



How To WPF
アプリケーション
Part2
By 中博俊



おさらい

- 前回はButtonとTextBoxを使った簡単なアプリケーションの作り方でした。
- INotifyPropertyChangedのインターフェイスを究めようという内容でした。
- NotifyPropertyChangedBaseは今回も出てきますが説明しないのでおさらい

NotifyPropertyChangedBase

```
public class NotifyPropertyChangedBase :  
    INotifyPropertyChanged {  
    public event PropertyChangedEventHandler  
        PropertyChanged;  
    protected void FirePropertyChanged(  
        string PropertyName) {  
        if (this.PropertyChanged != null) {  
            this.PropertyChanged(this,  
                new PropertyChangedEventArgs(PropertyName));  
        }  
    }  
}
```



今回のアジェンダ

- コンバータ コンバータ コンバータ
– 第2の肝であるコンバータをマスター
- ラジオボタン
– よく使うコントロールも覚えていきましょう



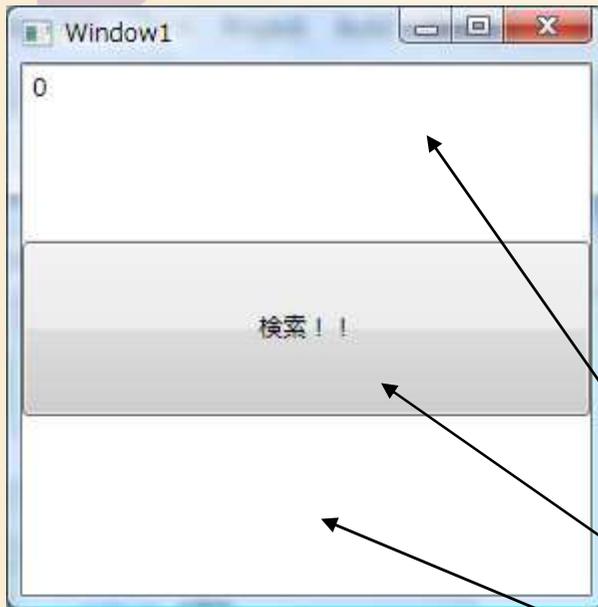
DEMO 1



わんくま同盟 名古屋勉強会 #2

とりあえず画面作っちゃいませよ

- 画面



```
<Window x:Class="WpfApplication1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:WpfApplication1="clr-namespace:WpfApplication1;assembly="
Title="Window1" Height="300" Width="300">
<Window.DataContext>
<WpfApplication1:DocumentA />
</Window.DataContext>

<Grid>
<Grid.RowDefinitions>
<RowDefinition />
<RowDefinition />
<RowDefinition />
</Grid.RowDefinitions>
<TextBox Text="{Binding Path=検索値}" />
<Button Content="検索!!" Click="検索" Grid.Row="1" />
<Label Content="{Binding Path=検索結果}" Grid.Row="2" />
</Grid>
</Window>
```



とりあえず画面作っちゃいませよ

- ドキュメントクラス



とりあえず画面作っちゃいませよ

- 検索部分

```
public void 検索()  
{  
    var rows = new Row[] {  
        new Row(){名前 = "えムナウ",年齢 = 18},  
        new Row(){名前 = "R田中",年齢 = 21},  
        new Row(){名前 = "中博俊",年齢 = 31}};  
  
    var 結果 =  
        (from x in rows  
         where x.年齢 > this.検索値  
         select x).FirstOrDefault();  
  
    this.検索結果 = string.Format("{0}様 {1}歳", 結果.名前,結果.年齢);  
}
```

※従来型の問題点
最終的に表示するための文字列に、処理で記述しなければならない。

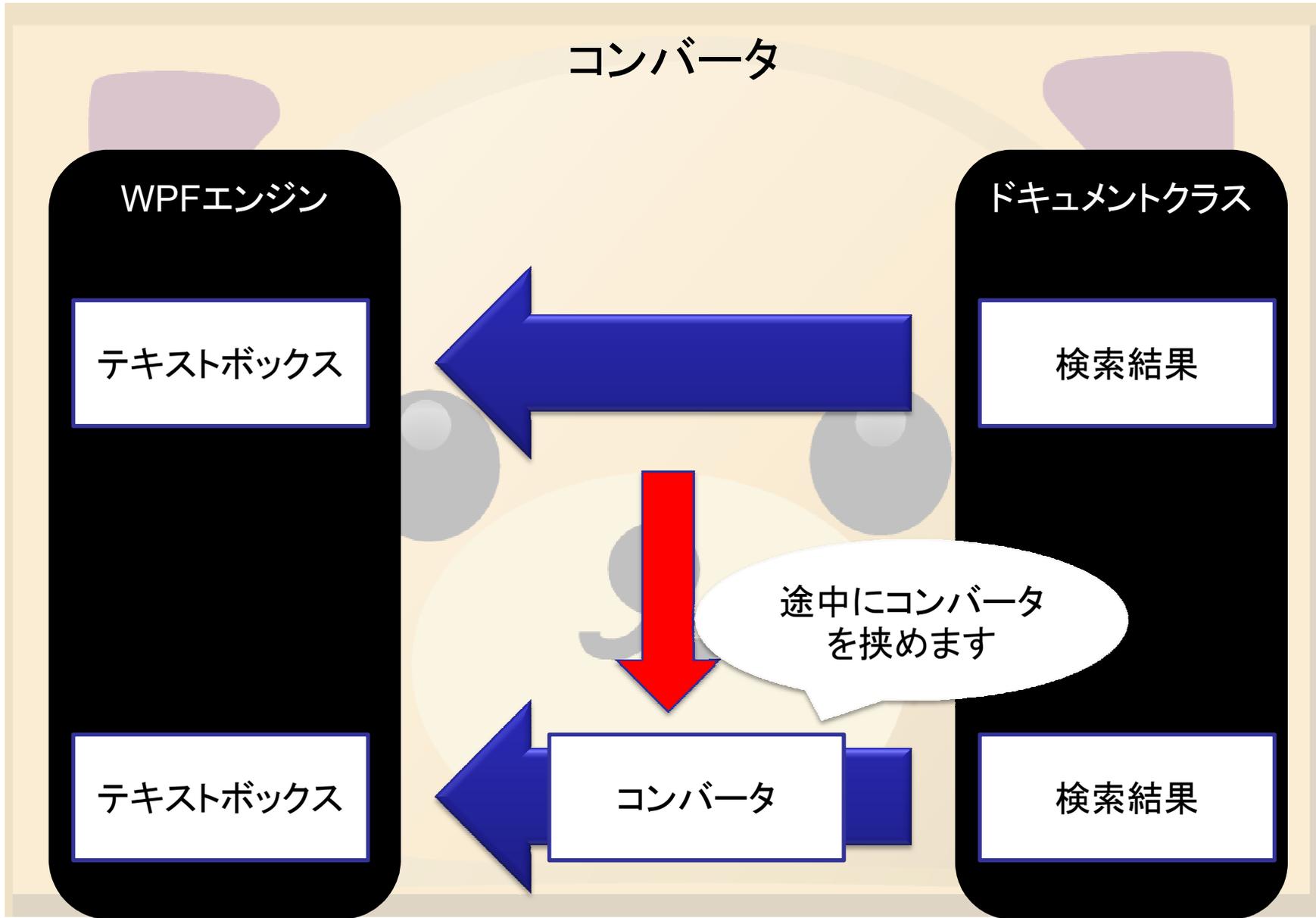


従来型の問題点

- XAML側
- `<TextBox Text="{Binding Path=検索値}"/>`
- Document側
- `public string 検索結果 { get { return _検索結果; } set { _検索結果 = value; this.FirePropertyChanged("検索結果"); } }`
- ただのStringになっちゃってますよね？
- この検索結果Rowを使って再度処理をしたければどうしましょう。
- 別途保存する？
- `public Row 検索結果Row;`



コンバータ



拍手の用意はいいですか？

DEMO2



わんくま同盟 名古屋勉強会 #2

コンバータのポイント

- **IValueConverter**を実装する
 - ほかにIMultiValueConverterというのものもある。
- 値を設定できない場合(null等)には
DependencyProperty.UnsetValueを返す。
- 気楽に作るといっぱいコンバータを作ってしまうがちなので注意
- オブジェクトをオブジェクトのまま利用できるようになるので、積極的に活用しましょう。

オブジェクトのオブジェクトをバインドするには

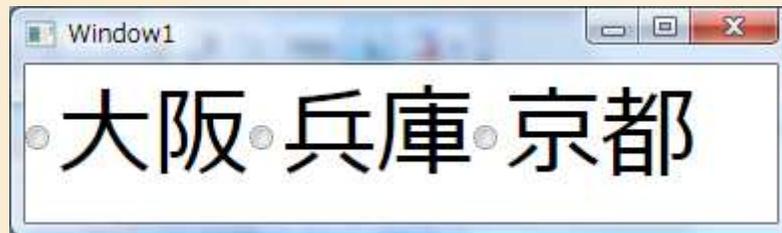
- 単純にオブジェクトのオブジェクトをバインドするには
- <Label Content="{Binding Path=検索結果.年齢}"/>
- <Label Content="{Binding Path=AAA[0].年齢}"/>
- これだけで実現できます。
- ただし検索結果がNullであったり、Listの0番目が存在しない場合などには例外が出たりするので要注意

DEMO2-2

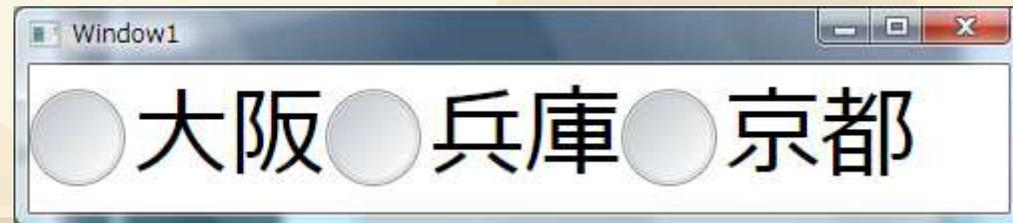


ラジオボタン

- ラジオボタンとは
- 複数の選択肢のうち1つの選択を強制させる



- ボタンが見にくいのでごちゃごちゃしてあります。



ラジオボタン

- それでは標準の動きを見てみましょう

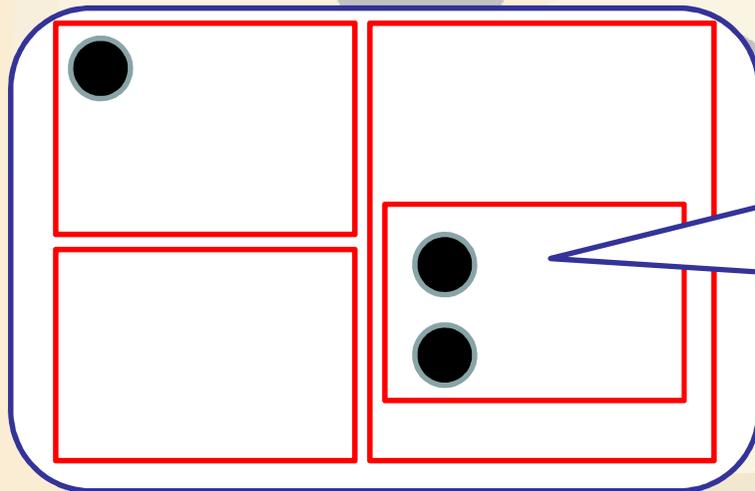
DEMO3



わんくま同盟 名古屋勉強会 #2

どうしてこんなことになるの？

- デザインと意味を分離するためです。
- 従来のWindowsFormsの場合にはパネルでくる必要がありました。
- それによりデザインの制約が発生します。



こんなUserControlの配置でラジオボタンを共通化させることができるようにした結果だと思われます。

じゃあバイディングで解決しましょうよ

DEMO4



わんくま同盟 名古屋勉強会 #2

やったことのポイント

- すべてのラジオボタンをグループ化しない
- 直接のバインディングはEnumを利用する
- コンバータでEnum \leftrightarrow Boolean変換を行う
 - このEnumBooleanConverterは汎用的に利用できる
- UserControlにはUserControl独自のドキュメントクラスを作成するとよい
- UserControlのDataContextは親コントロールから設定できる

まとめ

- コンバータはいろいろなところで役に立ちます
- このほかにもよく使うコンバータは汎用的に作成可能ですので、いろいろ用意しておくといいでしょう。
- ラジオボタンは結構コツがいる
 - 次回以降
 - コンボボックス、チェックボックス、リストビュー
 - まだまだ続きそうです

Enjoy WPF !!

