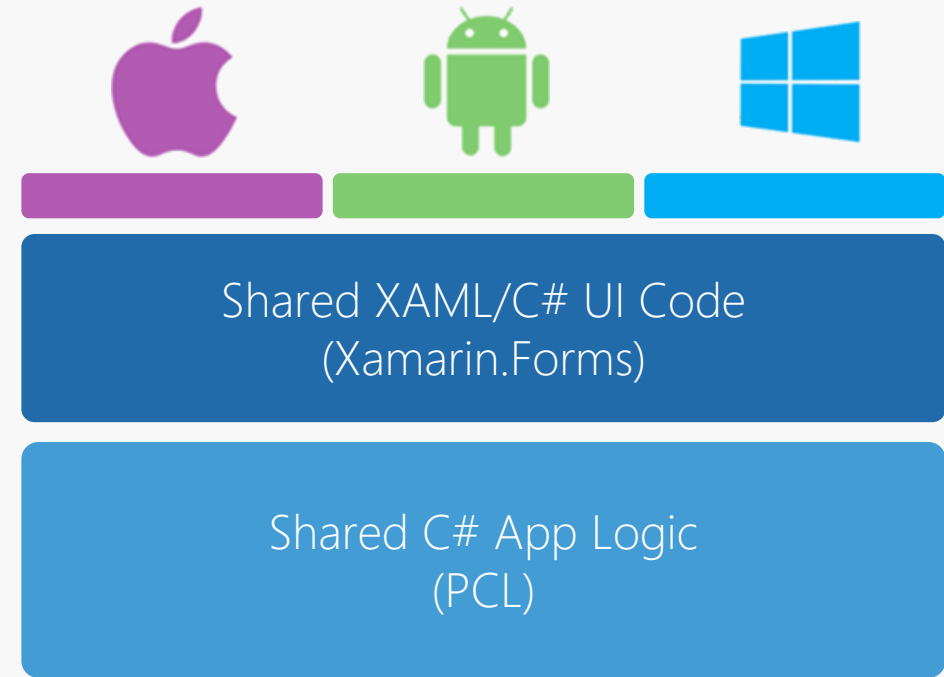
Xamarin Native

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



#Xamarinはいいぞ

リソース

公式ドキュメント

ペゾルド本（PDFが無料配布中）

日本語の情報

Japan Xamarin User Group

Build Insider

Qiita

田淵のブログ

各種ブログへのリンク

ありがとうございました



JAPAN
XAMARIN
USER
GROUP

Japan Xamarin User Group

田淵 義人

080-7015-3586

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

facebook: [xlsoft.ytabuchi](https://www.facebook.com/xlsoft.ytabuchi)

Blog: [Xamarin日本語情報](#)