Microsoft

MW08

de:code イベントアプリの作り方 ～ Xamarin.Forms で作っています ～

株式会社ジェーエムエーシステムズ
事業企画部 事業企画グループ
プロダクトスペシャリスト

川合　俊介

de:code

自己紹介

川合 俊介（かわい しゅんすけ）

株式会社ジェーエムエーシステムズ
　事業企画部 事業企画グループ
　プロダクト スペシャリスト

JMAS
JMA SYSTEMS Corporation

shunsuke.kawai.777

@_shunsuke_kawai

de:code

セッションゴール

- Xamarin.Forms でアプリを作るのは
  そんなに難しくない
- Xamarin.Forms で 開発する時の方法/テクニック
- Xamarin.Forms の向き不向きを感じてもらう
- Xamarin.Forms で開発したくなる

Agenda

- What's Xamarin
- Event app overview
- Xamarin.Forms UI
- Azure Mobile Engagement
- Native Binding

What's Xamarin

Xamarin とは

- C# によるクロスプラットフォーム開発環境。
- 2001 年に Mono プロジェクトとして発足し、後に Xamarin 社を設立、2016 年に Microsoft 社に買収された。
- 現在は Visual Studio に同梱され、OSS 化、ライセンスの無料化がされている。

⇒Xamarin(ザマリン) とはなんぞや

- http://qiita.com/amay077/items/38ee79b3e3e88cf751b9

2 つの開発手法

Traditional Xamarin approach (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

iOS C# UI

Android C# UI

Windows C# UI

Shared C# App Logic (PCL)

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)

2 つの開発手法

Traditional Xamarin approach (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

iOS C# UI

Android C# UI

Windows C# UI

Shared C# App Logic (PCL)

イベントアプリはこちらを採用

Xamarin.Forms

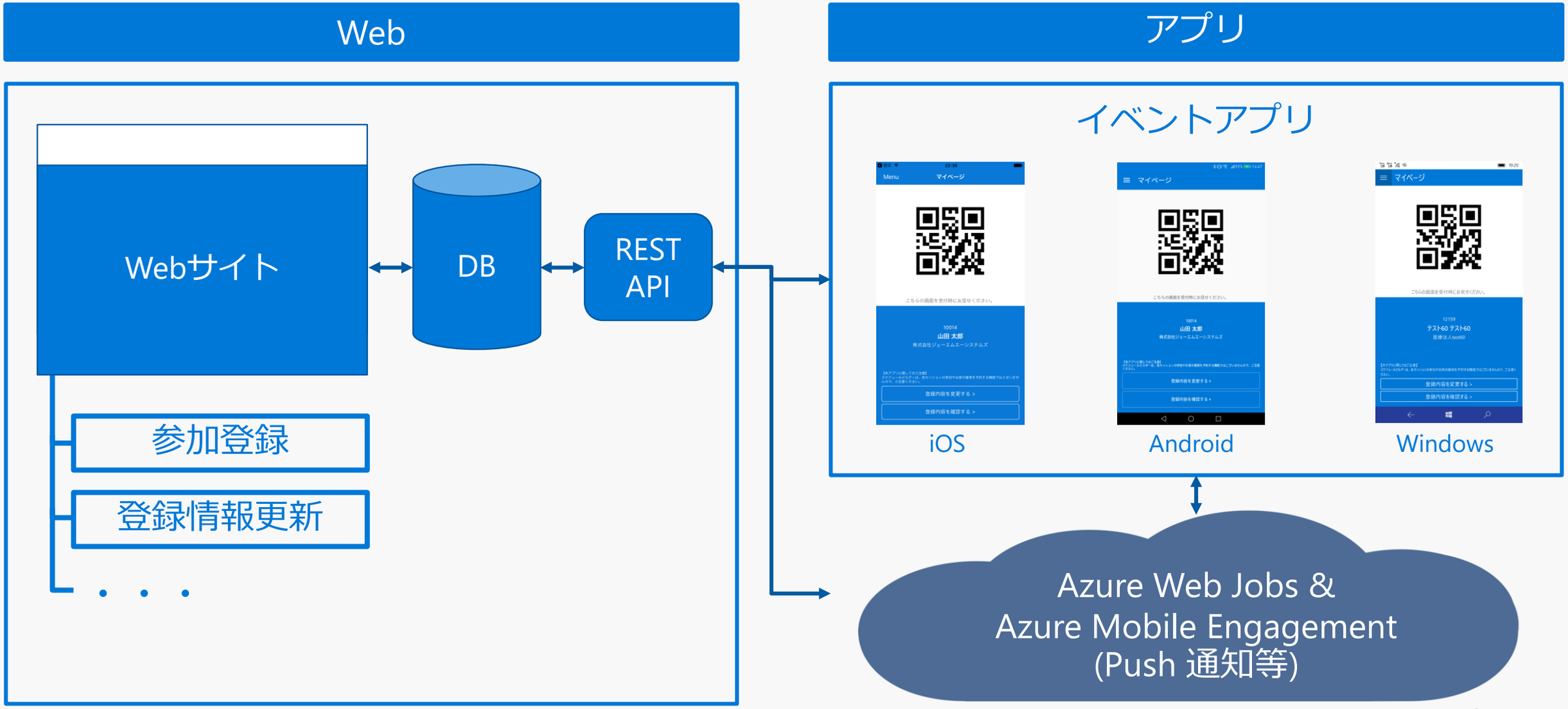
ロジックと UI を共通化
UI は各プラットフォームの
同じ役割の UI が自動マッピング

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

Shared C# App Logic (PCL)

Event app overview

イベントアプリシステム構成



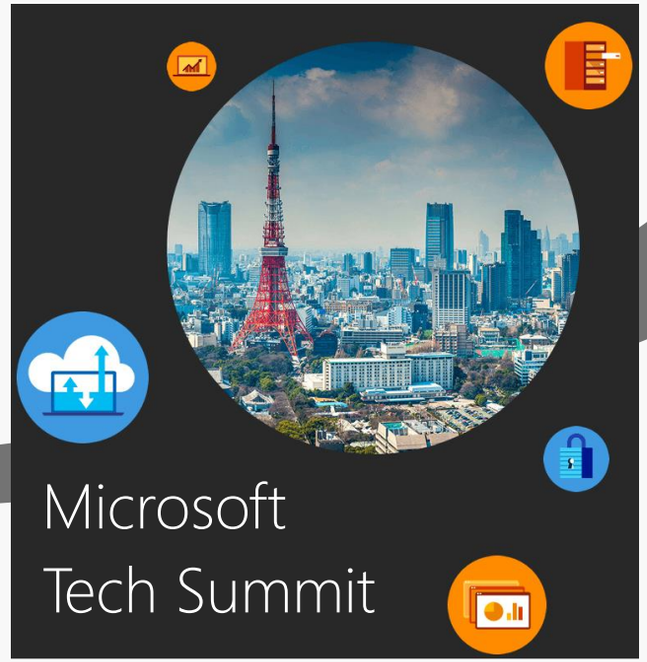
de:code

Xamarin.Forms を採用した理由

- アプリの特性上、複雑な UI、OS 独自の機能を使用する必要はあまり~~ない~~ なかった
- 開発メンバーは Windows 系開発者で iOS/Android の開発は未経験
- スケジュールがタイトだったため、できる限り共通化したい

公式イベントアプリの歴史

2016/9/6-7



2016/11/1-2

2017/5/23-24

イベントアプリ開発環境

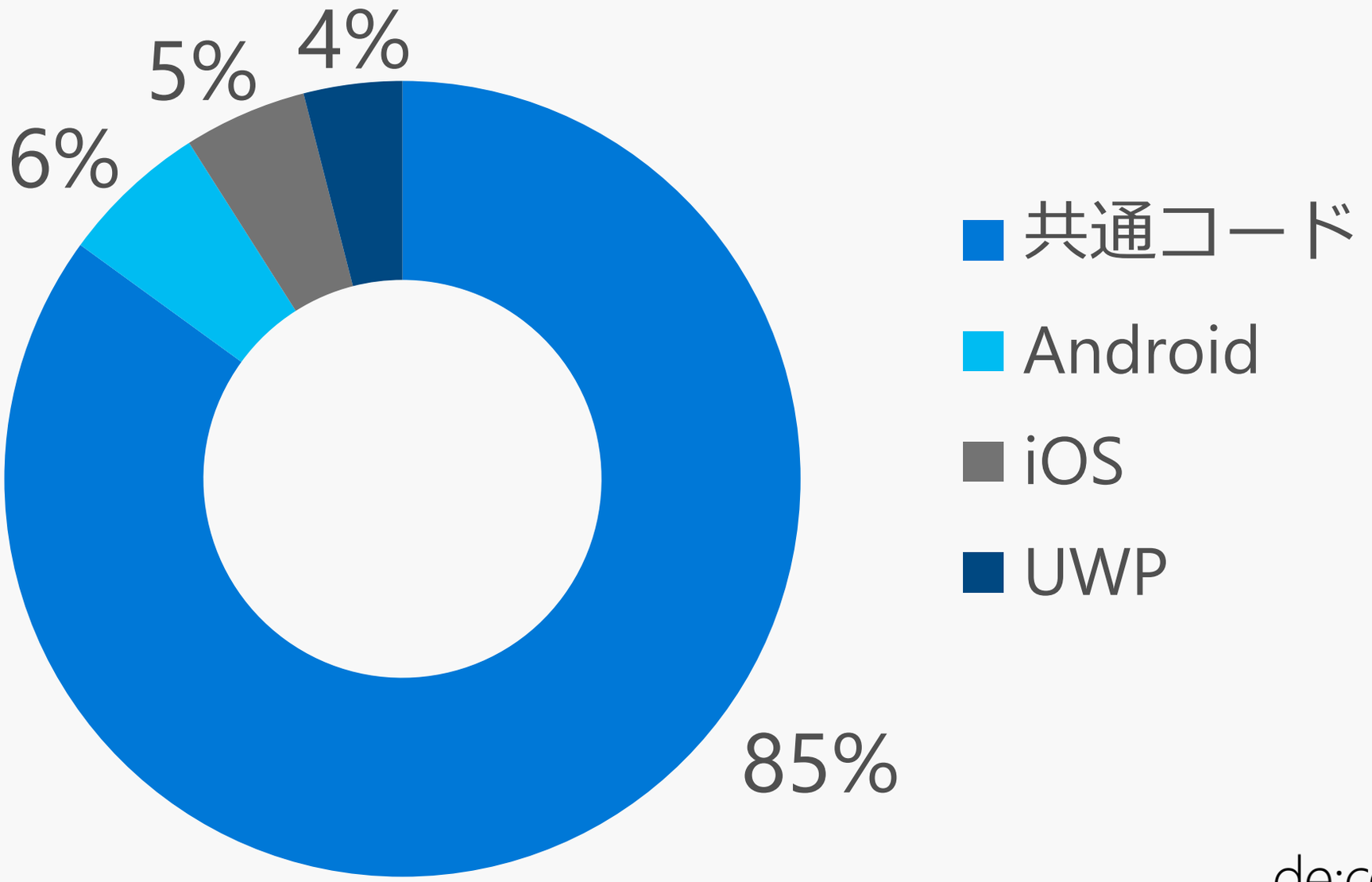
- Windows 10 Ver.1703
- Visual Studio Enterprice 2017 Ver.15.2 (26430.6)
- Xamarin.Forms Ver.2.3.4.247

- Prism Unity App(Xamarin.Forms) Ver.6.3.0

  ⇒ Prism for Xamarin.Forms入門

  http://www.nuits.jp/entry/2016/08/22/173858

コード共有率

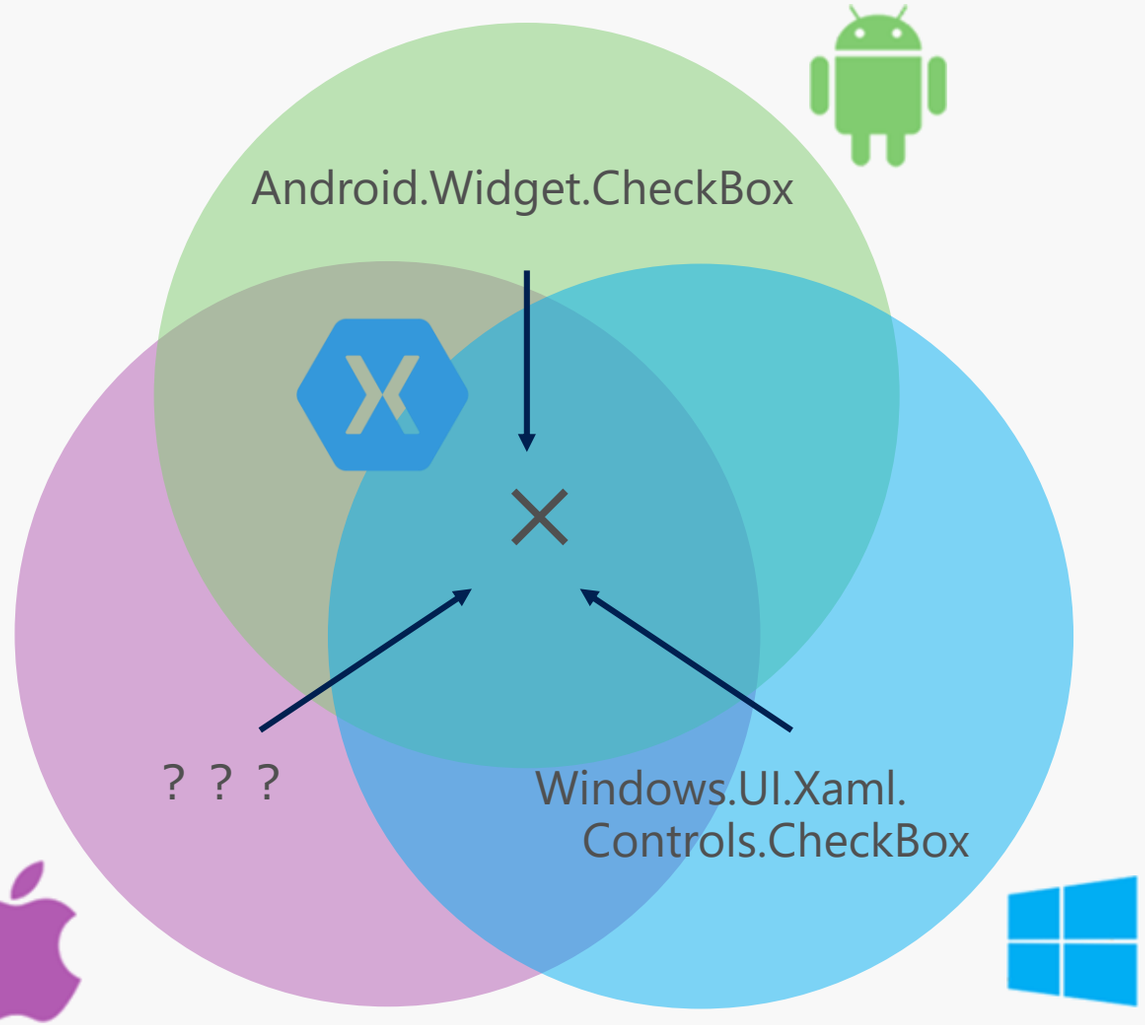


Xamarin.Forms UI

de:code

Xamarin.Forms の標準コントロール

各プラットフォームの最大公約数



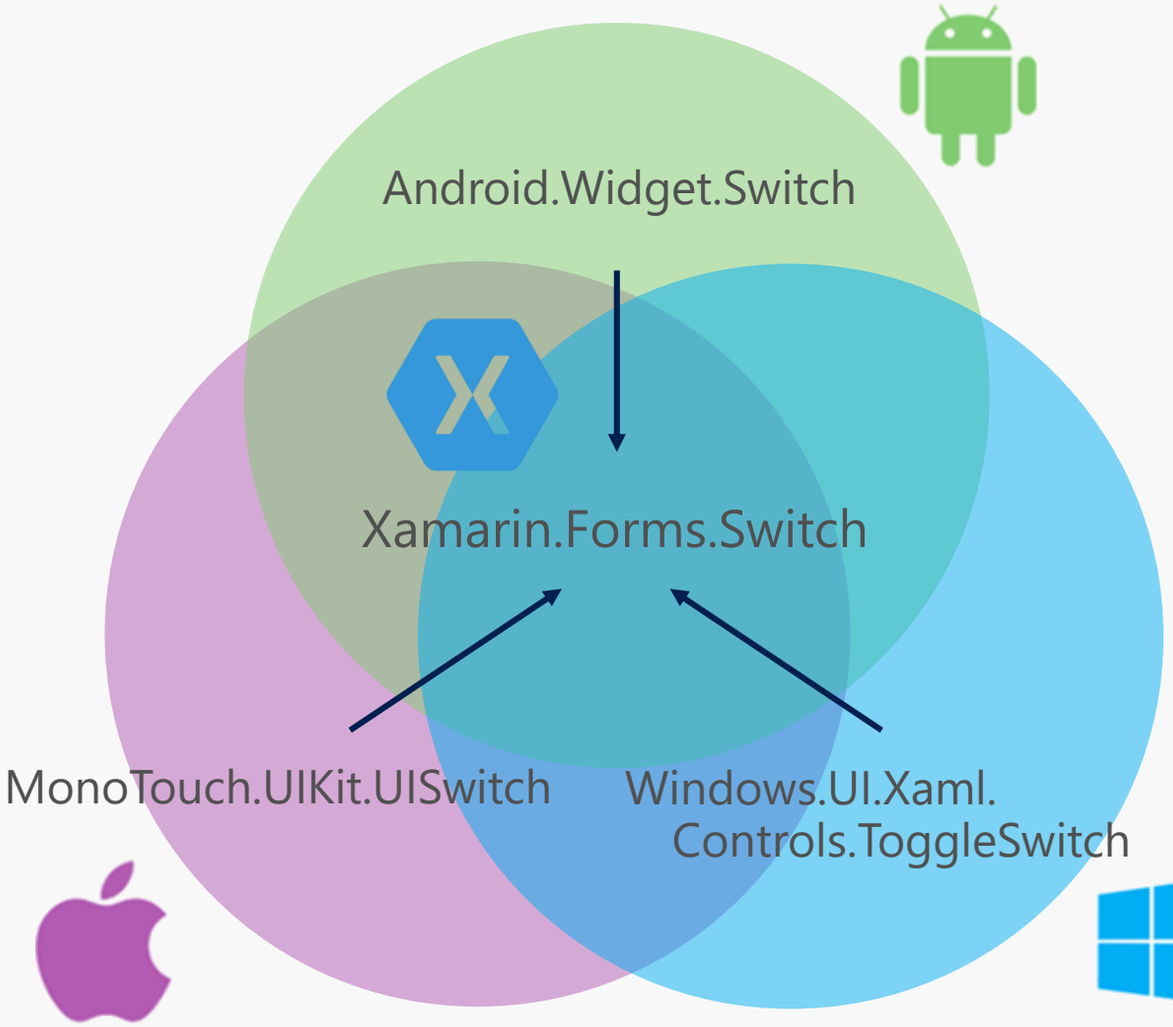
Check Box の場合

⇒iOS にはない

⇒Xamarin.Forms 標準コントロールにはない

じゃあどうするか？

⇒イベントアプリでは Switch で代替しました。

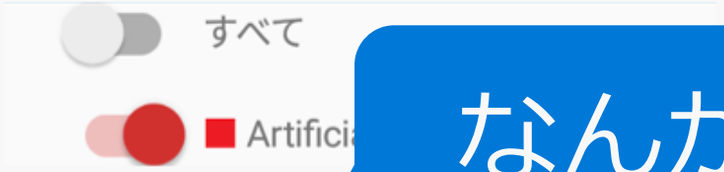
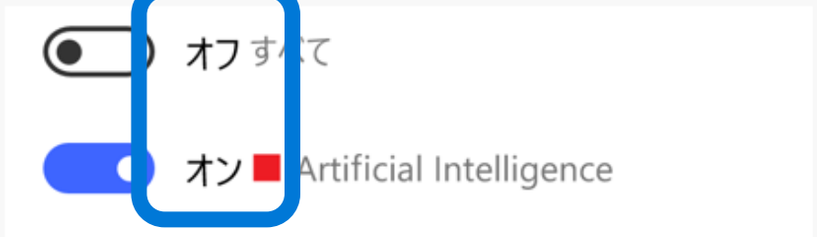


iOS	すべて Artificial Intelligence
Android	すべて Artificial Intelligence
UWP	オフ すべて オン Artificial Intelligence

ほとんど標準コントロールで大丈夫！

そう思っていた時期が私にもありました。

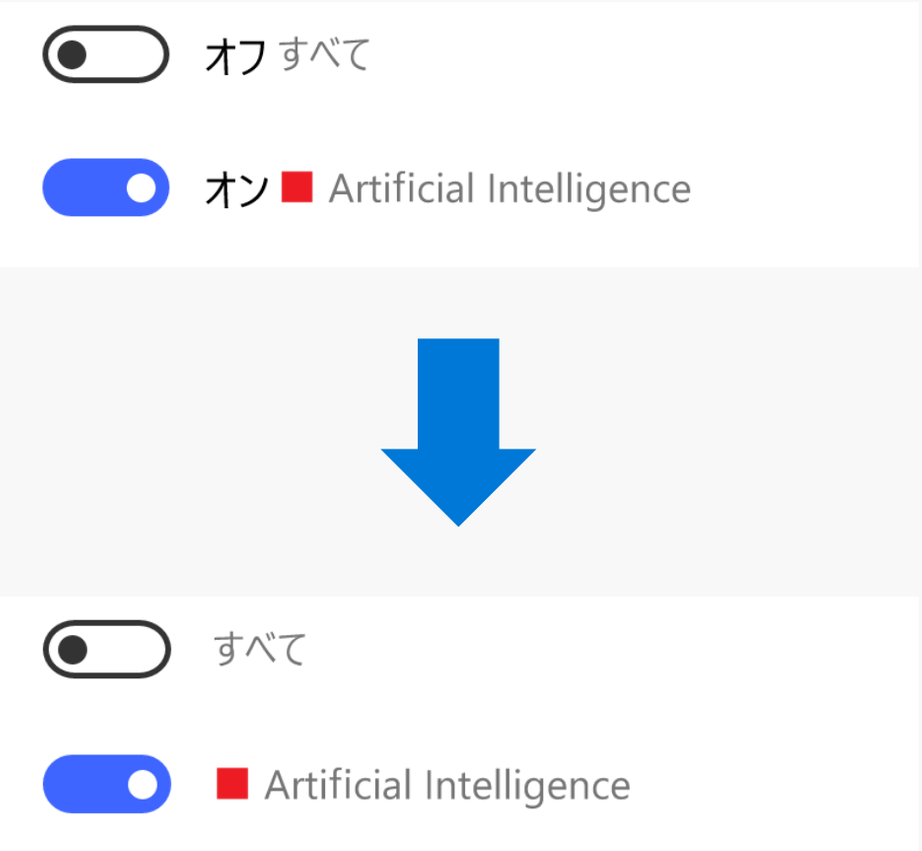
UWP の ToggleSwicth

iOS	
Android	
UWP	

なんか表示されてる…

イベントアプリでは

```
[assembly: ExportRenderer(typeof(Switch),
typeof(ToggleSwitchRenderer))]
namespace JMAS.MicrosoftDeCode2017.UWP.Renderers
{
    public class ToggleSwitchRenderer : SwitchRenderer
    {
        protected override void
OnElementChanged(ElementChangedEventArgs<Switch> e)
        {
            base.OnElementChanged(e);
            if (Control != null)
            {
                Control.OnContent = string.Empty;
                Control.OffContent = string.Empty;
            }
        }
    }
}
```



見た目はカスタマイズが必要

Xamarin.Forms の各プラットフォーム向けカスタマイズ手法

イベントアプリで主に使用している手法

手法	概要
Custom Renderer	• コントロールのクラスに適用 • カスタムコントロール的
Effects	• コントロールのインスタンスに適用 • Custom Renderer より手軽に利用可能
DependencyService	• 各プラットフォーム固有の機能を利用
Behaviors	• 機能を添付

サンプルアプリ

https://github.com/shunsuke-kawai/decode2017_MW08
(http://bit.ly/2rI6Lth)

Github 上で decode2017_MW08 で検索が早い

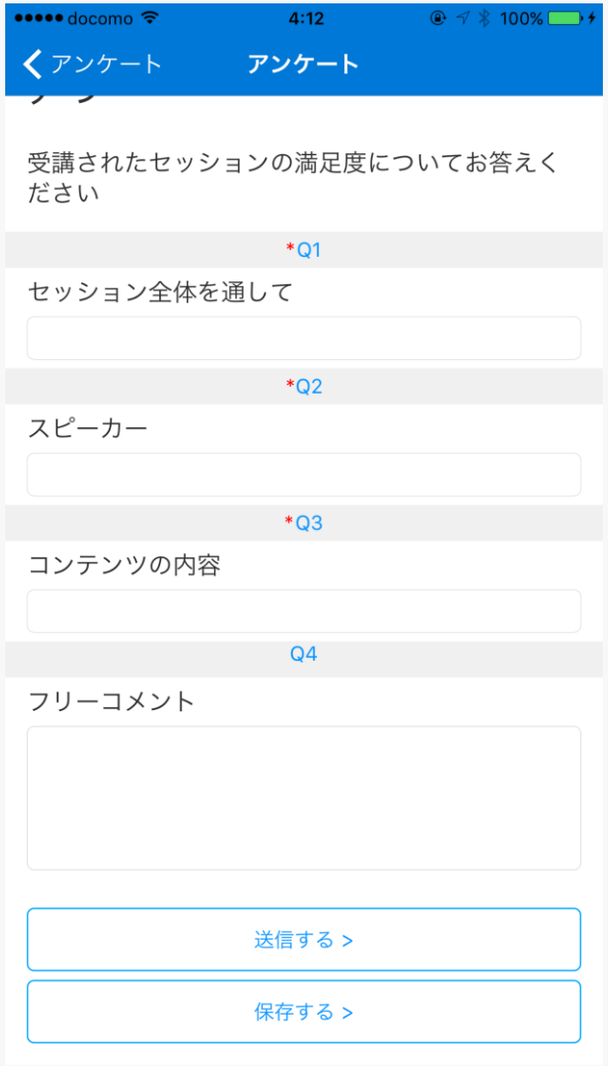
Custom Renderer
複数行テキスト 枠線

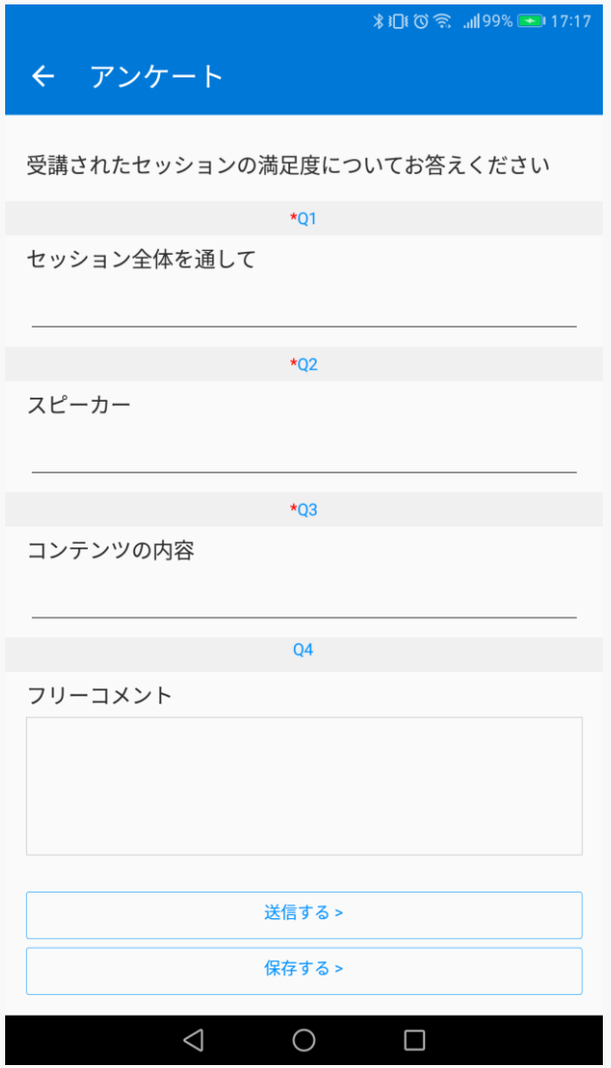
CustomEditorRenderer.cs

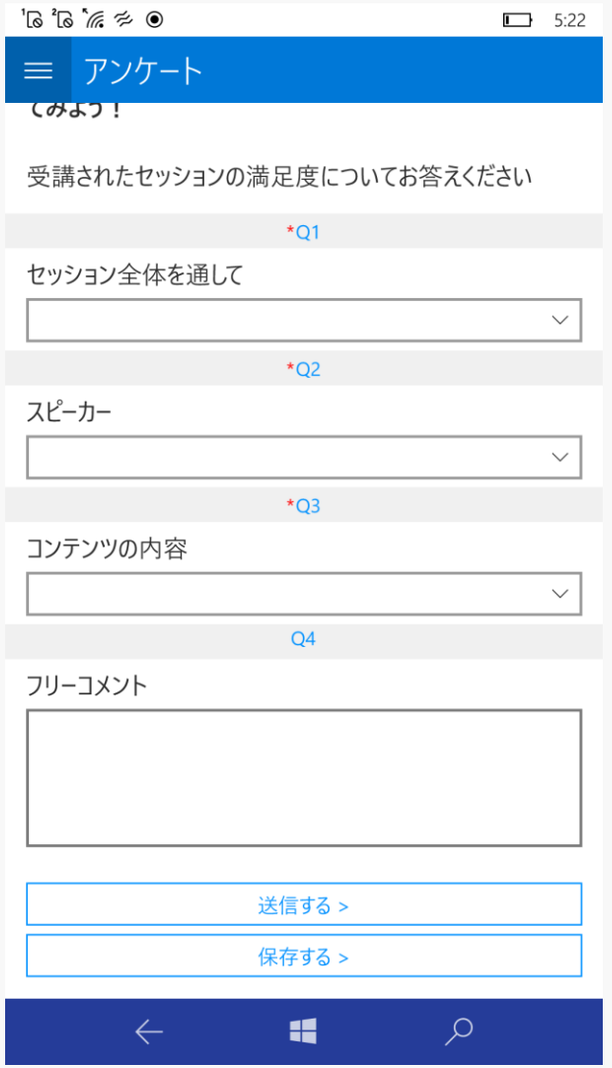
```
[assembly: ExportRenderer(typeof(CustomEditor), typeof(CustomEditorRenderer))]
namespace decode2017_MW08.Droid.Renderers
{
    public class CustomEditorRenderer : EditorRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<Editor> e)
        {
            base.OnElementChanged(e);
            var el = (CustomEditor)this.Element;
            var nativeEditText = (global::Android.Widget.EditText)Control;

            var shape = new ShapeDrawable(new Android.Graphics.Drawables.Shapes.RectShape());
            shape.Paint.Color = el.BorderColor.ToAndroid();
            shape.Paint.SetStyle(Paint.Style.Stroke);
            nativeEditText.Background = shape;
        }
    }
}
```

アンケート回答フリーテキスト







de:code

Custom Renderer
背景色透明 WebView

CustomWebViewRenderer.cs

```
[assembly: ExportRenderer(typeof(WebView), typeof(CustomWebViewRenderer))]
namespace decode2017_MW08.Droid.Renderers
{
    public class CustomWebViewRenderer : WebViewRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<WebView> e)
        {
            base.OnElementChanged(e);

            if (Element == null) return;

            Control.SetBackgroundColor(Element.BackgroundColor.ToAndroid());
            Control.ClearCache(true);
        }
    }
}
```

DependencyService
Local ファイル参照

HtmlPath_Droid.cs

```
public string GetHtmlPath()
{
    return "file:///android_asset/";
}
```

HtmlPath_UWP.cs

```
public string GetHtmlPath()
{
    return "ms-appx-web:///";
}
```

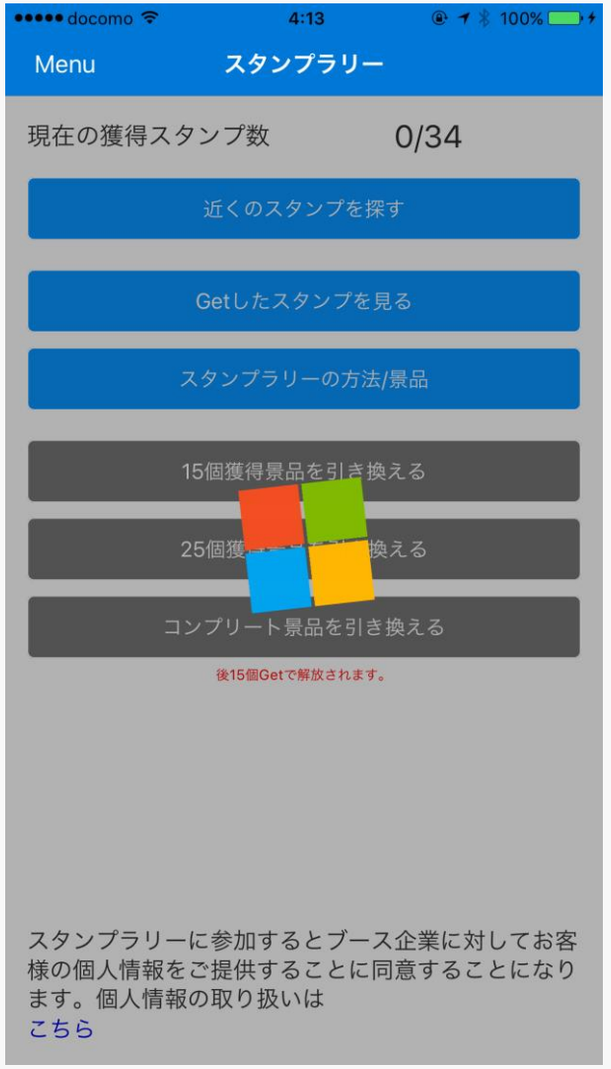
HtmlPath_iOS.cs

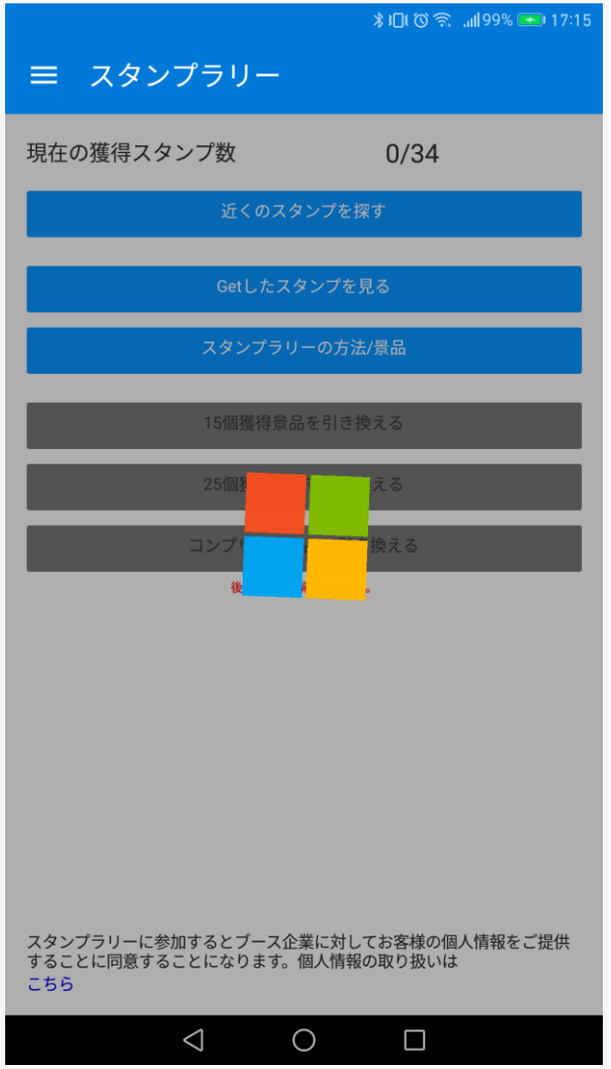
```
public string GetHtmlPath()
{
    return NSBundle.MainBundle.BundleUrl.ToString();
}
```

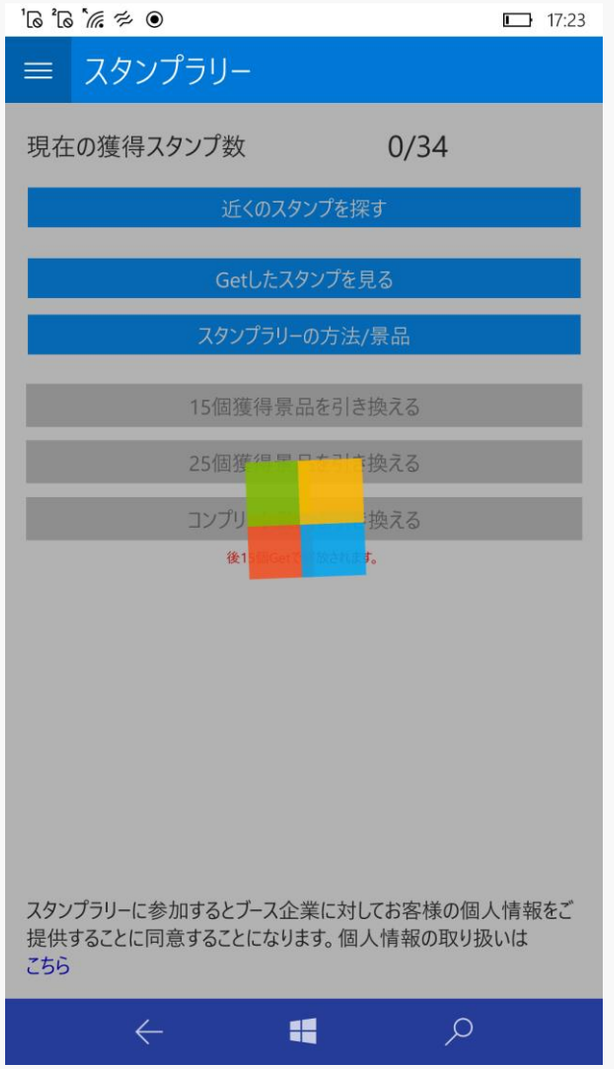
CustomIndicator.xaml.cs

```
public CustomIndicator()
{
    InitializeComponent();
    _htmlPath = DependencyService.Get<IHtmlPath>().GetHtmlPath() + "Sample_Spinner.html";
}
```

Custom Loading







de:code

Effects

Highlight 無効 ListView

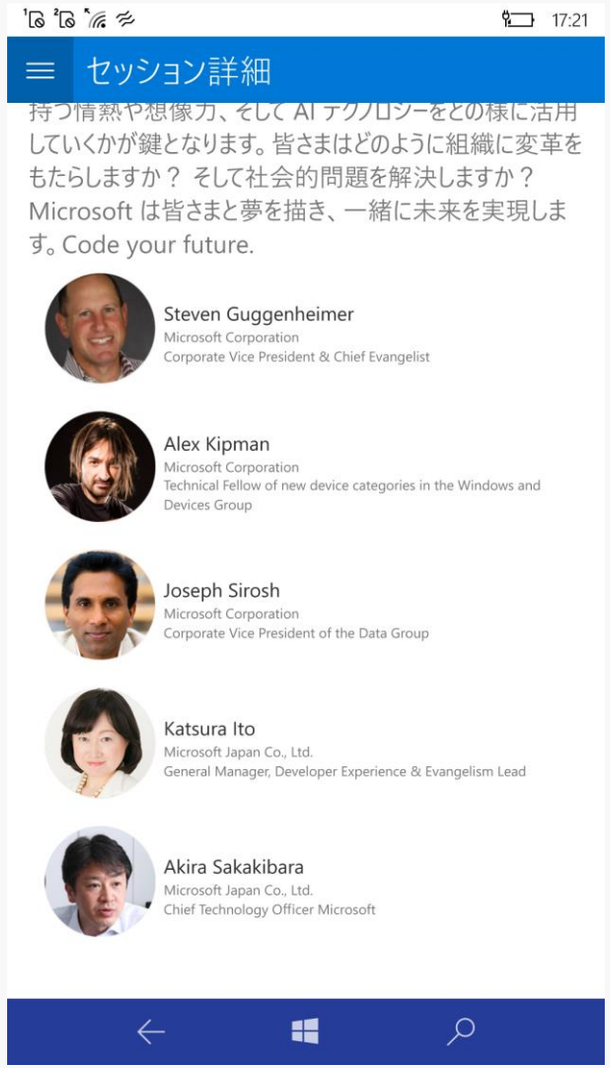
CustomIndicator.xaml.cs

```
[assembly: ResolutionGroupName("Xamarin")]
[assembly: ExportEffect(typeof(ListViewHasNoHighlightEffect), "ListViewHasNoHighlightEffect")]
namespace decode2017_MW08.Droid.Effects
{
    public class ListViewHasNoHighlightEffect : PlatformEffect
    {
        protected override void OnAttached()
        {
            var listView = Control as AbsListView;
            if (listView == null) return;
            listView.SetSelector(Resource.Drawable.NoHighlightViewCellBackground);
        }

        protected override void OnDetached()
        {
        }
    }
}
```

de:code

スピーカー一覧



de:code

Behaviors
指定アイテムまで自動スクロール ListView

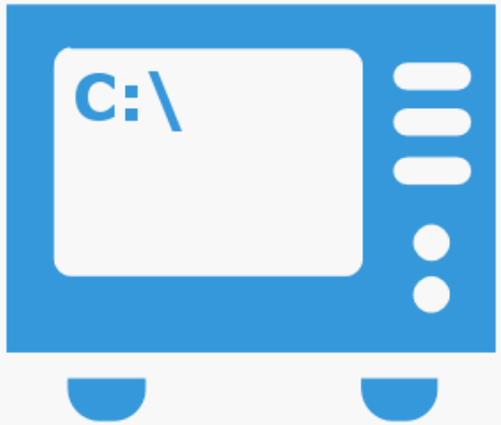
ScrollToBehavior.cs

```
public class ScrollToBehavior : BehaviorBase<ListView>
{
    public static readonly BindableProperty ScrollToItemProperty =
        BindableProperty.Create("ScrollToItem", typeof(object),
            typeof(ScrollToBehavior), null, BindingMode.TwoWay, null,
            ScrollToBehavior.OnScrollToItemPropertyChanged, null, null);

    private static void OnScrollToItemPropertyChanged(BindableObject bindable,
                                                      object oldValue, object newValue)
    {
        ～～略～～
    }
}
```

スタンプラリー

- 下記リポジトリの nuget パッケージを使用
  - https://github.com/microwavePC/Beahat
  - iBeaconを簡単に検知すること、またその結果をトリガーとすることが可能



その他

- 凹むボタン
  - 単純なカスタムコントロールだが、画像をボタンにしたい時に便利

UWP Xamarin.Forms の 2.3.4.247 時点バグ

- Alt 押すと ListView の文字が消える
  - **UWP label disappeared in ListView if update from view model (resize window will show again)**
  - https://bugzilla.xamarin.com/show_bug.cgi?id=44973

- Release ビルドで OnPlatform が効かない
  - **OnPlatform doesn't work on UWP when compiling with .NET Native**
  - https://bugzilla.xamarin.com/show_bug.cgi?id=55636

Xamarin.Forms UI

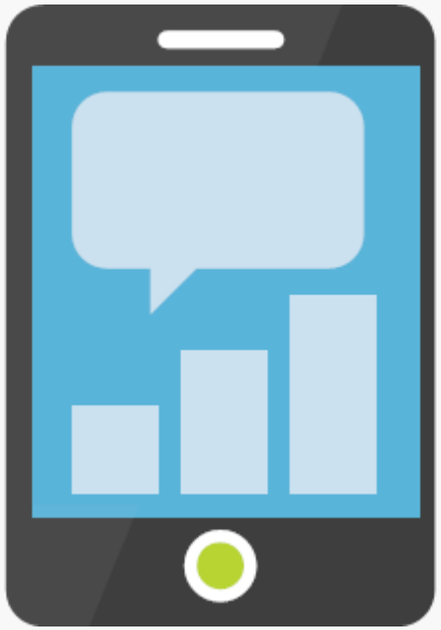
- 事前に検知できるケースはあまりない
  - 各プラットフォームで動作確認をして発覚するケースがほとんどだった
- 最初は難しそうと思っていたが、一回作ってみると仕組みが理解できる（ただし奥は深い）
- 各プラットフォームのネイティブの知識はやっぱり必要
  - 調べる時はネイティブ開発で調べてそれを C# に変換

Azure Mobile Engagement

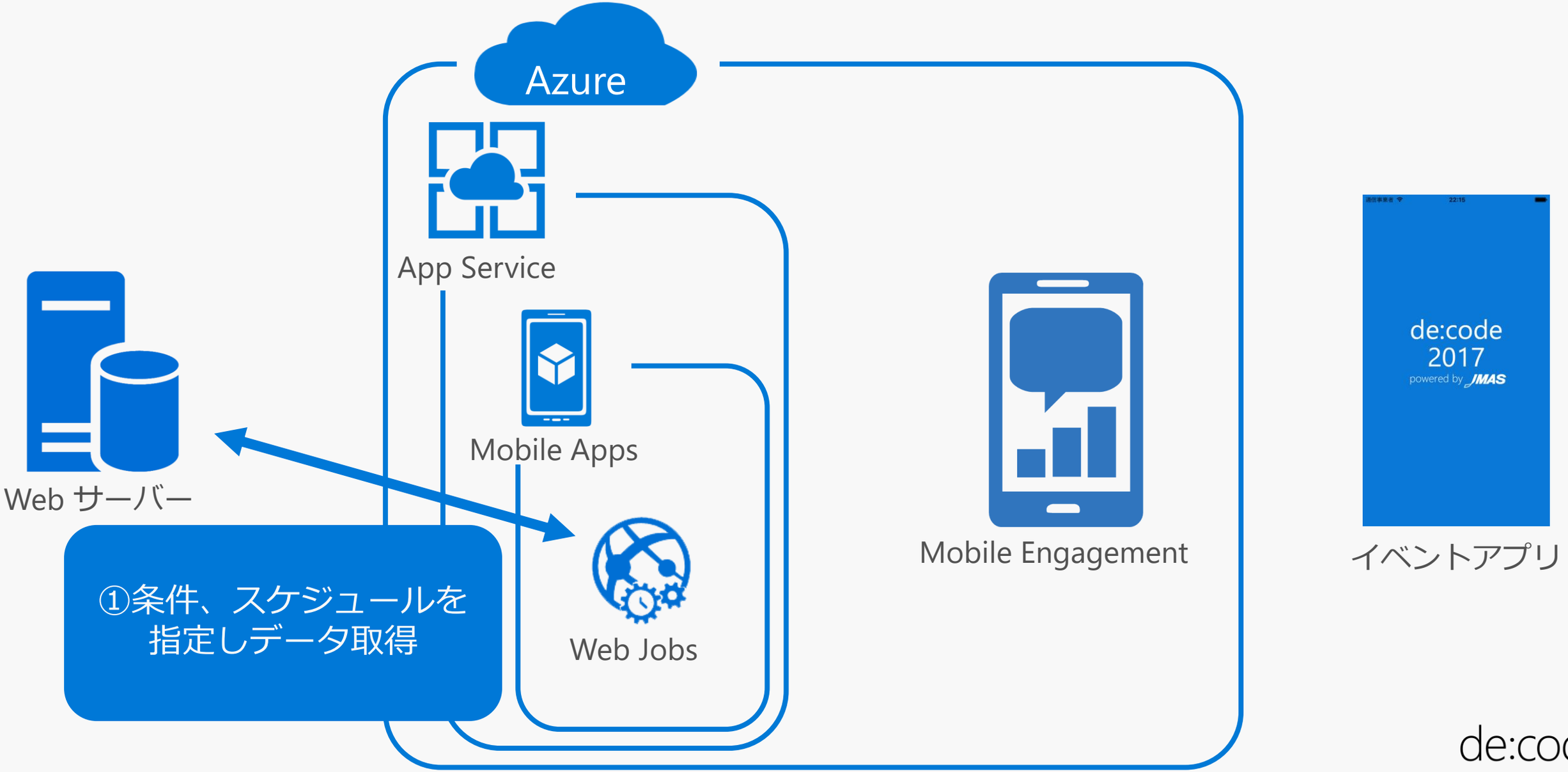
de:code

Azure Mobile Engagement

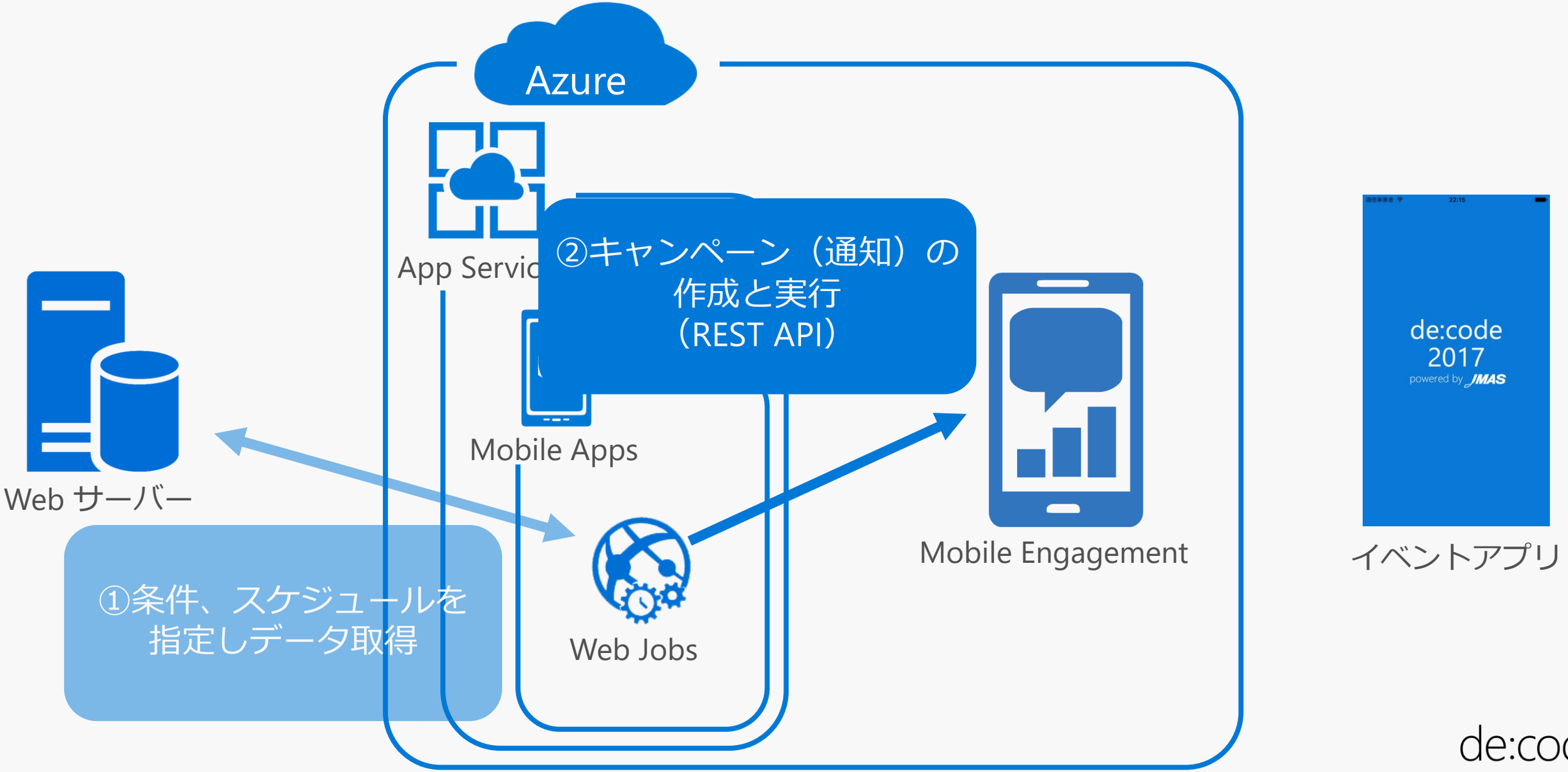
- データ主導のアプリ使用状況分析、リアルタイムでのユーザーのセグメント化、コンテキスト感知のプッシュ通知とアプリ内メッセージングが可能



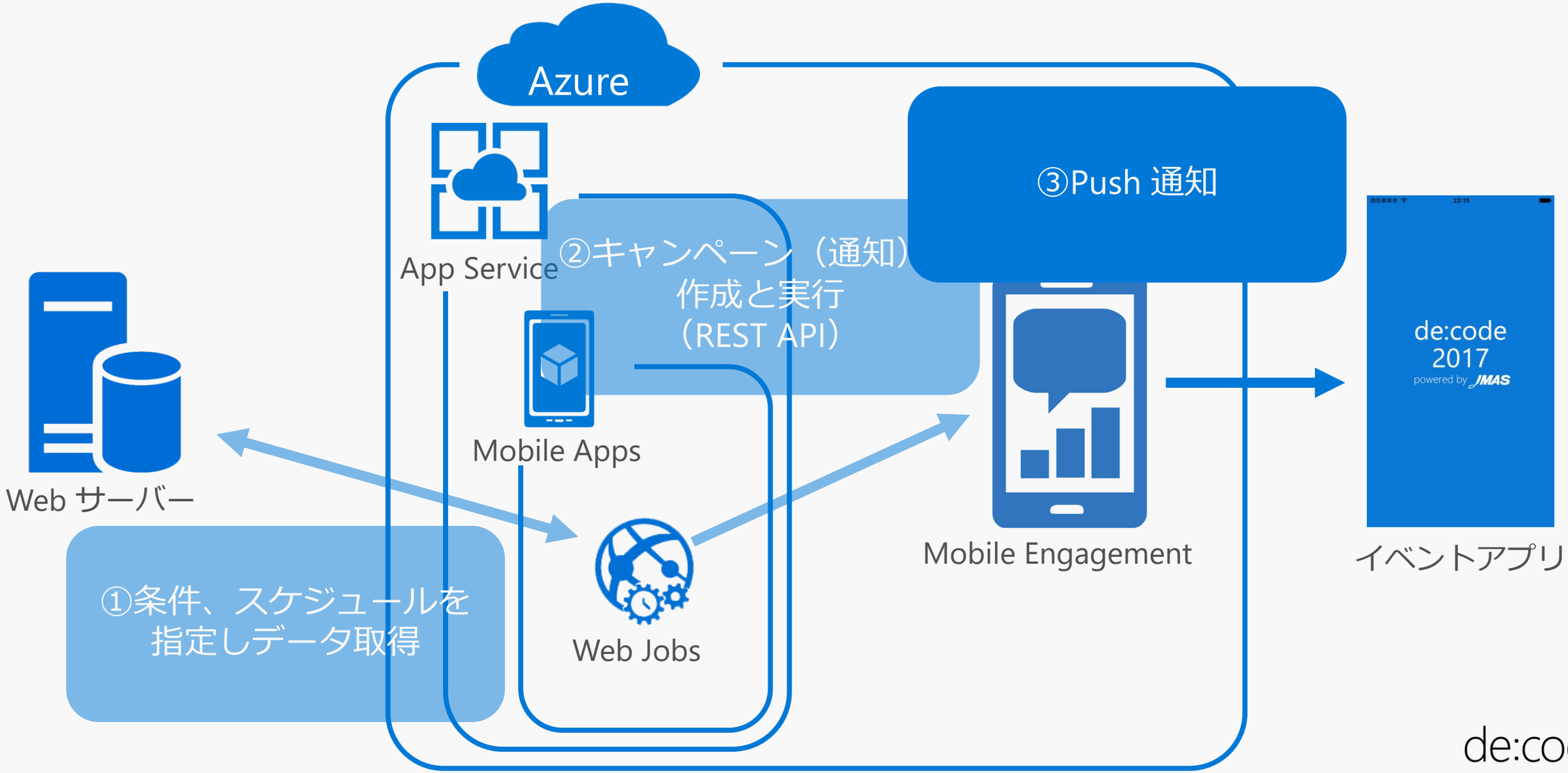
イベントアプリ Push 通知詳細



イベントアプリ Push 通知詳細

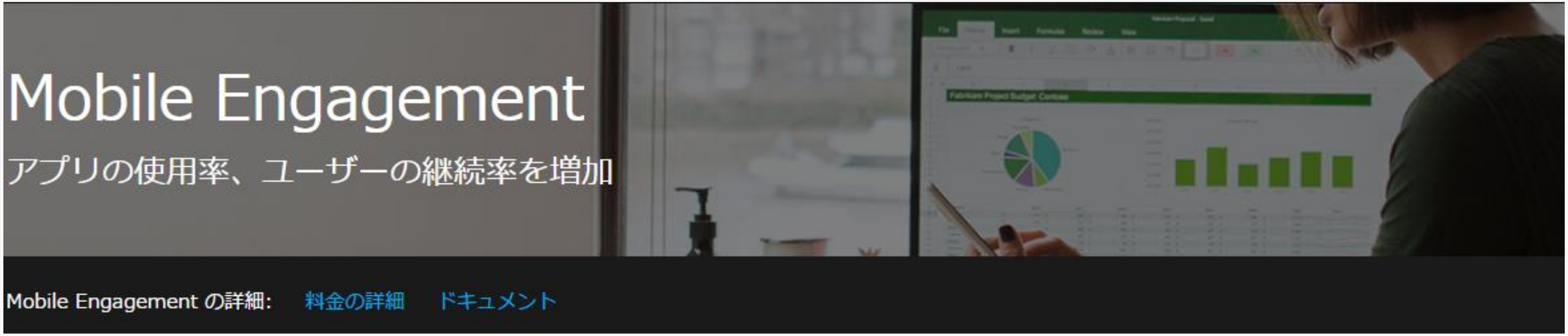


イベントアプリ Push 通知詳細



Demo

Azure Mobile Engagement



Mobile Engagement

アプリの使用率、ユーザーの継続率を増加

Mobile Engagement の詳細: 料金の詳細 ドキュメント

Azure Mobile Engagement サービスは停止となりますが、開発者がモバイル マーケティングに使用できるそのほかの強力かつ柔軟なサービスをご紹介します。

- HockeyApp クラッシュ レポートとユーザー メトリックスを、アプリ配布とユーザー フィードバックのプラットフォームに統合します。
- Notification Hubs アプリケーション開発者はターゲットを絞ったプッシュ通知を配信できます。
- Visual Studio Mobile Center モバイル アプリ管理戦略の要となる Visual Studio Mobile Center は、高度な分析、クラッシュ レポート、プッシュ通知、アプリ配布などを統合します。

Azure Mobile Engagement サービスは 2018 年 3 月に停止予定であり、現在は既存のお客様のみご利用いただけます。

https://azure.microsoft.com/ja-jp/services/mobile-engagement/

re Mobile Engagement サービスは 2018 年 3 月に停止予定であり

https://azure.microsoft.com/ja-jp/services/mobile-engagement/

Native Binding

Native Binding

- Android、iOS の Native ライブラリを Xamarin のアプリから呼び出せるようにする
  - https://developer.xamarin.com/guides/android/advanced_topics/binding-a-java-library/
  - https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/

Beacapp

Beacapp とは

**アプリの Beacon 対応を簡単・迅速に実現し、
現地でのユーザー体験を演出・効果測定できるクラウドサービスです。**

Beacon 対応することで

GPS よりもさらに近距離・高精度の位置情報を利用することができます。

Beacapp を活用すると

新規開発、既存アプリの対応にかかる費用や期間を大幅に圧縮できます。
コンテンツの変更反映も容易に可能。

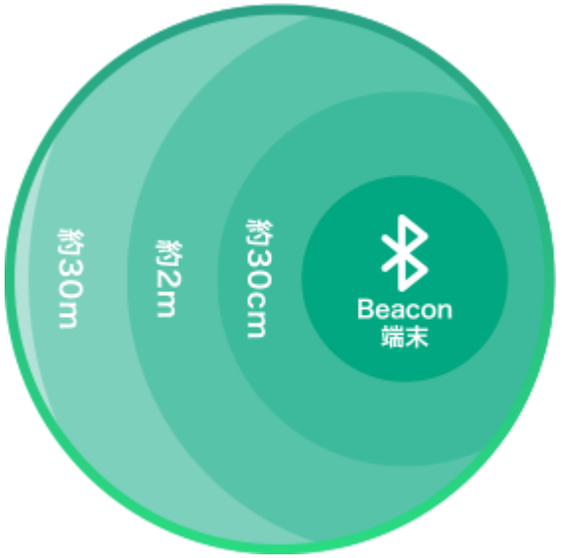
- スピード導入・開発コストの抑制を実現
- リアルタイムな情報更新（アプリアップデート不要）
- マルチ OS・マルチビーコン対応

法人向け実績多数

国内最大級の法人向けアプリ開発実績(100社、600アプリ以上) 、特殊な法人ユースケースに最適な企画開発、運用をお手伝いします。

- 管理機能を搭載（ログ収集やビーコン・イベント管理など）
- ワンストップ（ビーコン端末の現地設置～ログ解析）
- 開発から運用まで柔軟なサポート体制

beaconとは



BLE を採用
iOS7.1以上、Android4.3以上の
対象機種で利用可能

ビーコン(Beacon)の距離
30cm/2m/30m の3段階の
距離を検知

ビーコン端末の種類
1台あたり1,500円～8,000円

de:code

Demo

Native Binding

Native Binding

- iOS
  - CocoaPods との相性はいまいちかも
  - ヘッダファイルだけで Objective Sharpie を実行するのがオススメ
  - 自動生成する関数名が同じになることもあるので、調整が必要
  - LinkerFlags を指定する
    - コンパイルオプションに[-ObjC]を付けないとダメな箇所も
- Android
  - 直接呼び出すクラスのjarのみembeddedjarとし、ライブラリ内部で使用するjarはembeddedreferencejar
  - C#の予約語とライブラリのメソッド名がバッティングした場合、Metadata.xml きメソッド名変換規則を記載して対応

まとめ

セッションゴール

- Xamarin.Forms でアプリを作るのは
  そんなに難しくない
- Xamarin.Forms で 開発する時の方法/テクニック
- Xamarin.Forms の向き不向きを感じてもらう
- Xamarin.Forms で開発したくなる

プログラミングXamarin 上 Xamarin.FormsとC#によるクロスプラットフォームモバイルアプリ開発

2017/6/5 発売

696 ページ

de:code 会場にて先行販売！

¥ 6,480 ⇒ ¥ 5,800！

de:code

ユーザーグループへ参加しよう！

JAPAN
XAMARIN
USER
GROUP

#JXUG で Twitter で
つぶやくと誰かしら
反応してくれます！

http://jxug.org/

【 Session ID 】

MW08

Microsoft

アンケートにご協力ください。

■アプリのメニューから

アンケート ⇒ 「de:code イベントアプリの作り方」を選択して、回答・送信をお願いします。

de:code

Microsoft

ROOM D

Ask the Speaker のご案内

本セッションの詳細は、コミュニケーション
ルーム『Ask the Speaker』コーナー Room D
カウンタにてご説明させていただきます。
是非、お立ち寄りください。

de:code