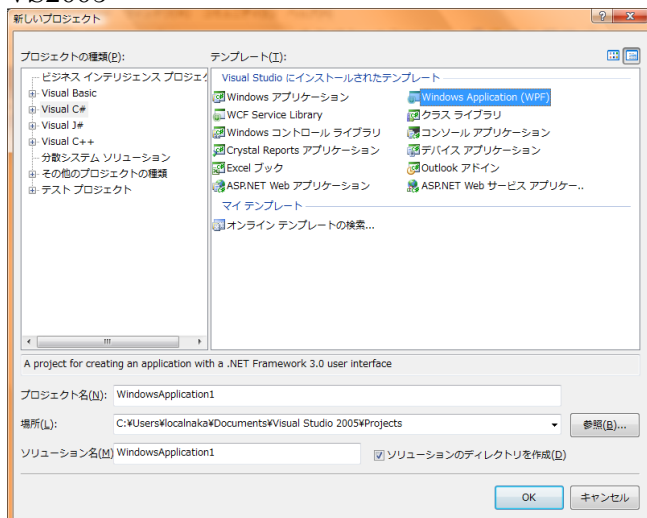


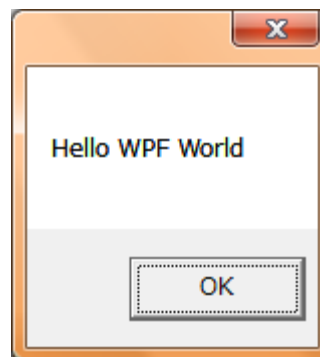
まずプロジェクトを作成します。

VS2005



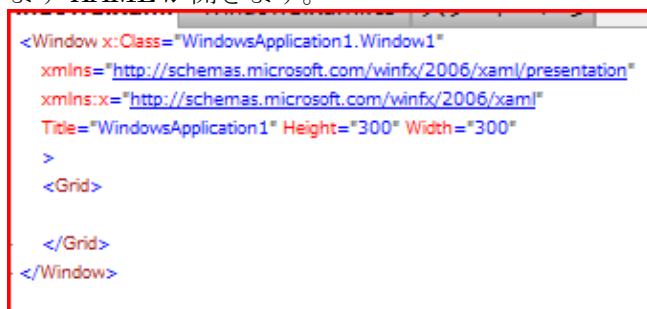
とりあえずメッセージボックスで出しましょう。

```
MessageBox.Show("Hello WPF World");
```

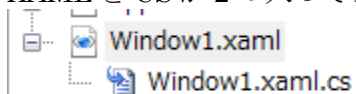


完成

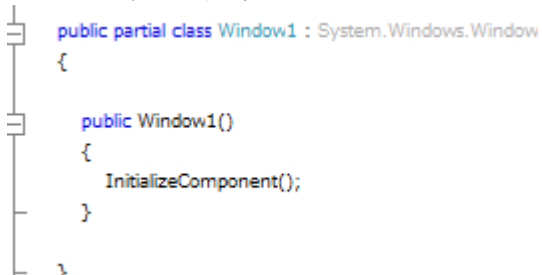
まず XAML が開きます。



XAML と CS が 2 つ入ってます。



ソースの中身は簡単



## DEMO2

Grid の中にボタンを配置します。  
ボタンの表面は **Content** で定義します。  
クリックイベントは **Click** で定義します。

```
<Grid>  
    <Button Content="Push Me!" Click="Button_Click"/>  
</Grid>
```

続いてソースに記載します。

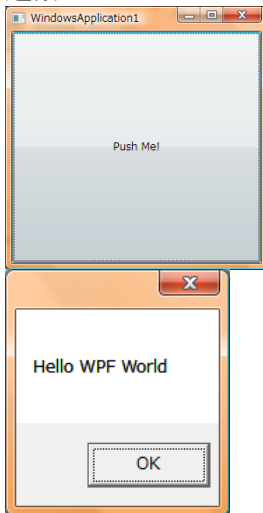
```
public void Button_Click ( object sender, RoutedEventArgs args)  
{  
    MessageBox.Show("Hello WPF World");  
}
```

この void 戻りの、object と、RoutedEventArgs というのはお約束になり、ほとんどのイベントがこれで受け取ることができます。

イベントの定義

Button1.OnClick += Button\_Click のようなコードは記載する必要がありません。

起動



## DEMO3

ドキュメントクラスを作成します。

WPF の場合には **public** なプロパティにする必要があるのですが、かならずプロパティにしてください。

```
public class Doc
{
    public string text1{get { return _text1; }set { _text1 = value; }}
    public string text2{get { return _text2; }set { _text2 = value; }}
    public string text3{ get { return _text3; } set { _text3 = value; }}
    private string _text1, _text2, _text3;
    public void Add() {this.text3 = this.text1 + this.text2; }
}
```

これで必要な実装クラスは完成です。  
これを画面に割り当てます。

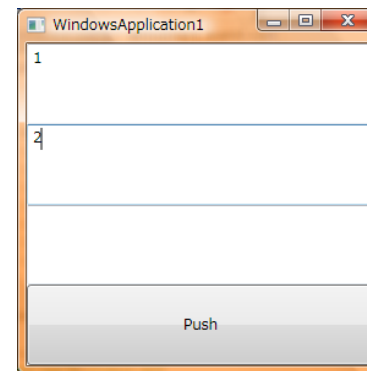
```
<Window x:Class="WindowsApplication1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:WindowsApplication1="clr-namespace:WindowsApplication1;assembly="
    Title="WindowsApplication1" Height="300" Width="300"
>
<Window.DataContext>
    <WindowsApplication1:Doc/>
</Window.DataContext>
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <TextBox Text="{Binding Path=text1}" />
    <TextBox Text="{Binding Path=text2}" Grid.Row="1" />
    <TextBox Text="{Binding Path=text3}" Grid.Row="2" />
    <Button Content="Push" Grid.Row="3" Click="Add" />
</Grid>
```

```
</Window>
```

ビューの処理としてドキュメントの **Add** を呼び出すように記載します。

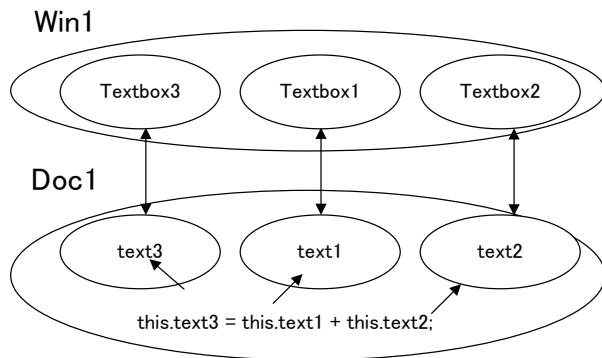
```
public void Add ( object sender, RoutedEventArgs args)
{
    Doc d = this.DataContext as Doc;
    if ( d != null )
    {
        d.Add();
    }
}
```

実行してみましょう。



ボタンを押しても反映されるはずが反映されません。  
デバッグしてみましょう。

デバッグではうまく値が反映されています。



このドキュメントからビューへの反映は、値が変わったことを通知する必要があります。

```
public class Doc : INotifyPropertyChanged
{
    public string text1 { get { return _text1; } set { _text1 = value; this.FirePropertyChanged("text1"); } }
    public string text2 { get { return _text2; } set { _text2 = value; this.FirePropertyChanged("text2"); } }
    public string text3 { get { return _text3; } set { _text3 = value; this.FirePropertyChanged("text3"); } }
    private string _text1, _text2, _text3;

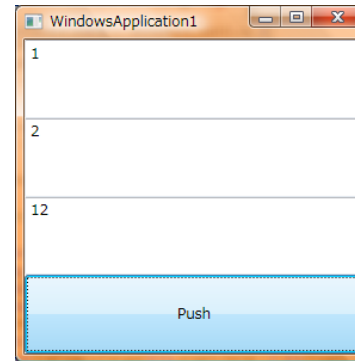
    public void Add() { this.text3 = this.text1 + this.text2; }

    public event PropertyChangedEventHandler PropertyChanged;

    protected void FirePropertyChanged(string propertyName) { if (this.PropertyChanged != null)
this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName)); }
}
```

この部分を追記します。

そして実行



拍手！！

## DEMO4

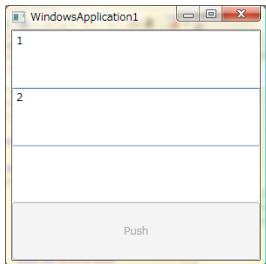
テキスト1とテキスト2に値が書かれていないとボタンを押せなくしたい。  
まずは XAML を変更します。

```
<Button Content="Push" Grid.Row="3" Click="Add" IsEnabled="{Binding Path=ButtonEnabled}"/>
```

ButtonEnabled にバインディングすることになります。

```
public bool ButtonEnabled
{
    get { return string.IsNullOrEmpty(this.text1) == false && string.IsNullOrEmpty(this.text2) == false; }
}
```

これで実行します。  
あれうまくいきません。  
入力しても有効にならないのです。



これは先ほど説明した NotifyPropertyChanged による通知を上げていないためです。

解決方法は 2 種類あります。

### 解決策 1

計算のもとになっている値の変更時に ButtonEnabled も変更通知を上げてもらう。

```
public string text1 { get { return _text1; } set { _text1 = value; this.FirePropertyChanged("text1");
this.FirePropertyChanged("ButtonEnabled"); } }
public string text2 { get { return _text2; } set { _text2 = value; this.FirePropertyChanged("text2");
```

```
this.FirePropertyChanged("ButtonEnabled"); } }
```

実行

うまくいきました。

### 解決策 2

Text1 と text2 の変更時に通知が上がるわけですから、その通知を利用します。

```
public Doc()
{
    this.PropertyChanged += delegate(object sender, PropertyChangedEventArgs e)
    {
        if (e.PropertyName == "text1" || e.PropertyName == "text2")
        {
            this.FirePropertyChanged("ButtonEnabled");
        }
    };
}
```

実行  
成功

## おまけの Demo5

```
<Button Grid.Row="4" Click="Add" >  
  <StackPanel>  
    <Image Source="{Binding Path=Text,ElementName=パス}" Width="40"/>  
    <TextBox x:Name="パス" Width="200"/>  
  </StackPanel>  
</Button>
```

Button の Content に 1 つのコントロールを配置できます。  
複数置きたければ Panel (枠) を置けば、その中に複数のコントロールを置くことができます。

Image とテキストボックスを配置します。

X:Name でコントロールに名前をつけます。

Image の Source にこのパスというコントロールの Text プロパティの中身とバインドするように設定します。

さて実行

