

Windows Client ハンズオン資料

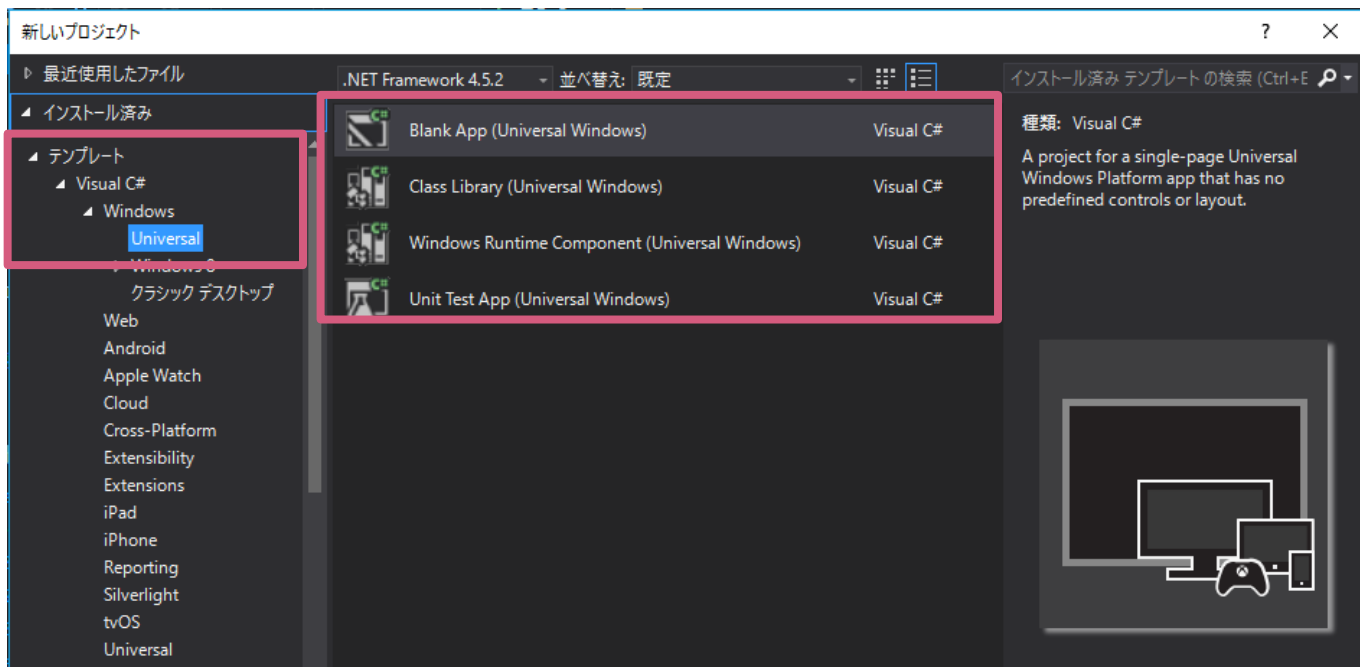
雑談対話を行うWindowsアプリの作成

ハンズオンに必要なもの

- パソコン (Windows10,Windows7,8.xのいずれかが必要です)
- Visual Studio 2015 Community Edition(無料)

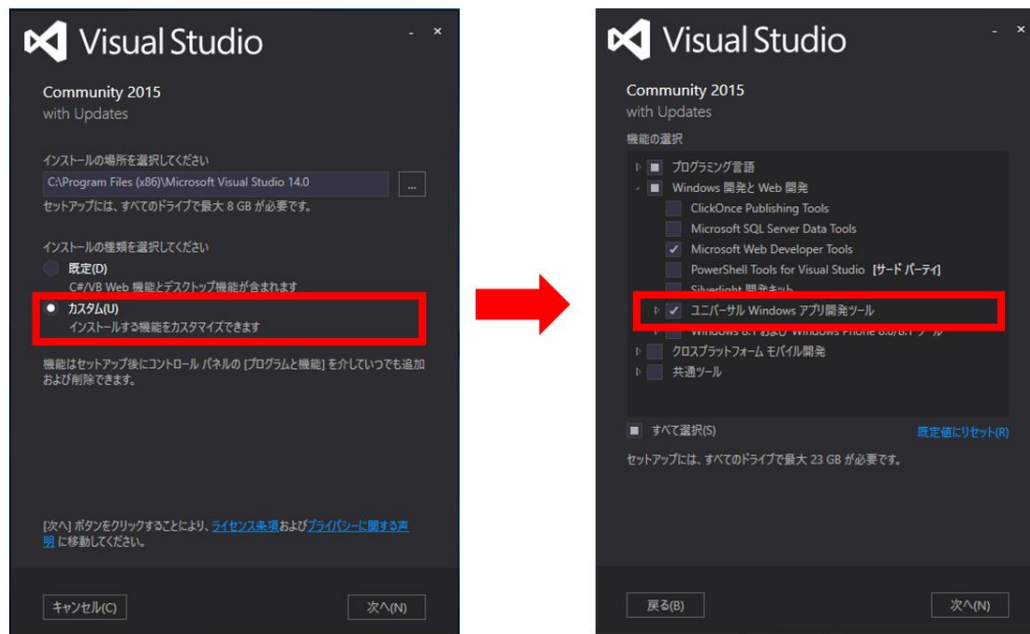
Universal Windows Platform利用確認

- Windows10の人 (Universal Windows Platformの場合)
- VisualStudio2015を立ち上げる->[ファイル]->[新規作成]->[プロジェクト]をおす
- [テンプレート]->[VisualC#]->[Windows]->[Universal]を選択
- テンプレートが4つであればOK



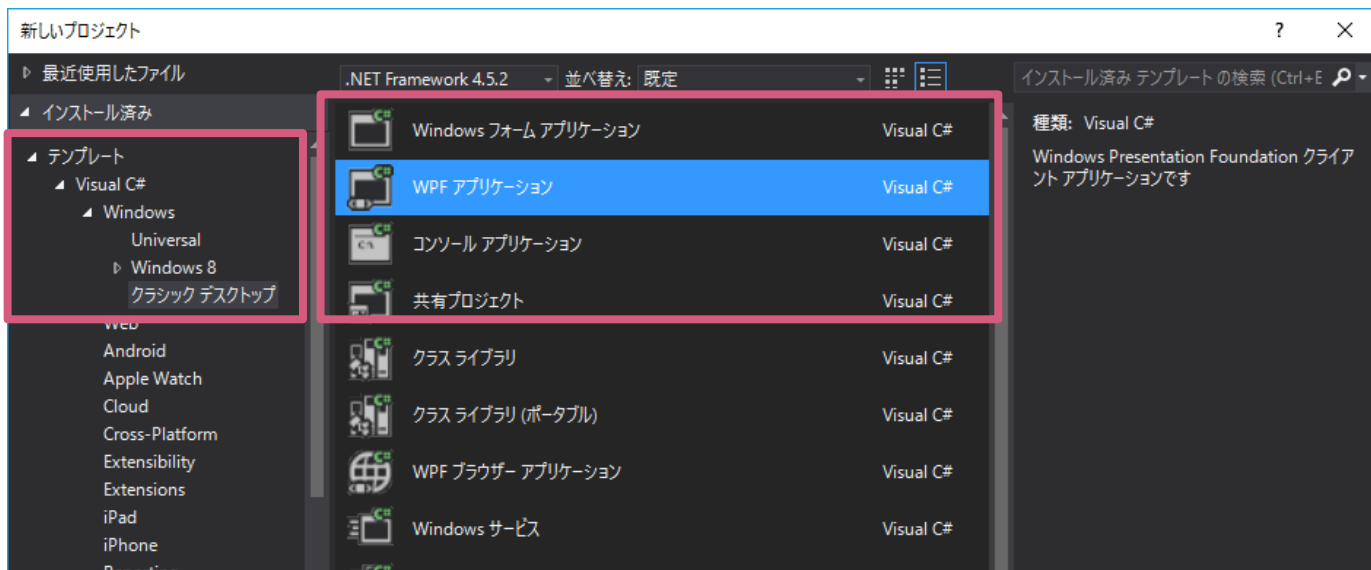
テンプレートが4つでない場合

- テンプレートが1つしかないなどの場合UWPのSDKが入っていない可能性があります
- コントロールパネルのプログラムと機能からインストーラを立ち上げ、[変更]からSDKをインストールしてください



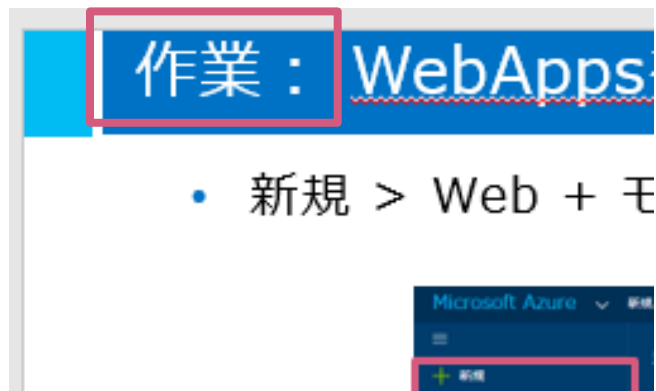
Windows Presentation Foundation 利用確認

- Windows7, Windows8.xの人はWPFを利用します
- WPFの場合、VisualStudioがインストールできていればOKです



資料の見方

- **作業資料**と**解説資料**の2種類があります
 - **作業資料**：具体的な手順が書かれています。手順に書いてあることを実行してください
 - **解説資料**：実行することはありません。解説です



資料の見方

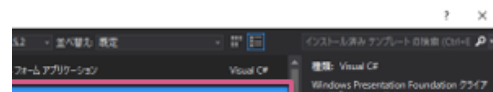
- **Windows10資料**と**Windows7,8.x資料**の2種類があります
 - ご自身が使っているOSに応じて任意に選択してください
 - 使っているOSでない資料は読み飛ばしてください

トを作成(Windows10の場合)

>[Windows]->[Universal]->[BlankApp]を選択

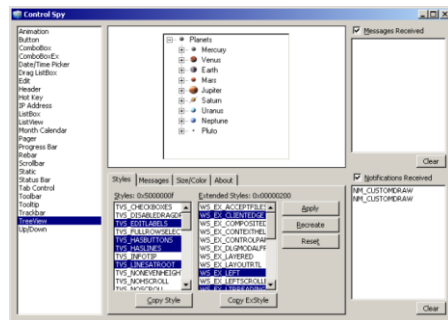
トを作成(Windows7,8.xの場合)

[Windows]->[クラシックデスクトップ]->[WPFアプリ]

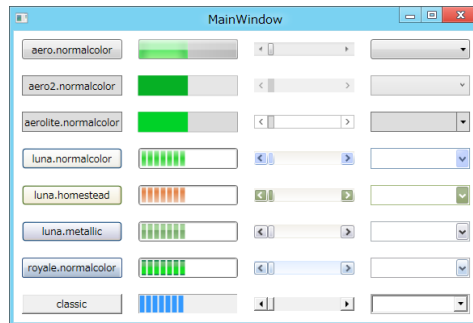


Windowsアプリ概要

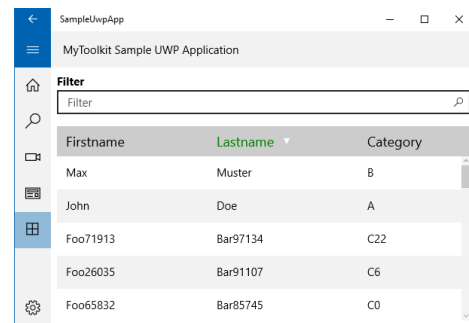
解説 : Windowsアプリの種類



Win32 API



WPF
Windows Presentation Foundation



UWP
Universal Windows Platform

時代

かなり昔

Windows7時代

Windows10時代

プログラミング言語

C or C++

XAML

C++

C#

Visual Basic

XAML

C++

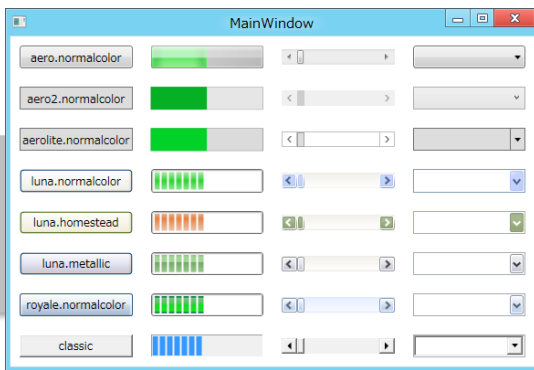
C#

Visual Basic

HTML

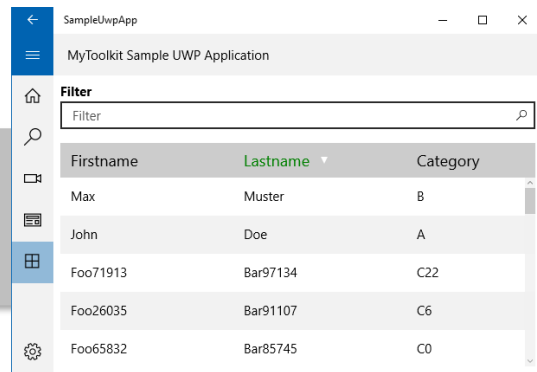
JavaScript

解説： Universal Windows Platform概要



WPF

Windows Presentation Foundation
～Windows 8.x



UWP

Universal Windows Platform
Windows10～

- UWPで大きく3つの変化

タッチインターフェース
に最適化

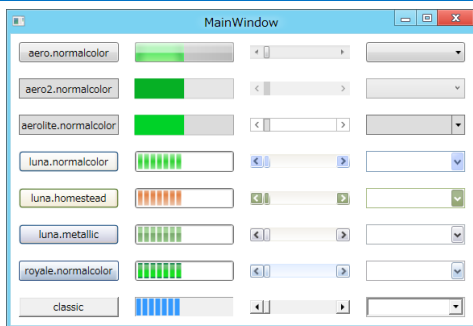
1つのアプリが
スマートフォンでも動く

Windows Storeによる
アプリの配信

解説： Universal Windows Platform概要

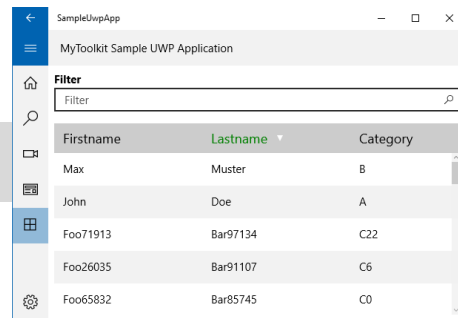


解説： Windowsアプリの基本的な作り方



WPF

Windows Presentation Foundation
～Windows 8.x



UWP

Universal Windows Platform
Windows10～

UI記述言語
XAML

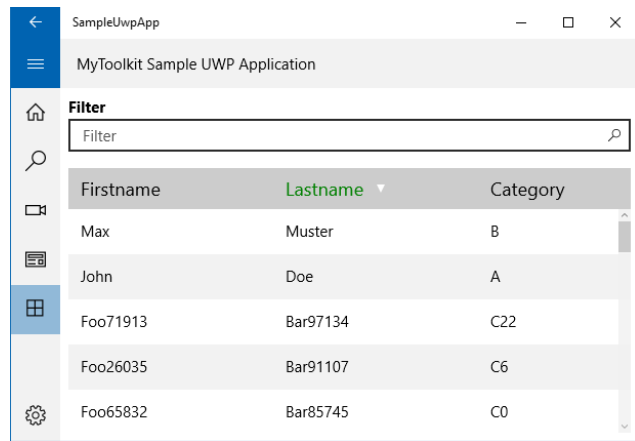


プログラミング言語

API

- WPFでもUWPでも、基本となる作り方は非常に似ている

解説： Windowsアプリの基本的な構造



Windowsアプリ

ユーザーインターフェース

```
<Page
  x:Class="App1.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:App1"
  xmlns:d="http://schemas.microsoft.com/expression/2010/framework/d"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">

  <Grid Background="{ThemeResource ApplicationBackground}">

  </Grid>
</Page>
```

XAML

ロジック

```
public sealed partial class MainPage : Page
{
    public MainPage()
    {
        this.InitializeComponent();
    }
}
```

プログラミング言語

- 見た目はXAML、ロジックは各種プログラミング言語

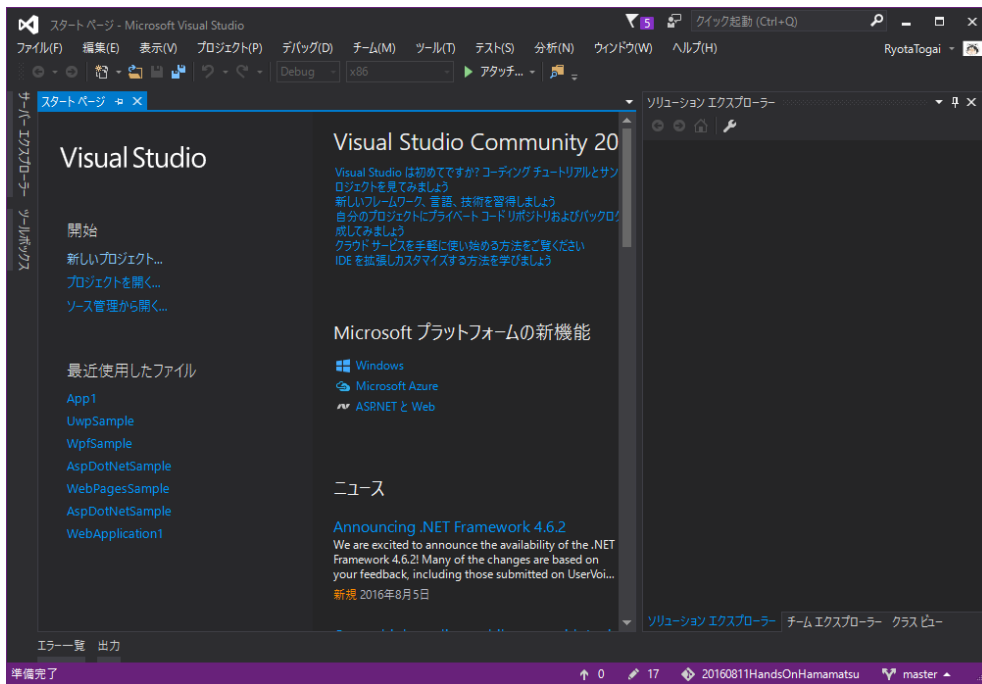
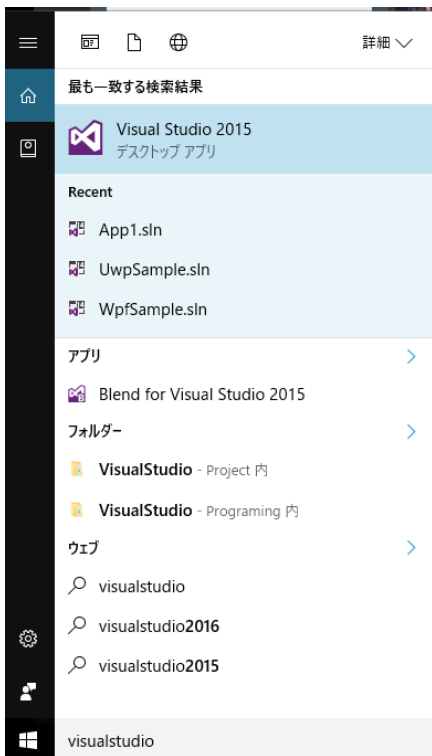
解説： Windowsアプリの基本的な作り方

- Windows10からはUWPが主流
- UWPは様々なメリットがあり、進化している
- WPFとUWPは基本的な作り方は同じ
- XAMLでUI記述、プログラミング言語でロジック記述

新しくプロジェクトを作成する

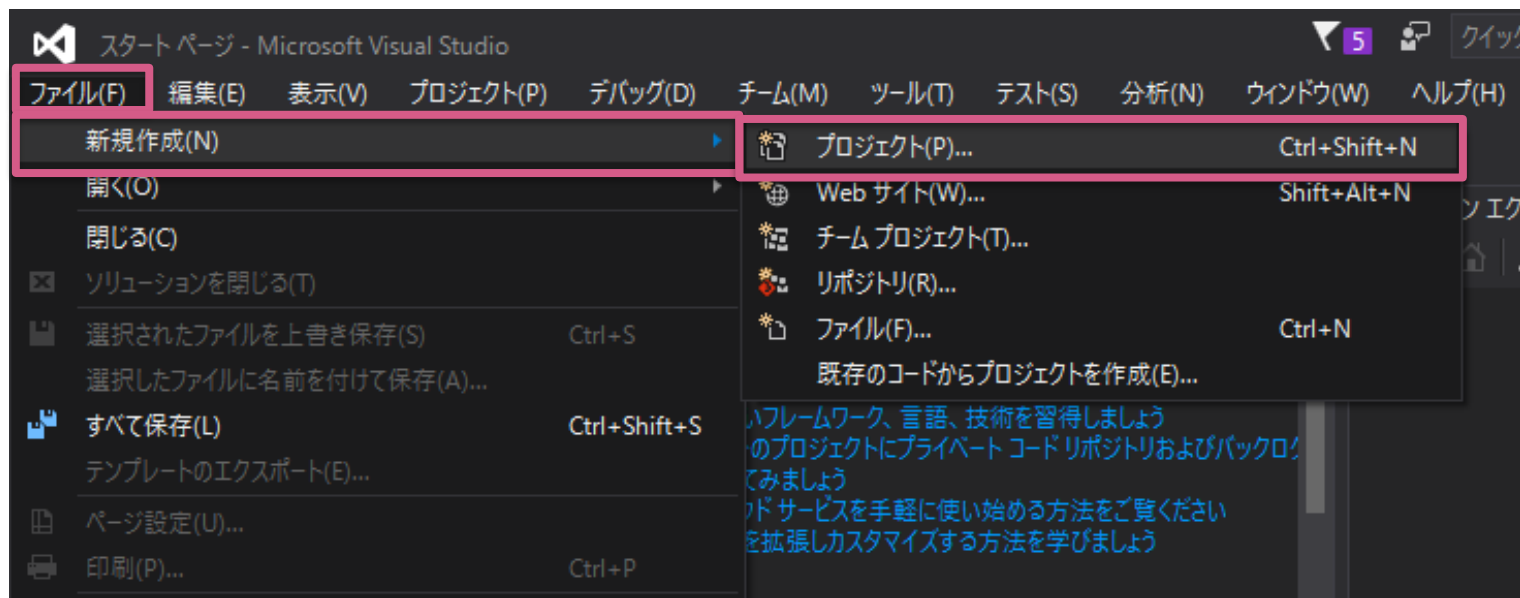
作業：VisualStudioを起動する

- [visualstudio]と検索して起動する



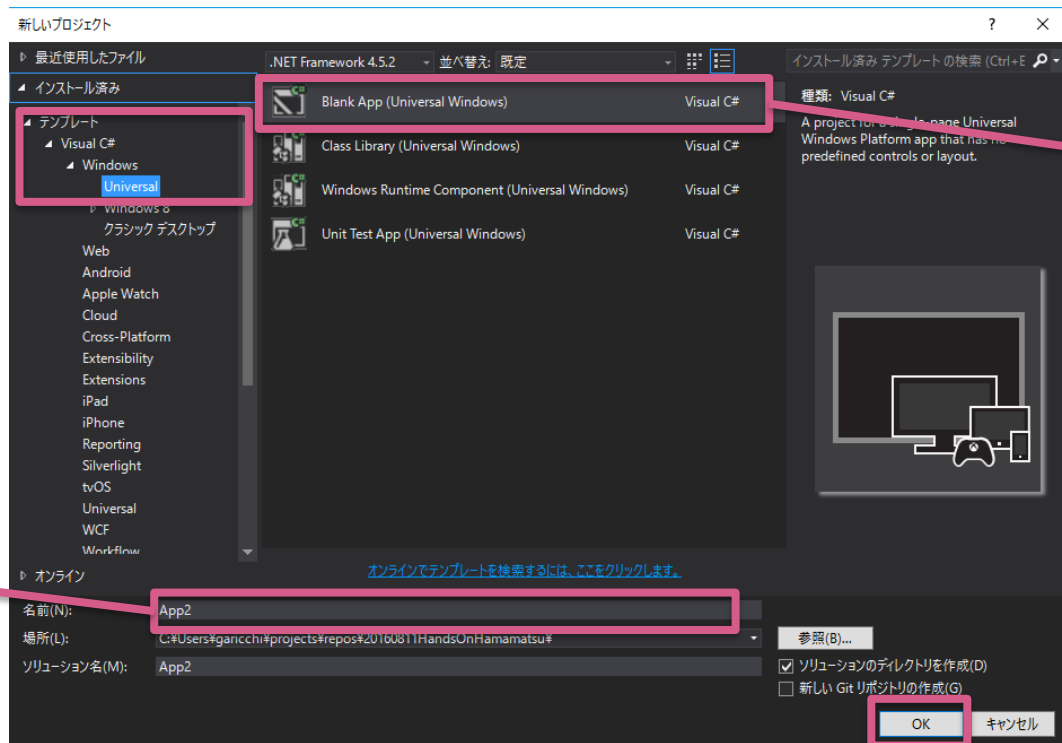
作業：新しくプロジェクトを作成する

- [ファイル]->[新規作成]->[プロジェクト]を押す



作業：プロジェクトを作成(Windows10の場合)

- [テンプレート]->[VisualC#]->[Windows]->[Universal]->[BlankApp]を選択
- 好きな名前をつけて[OK]を押す

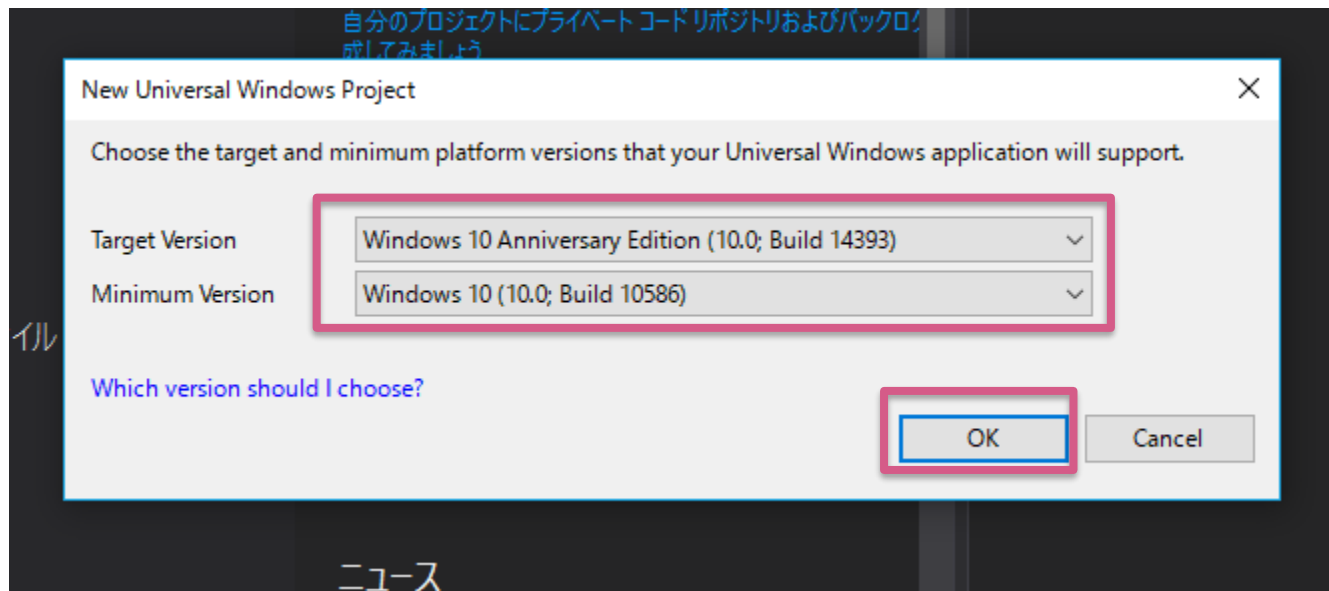


もしかしたら
[空のアプリ]かも

任意の名前をつける

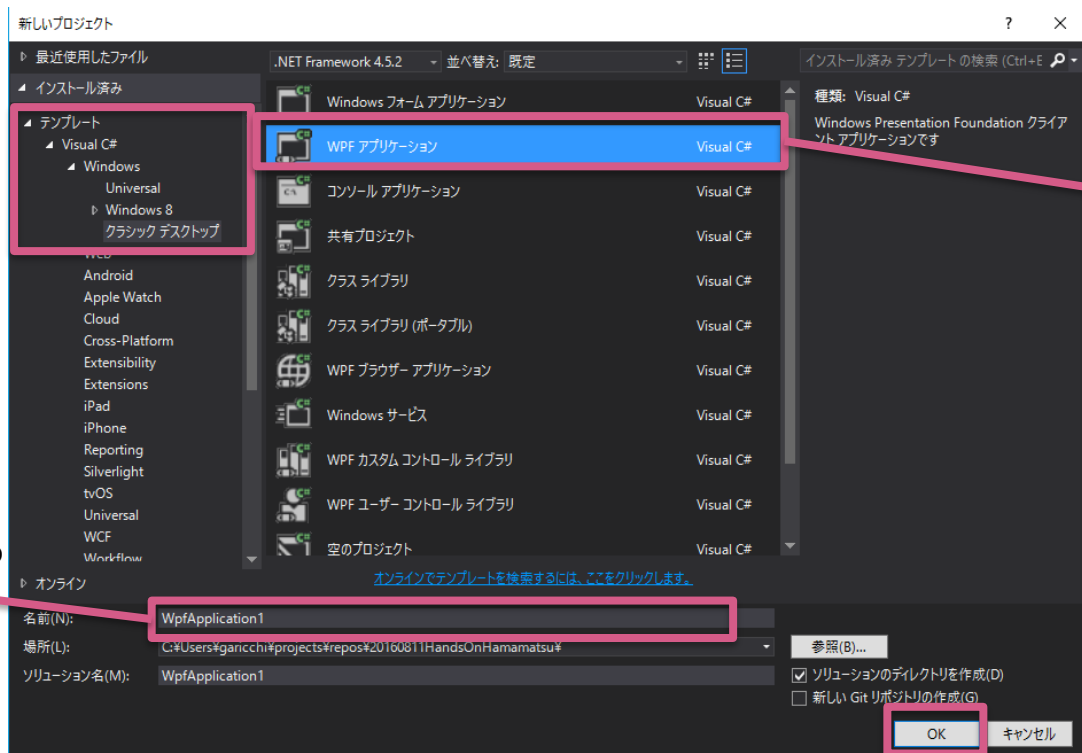
作業：プロジェクトを作成(Windows10の場合)

- ターゲットとするSDKのバージョンを設定
- 任意のバージョンを選択して[OK]を押す



作業：プロジェクトを作成(Windows7,8.xの場合)

- [テンプレート]->[VisualC#]->[Windows]->[クラシックデスクトップ]->[WPFアプリケーション]を選択
- 好きな名前をつけて[OK]を押す

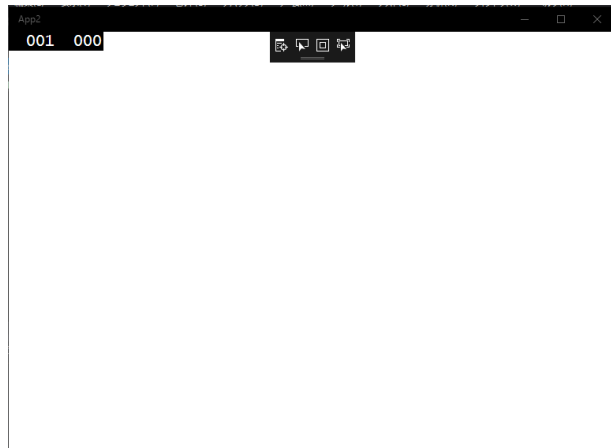
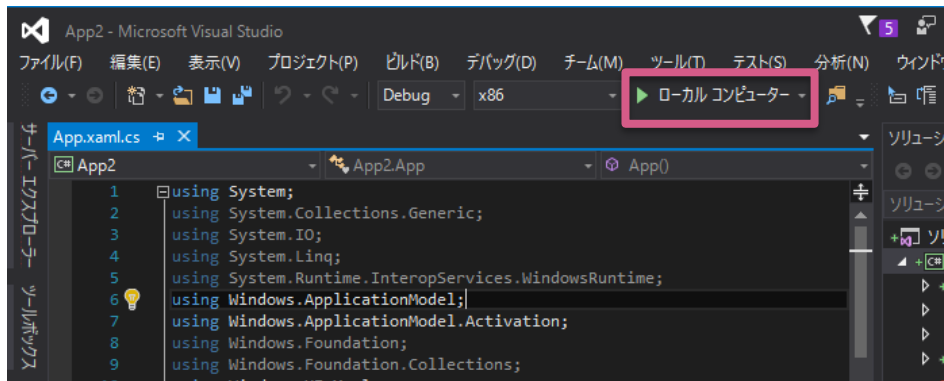


WPFアプリケーション
を選択

任意の名前をつける

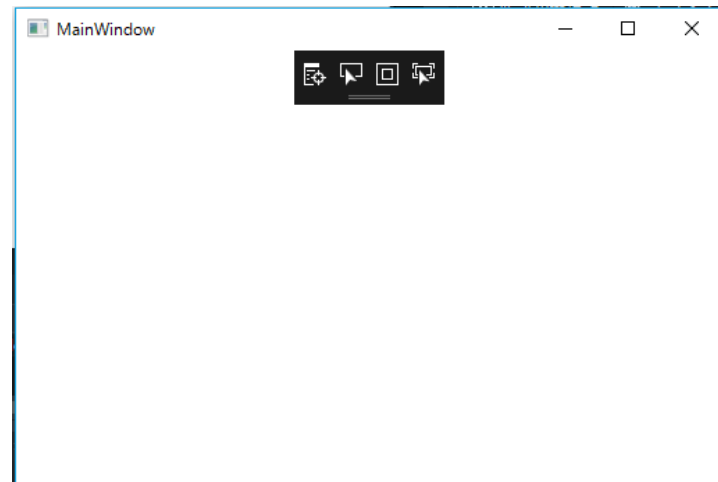
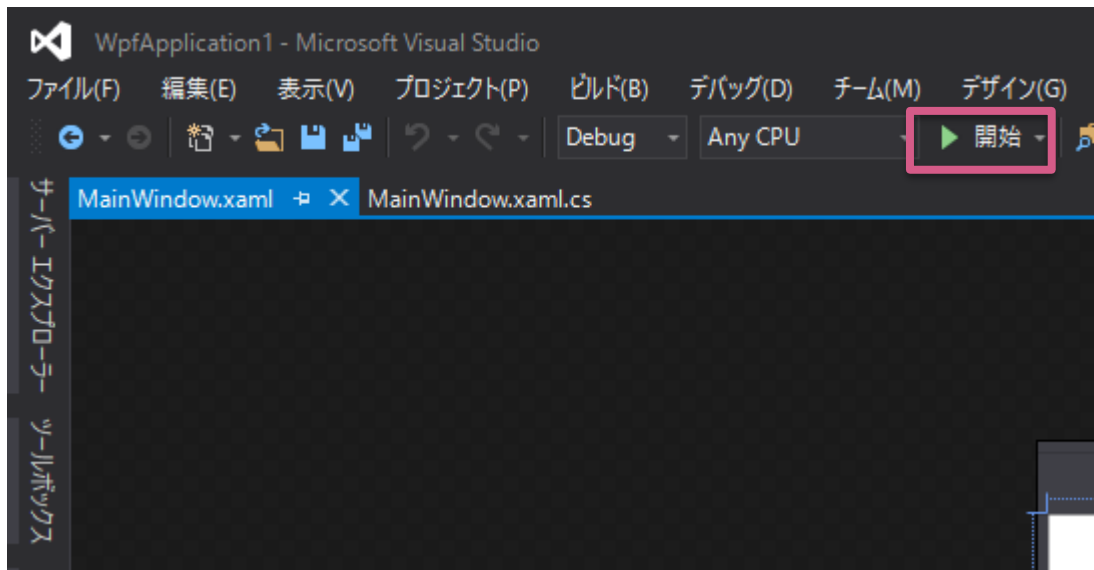
作業：アプリを実行する(Windows10の場合)

- 緑色の三角ボタン[ローカルコンピュータ]を押す



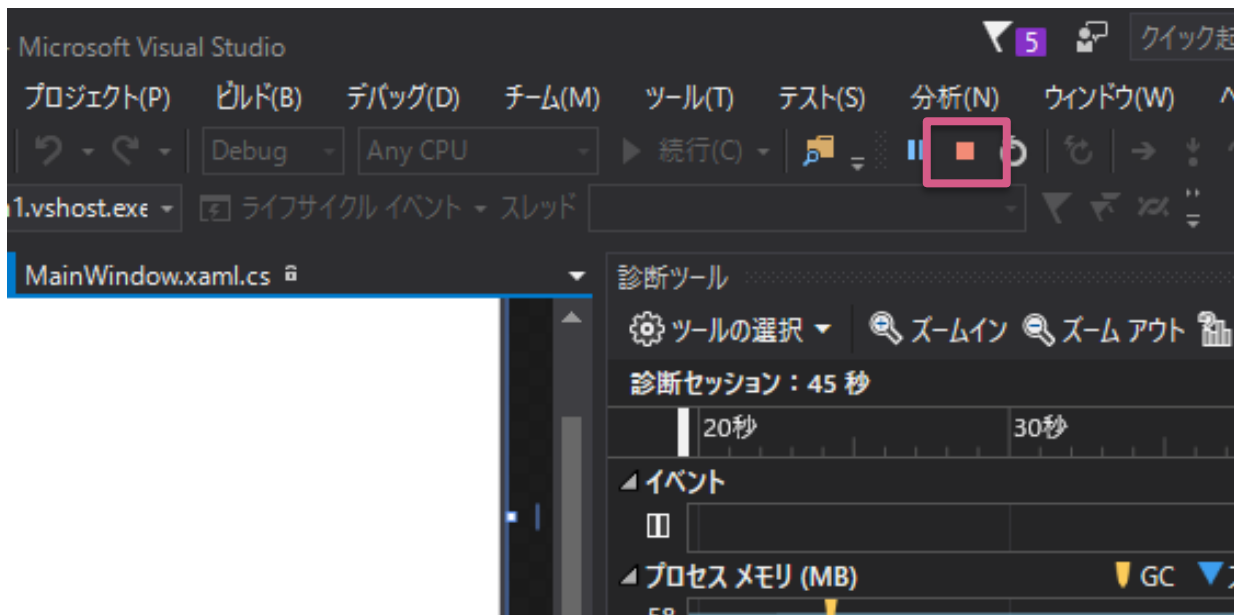
作業：アプリを実行する(Windows7,8.xの場合)

- 緑色の三角ボタン[開始]を押す



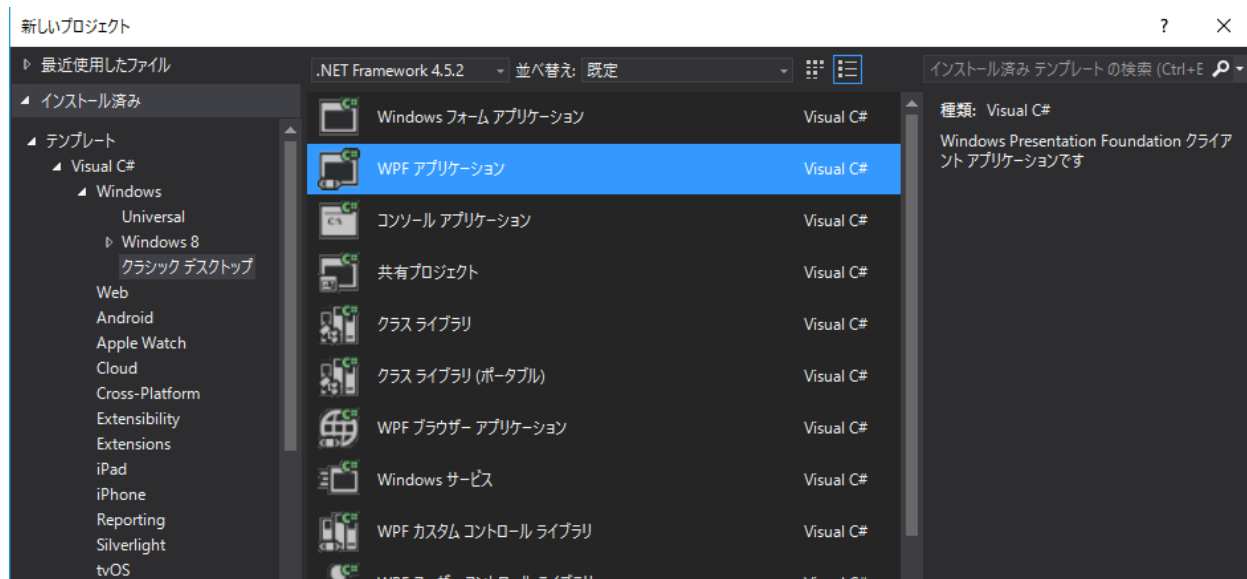
作業：アプリの実行を停止する

- 赤い■を押してアプリの実行を停止する



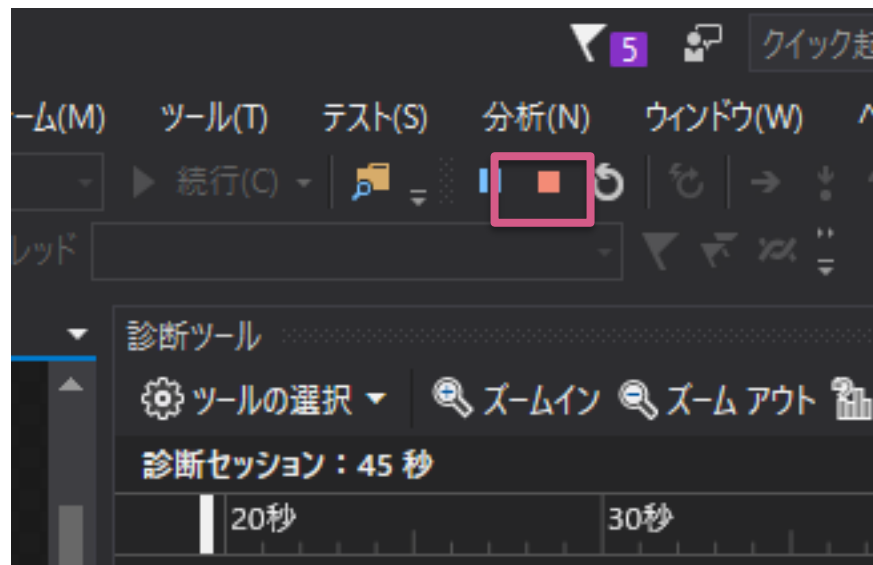
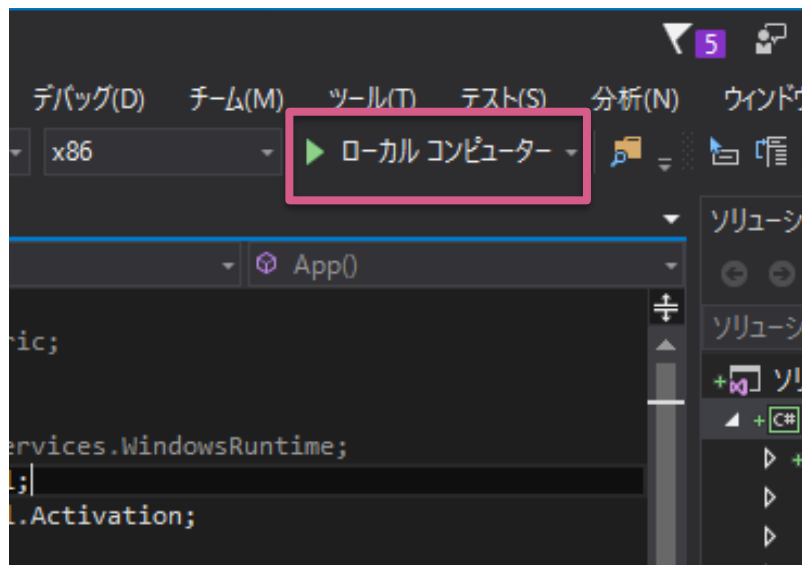
解説：VisualStudioによる新規プロジェクトの作成

- VisualStudioではどんなプログラムを作る場合でもファイルから新規プロジェクト作成を行う
- 1からプログラムを作るのは大変なので最初からテンプレートがたくさん用意されている



解説：VisualStudioによるアプリの実行、停止

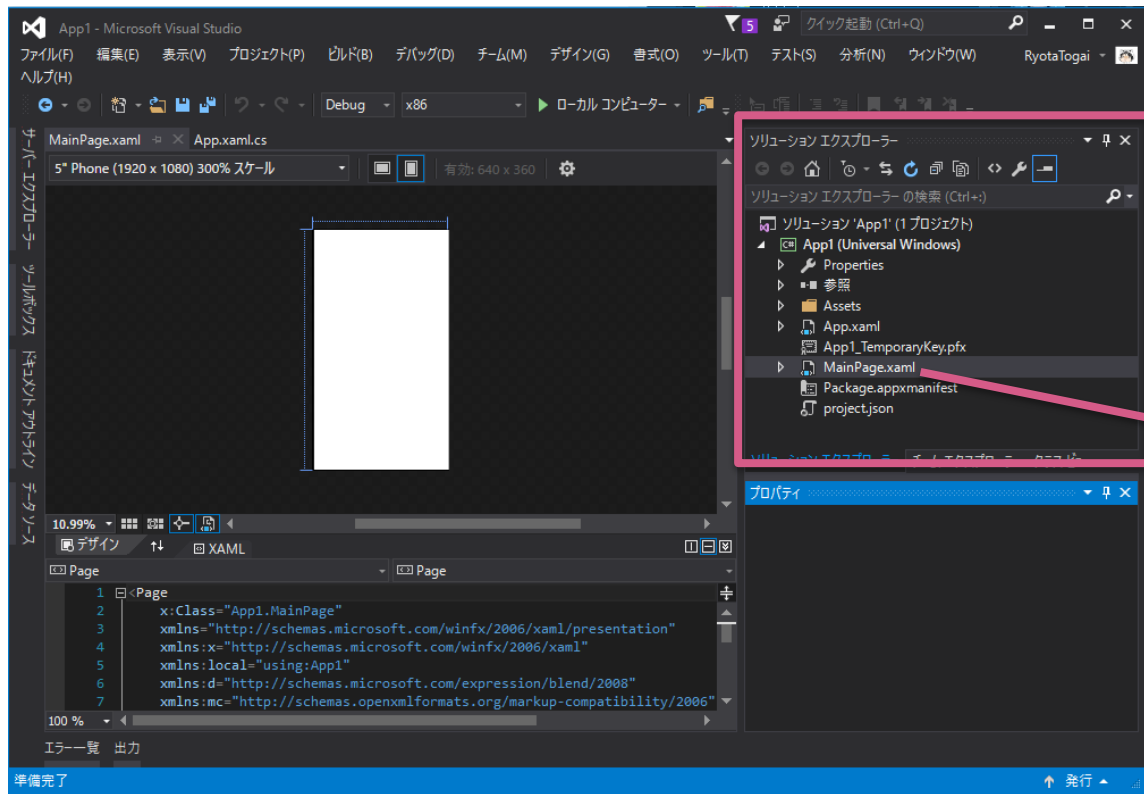
- VisualStudioではどんなプログラムも実行するときは緑の三角ボタンを押す
- 逆に停止させたいときは赤い四角ボタンを押す



Hello,Worldアプリを作る

作業：MainPage.xamlを開く

- ソリューションエクスプローラーから[**MainPage.xaml**]をダブルクリック



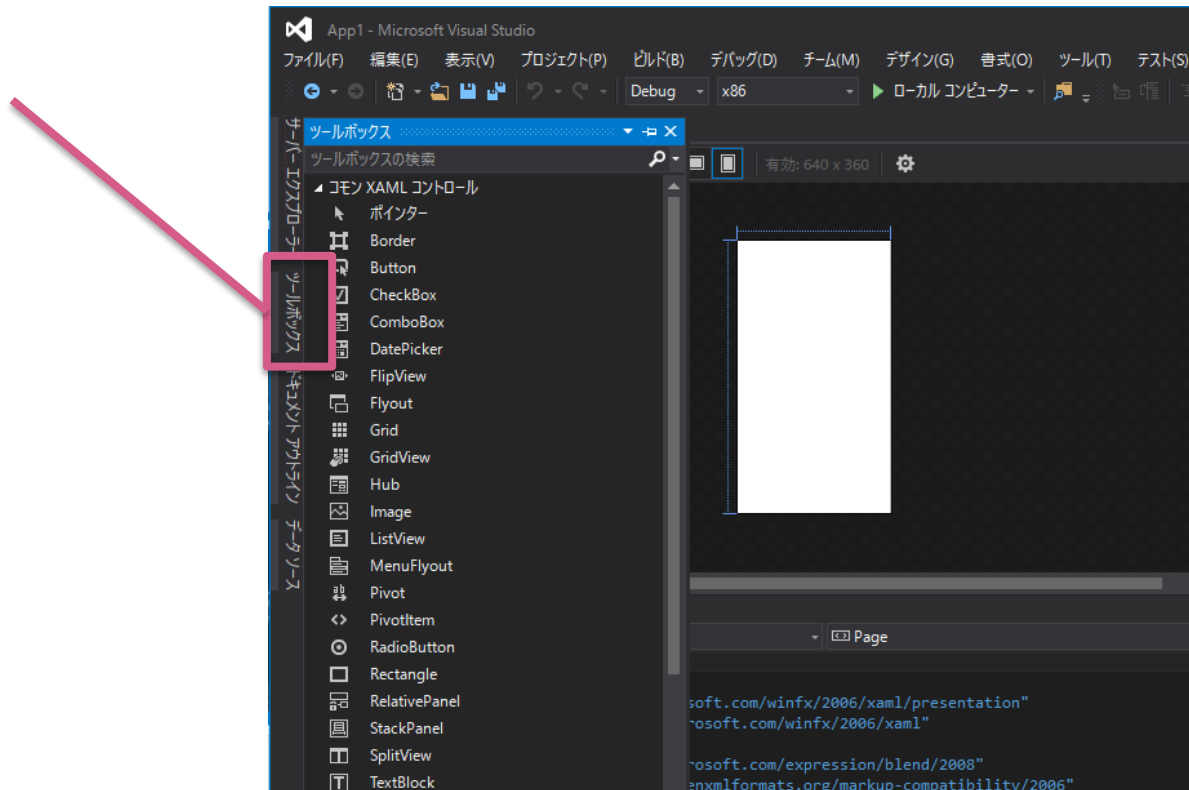
ソリューション
エクスプローラー

MainPage.xaml

作業：ツールボックスを開く

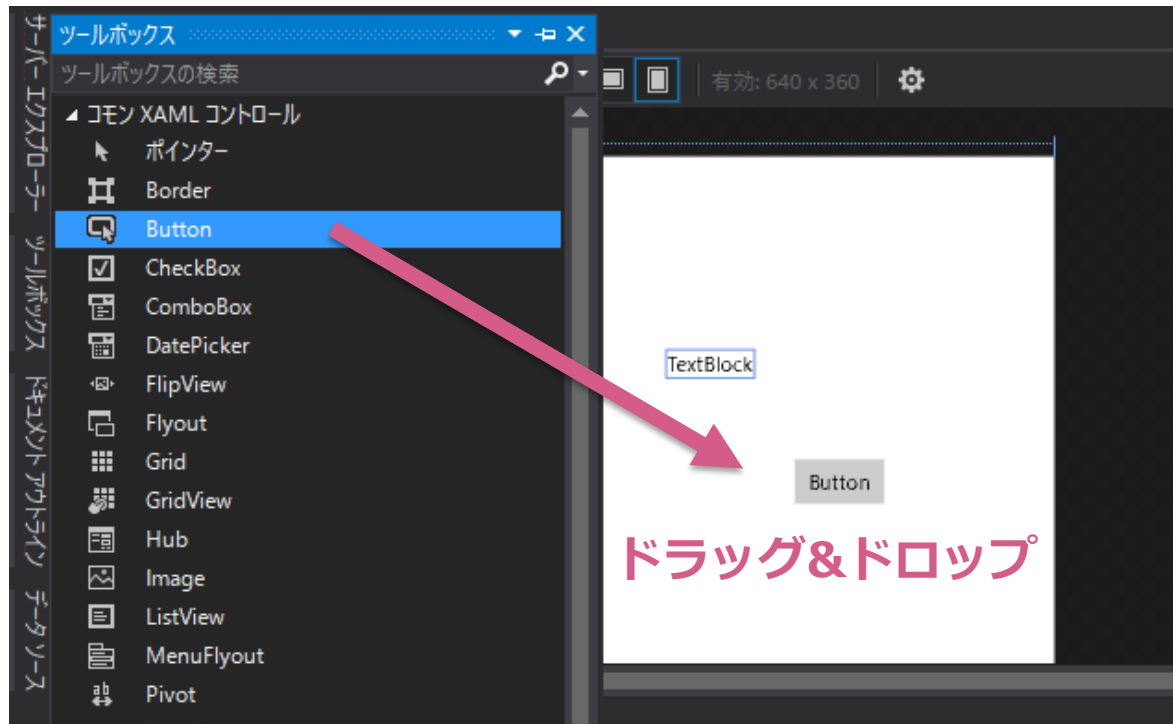
- 左にある[ツールボックス]を押してツールボックスを開きます

クリック



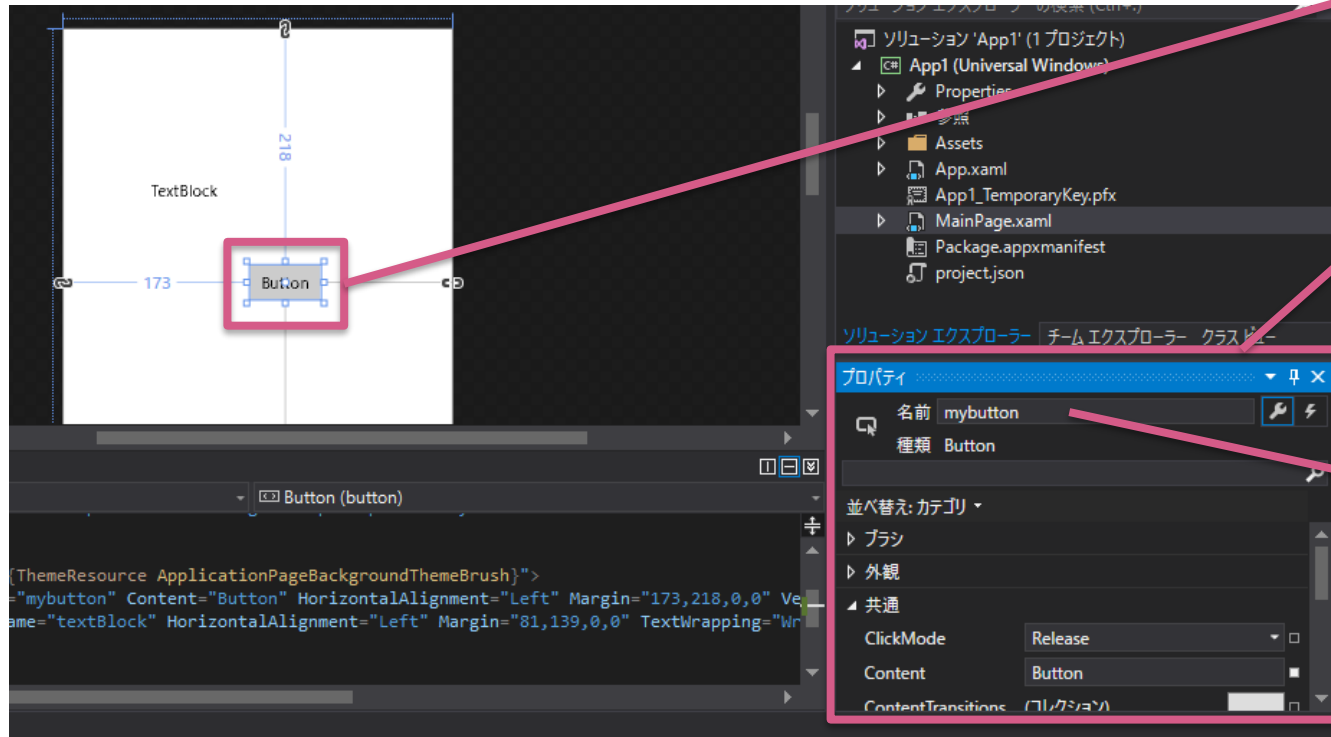
作業：コントロールを配置する

- ツールボックスから[**Button**]と[**TextBlock**]をMainPage.xamlのデザイナーにドラッグアンドドロップする



作業：ButtonのNameプロパティを変更する

- 配置したButtonのNameプロパティをmybuttonという名前にする



配置したButtonをクリックして選択

プロパティウインドウ

[名前]の項目を
mybuttonに変更する

作業：TextBlockのNameプロパティを変更する

- 配置したTextBlockのNameプロパティを**mytext**という名前にする

The screenshot displays the Visual Studio IDE with the following components:

- Design View:** A UI design showing a `TextBlock` control (highlighted with a red box) and a `Button` control. The `TextBlock` has a bounding box of approximately 81x139.
- Solution Explorer:** Located on the right, it shows the project structure for 'App1 (Universal Windows)'. The `MainPage.xaml` file is selected.
- Properties Window:** A floating window titled 'プロパティ' (Properties) is shown. It contains the following information:
 - 名前 (Name):** `mytext` (highlighted with a red box and an arrow pointing to the instruction).
 - 種類 (Type):** `TextBlock`
 - 並べ替え: カテゴリ (Sort: Category):** A dropdown menu.
 - ブラシ (Brush):** A dropdown menu.
 - 外観 (Appearance):** A dropdown menu.
 - 共通 (Common):** A section containing properties like `Text` (set to `TextBlock`), `ToolTipService.ToolTip`, and `DataContext`.
- Code View:** The bottom pane shows the XAML code for the `TextBlock` control, with the `Name` property set to `mytext`:

```
<TextBlock Name="mytext" HorizontalAlignment="Left" Margin="81,139,0,0" TextWrapping="Wrap" />
```

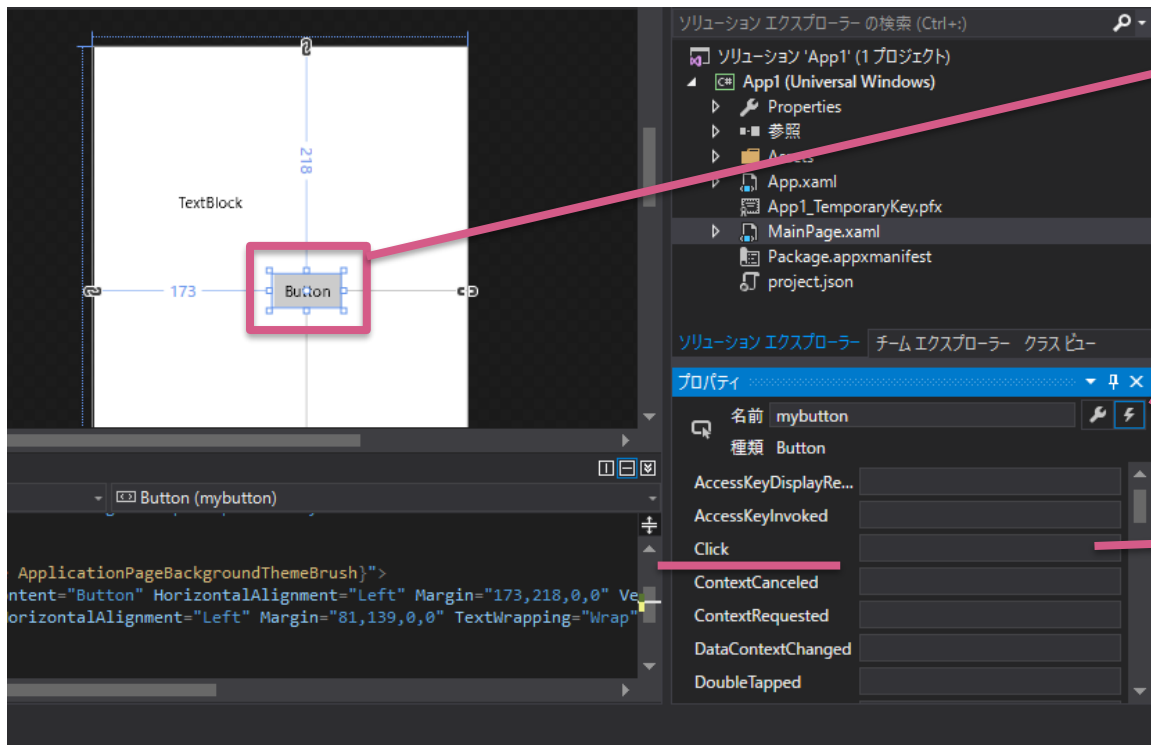
Annotations with red arrows point to the `TextBlock` in the design view, the `Properties` window, and the `名前` property field, all corresponding to the Japanese instructions on the right.

配置したTextBlockをクリックして選択

プロパティウインドウ

[名前]の項目を**mytext**に変更する

作業：ButtonのClickイベントにイベントハンドラを追加する



配置したButtonを
クリックして選択

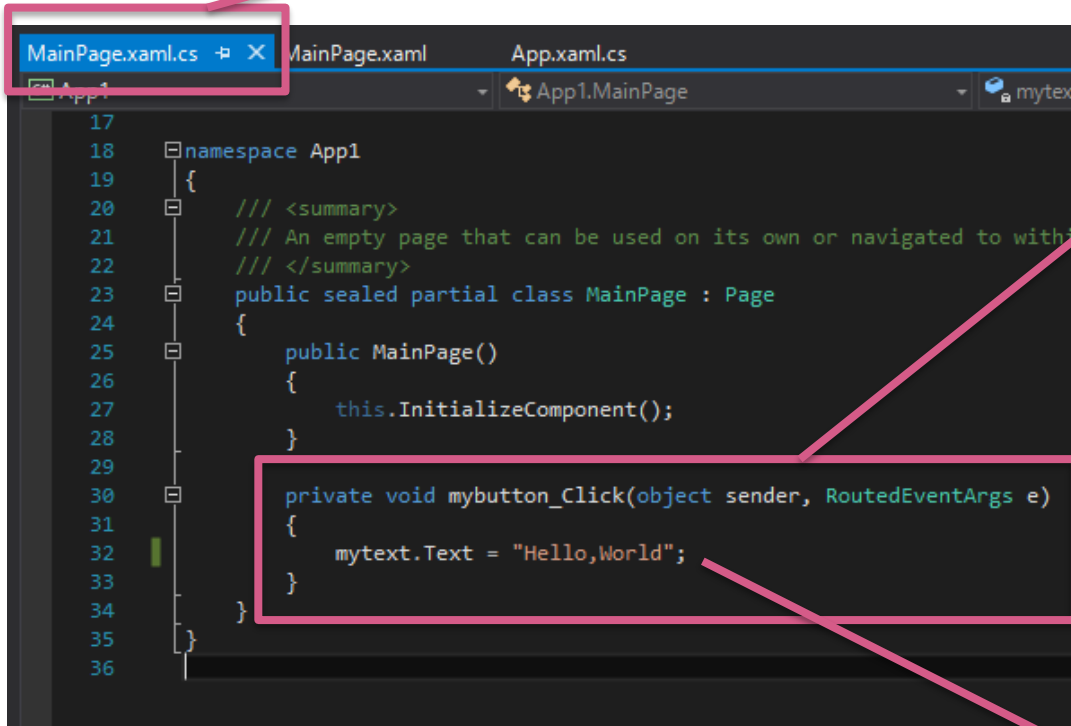


このマークを
クリックする

[Click]の右にある
テキストボックスを
ダブルクリックする

作業：ClickされたときにHello,Worldと表示するコードを書く

MainPage.xaml.csに自動的に移動したことを確認する



The screenshot shows the Visual Studio IDE with the 'MainPage.xaml.cs' file open. A pink box highlights the tab at the top. Another pink box highlights the 'mybutton_Click' method implementation. A pink arrow points from the text 'MainPage.xaml.csに自動的に移動したことを確認する' to the tab. Another pink arrow points from the text 'mybutton_Click関数があることを確認する' to the method implementation. A third pink arrow points from the text '以下のコードをmybutton_Click関数内に書く' to the line 'mytext.Text = "Hello,World";'.

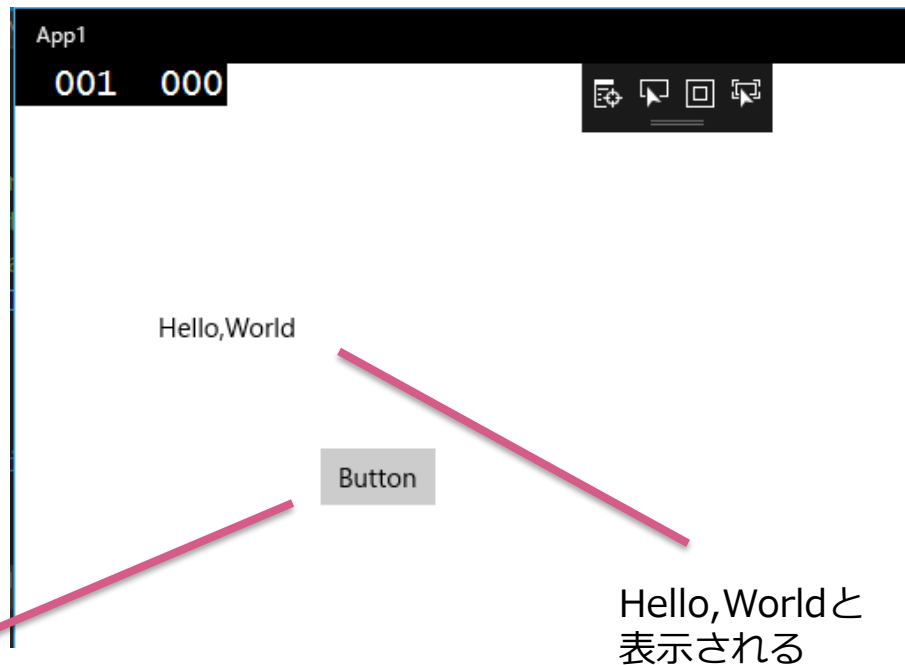
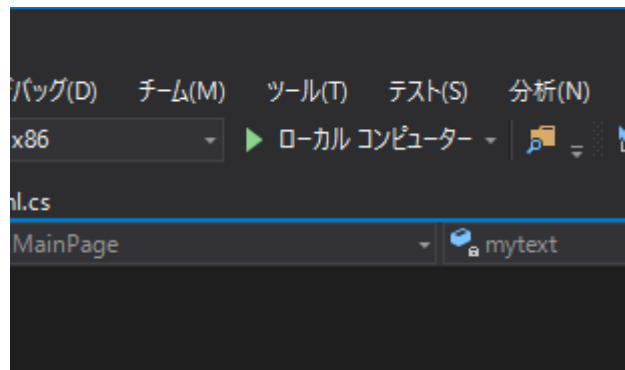
```
17
18 namespace App1
19 {
20     /// <summary>
21     /// An empty page that can be used on its own or navigated to with
22     /// </summary>
23     public sealed partial class MainPage : Page
24     {
25         public MainPage()
26         {
27             this.InitializeComponent();
28         }
29
30         private void mybutton_Click(object sender, RoutedEventArgs e)
31         {
32             mytext.Text = "Hello,World";
33         }
34     }
35 }
36
```

mybutton_Click関数があることを確認する

以下のコードをmybutton_Click関数内に書く

```
mytext.Text = "Hello,World";
```

作業：実行してHello,Worldを確認する



緑の三角[ローカルコンピュータ]を押して実行
WPFの人には[開始]ボタン

Buttonを押すと

Hello,Worldと
表示される

解説：XAML画面とデザイナー

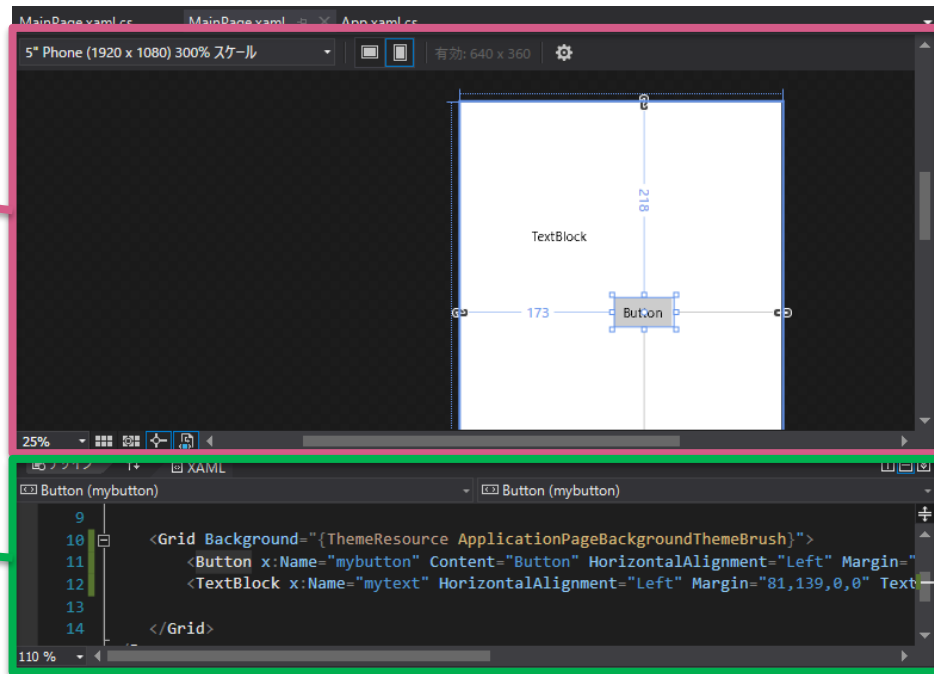
- アプリが実行されたとき、最初にロードされる画面は**MainPage.xaml**
- .xamlファイルを開いたとき、Visual Studioは**デザイナーも表示**してくれる
- デザイナーはxamlコードをアプリの画面に近い形で表示してくれる

XAMLデザイナー



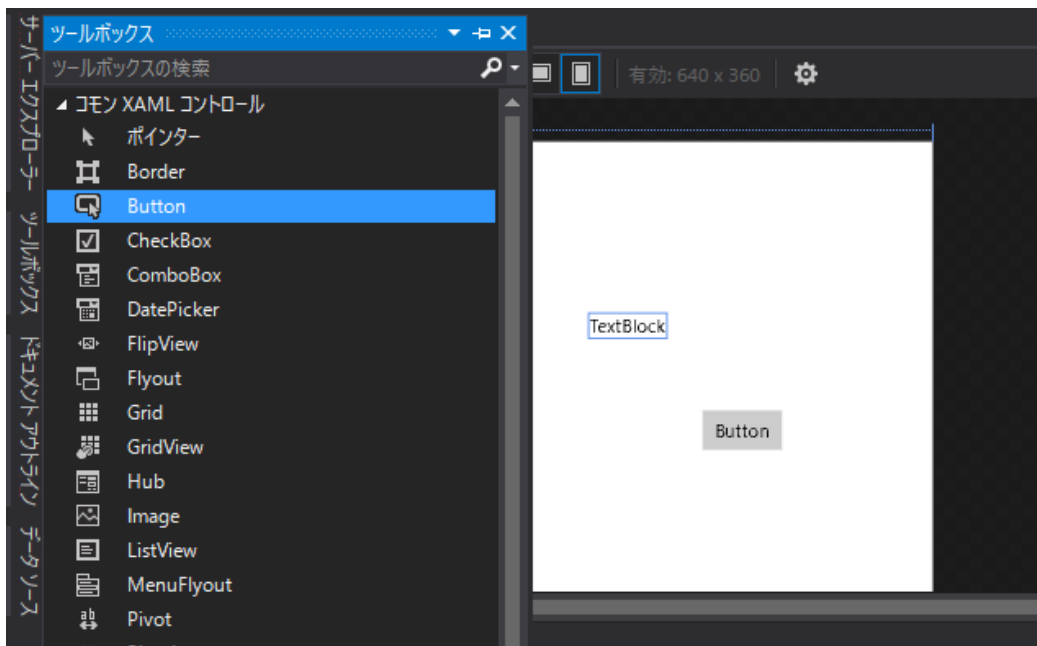
どちらも同じ
MainPage.xaml
を表示している

XAMLコード



解説：XAMLコントロールの配置

- ButtonやTextBlockなど、画面を構成する要素は[ツールボックス]からドラッグアンドドロップで配置できる

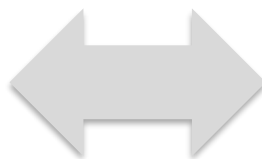


解説：XAMLコントロールのプロパティ編集

- XAMLコントロールの**プロパティ(設定)**を変更するには**コントロールを選択してプロパティウインドウを操作**する
- 今回操作した[名前]プロパティは[コントロールの名前]であり、**C#のコードとXAMLとの共通のID**となる

```
<Grid Background="{ThemeResource Appl
  <Button x:Name="mybutton" Content
  <TextBlock x:Name="mytext" Horizo
</Grid>
```

XAMLコード



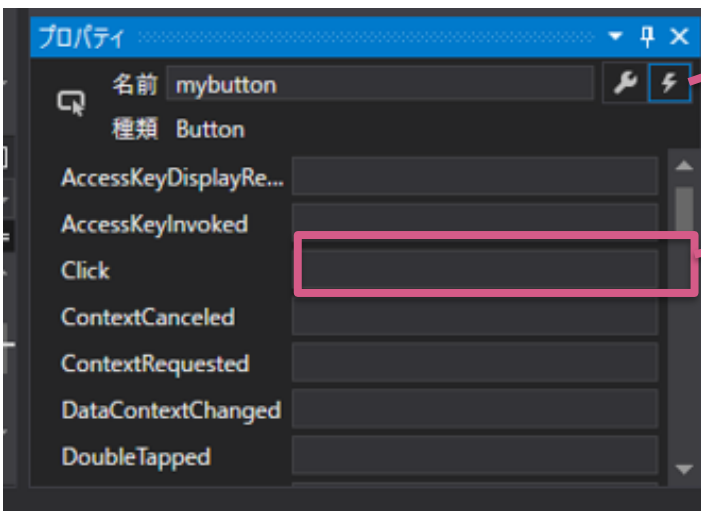
```
private void mybutton_Click(object sender, R
{
    mytext.Text = "Hello,World";
}
```

C#コード

TextBlockコントロールを差す
共通のID「mytext」

解説：イベントハンドラの登録

- XAMLコントロールにはユーザー操作によってさまざまな**イベントが発生する**(例 Clickイベント)
- イベントが発生したときにプログラムを実行するには**イベントハンドラ(イベントを受け取るC#のコード)を登録**する
- イベントハンドラを登録するにはイベントの右にあるテキストボックスをダブルクリックする

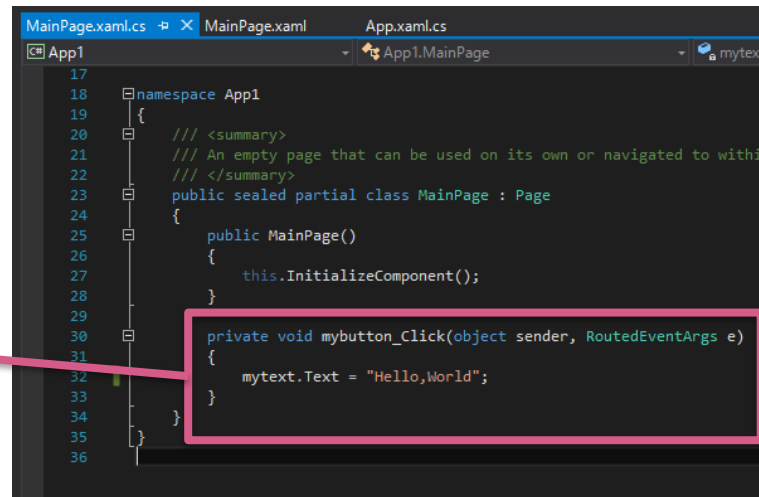


イベント一覧を
表示するボタン

ダブルクリックすると

イベントハンドラが
自動生成される

(Clickイベントが
発生すると
この関数が実行される)



解説：イベントロジックの記述

- イベントが発生したら、配置したTextBlockにHello,Worldと表示したい
- →TextBlockのプロパティにHello,Worldと入れればOK
- XAMLコントロールの表示や設定を変更するにはプロパティをC#のコードから変更する
- C#のコードからXAMLコントロールのプロパティを変更するには
- **[コントロールの名前].[プロパティ名] = 値**

```
mytext.Text = "Hello,World";
```

設定した名前

ドット

プロパティ名

値を代入

解説：XAML画面のロジック記述まとめ

～アプリ設計時～

①コントロールの配置

②共通のID
(Nameプロパティ)の設定

③イベントハンドラの登録

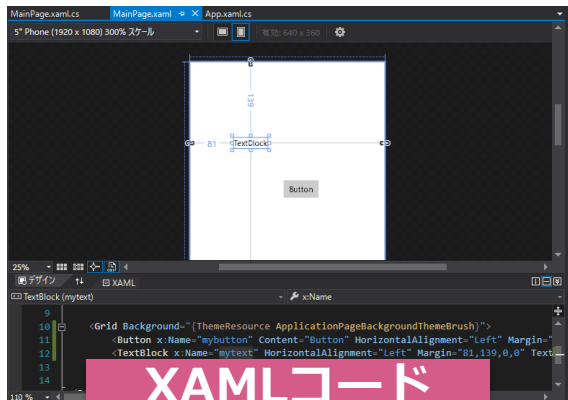
④プロパティ変更処理の記述

～アプリ実行時～

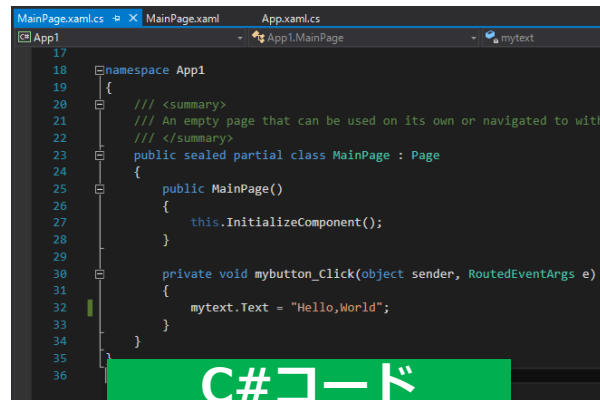
①ユーザーによってイベント発生

②イベントに登録された
イベントハンドラが呼び出される

③UIプロパティ変更処理によって
画面が書き換えられる



XAMLコード
(UI記述)



C#コード
(ロジック記述)

対話アプリを作る

作業：テンプレートのダウンロード

- docomoの雑談対話APIを利用して対話するアプリをつくります
- テンプレートを用意したのでここからダウンロードしてください

<https://github.com/mspjp/20160811HandsOnHamamatsu>

mspjp / 20160811HandsOnHamamatsu

Unwatch 8 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

No description or website provided. — Edit

12 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/mspjp/20160811HandsOnHamamatsu>

Open in Desktop Open in Visual Studio

Download ZIP

File	Commit
AspDotNetSample	add comment for View's JavaScript
AzureSample	remove mysql sample
UwpSample	add wpf project
WpfSample	fix bug of wpf
.gitattributes	add uwp project
.gitignore	Initial commit
LICENSE	Initial commit

ダウンロード

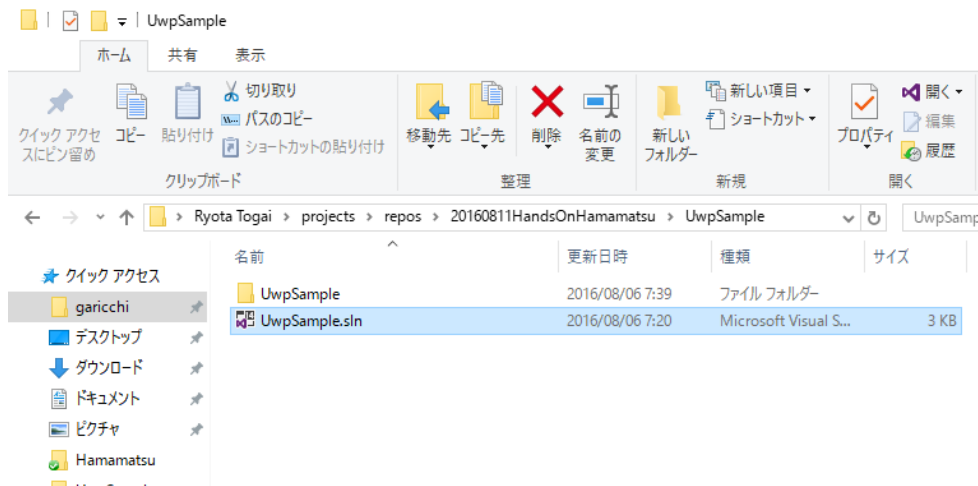
作業：対話アプリのプロジェクトを開く

Windows10の人は

20160811HandsOnHamamatsu/UwpSample/UwpSample.sln を開く

Windows7,8.xの人は

20160811HandsOnHamamatsu/WpfSample/WpfSample.sln を開く



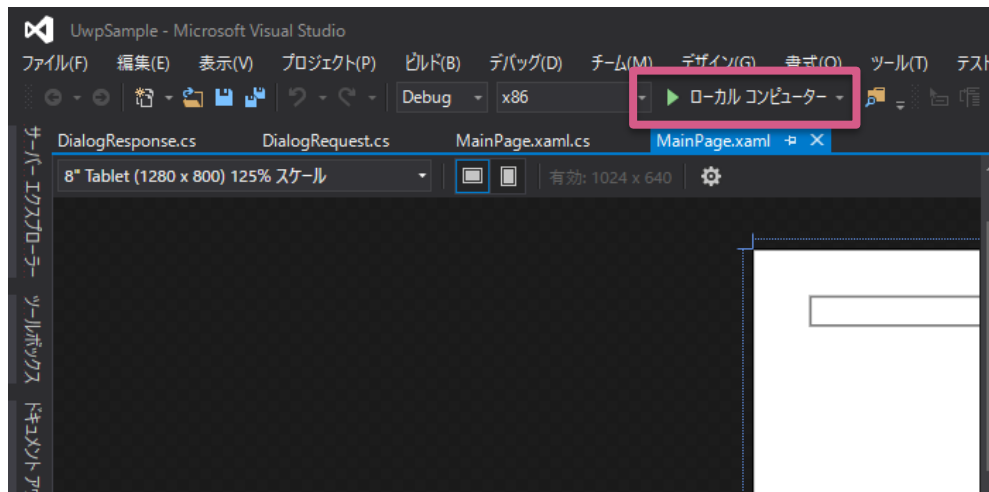
作業：対話アプリを実行する

Windows10の人は

[緑の三角]ローカルコンピュータ をクリック

Windows7,8.xの人は

[緑の三角]開始 をクリック



作業：対話アプリで遊ぶ

- TextBoxに発話を入れてSendボタンを押すとシステムと対話することができます

