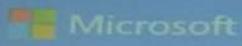


まとめ

- ・ WPF on .NET Core 3.0 で WPF が今後もひとまず安心
- ・ アクティブに開発するなら是非 .NET Core を検討してみよう
- ・ Windows 10 API も呼び出し可能なものが多いため
- ・ 最新 Windows 10 UI コントロールも
Windows UI Library 3.0 以降で使える

A young girl with dark hair tied back, wearing a bright pink short-sleeved top and a pink ruffled skirt, is smiling broadly and waving her right hand towards the camera. She is standing in a grassy field with a blurred building in the background.

楽しいデスクトップアプリ開発を



.NET Core 3.0 + Windows 10 で WPF 開発

日本マイクロソフト
Windows AppConsult
田 一希



自己紹介

日本マイクロソフト Windows AppConsult

大田 一希

- ・ SNS, Blog
Twitter: @okazuki
Blog: かずきのBlog@hatena
<https://blog.okazuki.jp>
- ・ 好き
WPF, UWP, Xamarin, Azure Functions, Vue.js
MHWI (MHW + MHWI = 1400h)、スマブラSP、
ウイッチャーライブ買いました
- ・ 苦手
型のない言語、パクチー、シューティングゲーム、音ゲー



自己紹介

日本マイクロソフト Windows AppConsult

大田 一希

- SNS, Blog

Twitter: @okazuki

Blog: かずきのBlog@hatena

<https://blog.okazuki.jp>

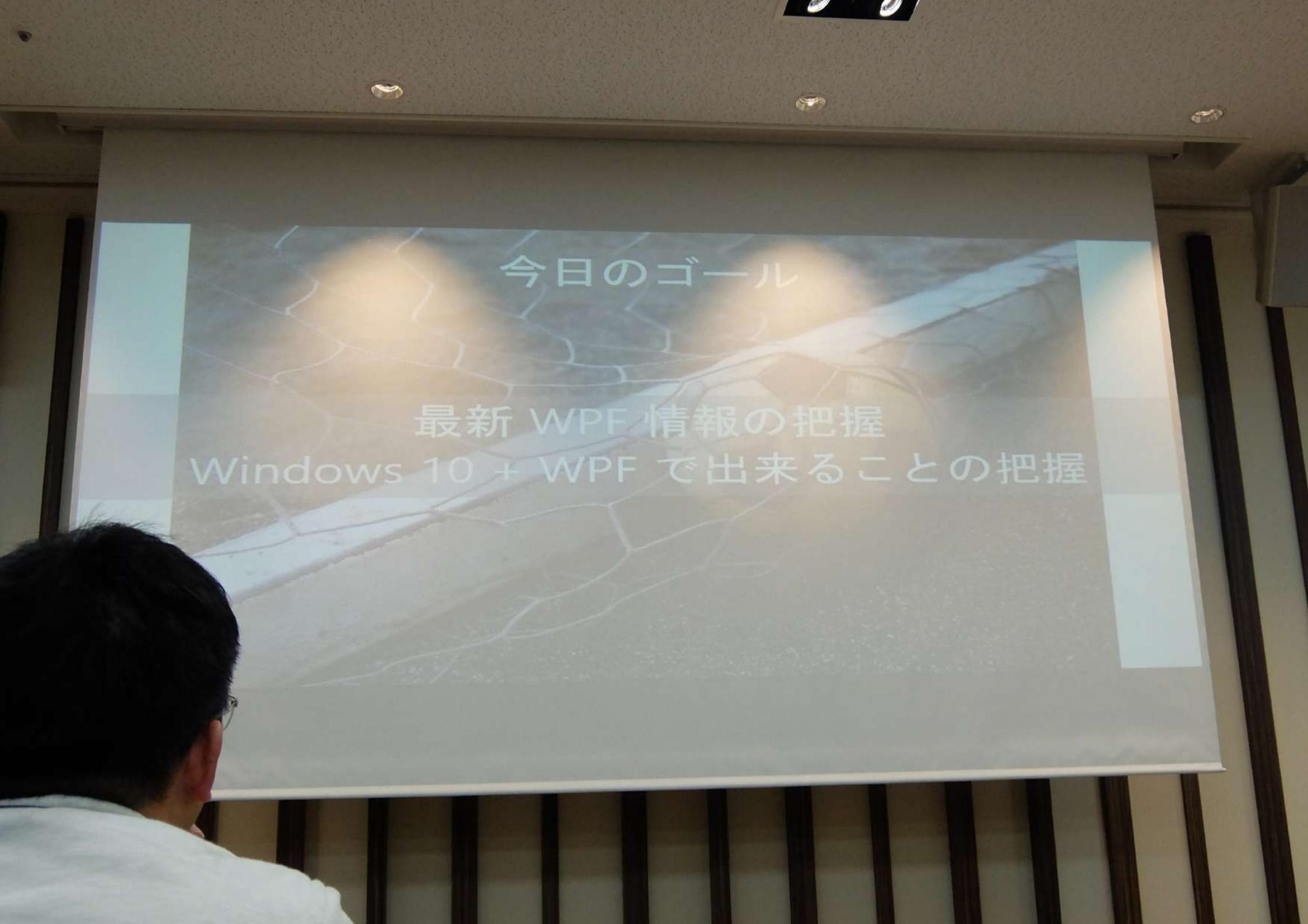
- 好き

WPF, UWP, Xamarin, Azure Functions, Vue.js
MHWI (MHW + MHWI = 1400h)、スマブラSP、
ウイッチャー3買いました

苦手

型のない言語、パクチー、シューティングゲーム、音ゲー





今日のゴール

最新 WPF 情報の把握

Windows 10 + WPF で出来ることの把握

WPF の歴史と今



WPF は...

- DirectX で画面を自分で描画してするフレームワーク
 - WPF 以前は Button や TextBox なども 1 つの Window
 - WPF は基本的に Window だけが Window
 - ComboBox のドロップダウンや右クリックメニューとかが例外
 - WPF 側で描画してるので好きに要素を合成できる

WinForms Button Image
WPF Image + Layout panel

新しいプロジェクトを構成します

Windows フォーム アプリケーション (.NET Framework)

Windows

デスクトップ

プロジェクト名(N):

WindowsFormsApp3

場所(L):

C:\Users\ki_ota\source\repos

ソリューション名(M):

WindowsFormsApp3

ソリューションとプロジェクトを同じディレクトリに配置する(O)

フレームワーク(F):

.NET Framework 4.7.2

次へ(N)

戻る(B)

メモ帳 表示設定 回観 常在



5.6... fx デザイン XAML

```
StackPanel
7     mc:Ignorable="d"
8     Title="MainWindow" Height="450" Width="800">
9     <Grid>
10    <Button>
11        <StackPanel>
12            <TextBlock Text="こんにちは" />
13            <Image />
14
15        </StackPanel>
16    </Button>
```

任务栏图标：VS Code, YouTube, Microsoft Word, Facebook, 其他未知图标。

```
5... fx デザイン XAML Text
6... 7 mc:Ignorable="d"
7... 8 Title="MainWindow" Height="450" Width="800">
8... 9 <Grid>
9... 10 <Button>
10... 11 <StackPanel>
11... 12 <TextBlock Text="こんにちは" />
12... 13 <Image />
<TextBlock Text="|~" />
</StackPanel>
</Button>
```



```
7     mc:Ignorable="d"
8     Title="MainWindow" Height="450" Width="800">
9     <Grid>
10    <Button>
11      <StackPanel>
12        <TextBlock Text="こんにちは" />
13        <Image />
14        <TextBlock Text=".NET Conf" />
15      </StackPanel>
16    </Button>
```



WPF 的には

今後も最新のプラットフォームで動作可能

- ・.NET Framework は 4.8 が最後のバージョンアップ
- ・今後の機能追加は .NET Core (.NET 5, 6, 7...) 系へ

WPF 的には

今後も最新のプラットフォームで動作可能

- ・.NET Framework は 4.8 が最後のバージョンアップ
- ・今後の機能追加は .NET Core (.NET 5, 6, 7...) 系へ

.NET Framework で今後考えられる事象

- ・対応しないライブラリの登場

WPF 的には

今後も最新のプラットフォームで動作可能

- .NET Framework は 4.8 が最後のバージョンアップ
- 今後の機能追加は .NET Core (.NET 5, 6, 7...) 系へ

.NET Framework で今後考えられる事象

- 対応しないライブラリの登場
- C# の最新言語仕様に対応しきれない

WPF 的には

今後も最新のプラットフォームで動作可能

- ・.NET Framework は 4.8 が最後のバージョンアップ
- ・今後の機能追加は .NET Core (.NET 5, 6, 7...) 系へ

.NET Framework で今後考えられる事象

- ・対応しないライブラリの登場
- ・C# の最新言語仕様に対応しきれない
- ・etc...

移行ドキュメント

- ・.NET Framework から .NET Core への移行

<https://docs.microsoft.com/ja-jp/dotnet/core/porting/>

The screenshot shows a Microsoft .NET documentation page. The top navigation bar includes links for Microsoft, .NET, About, Learn, Architecture, Docs, Downloads, Community, and Get Started. Below the navigation is a search bar and a sidebar with a table of contents for '从 .NET Framework 移植到 .NET Core'.

从 .NET Framework 移植到 .NET Core

2019/09/13 • 1 min read

.NET Framework 上で実行されるコードがある場合は、.NET Core 上でコードを実行することにも関心があるかもしれません。ここでは、移植プロセスの概要と、コードを .NET Core に移植するときに役立つツールの一覧を示します。

移植プロセスの概要

プロジェクトを .NET Core に移植する場合、次のプロセスを実行することをお勧めします。プロセスの各手順については、他の記事で詳しく説明します。

1. サードパーティの依存関係を識別し、理解します。

移行ドキュメント

- .NET Framework から .NET Core への移行

<https://docs.microsoft.com/ja-jp/dotnet/core/porting/>

The screenshot shows a web browser displaying the Microsoft .NET Core porting guide. The URL in the address bar is <https://docs.microsoft.com/ja-jp/dotnet/core/porting/>. The page title is ".NET Framework から .NET Core にコードを移植する". The main content area contains a summary of the porting process and a section titled "移植プロセスの概要" with a note about third-party dependencies. On the left, there is a sidebar with a navigation tree for the porting guide.

Microsoft | .NET About Learn Architecture Docs Downloads Community Get Started

≡ ブックマーク フィードバック ホーム 特別 テーマ 英語で読む

HOME > .NET > .NET Core のガイド

タイトルでファイルマーク (.NET Core API 参照)

· .NET Core の追加ツール

· .NET Framework からの移植

· .NET Framework からの移植

NET Core で使用できない.NET Framework テクノロジ

サードパーティの依存関係の分析

ライブラリの移植

NET Core のプロジェクトの整理

NET Core への移植で役立つツール

Windows 計算機バージョンの使用

.NET Framework から .NET Core にコードを移植する

2019/09/13 · 6 分

NET Framework 上で実行されるコードがある場合は、.NET Core 上でコードを実行することにも関心があるかもしれません。ここでは、移植プロセスの概要と、コードを .NET Core に移植するときに役立つツールの一覧を示します。

移植プロセスの概要

プロジェクトを .NET Core に移植する場合、次のプロセスを実行することをお勧めします。プロセスの各手順については、他の記事で詳しく説明します。

- サードパーティの依存関係を識別し、理解します。

移行ドキュメント

- Migrating WPF Apps to .NET Core

<https://docs.microsoft.com/ja-jp/dotnet/desktop-wpf/migration/convert-project-from-net-framework>

The screenshot shows a Microsoft Docs page titled "Migrating WPF Apps to .NET Core". The page is dated 2019/09/12 and has a reading time of 26 分. It features a sidebar with filtering options like "タイトルでフィルター" and "移行". The main content discusses the steps required to migrate a Windows Presentation Foundation (WPF) app from .NET Framework to .NET Core 3.0, using the Bean Trader sample app as an example. A note at the bottom states: "The Desktop Guide documentation is under construction."

Microsoft | .NET About Learn Architecture Docs Downloads Community Get Started 検索

Docs / .NET / デスクトップ ガイド - WPF / 移行

このコンテンツはお使いの言語では利用できません。英語版はこちらです。

タイトルでフィルター

- 概要
- 作業の開始
- 移行
 - .NET Framework との違い
 - Framework から Core への移行
 - 基礎
 - データの操作

Migrating WPF Apps to .NET Core

2019/09/12 • 累了までの所要時間: 26 分 • 3 人の編集者

In this article, we'll look at the steps necessary to migrate a Windows Presentation Foundation (WPF) app from .NET Framework to .NET Core 3.0. If you don't have a WPF app on hand to port but would like to try out the process, you can use the **Bean Trader** sample app available on [GitHub](#). The original app (targeting .NET Framework 4.7.2) is available in the `NetFx\BeanTraderClient` folder. This guide first explains the steps necessary to port apps in general and then walks through the specific changes that apply to the **Bean Trader** sample.

重要

The Desktop Guide documentation is under construction.

このページはお役に立ちましたか?

いいえ いいえ

この記事の内容

- About the sample
- Getting ready
- Migrating the project file
- Fix build issues
- Runtime testing

移行ドキュメント

- Migrating WPF Apps to .NET Core

<https://docs.microsoft.com/ja-jp/dotnet/desktop-wpf/migration/convert-project-from-net-framework>

The screenshot shows a Microsoft .NET documentation page. At the top, there's a navigation bar with links for Microsoft, .NET, About, Learn, Architecture, Docs, Downloads, Community, Get Started, and a search bar. Below the navigation bar, the URL is https://docs.microsoft.com/ja-jp/dotnet/desktop-wpf/migration/convert-project-from-net-framework. The main content area has a heading "Migrating WPF Apps to .NET Core" with a sub-headline "このコンテンツはお使いの言語ではありません。英語版はこちらです。". To the left, there's a sidebar with a filter section titled "タイトルでフィルター" and a list of categories: 概要, 作業の開始, 移行 (selected), .NET Framework との違い, Framework から Core への移行, 基礎, データの操作. The main article content discusses the steps to migrate a Windows Presentation Foundation (WPF) app from .NET Framework to .NET Core 3.0, mentioning the Bean Trader sample app available on GitHub. On the right, there's a sidebar with sections for "このページはお役に立ちましたか?", "この記事の内容", and a list of topics: About the sample, Getting ready, Migrating the project file, Fix build issues, Runtime testing.

.NET Framework からの移行時の苦しみ

- ・互換性があるのかわからない...
→ .NET Portability Analyzer あるよ！

.NET Framework からの移行時の苦しみ

- ・互換性があるのかわからない...
→ .NET Portability Analyzer あるよ！
- ・PlatformNotSupportedException が...
→ Microsoft.DotNet.Analyzers.Compatibility あるよ！

.NET Framework からの移行時の苦しみ

- ・互換性があるのかわからない...
→ .NET Portability Analyzer あるよ！
- ・PlatformNotSupportedException が...
→ Microsoft.DotNet.Analyzers.Compatibility あるよ！
- ・Blend SDK (Behavior) がディスコン
→ Microsoft.Xaml.Behaviors.Wpf を使おう
名前空間が異なるので修正は必須 (PathListBox とかは無い)

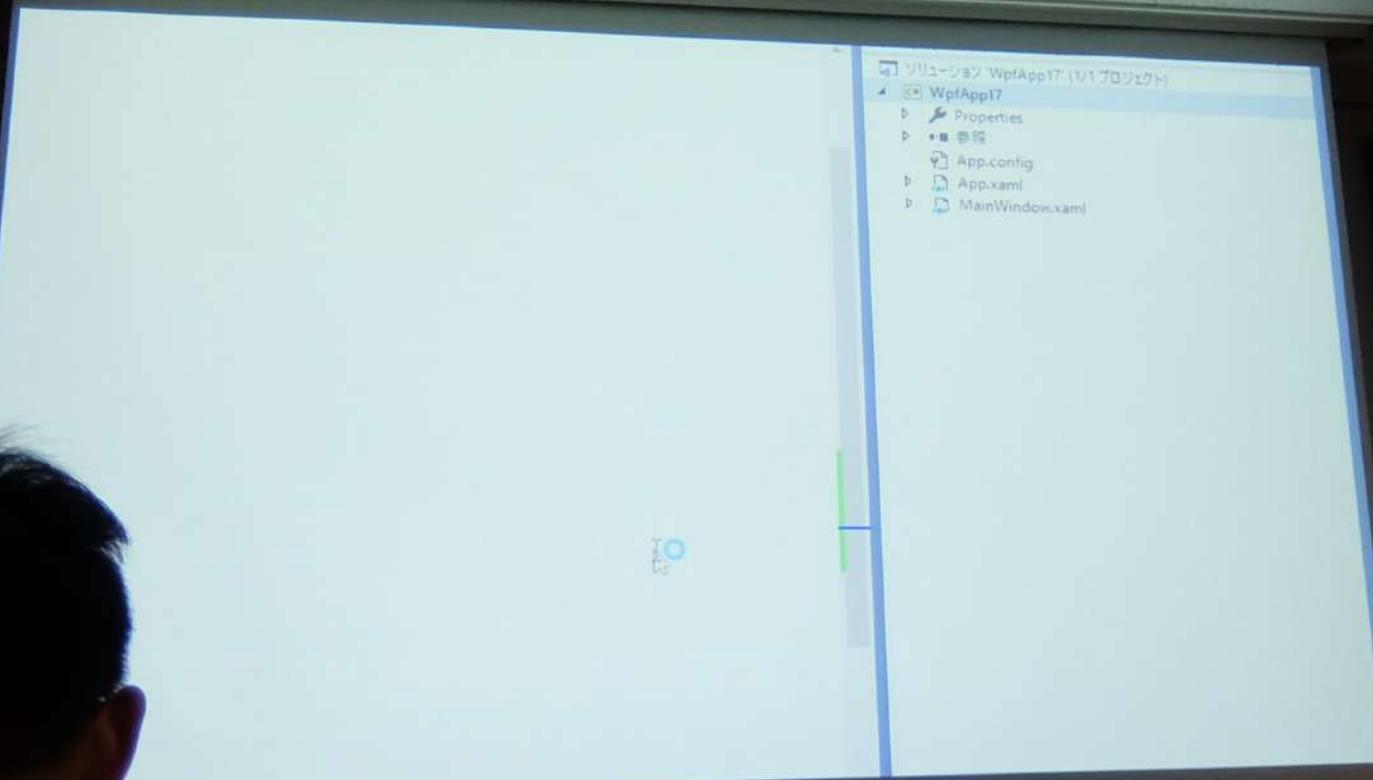
.NET Framework からの移行時の苦しみ

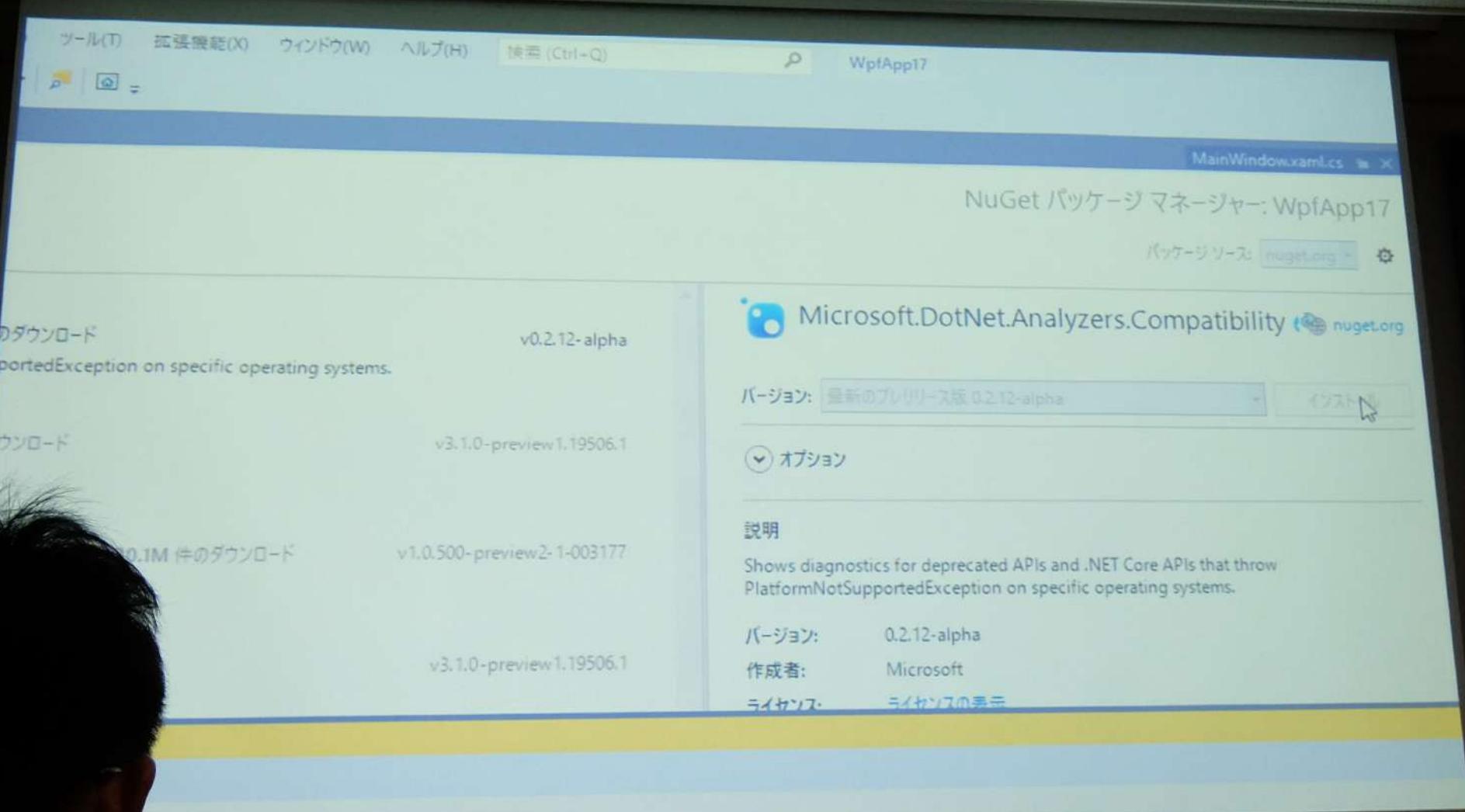
- ・互換性があるのかわからない...
→ .NET Portability Analyzer あるよ！
- ・PlatformNotSupportedException が...
→ Microsoft.DotNet.Analyzers.Compatibility あるよ！
- ・Blend SDK (Behavior) がディスコン
→ Microsoft.Xaml.Behaviors.Wpf を使おう
名前空間が異なるので修正は必須（PathListBox とかは無い）

Project ファイルの変更

- SDK Style に変換

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">
  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <UseWPF>true</UseWPF>
    <GenerateAssemblyInfo>false</GenerateAssemblyInfo>
  </PropertyGroup>
</Project>
```





Analyzers for consumers of Microsoft.CodeAnalysis NuGet package, i.e. extensions and applications built on to
package is included as a development dependency of Microsoft.CodeAnalysis NuGet package and does not ne

Microsoft.EntityFrameworkCore.Analyzers

作成者: Microsoft, 35.4M 件のダウンロード

CSharp Analyzers for Entity Framework Core.

全体

0 エラー

1 警告

7 メッセージ

ビルド + IntelliSense

コード 説明

CA001 AppDomain.CreateDomain(string) isn't supported on Linux, macOS, and Windows

System.Windows.RoutedEventHandler.#ctor(System.Object, System.IntPtr) ...

Windows.Controls.Primitives.ButtonBase.add_Click(System.Windows.RoutedEventHandler)

Windows.Window.#ctor ...

Windows.MessageBox.Show(System.String) ...

Windows.Application.set_StartupUri(System.Uri) ...

Windows.Application.LoadComponent(System.Object, System.Uri) ...

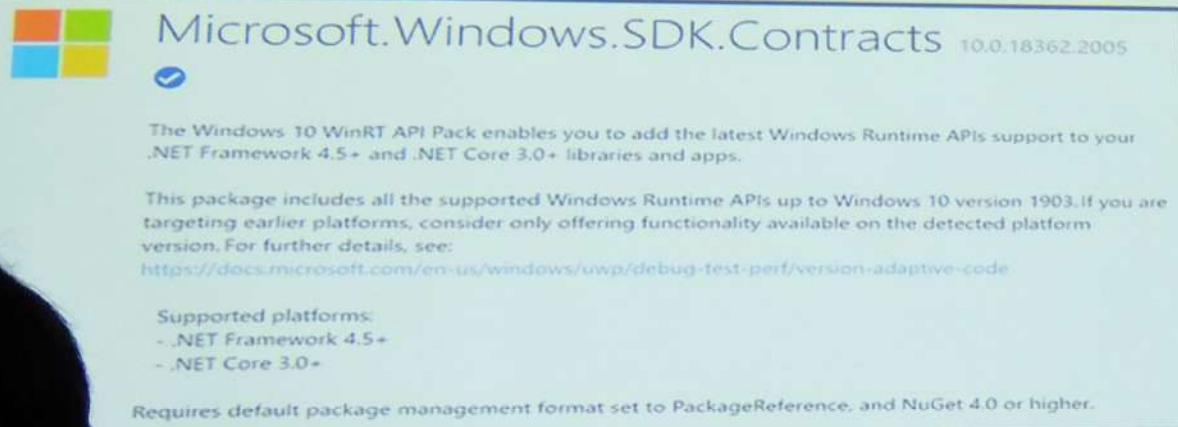
System.Configuration.SettingsBase.Synchronized(System.Configuration.SettingsBase) ...

Windows 10 の API 呼べる？

呼べます

- Microsoft.Windows.SDK.Contracts package

<https://www.nuget.org/packages/Microsoft.Windows.SDK.Contracts>



NuGet パッケージマネージャー: WpfAp

パッケージソース: nuget.org



Microsoft.Windows.SDK.Contracts



nu

バージョン: 最新の安定版 10.0.18362.2005

インスト

▼ オプション

説明

The Windows 10 WinRT API Pack enables you to add the latest Windows Runtime API support to your .NET Framework 4.5+ and .NET Core 3.0+ libraries and apps.

This package includes all the supported Windows Runtime APIs up to Windows 10 v1903. If you are targeting earlier platforms, consider only offering functionality available in Windows 8.1 and later.

パッケージ ソース: nuget.org

クリエイターズ

ソリュ



Microsoft.Windows.SDK.Contracts



nuget.org

バージョン: 最新の安定版 10.0.18362.2005

インストール

▼ オプション

説明

The Windows 10 WinRT API Pack enables you to add the latest Windows Runtime APIs support to your .NET Framework 4.5+ and .NET Core 3.0+ libraries and apps.

This package includes all the supported Windows Runtime APIs up to Windows 10 version 1903. If you are targeting earlier platforms, consider only offering functionality available in the detected platform version. For further details, see:

<https://docs.microsoft.com/en-us/windows/uwp/debug-test-perf/version-adaptive-code>

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+Shift+F)
- ソリューション 'WpfApp17' (1/1 プロジェクト)
 - WpfApp17
 - Properties
 - 依存関係
 - App.config
 - App.xaml
 - MainWindow.xaml
 - MainWindow.xaml.cs

```
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
public Mainwindow()
{
    InitializeComponent();
}

1 個の参照
private void Button_Click(object sender, RoutedEventArgs e)
{
    var g = Geolocator
}
}
```

```
24      public MainWindow()
25      {
26          InitializeComponent();
27      }
28
29      1 個の参照
30      private void Button_Click(object sender, R
31      {
32          var g = new Geolocator();
33      }
34  }
```

```
    InitializeComponent();  
}
```

1 個の参照

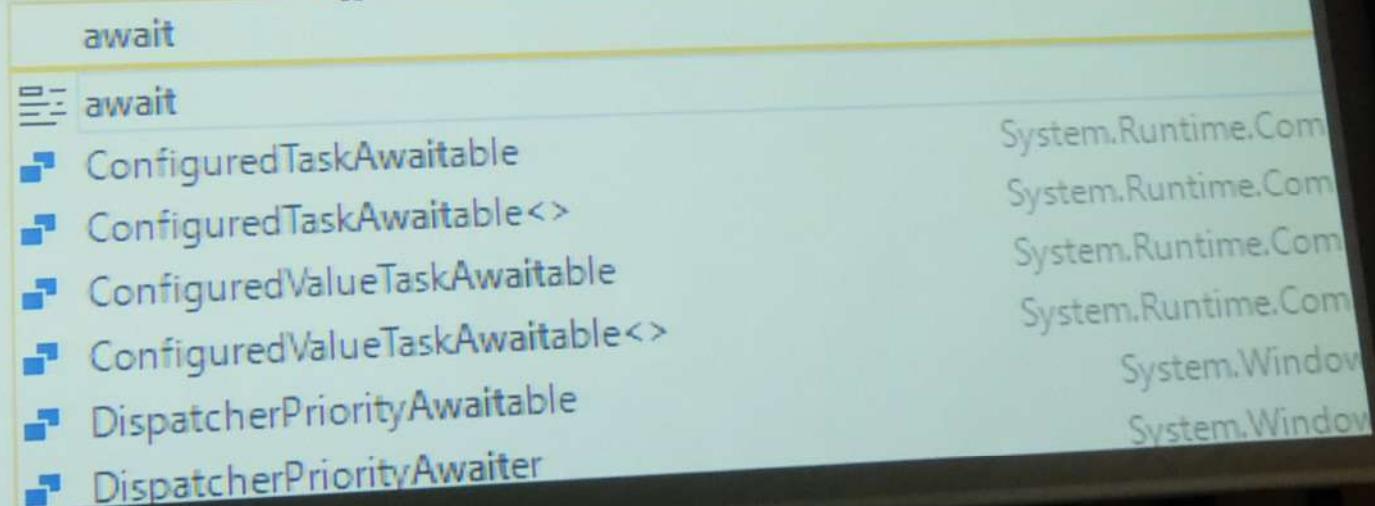
```
private async void Button_Click(object sender, RoutedEventArgs e)  
{  
    var g = new Geolocator();  
    var pos = new g.
```

I

```
    InitializeComponent();  
}
```

1 個の参照

```
private async void Button_Click(object sender, RoutedEventArgs e)  
{  
    var g = new Geolocator();  
    var pos = await g.
```



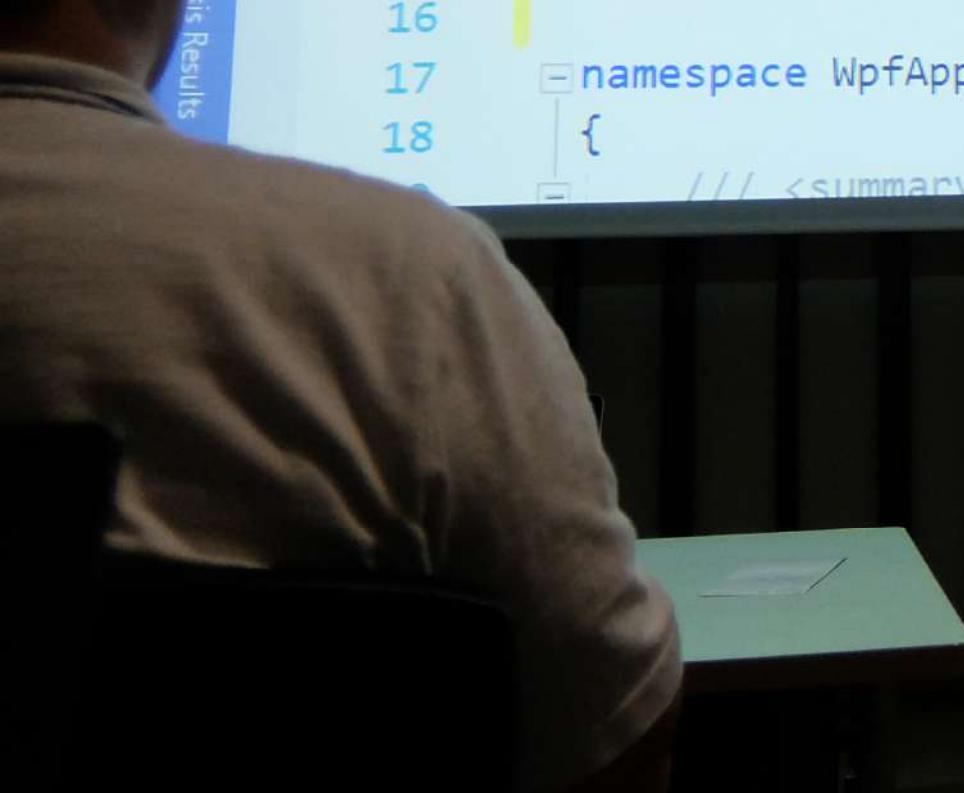
```
    InitializeComponent();  
}
```

1 個の参照

```
private async void Button_Click(object sender, RoutedEventArgs e)  
{  
    var g = new Geolocator();  
    var pos = await g.GetGeopositionAsync(); ~
```

```
c void Button_Click(object sender, RoutedEventArgs e)  
new Geolocator();  
= await g.GetGeopositionAsync();  
ox.Show($"{{pos.Coordinate.Point.Position.Latitude}}");
```

I



A screenshot of a Windows application window titled "MainWindow.xaml.cs*". The window shows C# code for a WPF application named "WpfApp17". The code includes several "using" statements and a namespace declaration. A tooltip is visible, showing the completion of the "Windows.Devices.Geolocation;" statement.

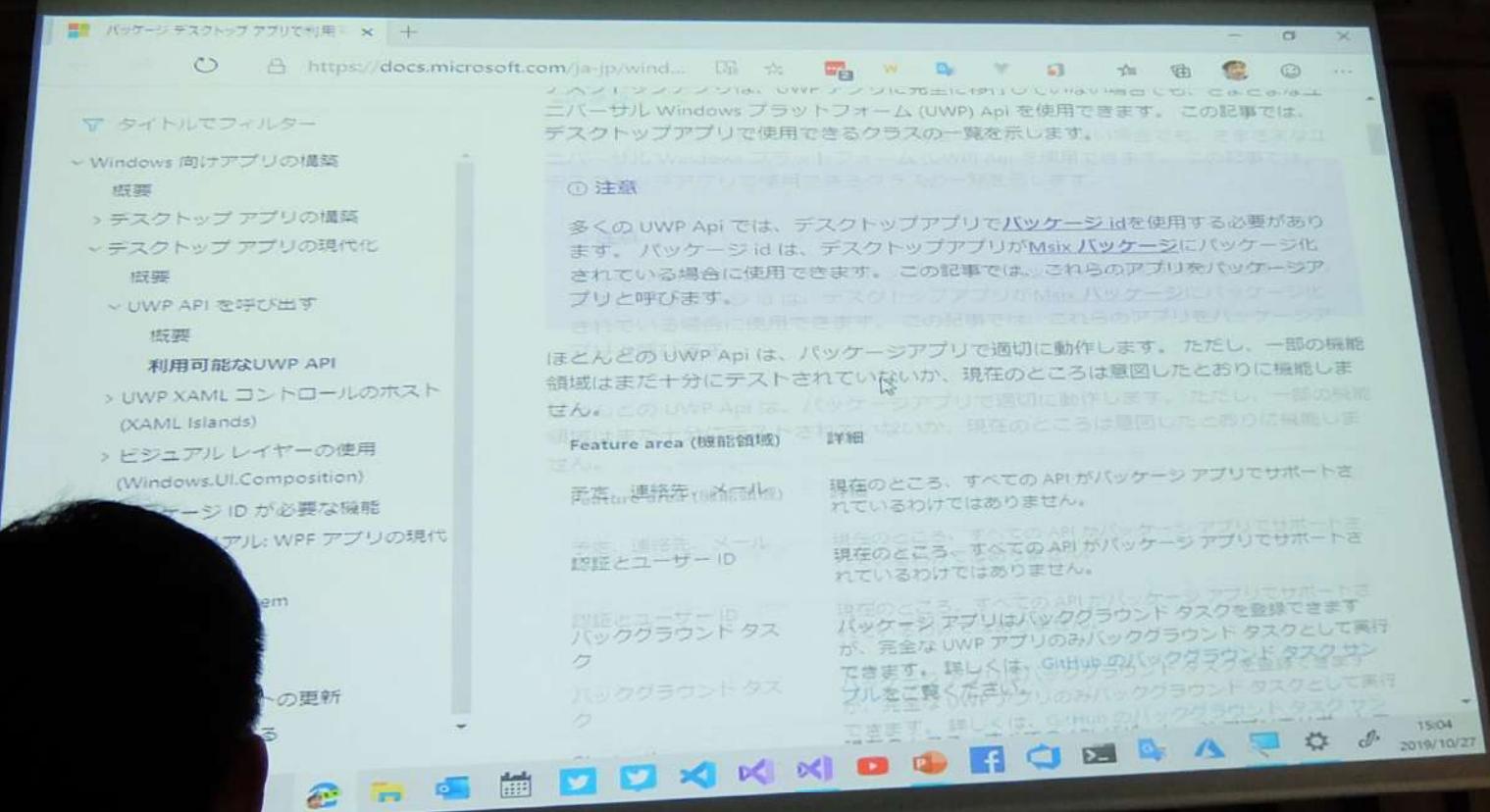
```
MainWindow.xaml.cs*
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Windows.Devices.Geolocation;
namespace WpfApp17
{
    /// <summary>
```



サポートされている Windows 10 API

- ・リストが確認できます

<https://docs.microsoft.com/ja-jp/windows/apps/desktop/modernize/desktop-to-uwp-supported-api>



サポートされている Windows 10 API

- ・リストが確認できます

<https://docs.microsoft.com/ja-jp/windows/apps/desktop/modernize/desktop-to-uwp-supported-api>

サポートされている API を簡単にまとめると…

- ・DualApiPartitionAttribute がついてる API
- ・Windows 10 の MSIX 形式に固めたアプリでサポートされる API

例えば...

- ・現在地取りたい
 - ・.NET Framework では GeoCoordinateWatcher が使えたが .NET Core にはない

MSIX の作り方

- Windows アプリケーション パッケージ プロジェクトを作ってパッケージ



Windows Application Packaging Project

A project that creates MSIX packages containing Windows applications for side-loading or distribution via the Microsoft Store

C#

Windows

UWP

Desktop

MSIX メリット

- ・メリット
 - ・管理者権限不要のインストール
 - ・レジストリーを汚さない
 - ・ストアへの提出も可能
 - ・ストア経由じゃない場合でもアプリの自動更新対応
(.NET Core は ClickOnce 非対応なので後継的な雰囲気)
 - ・呼べる Windows 10 の API が増える

MSIX デメリット

- ・デメリット
 - ・実質 Windows 10 のみ対応
 - ・サービスやカーネルドライバーのインストールには非対応
(バックグラウンドタスクなどで代用)
 - ・若干アプリの実行時の動作が変わる
 - ・AppData やレジストリーへの読み書きがリダイレクトされる
 - ・カレントディレクトリが exe のフォルダーではなくなる
 - ・etc...

MSIX デメリット

- ・デメリット
 - ・実質 Windows 10 のみ対応
 - ・サービスやカーネルドライバーのインストールには非対応
(バックグラウンドタスクなどで代用)
 - ・若干アプリの実行時の動作が変わる
 - ・AppData やレジストリーへの読み書きがリダイレクトされる
 - ・カレントディレクトリが exe のフォルダーではなくなる
 - ・etc...

最近の Windows 10 の UI コントロール事情

これまで

- ・ 最新コントロール追加したわ！※ただし UWP に限る
- ・ XAML Islands で Windows 10 May 2019 Update 以降は WPF などから UWP のコントロールが使える

最近の Windows 10 の UI コントロール事情

これまで

- ・最新コントロール追加したわ！※ただし UWP に限る
- ・XAML Islands で Windows 10 May 2019 Update 以降は WPF などから UWP のコントロールが使える

これから

UI コントロールのアップデートは OS から切り離して OSS 化

最近の Windows 10 の UI コントロール事情

これまで

- ・ 最新コントロール追加したわ！※ただし UWP に限る
- ・ XAML Islands で Windows 10 May 2019 Update 以降は WPF などから UWP のコントロールが使える

これから

UI コントロールのアップデートは OS から切り離して OSS 化

Windows UI Library 3.0 以降（2020年予定）でUWP 以外もサポート
(XAML Islands が組み込まれる)

最近の Windows 10 の UI コントロール事情

これまで

- ・ 最新コントロール追加したわ！※ただし UWP に限る
- ・ XAML Islands で Windows 10 May 2019 Update 以降は WPF などから UWP のコントロールが使える

これから

- ・ UI コントロールのアップデートは OS から切り離して OSS 化
 - ・ Windows UI Library 3.0 以降（2020年予定）でUWP 以外もサポート（XAML Islands が組み込まれる）
- Windows 10 Creators Update 以降をサポート

やり方

- UWP アプリプロジェクトを作成
 - Microsoft.Toolkit.Win32.UI.XamlApplication パッケージの追加
 - App クラスのベースクラスを XamlApplication に変更
 - コンストラクタで Initialize() を呼ぶ
- WPF プロジェクト
 - Microsoft.Toolkit.Wpf.UI.Controls パッケージの追加
 - UWP プロジェクトを参照に追加
 - InkCanvas, InkToolbar や WindowsXamlHost コントロールを使用して UWP のコントロールを表示

<https://docs.microsoft.com/ja-jp/windows/apps/desktop/modernize/host-standard-control-with-xaml-island>

アリカ

- ・ UWP アプリプロジェクトを作成
 - ・ Microsoft.Toolkit.Win32.UI.XamlApplication パッケージの追加
 - ・ App クラスのベースクラスを XamlApplication に変更
 - ・ コンストラクタで Initialize() を呼ぶ
- ・ WPF プロジェクト
 - ・ Microsoft.Toolkit.Wpf.UI.Controls パッケージの追加
 - ・ UWP プロジェクトを参照に追加
 - ・ InkCanvas, InkToolbar や WindowsXamlHost コントロールを使用して UWP のコントロールを表示

<https://docs.microsoft.com/ja-jp/windows/apps/desktop/modernize/host-standard-control-with-xaml-islands>

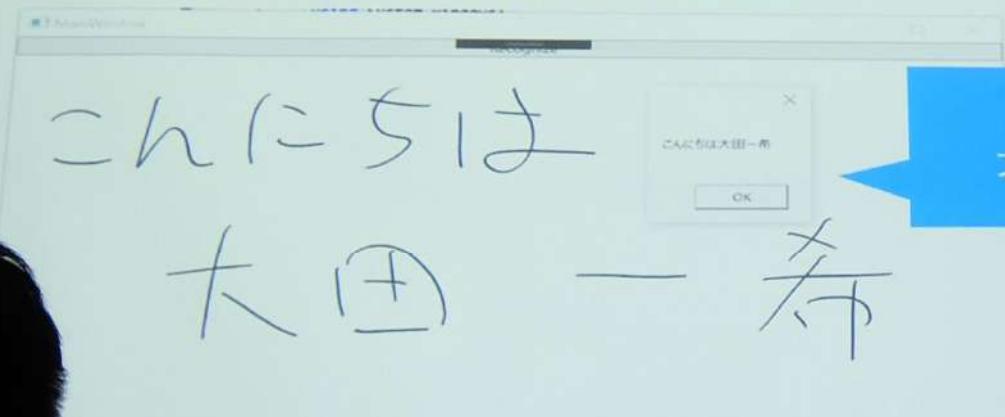
やり方

- ・ UWP アプリプロジェクトを作成
 - ・ Microsoft.Toolkit.Win32.UI.XamlApplication パッケージの追加
 - ・ App クラスのベースクラスを XamlApplication に変更
 - ・ コンストラクタで Initialize() を呼ぶ
- ・ WPF プロジェクト
 - ・ Microsoft.Toolkit.Wpf.UI.Controls パッケージの追加
 - ・ UWP プロジェクトを参照に追加
 - ・ InkCanvas, InkToolbar や WindowsXamlHost コントロールを使用して UWP のコントロールを表示

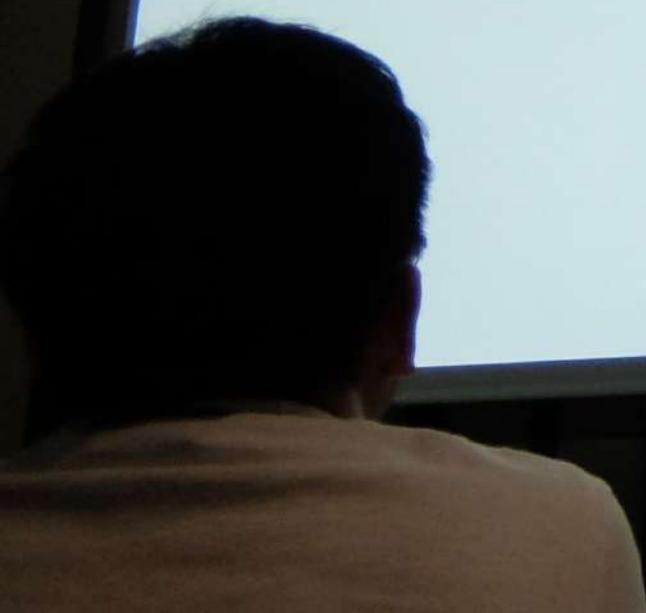
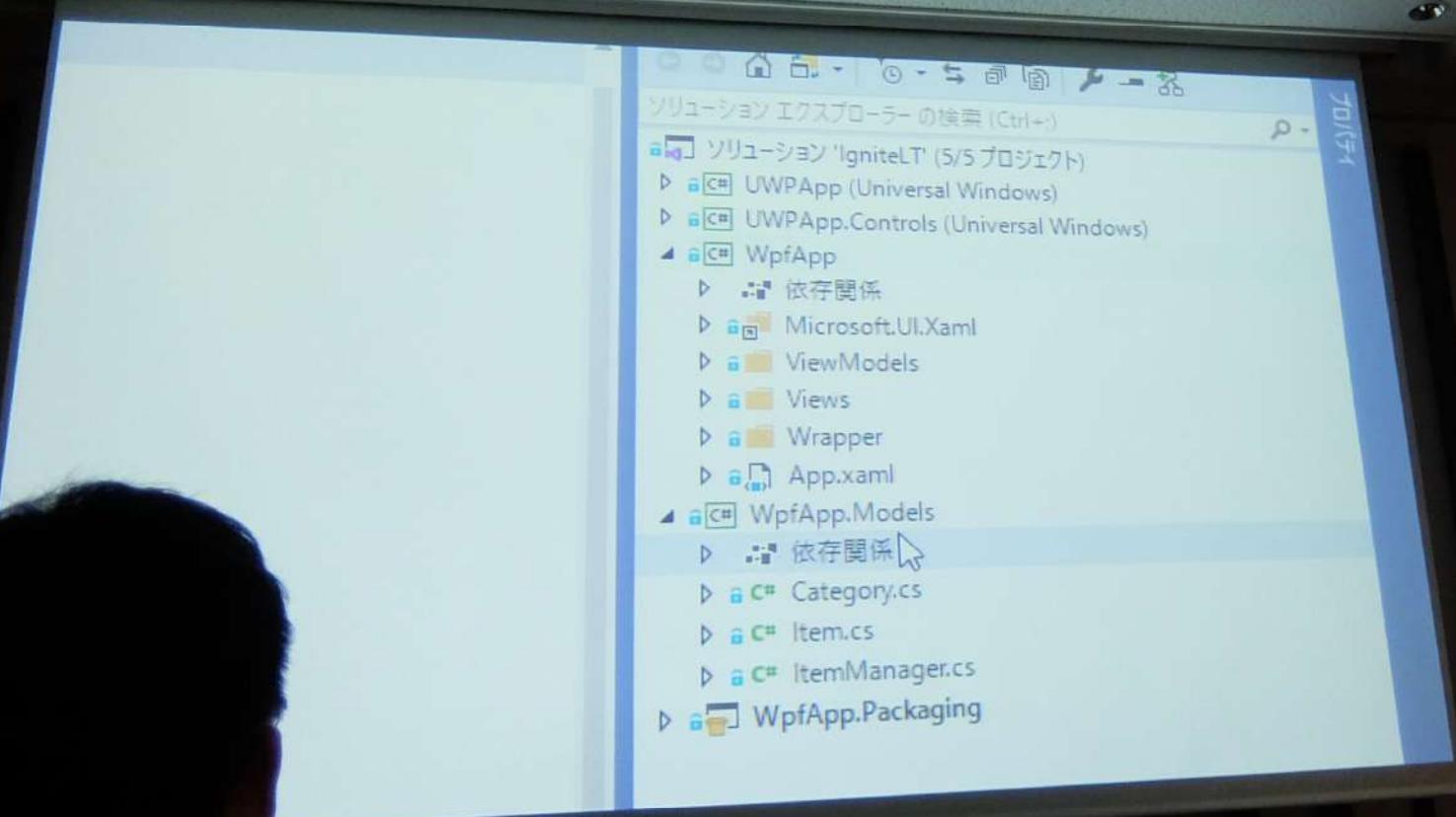
<https://docs.microsoft.com/ja-jp/windows/apps/desktop/modernize/host-standard-control-with-xaml-islands>

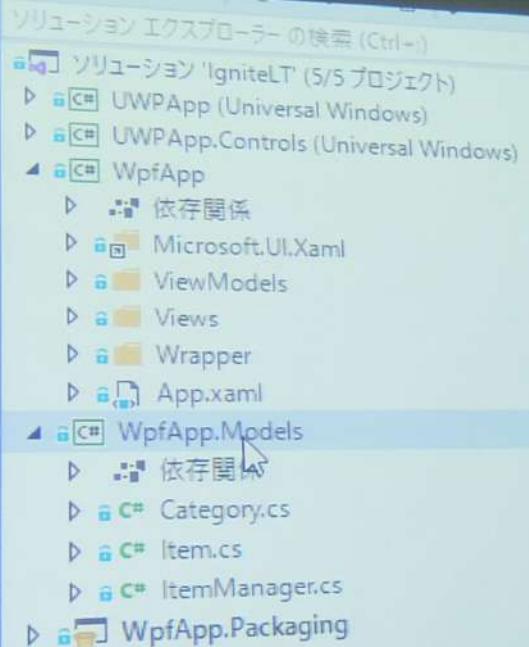
メリット

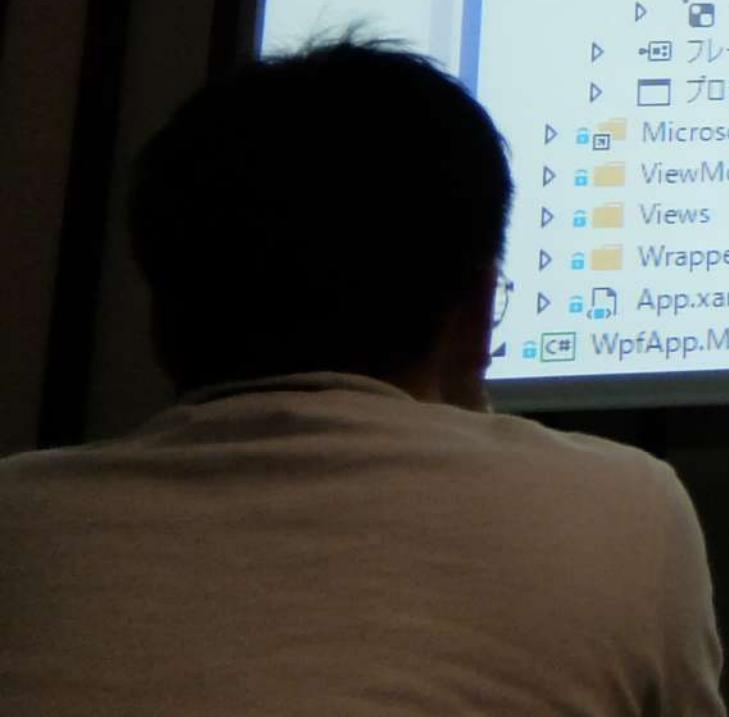
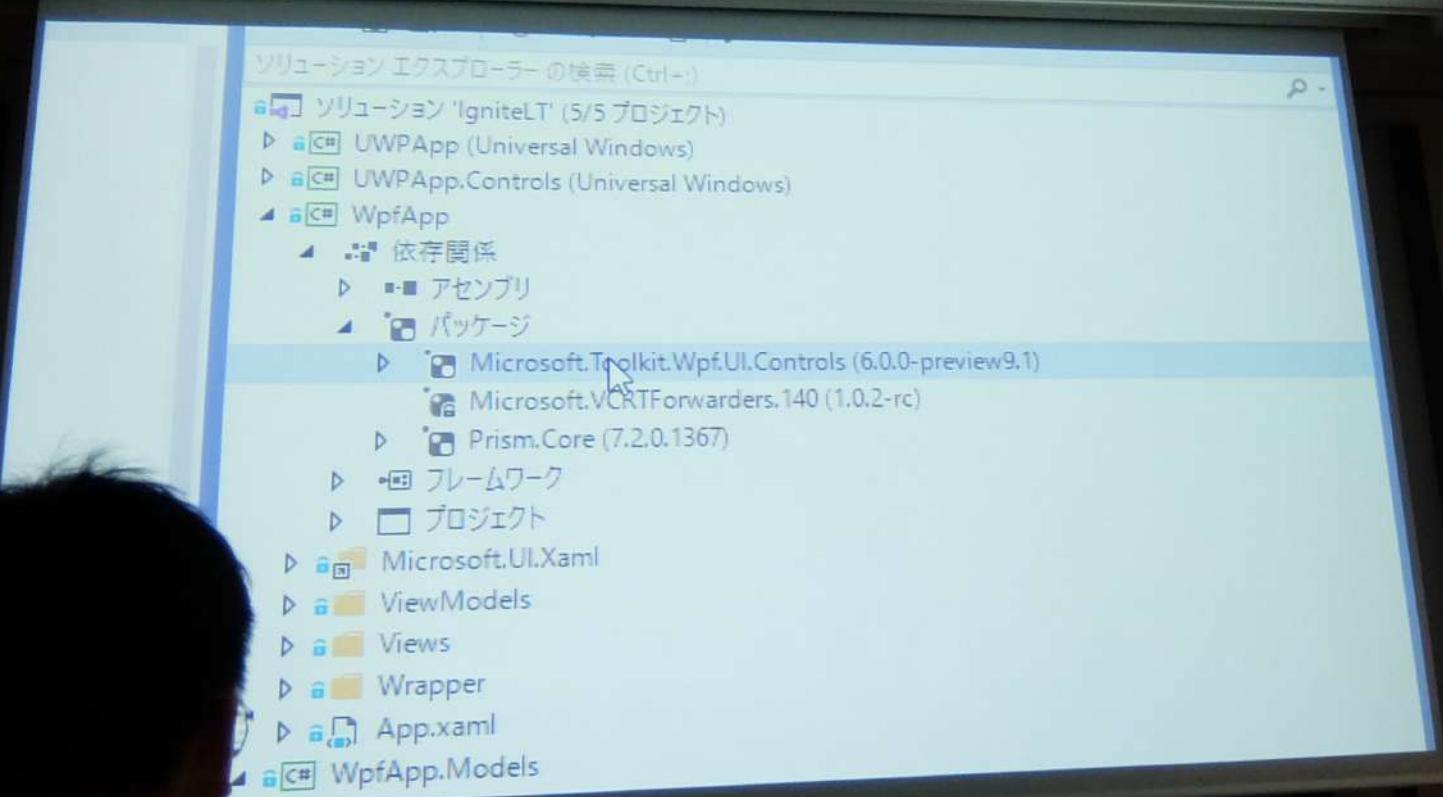
- ・最新コントロールが使える
 - ・Ink、タッチ、ペン、高 DPI 対応など
 - ・IE ベースじゃない WebView が使える



UWP の InkCanvas と
Windows 10 API の
文字認識の API を WPF から
使った例









A screenshot of a Windows desktop showing a Visual Studio IDE window. The window displays XAML code for a Universal Windows Platform (UWP) application.

The code in the main editor pane is:

```
1 <xaml:XamlApplication
2   x:Class="UwPApp.App"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:xaml="using:Microsoft.Toolkit.Win32.UI.XamlHost"
6   RequestedTheme="Dark"
7   xmlns:local="using:UwPApp">
8     <Application.Resources>
9       <XamlControlsResources xmlns="using:Microsoft.UI.Xaml.Controls" />
10    </Application.Resources>
11  </xaml:XamlApplication>
12
```

The Solution Explorer on the right shows the project structure:

- ソリューション 'igniteLT' (3/9/70/1279)
 - UwPApp (Universal Windows)
 - Properties
 - Assets
 - App.xaml
 - Package.appmanifest
 - UwPApp.Controls (Universal Windows)
 - Properties
 - TreeViewItemTemplateSelector.cs
 - UwpContentItemView.xaml
 - UwpItemsTreeView.xaml
 - WpApp
 - Assets
 - App.xaml
 - Microsoft.Toolkit.Wpf.UILevelControl (6.0.0-pre)
 - Microsoft.Toolkit.Windows140 (1.0.2-rc)
 - Prism.Core (7.2.0.1367)
 - Wrapper
 - Views
 - ContentItemView.xaml
 - ItemsTreeView.xaml
 - MainWindow.xaml
 - TreeViewItemTemplateSelector.cs
 - ViewModels
 - Wrappers
 - App.xaml
 - WpApp.Models
 - WpApp.Packaging

まとめ

- ・ WPF on .NET Core 3.0 で WPF が今後もひとまず安心
- ・ アクティブに開発するなら是非 .NET Core を検討してみよう
- ・ Windows 10 API も呼び出し可能なものが多い
- ・ 最新 Windows 10 UI コントロールも
Windows UI Library 3.0 以降で使える