


# Xamarin.Forms概要

jxug #1 2014.7.5

@amay077

# 自己紹介

-  奥山 裕紳 (OKUYAMA Hironobu) - @amay077
- 位置情報エンジニア、モバイルアプリエンジニア
- VB6 → VC6 → C#2.0 → (闇) → Java/Obj-C → C#5.0
- Cosmoroot,Inc(Nagoya,Tokyo)
- 地理情報、オープンデータ、C#、Android、iOS、**Xamarin**

# 個人活動

- Androidアプリ - HexRinger、しゃべPOI、etc
- MA9 いのししー頭賞 - 富士フォト
- オープンデータ関連コンテスト参加
- うずらインキュベータ(愛知県豊橋市のIT勉強会)
- 「Xamarin でググレ！」と言ってみたい

Web Technology

Enterprise

Small & Medium

Consumer

> Mobile

# BUILD Insider で

## Xamarin逆引きTips

# 「Xamarin 逆引き Tips」

ダイアログ（=iOSのUIAlertView／AndroidのAlertDialog）や、処理実行中を示すBusyインジケータを、Xamarin.Formsで表示する方法を解説する。これらは共通のAPIを使って実装できる。

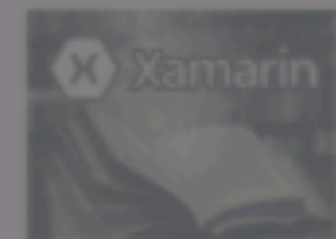
# 書いています

Mobile C# マルチプラットフォーム Android iOS Xamarin Mac Windows Xamarin Studio Xamarin.Forms

2014年7月2日(水曜日)

## Xamarin.Formsからプラットフォーム固有の機能を利用するには？ (DependencyService利用)

UIを共通化するフレームワーク「Xamarin.Forms」で、「DependencyService」機能を使用してiOS／Androidの各プラットフォーム固有の機能を実装する方法を解説する。



# [PR]株式会社コスモルート

## クラウドサービスプロバイダ

- ロジネビュラ:クラウド型倉庫管理システム
- GeQuu:リアルタイムセンサデータ収集プラットフォーム
- ネプラ:基幹業務向けPaaS
- AWS テクノロジーパートナー認定
- AWSジャパンツアー2014夏(8.1名古屋)に社長が登壇します

# 目次

1. Xamarin.Forms とはなにか？
2. プラットフォーム機能との併用
3. MVVM な機能
4. ためしにアプリを作ってみた
5. まとめ

# 1. Xamarin.Forms とはなにか？

# そもそも Xamarin(.iOS or .Android) とは

- ネイティブ(iOS or Android)APIの **C#のラッパー** +  
.NET BCL の iOS or Android版
- .NET BCL 部分はクロスPFで使いまわせる→PCLとか
- ネイティブAPIのラッパー部分は**使いまわせない**
  - 画面
  - カメラ、GPS、Push、アプリ連携、Beacon、etc...



# Xamarin.Forms とは

- ワンソースで複数PFの画面を作成できるフレームワーク
- iOS6～, Android4～, Windows Phone8～ に対応
- PCL と Shared Project で作れる
- C# or XAML(WPFとは非互換) で書く
  - UIデザイナーは(まだ?)ない
- MVVM な機能も含む

# Xamarin.Forms とは

- UIレンダリングはネイティブで行われる
  - たとえば `Xamarin.Forms.Label` は Android では `TextView`, iOS では `UILabel` が使われる
- Titanium Mobile と同じ仕組み
- Adobe AIR や Delphi XE は独自レンダリング
- PhoneGap は HTML なので比較対象外

# 画面の定義(C#)

```
public class App {  
    public static Page GetMainPage() {  
        return new ContentPage {  
            Content = new Xamarin.Forms.DatePicker {  
                VerticalOptions = LayoutOptions.CenterAndExpand,  
                HorizontalOptions = LayoutOptions.CenterAndExpand,  
            },  
        };  
    }  
}
```

# 画面の定義(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="A2_Xaml.MainPage">
  <ContentPage.Content>
    <Label Text="Hello Xaml"
      VerticalOptions="Center"
      HorizontalOptions="Center" />
  </ContentPage.Content>
</ContentPage>
```

おことうわり

**XAMLの説明はこれで終わりです**

Windows Phone,

説明できません

端末をくだ(ry

# エントリーポイント (iOS)

```
//using Xamarin.Forms;
```

```
public class AppDelegate : UIApplicationDelegate {  
    public override bool FinishedLaunching(  
        UIApplication app, NSDictionary options) {  
  
        window = new UIWindow(UIScreen.MainScreen.Bounds);  
  
        Forms.Init();  
        window.RootViewController = App.GetMainPage()  
            .CreateViewController(); // 1.  
  
    }  
}
```

```
// 以下省略
```



# エントリーポイント (Android)

```
//using Xamarin.Forms.Platform.Android;
```

```
//using Xamarin.Forms;
```

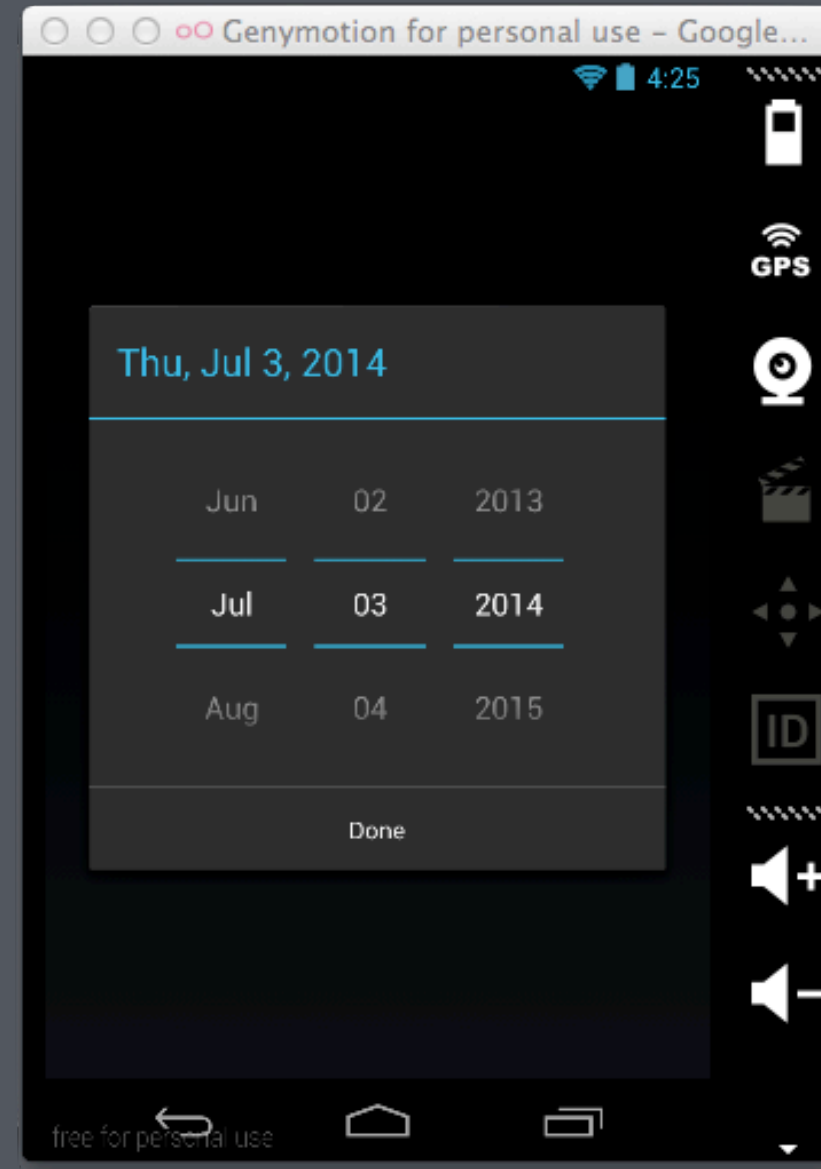
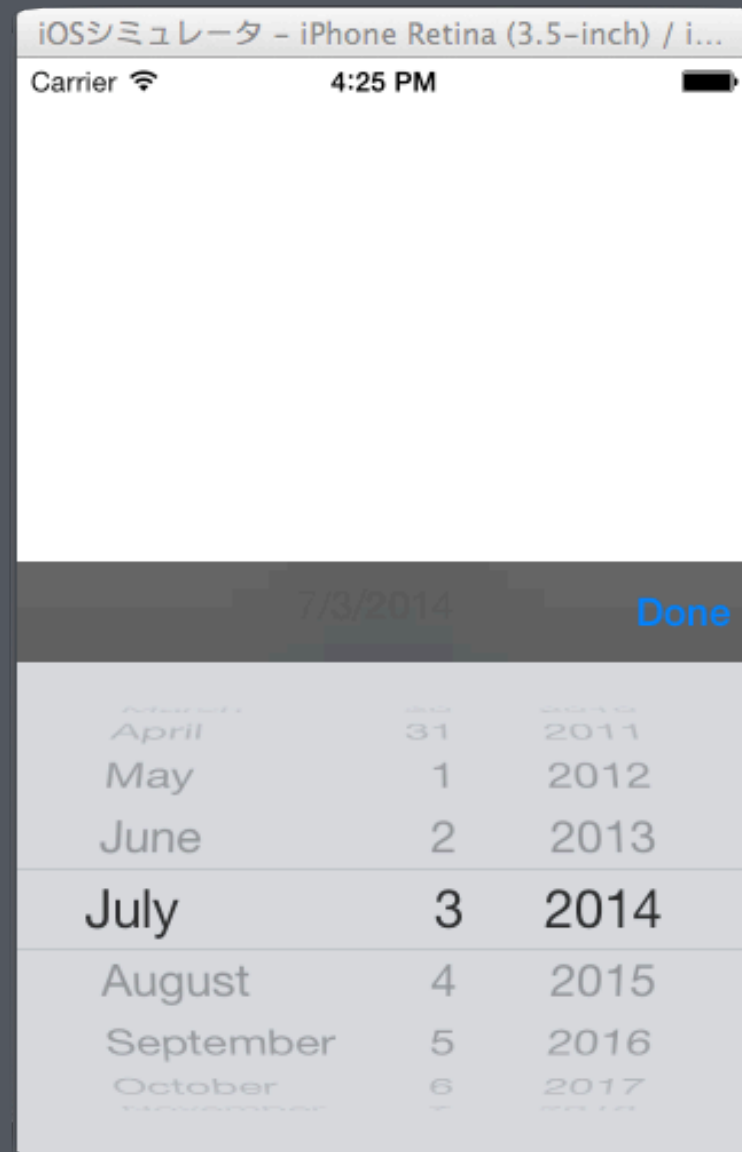
```
public class MainActivity : AndroidActivity { // 1.  
    protected override void OnCreate(Bundle bundle) {  
        base.OnCreate(bundle);
```

```
        Forms.Init(this, bundle); // 2.
```

```
        SetPage(App.GetMainPage()); // 3.
```

```
// 以下省略
```

# 実行結果



# 構成要素

- Page = 画面
- Layout = View のコンテナ(Viewでもある)
- View = コントロール

# Page



ContentPage



MasterDetailPage



NavigationPage

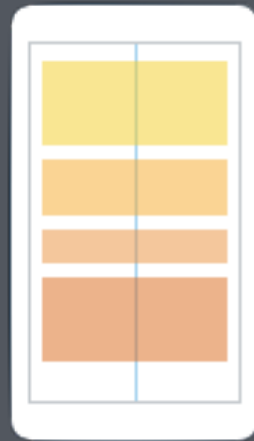


TabbedPage

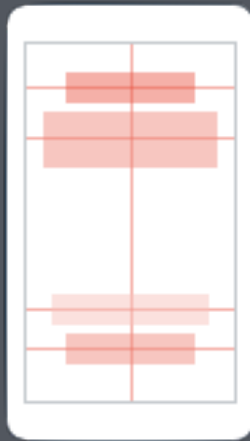


CarouselPage

# Layout



StackLayout



AbsoluteLayout



RelativeLayout



GridLayout



ContentView



ScrollView



Frame

# View

ActivityIndicator

BoxView

Button

DatePicker

Editor

Entry

Image

Label

ListView

Map

OpenGLView

Picker

ProgressBar

SearchBar

Slider

Stepper

TableView

TimePicker

WebView

EntryCell

ImageCell

SwitchCell

TextCell

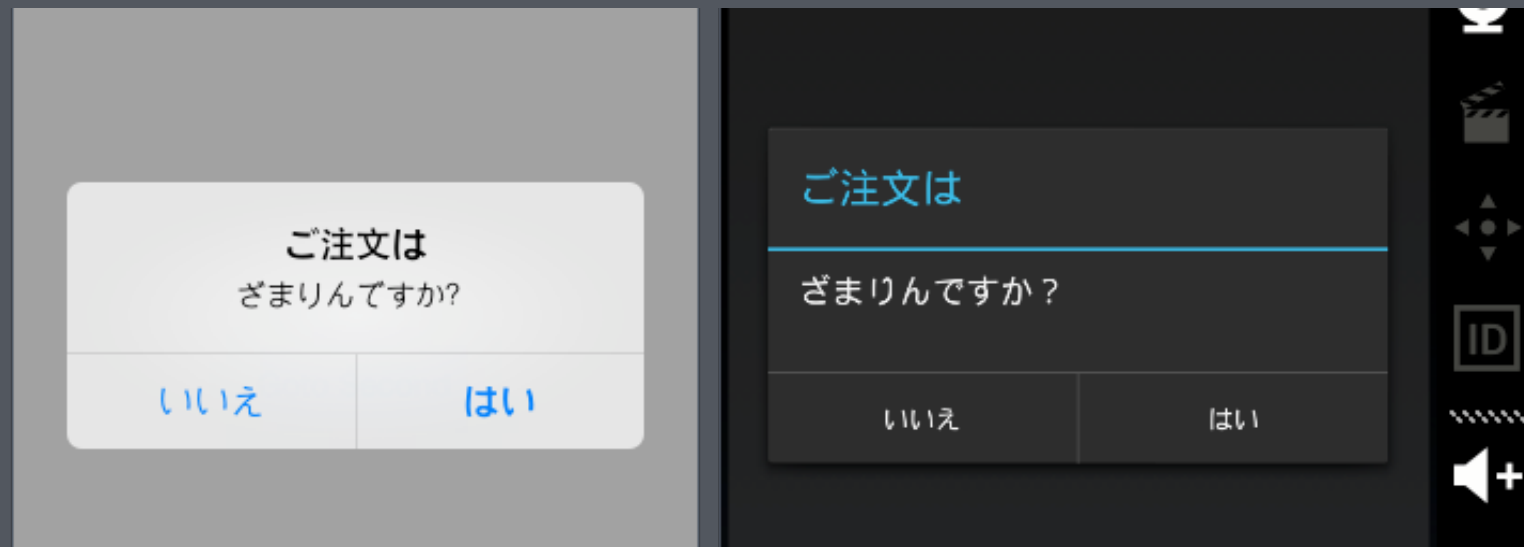
ViewCell

# 画面遷移

```
public class App {  
    public static Page GetMainPage() {  
        var btn = new Button { Text = "Goto Second" };  
  
        var navPage = new NavigationPage(new ContentPage { // 1.  
            Content = btn,  
        });  
  
        btn.Clicked += (sender, e) =>  
            navPage.PushAsync(App.GetSecondPage()); // 2.  
  
        return navPage;  
    }  
}
```

# アラートダイアログ

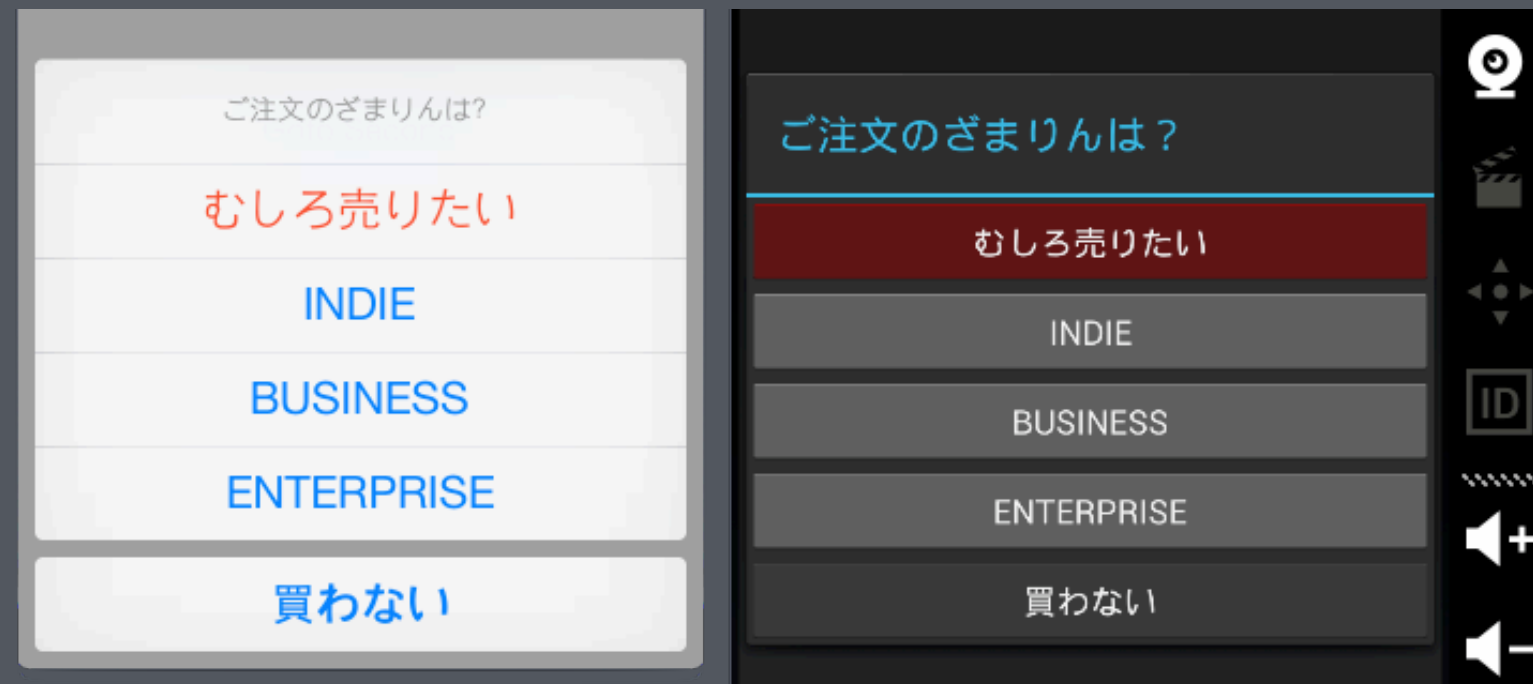
```
btn.Clicked += async (sender, e) => {  
    if (await page.DisplayAlert("ご注文は",  
        "ざまりんですか?", "はい", "いいえ")) {  
        btn.Text = "あざす";  
    }  
};
```





# 選択ダイアログ

```
var selectedText = await page.DisplayActionSheet(  
    "ご注文のざまりんは?",  
    "買わない", "むしろ売りたい",  
    new string[] { "INDIE", "BUSINESS", "ENTERPRISE" });  
btn.Text = selectedText; // Android で BACK すると null が来る
```



# 分岐処理

// プラットフォーム毎の処理

```
Device.OnPlatform(() => /* Action for iOS */,  
                  () => /* Action for Android */,  
                  () => /* Action for Windows Phone */);
```

// プラットフォーム

```
Device.OS = { iOS | Android | WinPhone }
```

// Phone or Tab

```
Device.Idiom = { Phone | Tablet }
```

「これ使ったら負け」感...。

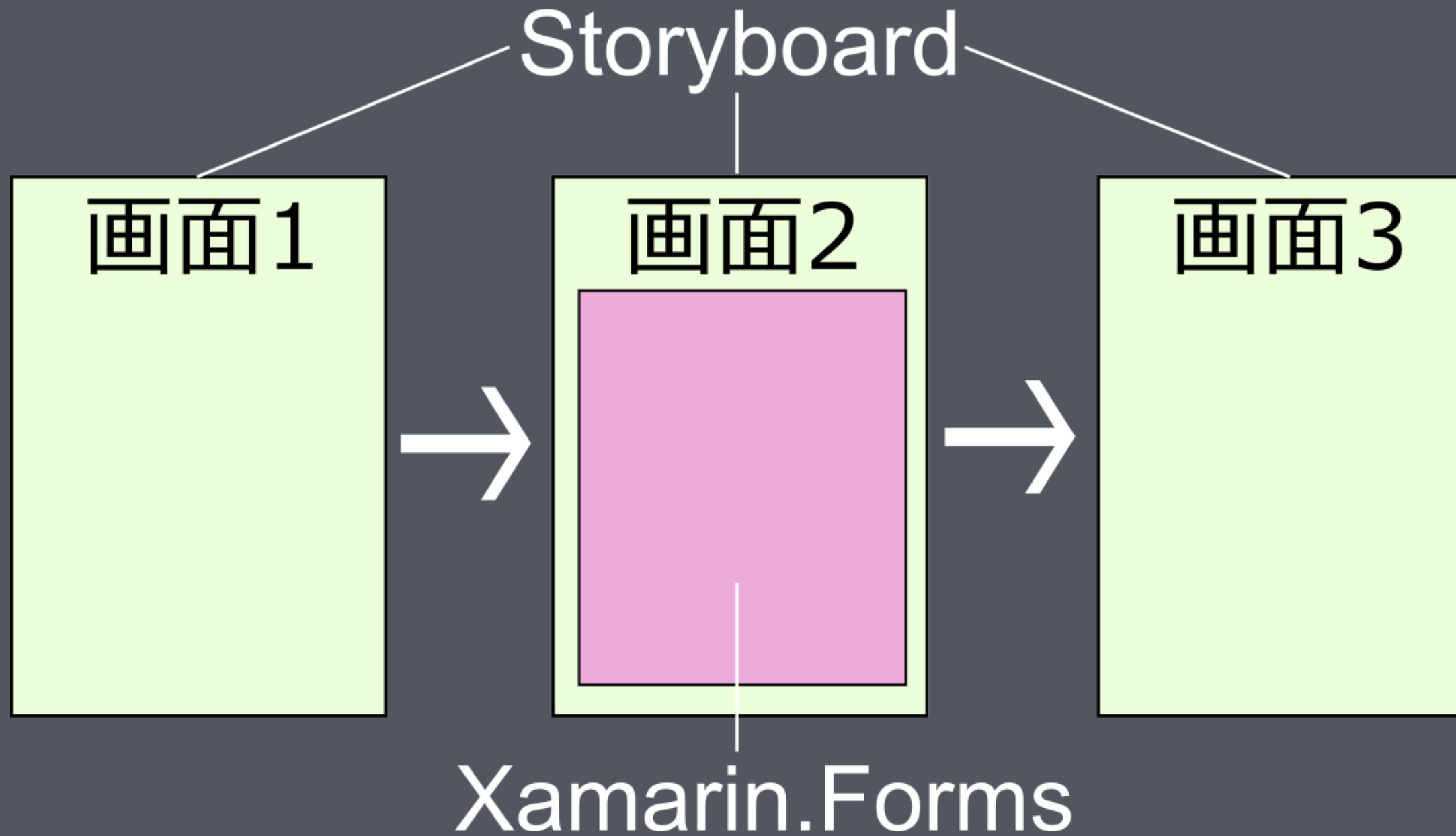
## 2. プラットフォーム機能との併用

# プラットフォーム機能を使いたい時とは？

- いきなり Forms に移行するとかムリ
  - 例: .axml/.storyboard アプリの一部に Forms を使う
- Forms に足りない機能がある
  - 既存のViewの拡張(例:ButtonにLongTapイベントを付ける)
  - まったく新しいViewを作る
- 画面以外の機能を使いたい(GPSとか)

.axml/.storyboard アプリの一部に Forms を使う

# [iOS]Storyboard の一部に Forms



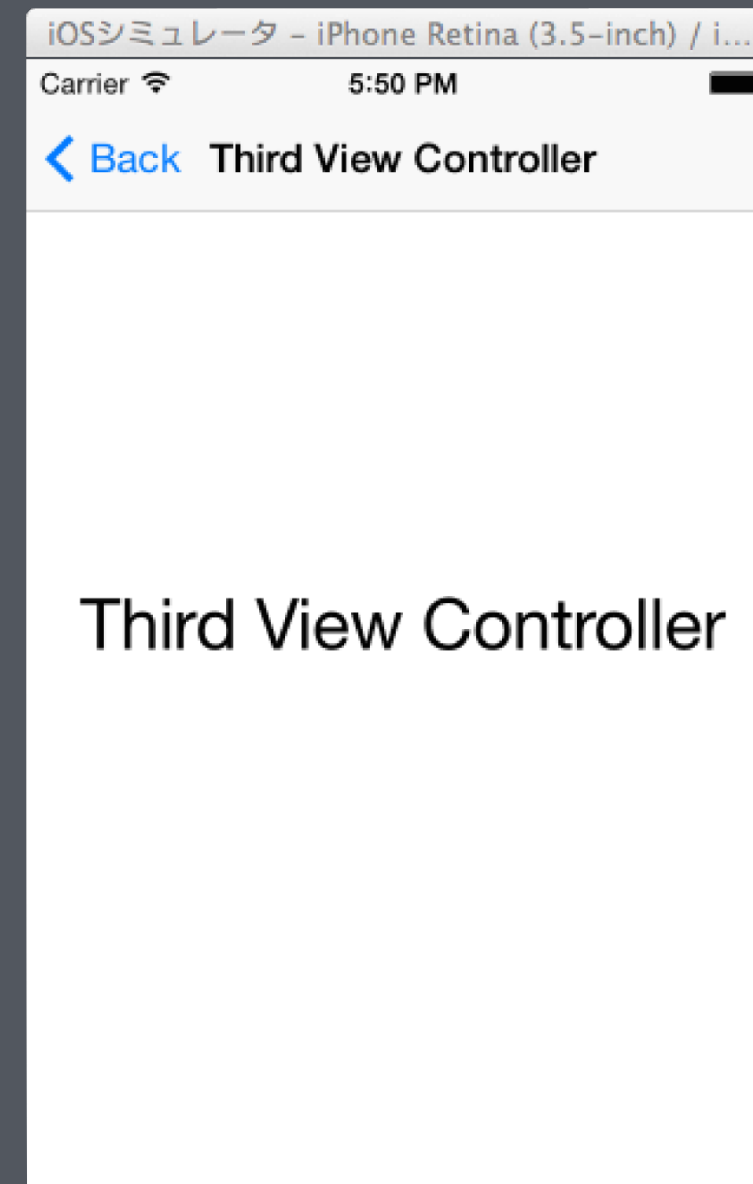
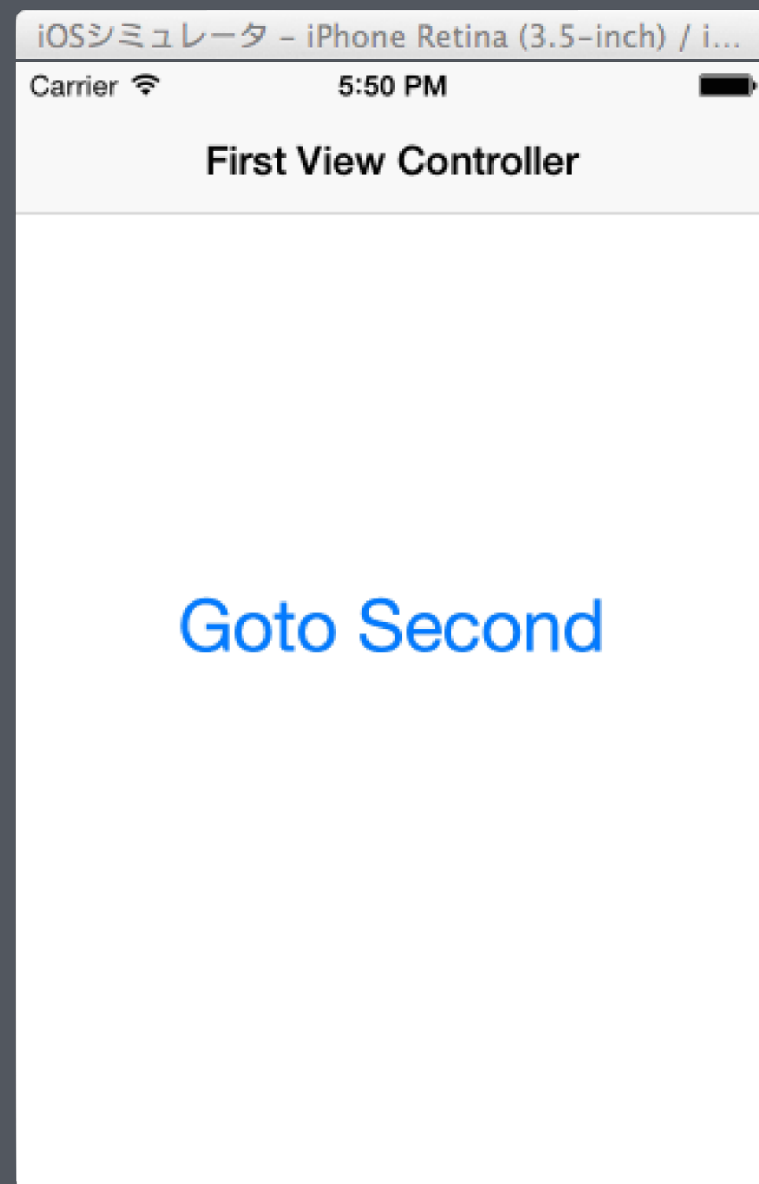
# Forms を適用したい ViewController の実装

```
partial class SecondViewController : UIViewController {
    public override void ViewDidLoad() {
        base.ViewDidLoad();
        Forms.Init(); // 初期化

        // Page 側のボタンのイベントハンドラ(画面遷移)
        var page = new MainPage();
        page.GotoThirdButton.Clicked += (s, e) => PerformSegue("goto_third", this);

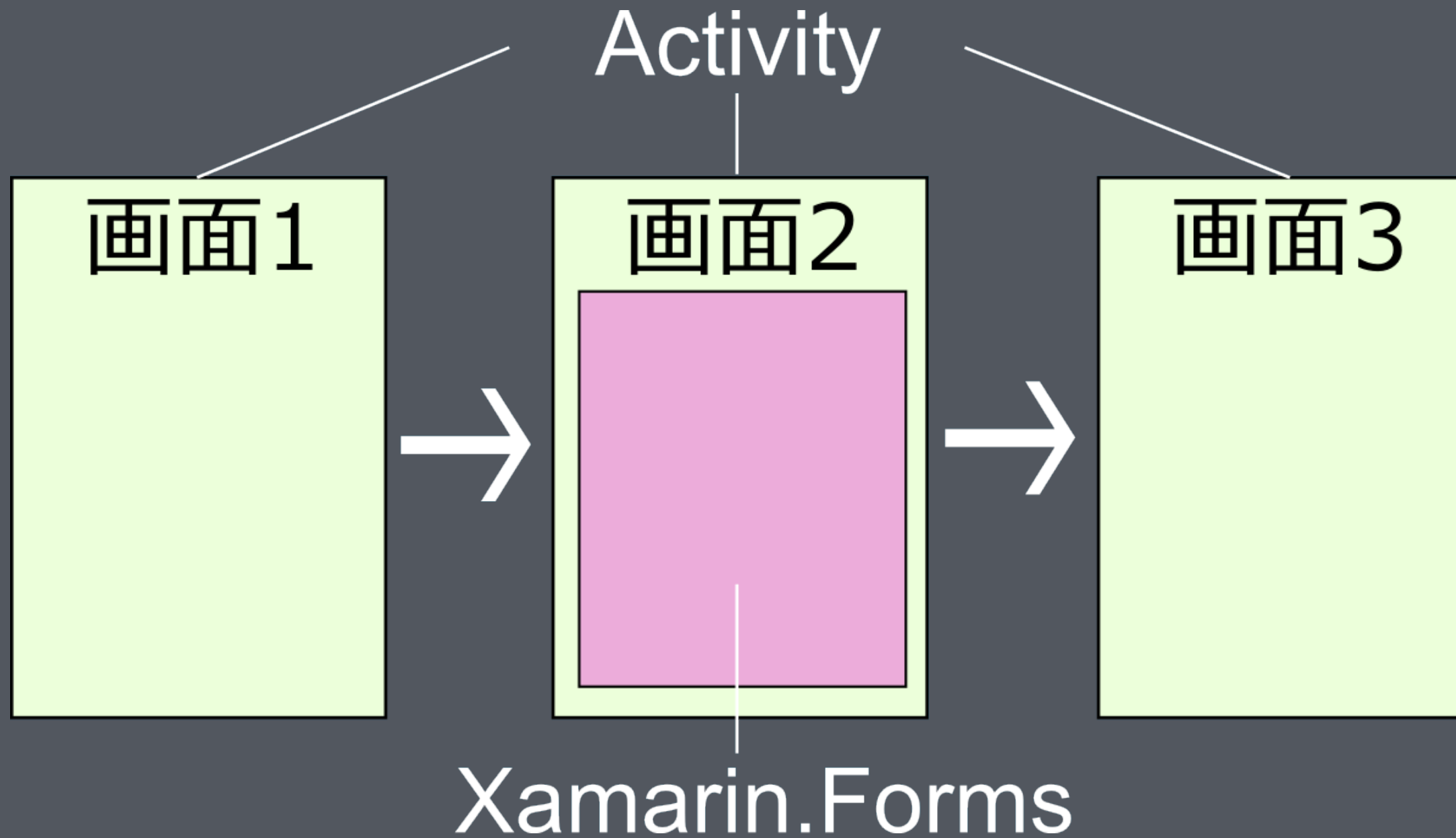
        var vc = page.CreateViewController();
        AddChildViewController(vc);
        View.Add(vc.View);
        vc.DidMoveToParentViewController(this);
    }
}
```

# [iOS]実行結果





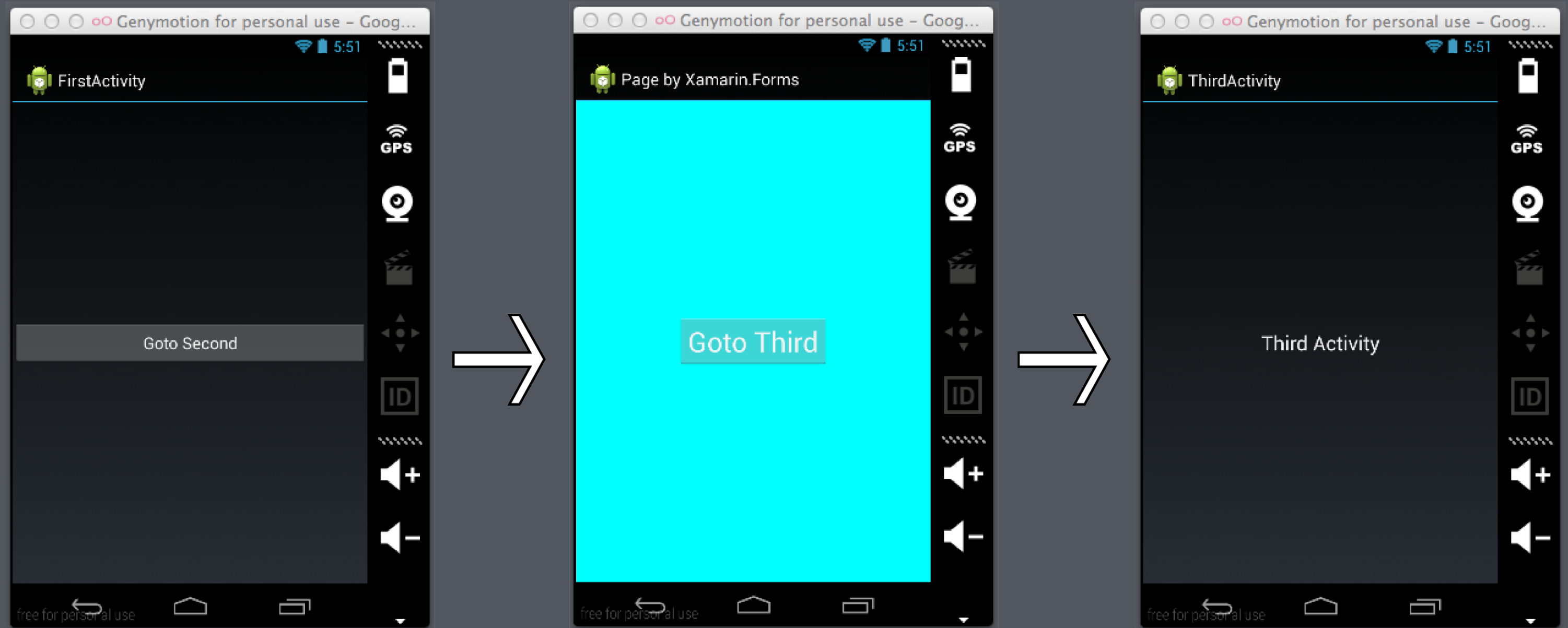
# [Android]Activity の一部に Forms



# Forms を適用したい Activity の実装

```
public class SecondActivity : AndroidActivity {  
    protected override void OnCreate(Bundle bundle) {  
        base.OnCreate(bundle);  
        Forms.Init(this, bundle); // 初期化  
  
        // Page生成  
  
        var page = new MainPage();  
        // Page 側のボタンのイベントハンドラ(画面遷移)  
  
        page.GotoThirdButton.Clicked += (sender, e) =>  
            StartActivity(typeof(ThirdActivity));  
        // Pageを設定  
  
        this.SetPage(new NavigationPage(page));  
    }  
}
```

# [Android]実行結果



# Forms の部品を拡張する

例:Forms.ButtonにLongTapイベントを追加する

# [Forms]Buttonを拡張したMyButtonを用意

```
public class MyButton : Xamarin.Forms.Button {  
    // イベント  
  
    public event EventHandler LongTap;  
    // イベント発火用  
  
    public void OnLongTap() {  
        if (this.LongTap != null)  
            this.LongTap(this, new EventArgs());  
    }  
}
```

# [Android]ButtonRenderer を拡張する

// ↓↓ 超重要！ ↓↓

```
[assembly:ExportRenderer(typeof(MyButton), typeof(MyButtonRenderer))]
```

```
public class MyButtonRenderer : ButtonRenderer {  
    protected override void OnElementChanged(  
        ElementChangedEventArgs<Button> e) {  
        base.OnElementChanged(e);  
        var formsButton = e.NewElement as MyButton; // FormsのButton  
        var droidButton = this.Control; // AndroidのButton  
        droidButton.LongClick += (sender, _) =>  
            formsButton.OnLongTap();  
    }  
}
```

# [iOS]ButtonRenderer を拡張する

// ↓↓ 超重要! ↓↓

```
[assembly:ExportRenderer(typeof(MyButton), typeof(MyButtonRenderer))]
```

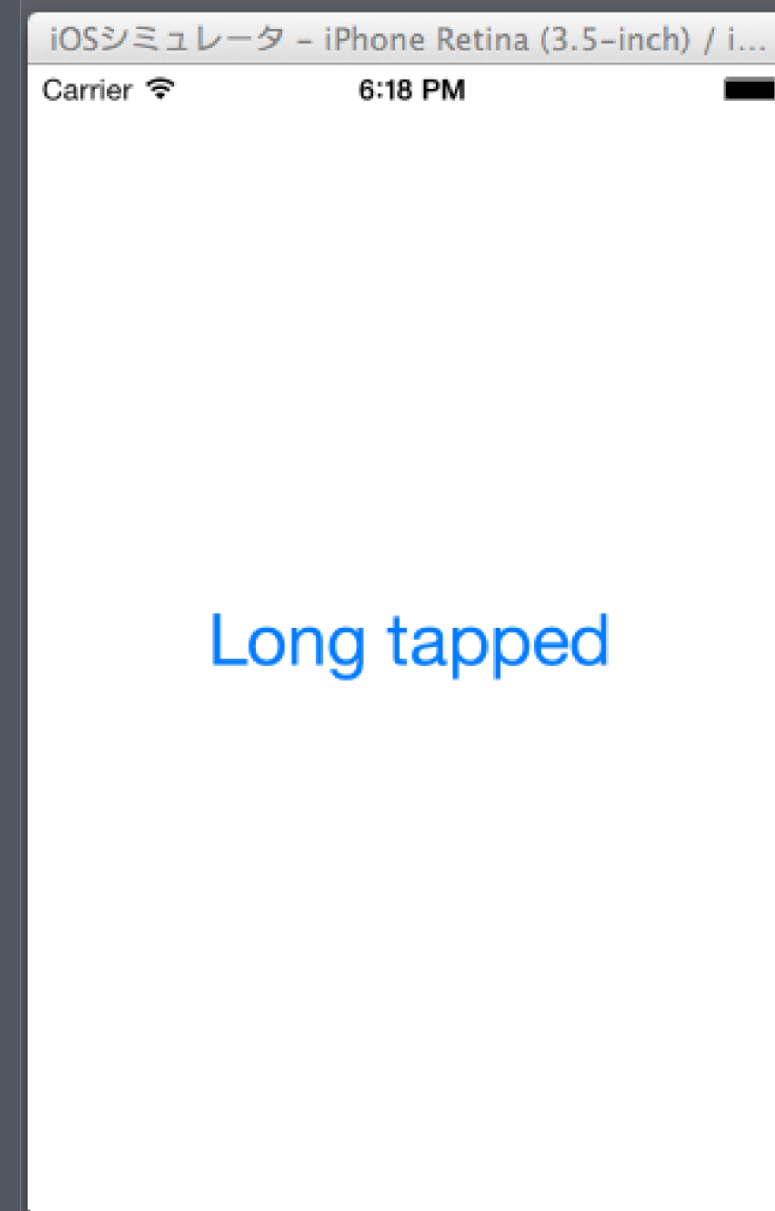
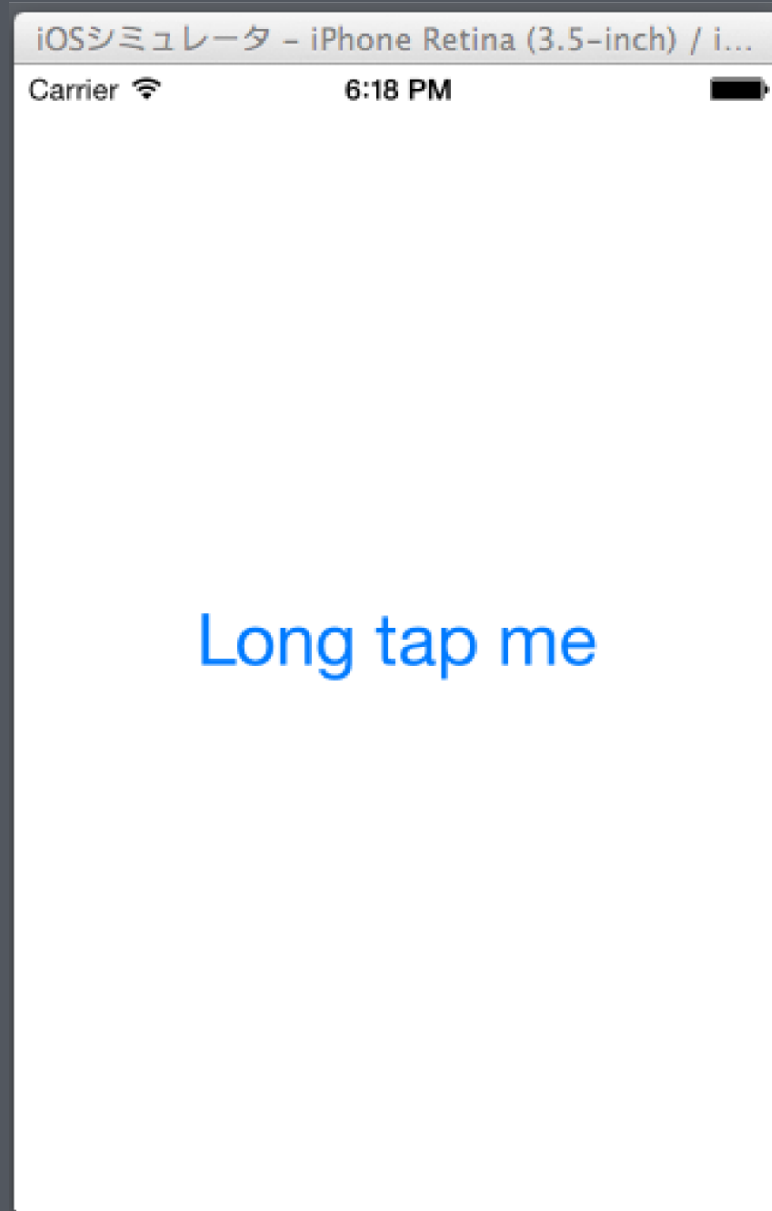
```
public class MyButtonRenderer : ButtonRenderer {  
    protected override void OnElementChanged(  
        ElementChangedEventArgs<Button> e) {  
        base.OnElementChanged(e);  
        UIButton iosButton = this.Control; // iOSのButton  
        iosButton.AddGestureRecognizer(new UILongPressGestureRecognizer(x => {  
            if (x.State == UIGestureRecognizerState.Recognized)  
                (e.NewElement as MyButton).OnLongTap();  
        }));  
    }  
}
```

# [Forms]MyButton を使う

```
public class App {  
    public static Page GetMainPage() {  
        var button = new MyButton { Text = "Long tap me" };  
  
        button.LongTap += (sender, e) =>  
            button.Text = "On long tapped";  
  
        return new ContentPage { Content = button };  
    }  
}
```



# [iOS]実行結果



まったく新しいViewをつくる

# [Forms]Viewを拡張したMyMapViewを用意

```
// Forms:MyMapView.cs
public class MyMapView : Xamarin.Forms.View
{

}
```

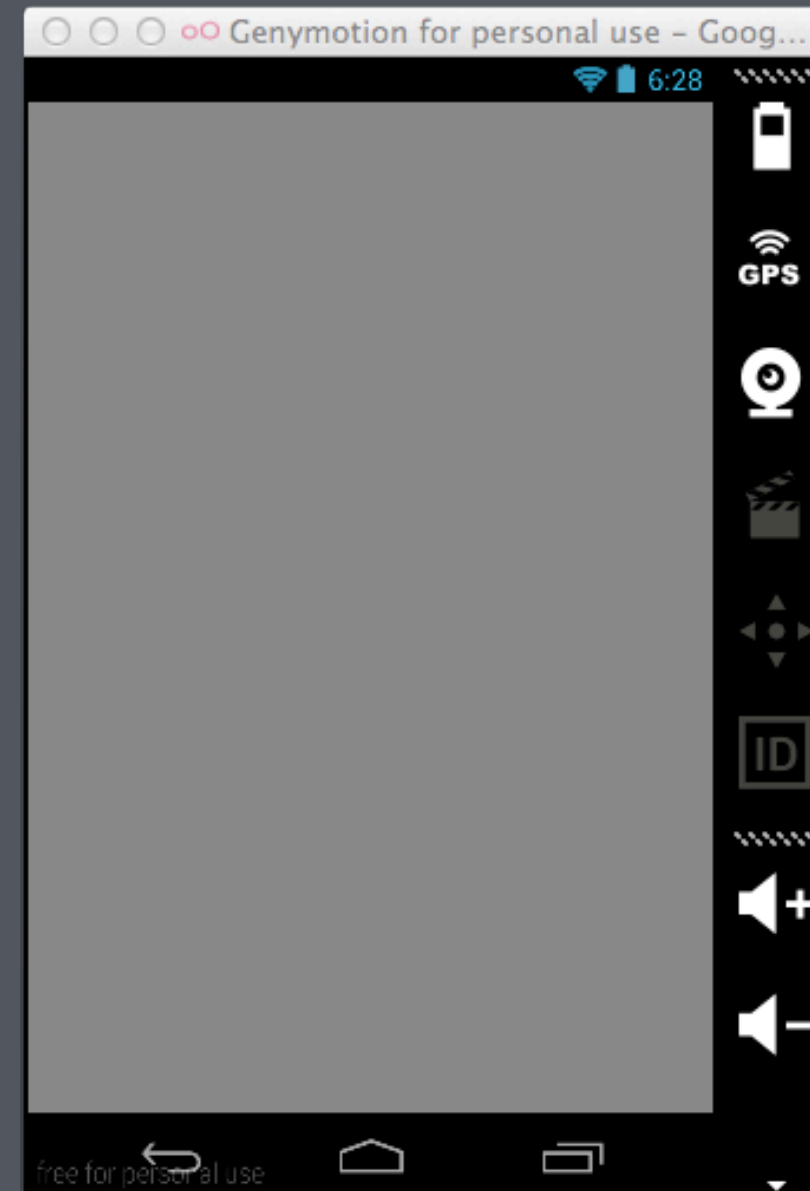
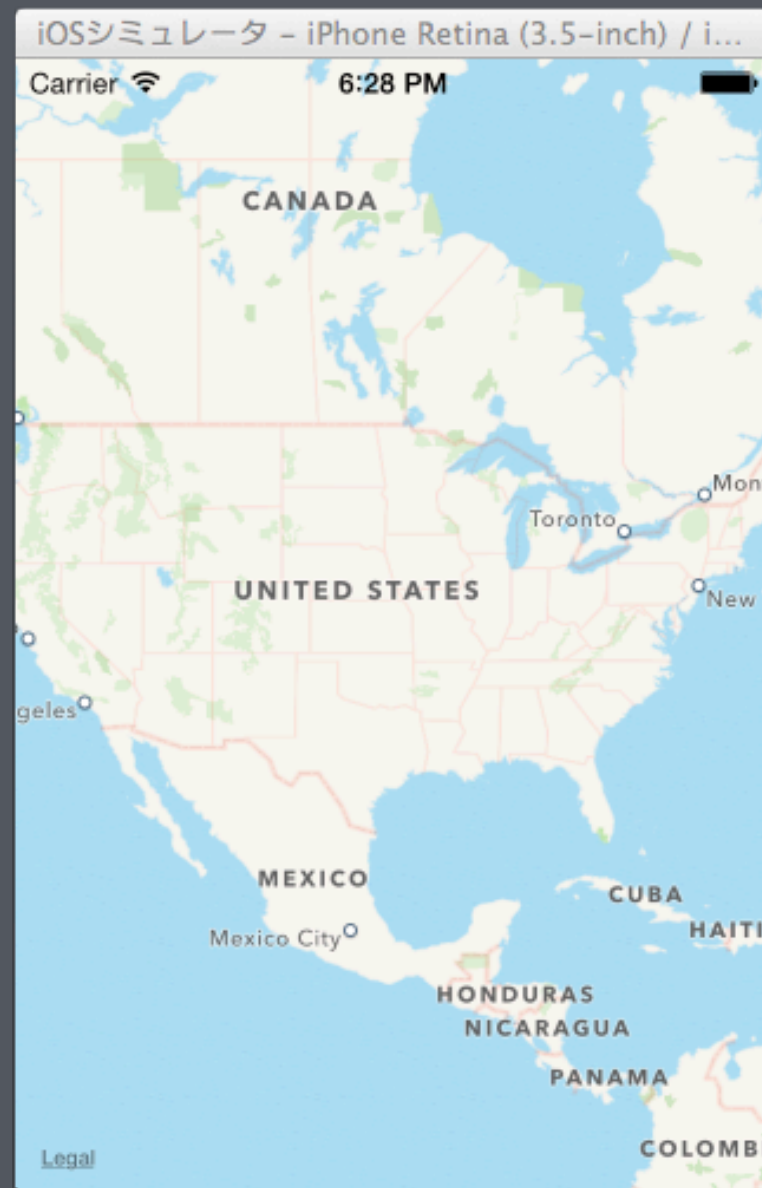
# [iOS]ViewRenderer を拡張する

```
//iOS:MyMapViewRenderer.cs
[assembly:ExportRenderer(typeof(MyMapView), typeof(MyMapViewRenderer))]

public class MyMapViewRenderer :
    ViewRenderer<MyMapView, MonoTouch.MapKit.MKMapView> {
    protected override void OnElementChanged (
        ElementChangedEventArgs<MyMapView> e) {
        base.OnElementChanged(e);

        // 独自のコントロールに置き換えちゃう
        SetNativeControl(new MonoTouch.MapKit.MKMapView());
    }
}
```

# [iOS,Android]実行結果



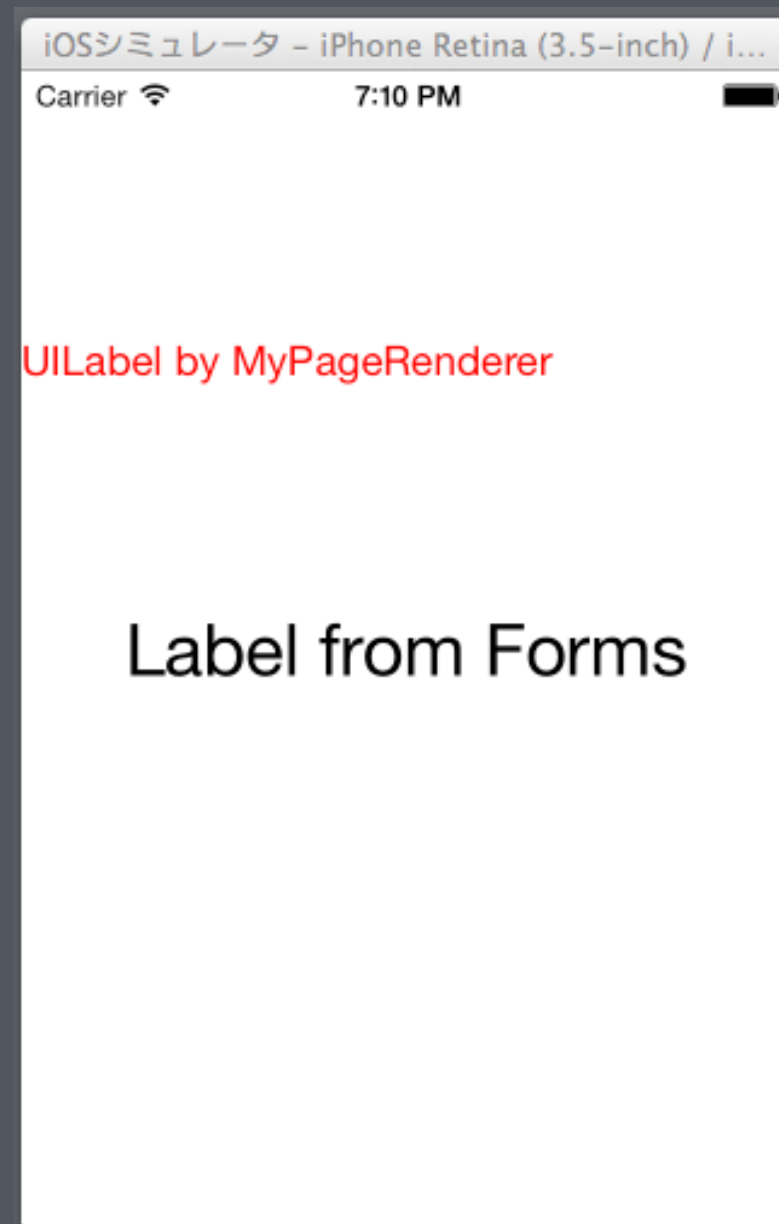
# Pageもさわれる(iOSでの例)

Forms の画面に、さらに UILabel を追加する

```
[assembly:ExportRenderer(typeof(SecondPage), typeof(MyPageRenderer))]
```

```
public class MyPageRenderer : PageRenderer {  
    protected override void OnElementChanged(  
        VisualElementChangedEventArgs e) {  
        base.OnElementChanged(e);  
  
        NativeView.Add (new UILabel(new RectangleF(0, 100, 300, 40)) {  
            Text = "UILabel from MyPageRenderer"  
        });  
    }  
}
```

# [iOS]実行結果



# Xamarin.Forms + ネイティブ画面開発

- ExportRenderer ととても便利
- ネイティブ側の実装も 同じ世界の中でできる！
  - XS or VS でデバッグできる
  - ネイティブAPIのラッパーのおかげ
  - Titanium や AIR ではこうはいかない



画面以外のプラットフォーム固有機能を使う

例： 「端末の名称」 を取得する

# [Forms]共通のインターフェースを定義する

```
public interface IDeviceInfo {  
  
    string DeviceName { get; }  
  
}
```

# [iOS]プラットフォーム側の実装

```
// DeviceInfo_iOS.cs
```

```
// これ重要！
```

```
[assembly: Xamarin.Forms.Dependency(typeof(DeviceInfo_iOS))]
```

```
public class DeviceInfo_iOS : IDeviceInfo {
```

```
    public string DeviceName {
```

```
        get {
```

```
            return UIDevice.CurrentDevice.Name; // iOSの実装
```

```
        }
```

```
    }
```

```
}
```

# [Andriid]プラットフォーム側の実装

```
// DeviceInfo_Android.cs
```

```
// これ重要！
```

```
[assembly: Xamarin.Forms.Dependency(typeof(DeviceInfo_Android))]
```

```
public class DeviceInfo_Android : IDeviceInfo {
```

```
    public string DeviceName {
```

```
        get {
```

```
            return Android.OS.Build.Device; // Android の実装
```

```
        }
```

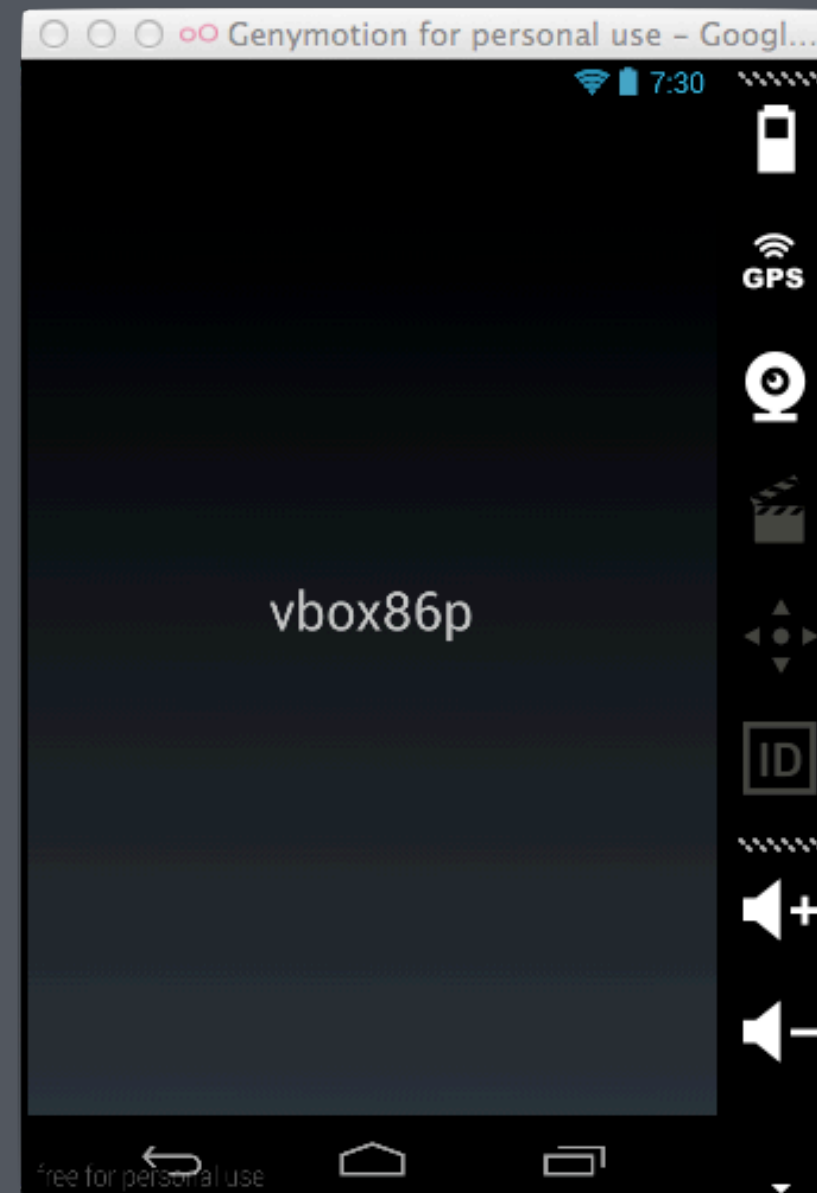
```
    }
```

```
}
```

# [Forms]IDeviceInfo を使う

```
public class App {  
    public static Page GetMainPage() {  
        var label = new Label { Text = "Get device name" };  
  
        // 実行しているプラットフォームの DeviceInfo を生成  
        // 対応していない時は null  
  
        var info = DependencyService.Get<IDeviceInfo>();  
        if (info != null)  
            label.Text = info.DeviceName;  
  
        return new ContentPage { Content = label };  
    }  
}
```

# [iOS, Android]実行結果



# 3. MVVM な機能



# ViewModel - INotifyPropertyChanged

```
public class MyViewModel : INotifyPropertyChanged {  
    public event PropertyChangedEventHandler PropertyChanged;  
  
    string _myName;  
    public string MyName {  
        get { return _myName; }  
        set {  
            if (_myName == value) return;  
            _myName = value;  
            // だいたい端折ってますが  
            PropertyChanged(this, new PropertyChangedEventArgs("MyName"));  
        }  
    }  
} ..続く..
```



# ViewModel - ICommand

...続き...

```
ICommand _resetCommand;  
public ICommand ResetCommand {  
    get {  
        return _resetCommand ?? (_resetCommand =  
            new Xamarin.Forms.Command(() => {  
                MyName = "xamariiiiin!!";  
            }));  
    }  
}
```

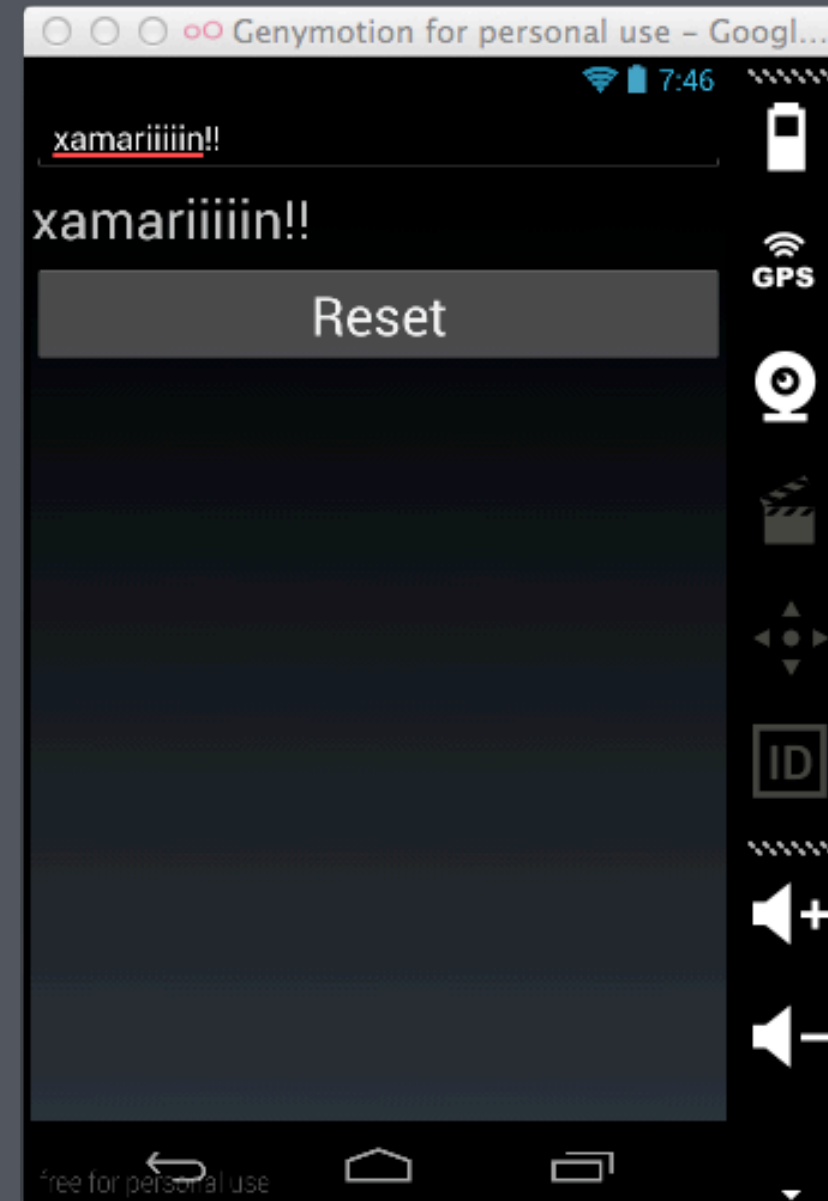
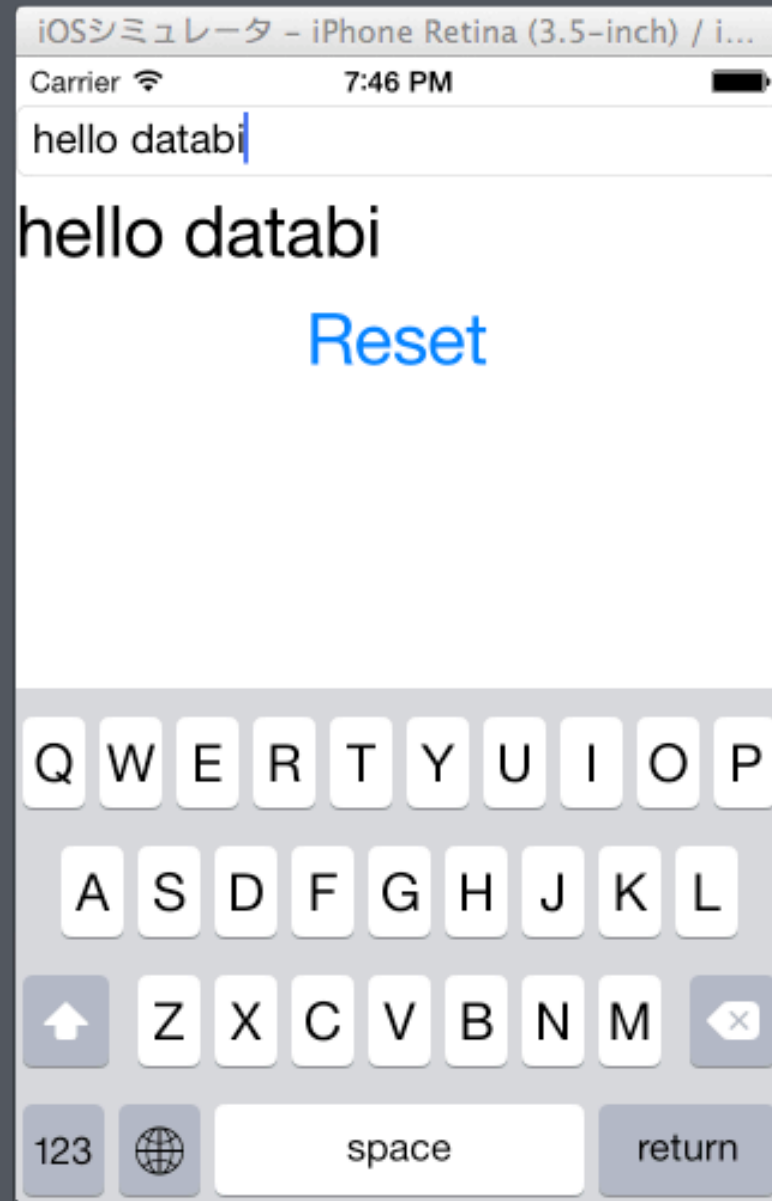
MS系のみなさまにはお馴染みのコードかと存じます。

# DataBinding

```
public class MyPage : ContentPage {  
    public MyPage() {  
        var editBox = new Entry { };  
        var label = new Label { };  
        var button = new Button { Text = "Reset" };  
  
        this.BindingContext = new MyViewModel();  
        editBox.SetBinding(Entry.TextProperty, "MyName");  
        label.SetBinding(Label.TextProperty, "MyName");  
        button.SetBinding(Button.CommandProperty, "ResetCommand");  
    }  
}
```

普通にできます。

# [iOS, Android]実行結果



# Messenger (MessagingCenter)

// メッセージ待ち受け側

```
MessagingCenter.Subscribe<TSender, string>(
    receiver, "messageId", (sender, param) =>
        Toast.MakeText(this, param, ToastLength.Long).Show());
};
```

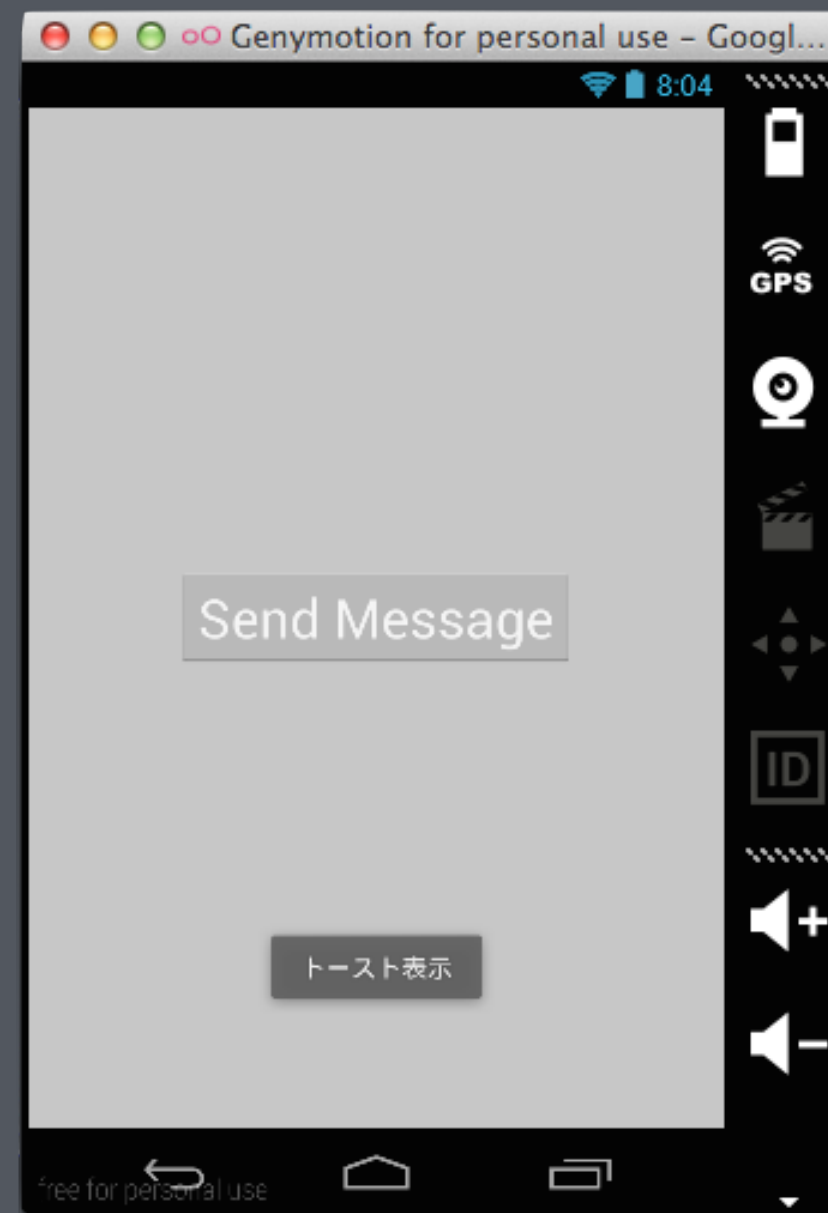
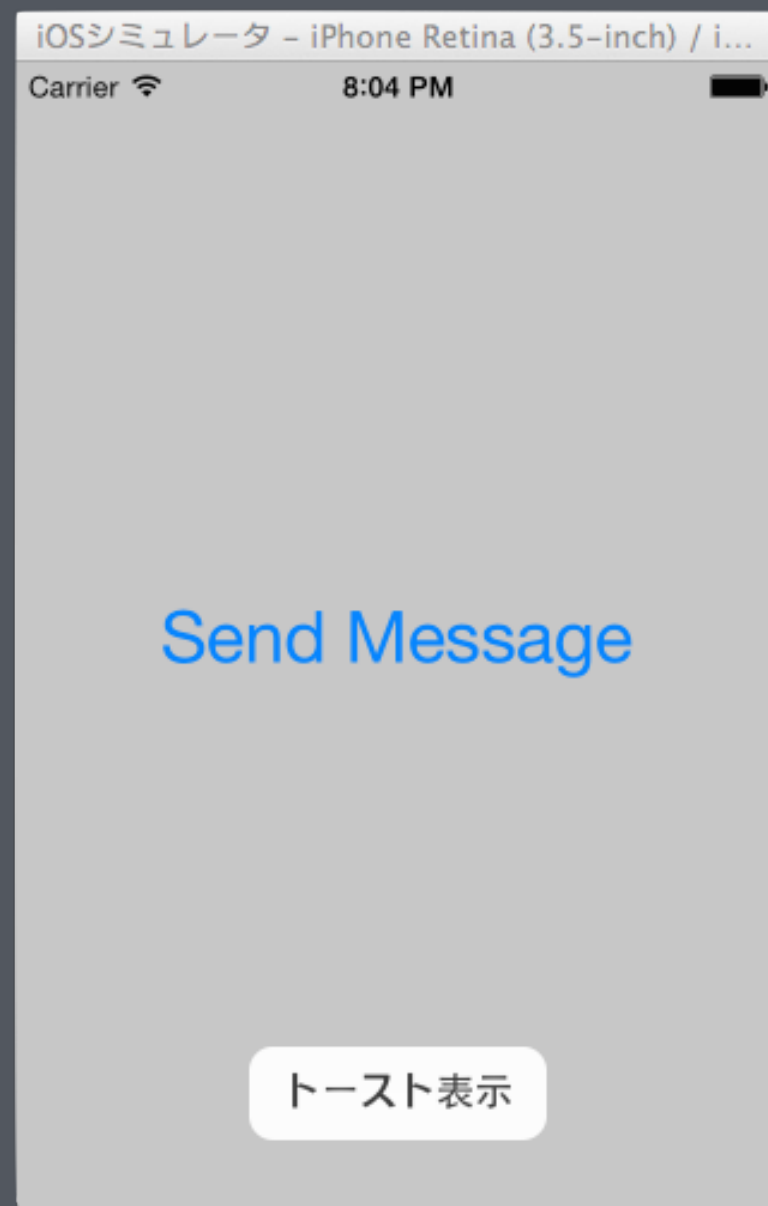
-----

// メッセージ送信側

```
MessagingCenter.Send(sender, "messageId", "param");
```

DisplayAlert など使われているもよう。

# [iOS, Android]実行結果



# 他の MVVM ライブラリと混ぜられる？

- MvvmCross とか
- ICommand, INotifyPropertyChanged なVMは使えそう
- FormsのVMは、Xamarin.Forms.Commandとか  
MessagingCenterなど使ってる可他ではダメ
- Modelも、DependencyServiceを使ってる可他ではダメ
- 上記を抽象化すれば使えそうだけど、そこまでやる？

# Forms 非対応PFとまぜられる？

- WPF、WinStore、Xamarin.Mac、Sil(ry
- Xamarin.Forms 名前空間を使ってる所はまぜられない
- 前頁の通り、ICommand, INotifyPropertyChanged なVMは使えるかも
- 実質、Model内のビジネスロジックくらい
- Windows Univarsal Apps はわかりません

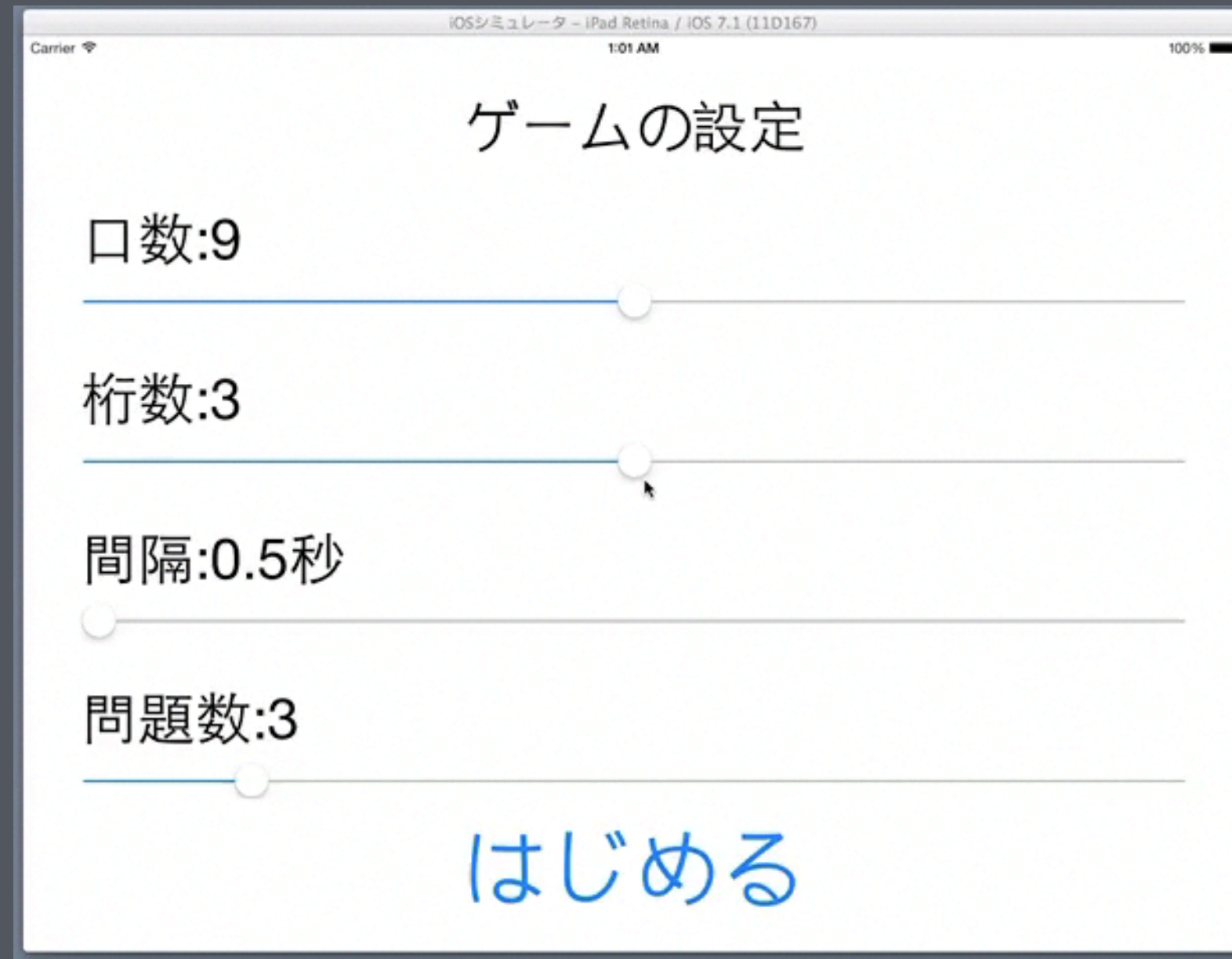
## 4. ためしにアプリを作ってみた



# PakaPakaCalc by Xamarin.Forms

- 「フラッシュ暗算」のアプリ
- iPhone、iPad、Android 対応(WPもってない)
- 画面はすべて Forms 側で実装
- XAML化に備えて MyView.cs と MyView.designer.cs に分離
- アプリケーション情報の保存は PCLStorage + JSON.NET

# DEMO



# 4. まとめ

# Xamarin.Forms まとめ

- iOS/Android/WP で、ワンソースでUIを構築できる
- 足りない機能もあるけど、拡張がとても簡単
- MVVM が普通に使える
- ロックイン注意、というか「モバイル向け」と割りきって
- はやくUIデザイナーおねがいします

# Links

- [Xamarin.Forms | Xamarin](#)
- [Xamarin.Formsの基本構想と仕組み - Build Insider](#)
- [Xamarin逆引きTips - Build Insider](#)
- [xamarin/xamarin-forms-samples](#)
- [XForms/Xamarin-Forms-Labs](#)
- [NuGet Gallery | Xamarin.Forms](#)

# ありがとうございました

この資料は Markdown + Deskset で作成しました

資料およびサンプルは [github.com/amay077/jxug\\_1\\_xamarin\\_forms\\_talk](https://github.com/amay077/jxug_1_xamarin_forms_talk)

# おまけ：Tips じゃない Tips

- NuGet から頻繁に更新を（今の最新は 1.1.1.6206、プロジェクト作成直後は古い）
- 画面サイズから文字サイズをスケーリング(手動)
- iOSのステータスバーの為の `Device.OnPlatform(20,0,0)`  
...ひどい
- Xamarin.Formsでキーボードに隠れないようにレイアウトをずらすハック by @WilfremLuminous さん