

えムナウのC#
プログラミングのページ



WPF

Bindingの威力

えムナウ(児玉宏之)

Microsoft MVP for Visual-
Developer C# 2005/01-2007/12



わんくま同盟 大阪勉強会 #14

アジェンダ

- はじめに
- Bindingの概要
- データソース
- データ変換・データ検証

はじめに

- Windows Presentation Foundation (WPF) データ バインディングは、アプリケーションがデータを提供し、柔軟な UI 表現、ビジネスロジックと UI の明確な分離を実現します。
- データフローの方向やソースの更新の要因を選択し表示のためのデータ変換や格納時のデータの検証を行えます。
- コレクションへのバインドで並べ替え、フィルタ処理、グループ化を行えます。

Bindingの概要

- Bindingオブジェクト

```
<Window.Resources>
```

```
  <src:Person x:Key="myDataSource" PersonName="えムナウ"/>
```

```
</Window.Resources>
```

```
<TextBlock Text="{Binding Source={StaticResource  
  myDataSource}, Path=PersonName}"/>
```

TextBlock

MyDataSource
(Personクラス)

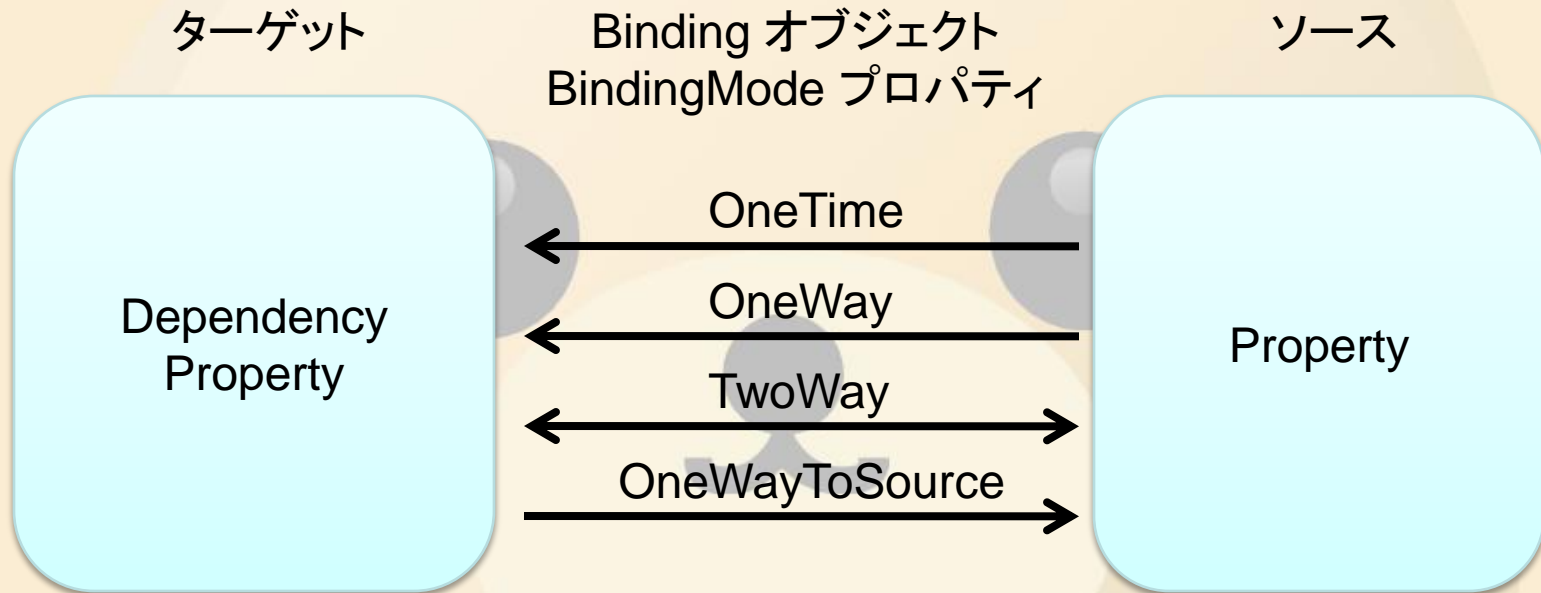
Text
プロパティ

Binding オブジェクト

PersonName
プロパティ

Bindingの概要

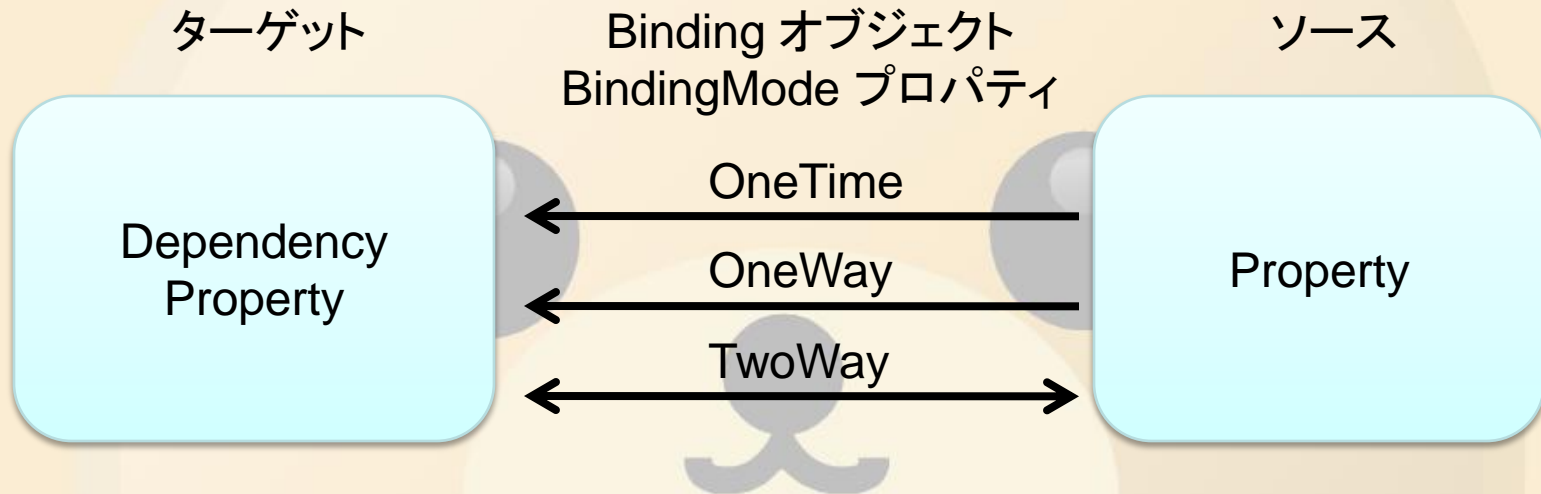
- データフローの方向



BindingModeでDefault はテキスト ボックスやチェック ボックスなど編集可能な場合はTwoWay、それ以外のほとんどのプロパティはOneWay。

Bindingの概要

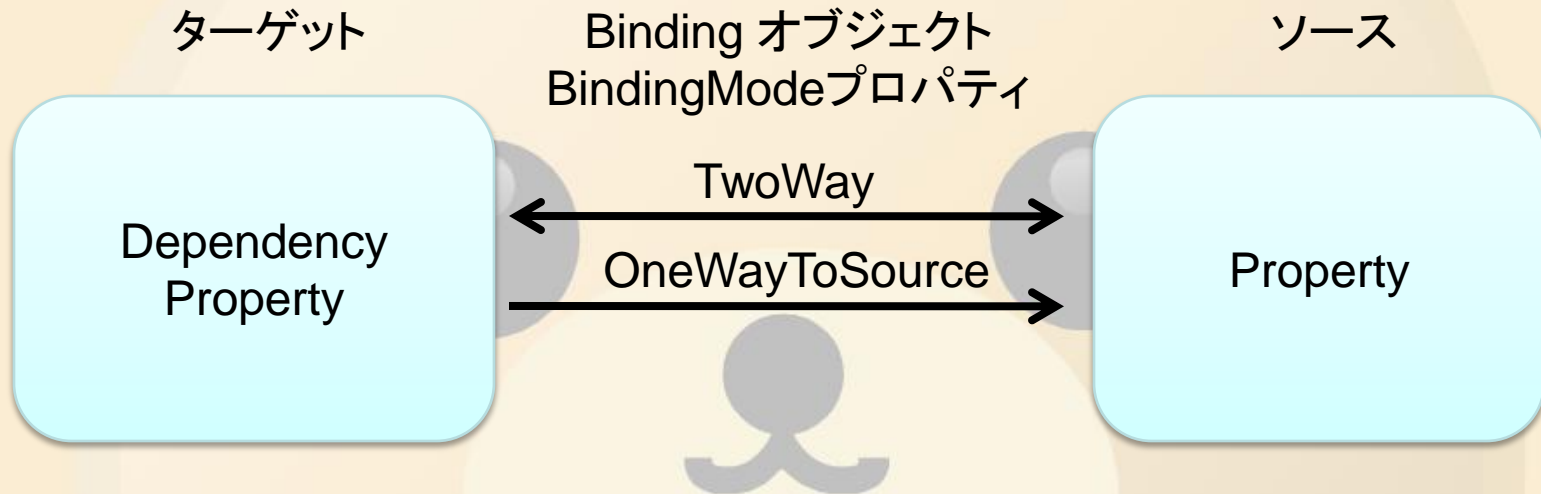
- ターゲット更新要因



OneTimeはアプリケーションの起動時またはデータ コンテキストの変更時
OneWay・ TwoWayはプロパティ変更時

Bindingの概要

- ソースの更新要因



Binding.UpdateSourceTrigger プロパティ
PropertyChanged の場合ターゲットプロパティ変更時、
LostFocus の場合ターゲットがフォーカスを失った時、
Explicit の場合アプリケーションが UpdateSource を呼び出した時。

Bindingの概要

- **バインディング ソース指定方法の種類**
 - Binding.Sourceプロパティ
ソースを直接指定します
 - Binding.RelativeSourceプロパティ
ターゲットの位置を基準にしてソースを指定します
 - Binding.ElementNameプロパティ
別の要素のプロパティをソースに指定します
 - 自分や親要素のDataContextプロパティ
複数のプロパティを共通のソースに指定します

Bindingの概要

• DataContext

顧客ListView.DataContext
(親要素)

代入

顧客クラス
インスタンス

TextBlock.Text
プロパティ
TextBlock.Text
プロパティ
TextBlock.Text
プロパティ
TextBlock.Text
プロパティ
TextBlock.Text
プロパティ
TextBlock.Text
プロパティ
TextBlock.Text
プロパティ

Binding オブジェクト

Binding オブジェクト

Binding オブジェクト

Binding オブジェクト

Binding オブジェクト

Binding オブジェクト

ID
プロパティ
ふりがな
プロパティ
氏名
プロパティ
誕生日
プロパティ
都道府県ID
プロパティ
郵便番号
プロパティ

Bindingの概要

- **MultiBinding**オブジェクト

複数のBindingオブジェクトから渡るデータを元に、Converterでターゲットに渡す値を決定します。

- **PriorityBinding**オブジェクト

複数のBindingオブジェクトをリスト内の順番にたどって最初の有効なBindingオブジェクトを使用します。

DependencyProperty.UnsetValue を返す場合が無効値です。

Bindingの概要

DEMO

データソース

- Expression Blendのデータソース

- XML データ ソース

- XML 形式のデータを提供できるローカルまたはリモートの XML ファイル

- 共通言語ランタイム (CLR) オブジェクト データ ソース

- ObservableCollection クラスが提供
- IEnumerable インターフェイスと INotifyCollectionChanged インターフェイスを実装したクラス

データソース

- WPFのバインディングソース
 - Expression Blendのデータソースの2つ
 - ADO.NET データ
 - ADO.NET DataView は、IBindingList を実装し、バインディング エンジンがリスンする変更通知を提供
 - DependencyObject
 - 任意の DependencyObject の依存関係プロパティにバインド、コントロールのプロパティ間のバインディングに主に使用
 - Blend ではプロパティウィンドウからデータバインドで設定可能

データソース

- Expression BlendのDataSetの対応
 - 型付きのDataSetでもBlendではデータソースとして認識されない
 - ObservableCollection<DataSet1.顧客Row> とかを作成すれば認識される
 - しかし、VisualStudioでプログラムからDataContext に設定すれば利用できる

データソース

- GridView

- ListView の表示モードを指定する View プロパティに GridView オブジェクトを設定することで、DataGridView に近い表現と入力が可能
- GridViewColumn でカラムを指定する
 - CellTemplate や HeaderTemplate でヘッダーやセルの表示形式を変更
 - HeaderComponentStyle でヘッダーの表示形式を変更

データソース

- DataTemplate

- データ オブジェクトの視覚化を表現するのに作成する
- DataGridView ではできなかった2段表示も可能


```
<DataTemplate x:Key="氏名HeaderCell">  
  <StackPanel>  
    <TextBlock Text="ふりがな" HorizontalAlignment="Center"/>  
    <TextBlock Text="氏名" HorizontalAlignment="Center"/>  
  </StackPanel>  
</DataTemplate>
```


データソース

- ControlTemplate

- コントロールのビジュアル構造を標準から変更するときを使用
- 少し拡張する場合にも

```
<ControlTemplate x:Key="validationTemplate">  
  <DockPanel>  
    <AdornedElementPlaceholder/>  
    <Image Source="Images¥error.ico" Width="20" Height="20"/>  
  </DockPanel>  
</ControlTemplate>
```



データソース DEMO

データ変換・データ検証

- データ変換
 - 今までは WindowsForms では Format プロパティを指定することでデータ変換を処理
 - WPFはデータ変換用のクラスを準備
- データ変換パラメータ
 - Binding.ConverterParameter でパラメータを指定
 - データ変換時にパラメータを指定できることでのいろいろな応用が可能になる

データ変換・データ検証

- データ変換の実装

- IValueConverter インターフェースを実装して、
Convert(ソースからターゲットに変換)・
ConvertBack(ターゲットからソースに変換)のメソッドを作成

```
public object Convert(object value, Type targetType,  
object parameter, System.Globalization.CultureInfo  
culture)
```

```
public object ConvertBack(object value, Type  
targetType, object parameter,  
System.Globalization.CultureInfo culture)
```

データ変換・データ検証

- クイズです。何のコンバーターでしょうか？

```
public object Convert(object value, Type targetType, object parameter,
    System.Globalization.CultureInfo culture)
{
    string valuetext = ((int)value).ToString();
    if (valuetext == (string)parameter) return true;
    return false;
}
public object ConvertBack(object value, Type targetType, object parameter,
    System.Globalization.CultureInfo culture)
{
    if ((bool)value)
    {
        int selectvalue;
        int.TryParse((string)parameter, out selectvalue);
        return selectvalue;
    }
    return DependencyProperty.UnsetValue;
}
```

データ変換・データ検証

- データ検証

- Binding.ValidationRules コレクションの中に、ValidationRule クラスから派生させたデータ検証用のクラスを作成
- データ検証用のクラスで Validate メソッドをオーバーライド

```
public override ValidationResult Validate(object value, CultureInfo cultureInfo)
```

データ変換・データ検証

- 検証結果の表示

- TextBox.Validation.ErrorTemplate に ControlTemplate を割り当てて、エラーがあったときだけエラー表示を拡張
- Validation.HasError や Validation.Errors も利用可能
- Validation.ErrorTemplate などは添付プロパティなので TextBox の HELP にのっていないので注意が必要

データ変換・データ検証

DEMO

