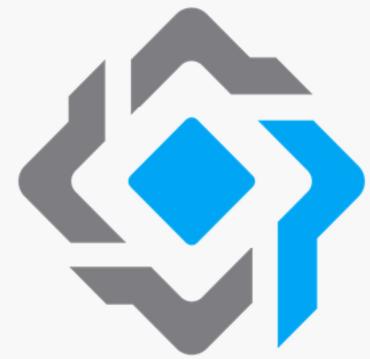


# Why Prism for Xamarin.Forms



JAPAN  
XAMARIN  
USER  
GROUP

2016.10.28 JXUGC #18

© 2016 Japan Xamarin User Group

本ハンズオンは2016.10.28に実施された「JXUGC #18 Xamarin.Forms & Prism & Azure Mobile Apps を使いこなそう」にて発表した資料と、ライブコーディングが元の資料となっています。  
まずはこの資料にざっと目を通したうえで、ハンズオンを進めてください。

# Today's Goal

次のふたつを理解していただくこと

- なぜ Prism を使うべきか？
- だれが Prism を使うべきか？



© 2016 Japan Xamarin User Group

2

さて、本資料でお伝えしたい内容は次の2点です。

- ・なぜ Prism を使うべきか？
- ・そして誰が Prism を使うべきか？

# Agenda

- Introduction
- What is Prism? & What do you get?
- Why Prism for Xamarin.Forms?
- Hands-On

© 2016 Japan Xamarin User Group

3

アジェンダはこちらの通りです。

# Introduction

- Xamarin.Formsをつかう  
→ MVVMパターンにしよう  
→ 素のままだとつらい！

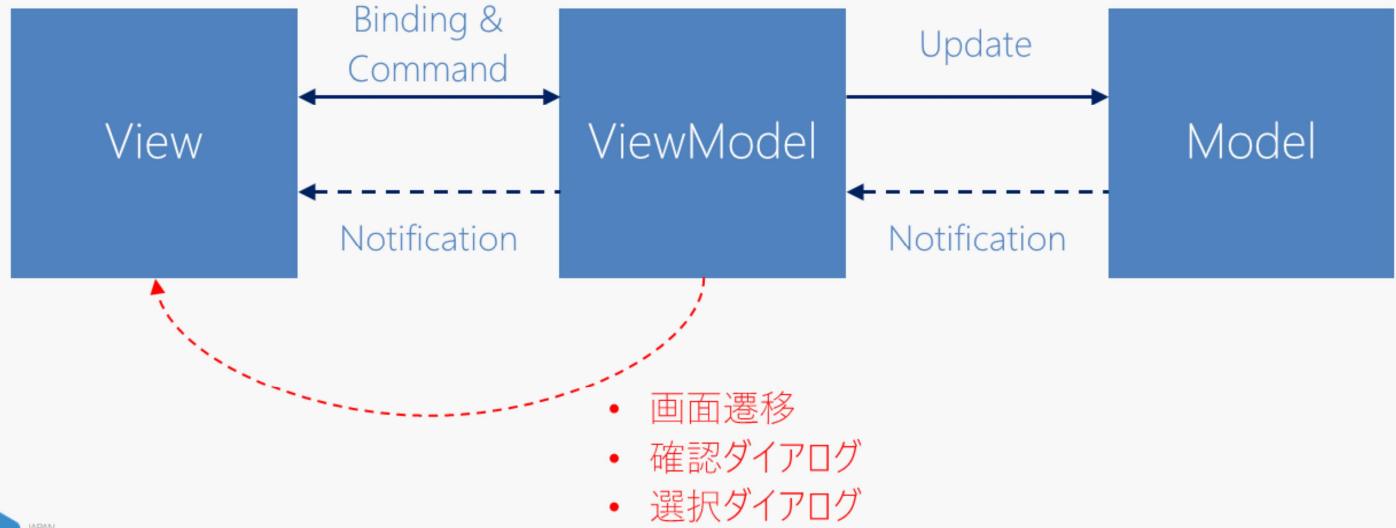


© 2016 Japan Xamarin User Group

さて Xamarin.Forms で開発しようとした場合、特別な理由がない限りは、MVVM パターンを利用することになると思います。

MVVM の利用系を、Xamarin.Forms 単独で実現するのはなかなか大変な話でもあります。  
これは Xamarin.Forms 機能不足であるということではなく、担うレイヤーが異なることが要因です。

# MVVM Patternで、辛くなりやすいところ



© 2016 Japan Xamarin User Group

5

特につらいのは、VMからViewを操作するケースです。  
画面遷移や、確認ダイアログ・選択ダイアログなどです。

# 何つかおう？

- Xamarin.Formsをつかう  
→ MVVMパターンにしよう  
→ 素のままだとつらい！  
→ MVVM支援ライブラリつかいたい！

## Prism or MVVM Light Toolkit ?



© 2016 Japan Xamarin User Group

このため通常は何かライブラリ使おうという話になります。

Xamarinの場合は代表的な選択肢として

- Prismか
  - MVVM Light
- の何れかが多いと思います。  
ではどちらを利用すべきでしょうか？

# Prism or MVVM Light Toolkit

MVVM Light Toolkit  
シンプルで分かりやすく  
取り組みやすそう

Prism  
凄いんだろうけどむず  
かしそう



© 2016 Japan Xamarin User Group

7

PrismとMVVM Lightを比較した場合、多くの方はこう思っているのではないでしょうか？

- Prismは凄いんだけど難しそうだな
- MVVM Lightの方が簡単そうだ、ライトっていうし

実は私も、昔全く同じ勘違いをしていました。

# Agenda

- Introduction
- What is Prism? & What do you get?
- Why Prism for Xamarin.Forms?
- Hands-On

© 2016 Japan Xamarin User Group

8

では何が勘違いなのか？

そこ理解していただくためにまずは、Prismとは何か、簡単にお話ししたいと思います。

# What is Prism?

- XAML Application Framework
- Guidance
- Patterns & Practices
- Testable & Maintainable
- Open Source
- .NET Foundation



© 2016 Japan Xamarin User Group

9

製作者であるBrian Lagunas氏は次のようにおっしゃっています。

プリズムはXAMLアプリケーションフレームワークです。

そして、プリズムはガイダンスであり、パターンやプラクティスの集合でもあります。

プリズムを使うと、アプリケーションは自然とテストしやすく、変更も容易になります。

# What do you get?

- MVVM Support
- Commanding
- Messaging
- Navigation
- Page Dialog Service
- Dependency Injection
- Logging



© 2016 Japan Xamarin User Group

10

プリズムは利用者に  
MVVMのSupportを提供します。

- Commanding
  - Messaging
  - Navigation
  - Page Dialog Service
  - Dependency Injection
- そしてLoggingです

# Agenda

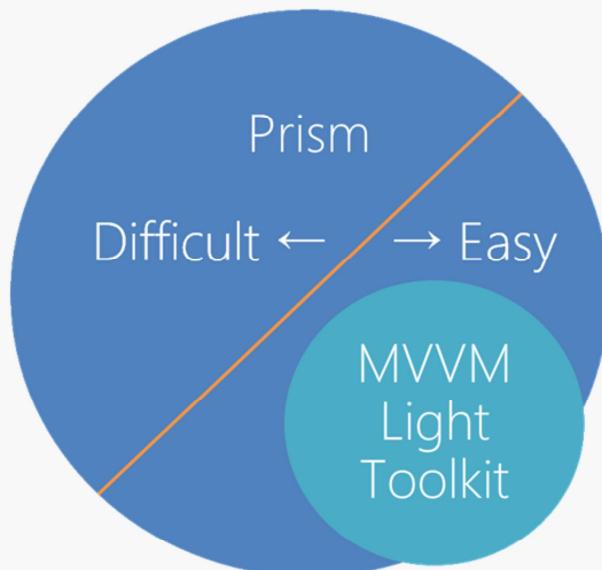
- Introduction
- What is Prism? & What do you get?
- Why Prism for Xamarin.Forms?
- Hands-On

© 2016 Japan Xamarin User Group

11

さて、実のところPrismが難しそうというのは  
半分正しく、半分誤りです

# Prism & MVVM Light Toolkit



© 2016 Japan Xamarin User Group

12

PrismはMVVM Lightと比較すると、非常に広い範囲をサポートしています。

しかも一部は確かに使いこなすのが難しいです。

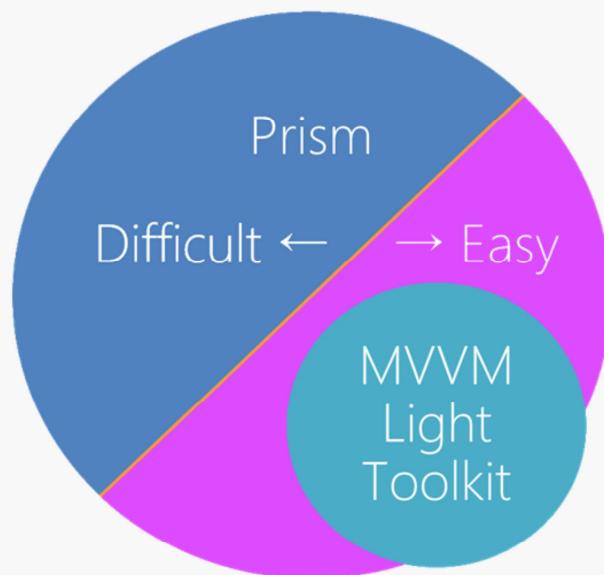
アーリーアダプターの方は大変苦労されたと思います。

しかし現在は、状況が異なります。

現在は日本語の情報もだいぶ揃ってきています。

そして実は多くの機能は簡単に利用することができます。

# Prism & MVVM Light Toolkit



© 2016 Japan Xamarin User Group

13

そして大切なのは、MVVM Lightには含まれていなくて、実は使うと幸せになれるという機能が特にXamarin向けのPrism for Xamarin.Formsに、特に多くあるということです。

# What do you get?

- MVVM Support
- Commanding
- Messaging
- Navigation
- Page Dialog Service
- Dependency Injection
- Logging



© 2016 Japan Xamarin User Group

14

具体的には

- Navigation
  - Page Dialog Service
  - Dependency Injection
- この辺りが非常に強力です。

ただ私は、こういった個別の機能よりも大切なものがPrismには含まれていると考えています。

# What is Prism?

- XAML Application Framework
- Guidance
- Patterns & Practices
- Testable & Maintainable
- Open Source
- .NET Foundation

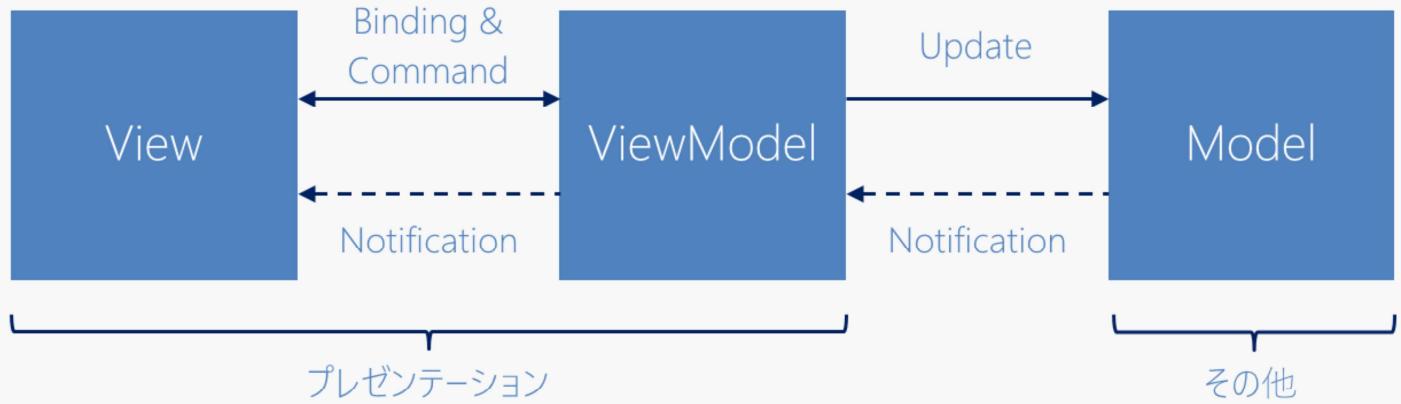


© 2016 Japan Xamarin User Group

15

それはPrismが  
・ガイダンスであり  
・パターンやプラクティスの集合  
であるということです。  
そしてそれらが、みなさんのアプリにテスタビリティやメンテナンスビリティをあたえてくれる  
これこそがPrismの真価だと思っています。

# よく見かけるMVVMの図

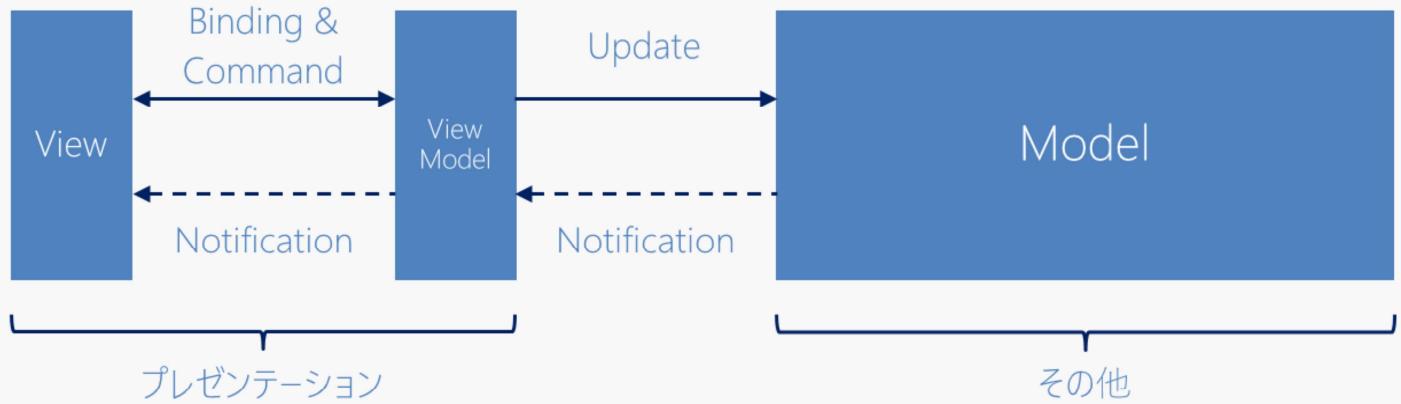


© 2016 Japan Xamarin User Group

16

アプリのデザインというと  
MVVMばかりがフォーカスされがちです。  
これには理由があって、プレゼンテーション層は  
・専門性が高かったり  
・テストが困難だったりすることが  
多いからです。  
そして、一般論として議論しやすいからでもあります。

# 実際の割合



© 2016 Japan Xamarin User Group

17

ところが、実際のアプリケーションの比率をみると  
ちゃんとMVVMしていればVやVMより  
Modelの方が圧倒的に分厚くなります。

もちろん、例外として、ヘビーなViewというのも存在します。

しかし基本はViewやViewModelというのは可能な限り薄くしてModelを厚く設計するべきです。  
そもそも、それがMVVMも設計原則でもあります。

# モバイル & クロスプラットフォーム開発

Modelにも

- 専門性の高い領域
- テストが難しい領域

が多数存在します



© 2016 Japan Xamarin User Group

さて、特にXamarinのようなモバイルでかつクロスプラットフォームの場合、実はモデルの中にも

- ・専門性の高い領域や
  - ・テストが難しい領域
- が多数存在します。

# モバイルクロスプラットフォームは課題の山

- プラットフォーム依存領域
- 時間
- 非同期処理
- プッシュ通知
- センサー類（位置情報、加速度、カメラ）



© 2016 Japan Xamarin User Group

例えば

プラットフォーム依存領域や、時間、非同期処理、プッシュ通知  
位置情報や加速度、カメラといったデバイスやセンサー類です。

そしてこれらの課題はMVVMでは解決できません

# MVVM is 何？



© 2016 Japan Xamarin User Group

20

そもそもMVVMはXAMLアプリケーションを構築する  
神器みたいに言われることが多いですが一体何なのでしょうか？

# MVVM is PDS

Presentation Domain Separation : PDS

MVVM

MVPVM

MVC



© 2016 Japan Xamarin User Group

21

MVVMというのは、Presentation Domain Separation(PDS)と呼ばれる  
「プレゼンテーション層」と「それ以外」を分離するための仕組みの実現手段の一つです。  
PDSにはMVVM以外にも、MVPVMやMVCなど多数の「手段」が存在します。

# PDS is SoC

Separation of Concerns : SoC

Presentation Domain Separation : PDS

MVVM

MVPVM

MVC

© 2016 Japan Xamarin User Group

22

そして実のところ、PDS自体もそれを包括するSeparation of Concerns(SoC)の一つの手段でしかありません。

アプリケーションは一つの大きな塊で構築するより、小さく分けて開発して、それらを統合した方が良い。

この考え方の方と共有できるのではないでしょか？

そして大雑把にいうと、SoCとは、その考え方の事です。

つまり

- ・アプリケーションを分割して構築しようというSoCの考え方の内
- ・プレゼンテーション層のみに焦点を当たPDSという考え方の
- ・一つの手段がMVVMである

という事です。

こういうと、MVVMというのはアプリやシステム構築のほんの一部分を解決する手段でしかないことが理解いただけるかと思います。

# SoC Overview

## Separation of Concerns : SoC

### Inversion of Control : IoC

Dependency Injection  
: DI

Service Locator  
(DependencyService)

### Presentation Domain Separation : PDS

MVVM

MVPVM

MVC



© 2016 Japan Xamarin User Group

23

ちなみにSoCを実現する手段として有名な概念に

・Inversion of Control (IoC)

というものが存在し、そのIoCの実現手段のとして特に有名なのが

・Dependency Injection (DI)

・Service Locator

の2種類があります。

そして、PrismではDIを最大限に有効活用することで、MVVMにとどまらない、アプリケーション全体の課題解決に対するガイダンスを示してくれています。

# Why Prism for Xamarin.Forms

- PrismはMVVMの課題だけでなく、これらの課題に対しても、ガイダンスを提供します
- Prismはアプリケーションを開発する上での、パターンとプラクティスの集合です
- そしてこれらは、テスト容易性と保守容易性を提供します



© 2016 Japan Xamarin User Group

つまりPrismは単純にMVVMをサポートするだけでなく、こういった課題を解決するパターンやプラクティスが含まれているわけです。

# だれがPrismを使うべきか？

「MVVM初めてやるけど、Prismって難しそう」という人ほど、使うべきです。



© 2016 Japan Xamarin User Group

だからこそ、私はMVVM初心者の方にこそPrismを使って欲しいと考えています。

MVVM Lightは確かにライトで取り組みやすいです。

しかし、その分、自分で決定し解決しないといけない問題も多数あります。

まずXamarin.Formsを触ってみて、感触を理解できたら次のステップとして、私はPrismをお勧めします。

# #Hands-On

© 2016 Japan Xamarin User Group

26

それではハンズオンに進みましょう

# ハンズオン概要

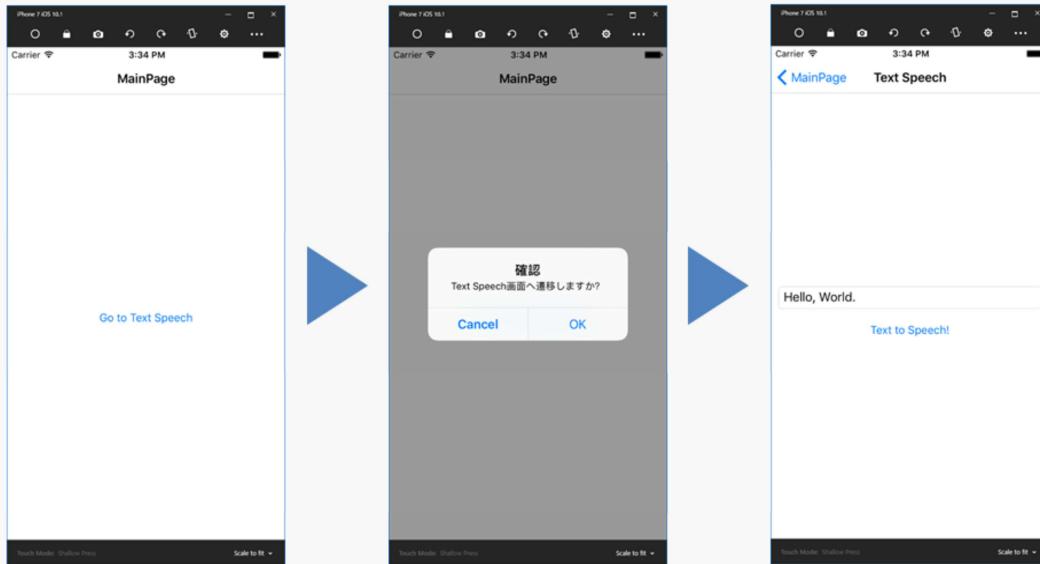
- 素のXamarin.Formsのみで作ったアプリを Prismを適用してリファクタリングします
- TDD (Test First) でいきます
- TestではMoqを利用します
- ReSharper先生最高！



© 2016 Japan Xamarin User Group

ハンズオンですが、素のXamarin.Formsのみで作ったアプリをPrismを適用してリファクタリングします。  
その際、Test Firstでいきます。  
またTestではMoqというライブラリを利用します  
これが非常に良くできたライブラリなのでぜひ皆さんに紹介したいと思います。

# Hands-Onアプリケーション概要



© 2016 Japan Xamarin User Group

28

さて、アプリケーション構成としては単純です。

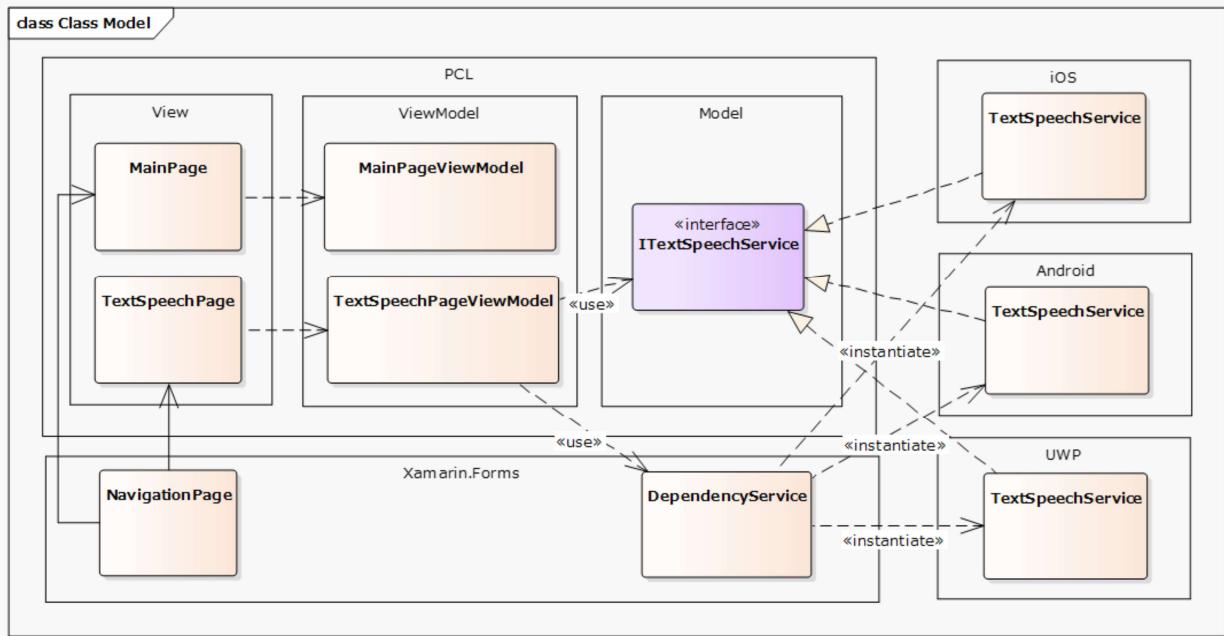
アプリケーションを起動するとメニュー画面が表示されます。

メニューでボタンを押下すると、確認ダイアログが表示され、OKを押すと次画面へ遷移します。

次画面ではボタンを押下すると、テキストボックスに入力された文字列が音声として読み上げられるというアプリケーションです。

画面遷移には`NavigationPage`を利用しているため、実装上Viewは3つ存在します。

# ハンズオンアプリクラス構成



29

まずアプリケーションには3つのViewがあります。

画面遷移にNavigationPageを利用するため、Xamarin.Formsの提供するNavigationPageが一つ。

さらにメニューに該当する MainPageと、TextSpeechPageです。

そしてそれぞれに該当するViewModelが存在し、TextSpeechPageからはITextSpeechServiceを通して文字列から音声に変換します。

ITextSpeechServiceの実態は、個々のプラットフォーム別に実装されており、ViewModelからDependencyServiceを通して利用しています。

一見、問題ないように見えますが、二つの設計上の課題があります。

それでは実際いよいよハンズオンに移動しましょう。

<https://github.com/jxug/PrismAndMoqHansOn/blob/master/docs/01.HandsOn-Overview.md>

# #Hands-On

© 2016 Japan Xamarin User Group

30

というわけで、デモに行きます

# まとめ

いい入門サイトをたまたま知っています！

【Xamarin】Prism.Forms入門

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



Xamarin関わらず多数の日本語のPrism情報があります

Prism自習用リポジトリ

<https://github.com/runceel/PrismEdu>



© 2016 Japan Xamarin User Group

31

# #Xamarinはいいぞ

© 2016 Japan Xamarin User Group

32

# #Prismもいいぞ"

© 2016 Japan Xamarin User Group

33