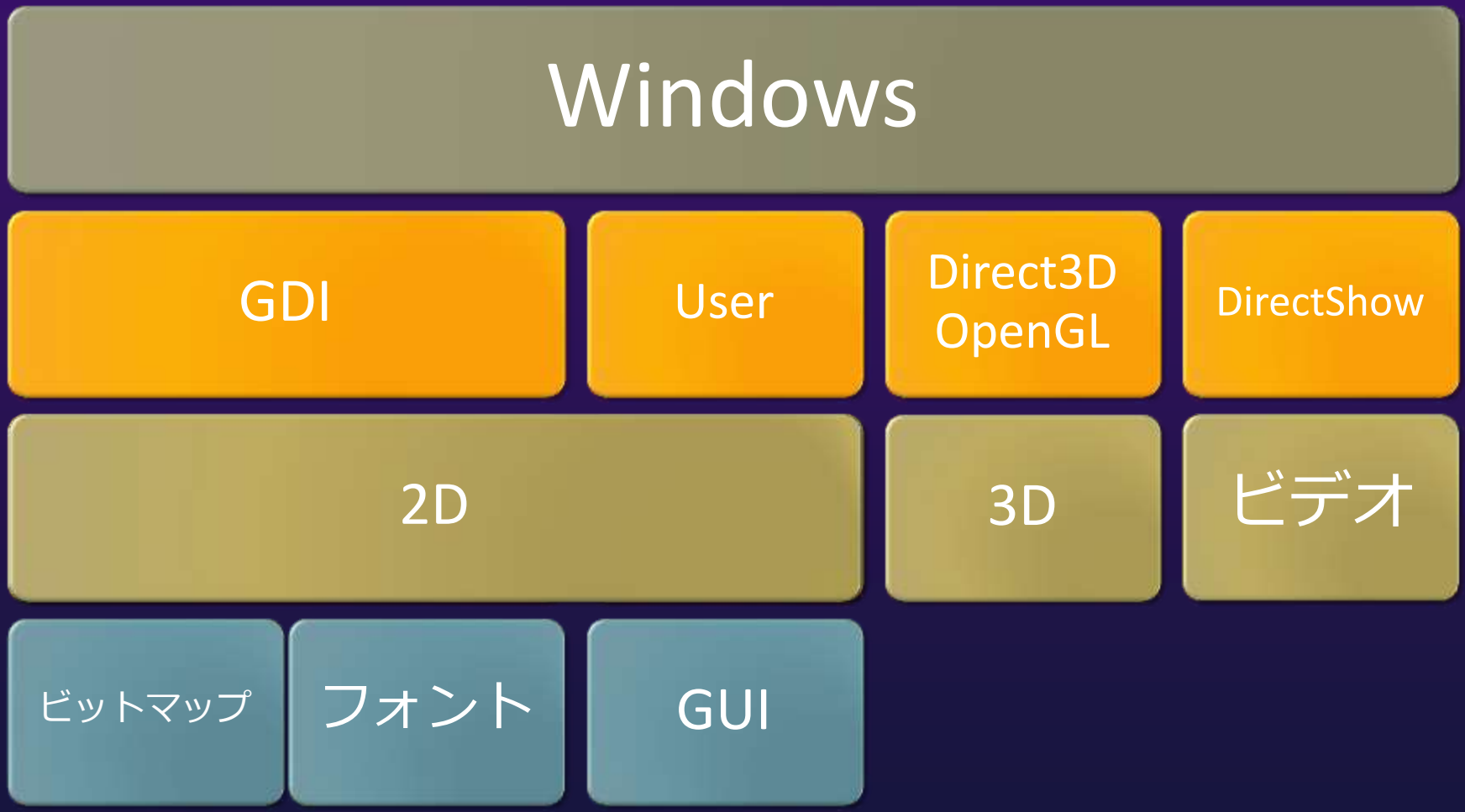
やっぱり新しいものはいいのだ♪ WPF／Silverlightで変わるUI開発の世界

グレープシティ株式会社
八巻 雄哉

WPFとは

- .NET Framework 3.0で追加されたWindowsアプリケーションのためのGUIサブシステム

WPF登場の背景


Windows
GDI
User
Direct3D
OpenGL
DirectShow
2D
3D
ビデオ
ビットマップ
フォント
GUI

WPF登場の背景

本日のメインは業務アプリ

- うちの業務アプリでは「3D」や「動画」は使わないのですが・・・。
- 大丈夫です。どこの業務アプリでも「3D」や「動画」は使いません。

初めての刷新

- WPF UIフレームワークは、全く新しいマイクロソフトの最新UI基盤
- 20年来のWin16/32ベースのGUIから大きく進化
- 業務アプリでも大きな恩恵
  - 画面（GUIコントロール、2D描画）
  - アプリケーション開発

Silverlightとは

- WPFのクロスプラットフォーム実装
  - 旧称：WPF/E(Everywhere)
- 実装はサブセット（限定部分）
  - Silverlightにしかない機能もある
- Webブラウザプラグインとして動作

ご注意

- 用語「WPF UIフレームワーク」は、WPFとSilverlightで共通仕様となっているUI基盤を指します。
- 本セッションでご紹介するすべての内容は、WPFとSilverlightの両方で同様に動作します。

WPF／Silverlight UIフレームワーク入門 － @IT

http://www.atmarkit.co.jp/fdotnet/vblab/uiframework_index/

- **第1回 WPFとSilverlightをまとめて習得しよう**
  - 1．XAML構文の基礎
  - 2．Panelによるレイアウト（StackPanel／Canvas）
  - 3．Panelによるレイアウト（Grid／そのほかのPanel）
  - 4．コントロールの種類
- **第2回 データの表示と入力に必要な知識**
  - 1．データ・バインディングの基本
  - 2．より実践的なデータ・バインディング
  - 3．双方向データ・バインディング
  - 4．コンバータによるデータ変換／コレクション・オブジェクトへのバインド
  - 5．表示をカスタマイズできるデータ・テンプレート
- **第3回　“見た目”を決めるリソースとスタイル**
  - 1．リソースの概要／静的リソース参照
  - 2．動的リソース参照／リソース・ディクショナリ・ファイル
  - 3．スタイルの基礎／スタイルの共有と継承
- **第4回　“見た目”を決めるコントロール・テンプレート**
  - 1．コントロール・テンプレートの概要
  - 2．コントロール・テンプレートによる外観の定義方法
  - 3．コントロール・テンプレート内で動的に外観を定義する方法

本日のデモ環境

- Visual Studio 2010 Beta 2
- .NET Framework 4 Beta 2
  - WPF 4.0 Beta 2
- Silverlight 4 Beta

アジェンダ

- XAMLの基本
- レイアウトとスケーリング
- コントロールの種類
- データバインディング
  - コンバーター
  - データテンプレート

XAML Fundamentals

XAMLの基本

XAMLとは

- XMLをベースに
  独自の拡張を施した宣言型言語
- WPF、SilverlightのUI作成に使用
- メリット
  - 可読性
    - 人（デザイナー）
    - ツール（デザイナー）

コードとXAMLの比較

```
Dim Canvas1 As New Canvas()
Dim Button1 As New Button()
Button1.Content = "ボタン“
Button1.FontSize = 24
Canvas.SetTop(Button1, 50)
Canvas.SetLeft(Button1, 50)
Canvas1.Children.Add(Button1)
```

可読性に注目！

```
<Canvas>
  <Button Content="ボタン" Canvas.Top="50" Canvas.Left="50" FontSize="24"/>
</Canvas>
```

押さえておきたいポイント

- XAMLは.NET Frameworkにおける
  クラスのインスタンス化を
  宣言的に記述したもの
- XAMLの親子関係は、
  そのままUIの親子関係となる

Tech·Ed 2008 Yokohama
「開発者なら知っておきたい XAML の書き方」セッション資料
http://d.hatena.ne.jp/Yamaki/20080901/1220243792

Layout and Scaling

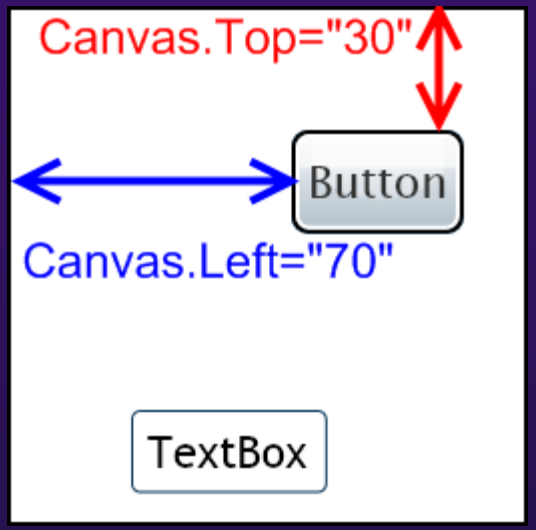
レイアウトとスケーリング

レイアウトの種類

		Windowsフォーム	WPF UIフレームワーク
絶対配置	座標指定	デフォルト （パネルなし）	Canvas
相対配置	水平方向または 垂直方向への整列	FlowLayoutPanel	StackPanel
	上下左右の端に ドッキングして整列	Dockプロパティ （パネルなし）	DockPanel
	列と行で構成される 格子状の領域へ配置	TableLayoutPanel	Grid UniformGrid

Canvas

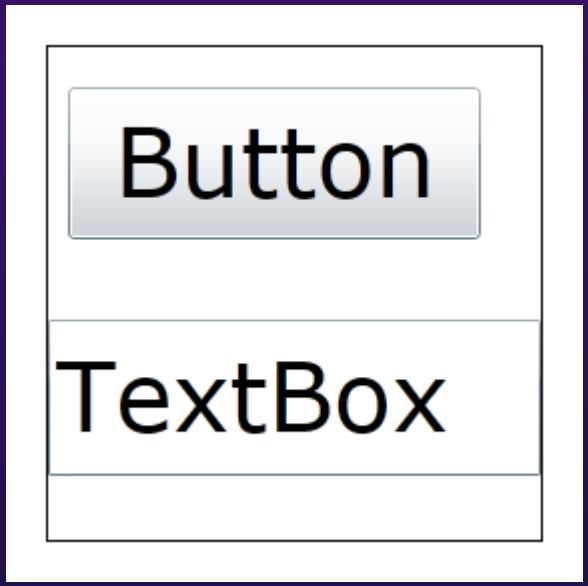
- 添付プロパティ
  - Left
    - 要素の左端と
      親Canvasの左端との距離
  - Top
    - 要素の上端と
      親Canvasの上端との距離




```
<Canvas Background="White">
   <Button Content="Button" Canvas.Top="30" Canvas.Left="70"/>
   <TextBox Text="TextBox" Canvas.Top="100" Canvas.Left="30"/>
</Canvas>
```

StackPanel

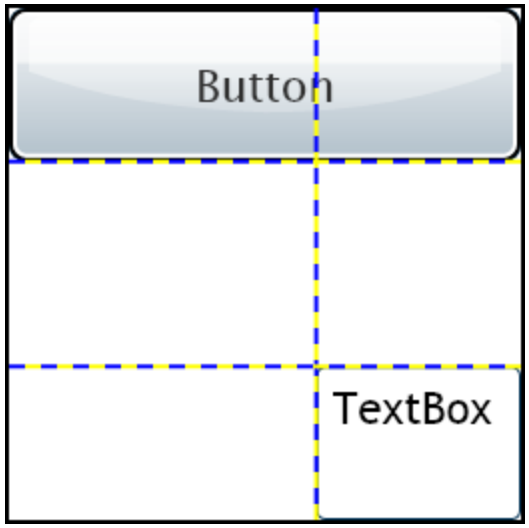
- Orientationプロパティ
  - 子要素が配置される方向（水平か垂直）




```
<StackPanel Orientation="Vertical">
   <Button Content="Button" Margin="10,20,30,40"/>
   <TextBox Text="TextBox" Margin="0"/>
</StackPanel>
```

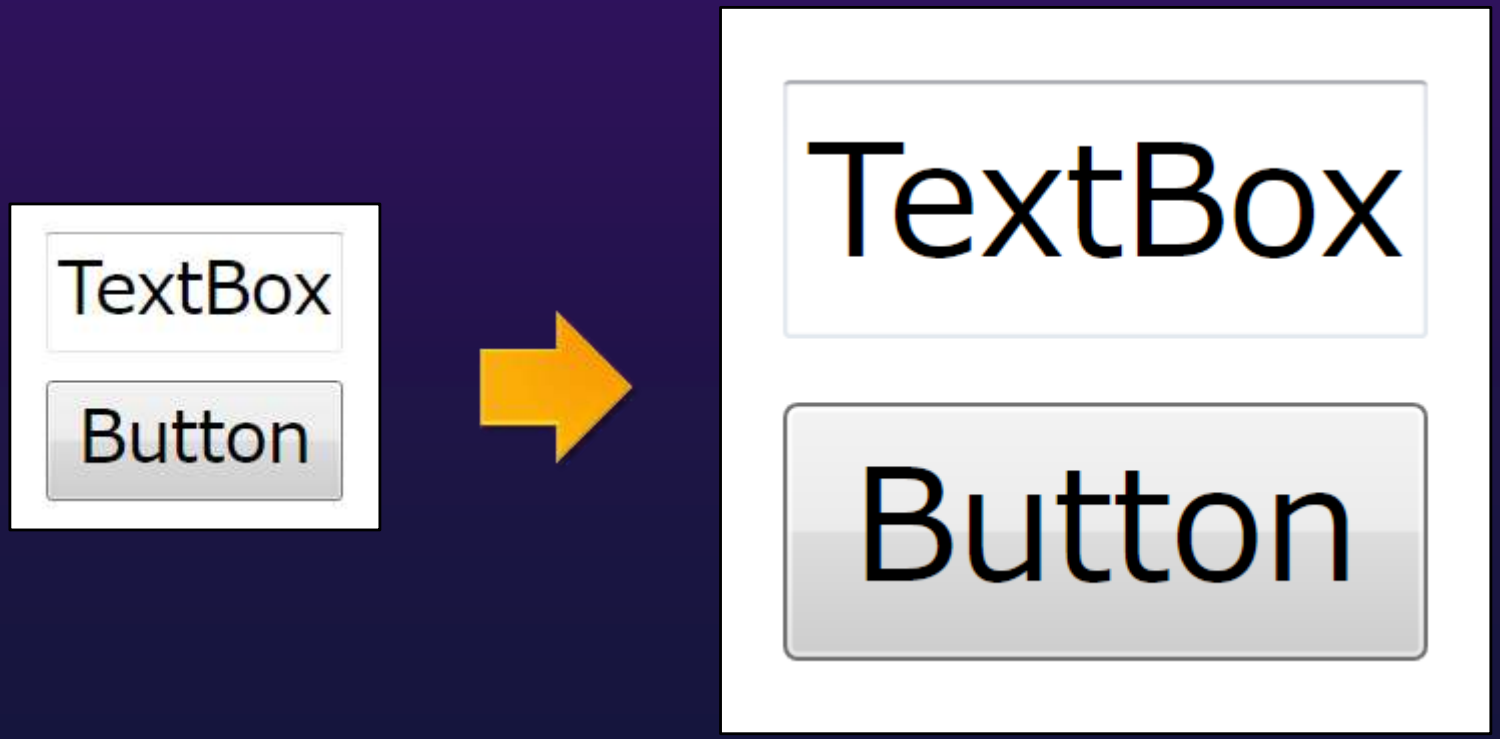
Grid

```
<Grid Background="White" ShowGridLines="True">
   <Grid.ColumnDefinitions>
      <ColumnDefinition Width="6*"/>
      <ColumnDefinition Width="4*"/>
   </Grid.ColumnDefinitions>
   <Grid.RowDefinitions>
      <RowDefinition Height="3*"/>
      <RowDefinition Height="4*"/>
      <RowDefinition Height="3*"/>
   </Grid.RowDefinitions>
   <Button Content="Button" Grid.ColumnSpan="2"/>
   <TextBox Text="TextBox" Grid.Column="1" Grid.Row="2"/>
</Grid>
```



Viewbox

- Viewboxコントロールの大きさに合わせて子要素が拡縮される



Control Types

コントロールの種類

コントロール

- Calender
- ContentControl
  - ButtonBase
    - Button
    - RepeatButton
    - ToggleButton
      - Checkbox
      - RadioButton
  - ComboBoxItem
  - Label
  - ListBoxItem
  - ScrollViewer
  - TabItem
- DataGrid
- DatePicker

- ItemsControl
  - Selector
    - ComboBox
    - ListBox
    - TabControl*1
  - TreeView
- PasswordBox
- RangeBase
  - ProgressBar
  - ScrollBar
  - Slider
- Thumb
  - GridSplitter*2
- UserControl
- TextBox

*1 SilverlightではItemsControlの派生クラス
*2 SilverlightではControlの派生クラス

従来の親子関係

- Windowsフォーム
  - Form、FlowLayoutPanel、GroupBox、Panel、SplitContainer、TabControl、TableLayoutPanelなど
- ASP.NET（Webフォーム）
  - FormView、 ListView、Page、Panel、PlaceHolder、Repeater、UpdatePanel、UpdateProgressなど
- コンテナ系のコントロールだけが
  子コントロールを持つ

WPF UIフレームワークの親子関係

- Buttonコントロールなども
  子要素を持つ

要素の種類	子要素の種類	子要素の数
	該当する代表的な要素	
Panel	UIElementオブジェクト	複数
	Canvas、Grid、StackPanelなど	
ContentControl	文字列、オブジェクト	単一
	Button、CheckBox、RadioButtonなど	
ItemsControl	文字列、オブジェクト	複数
	ListBox、ComboBox、TabControlなど	

ContentControl

- アイコン付ボタンの例


```
<Button>
 <StackPanel Orientation="Horizontal">
  <Image Source="Help.gif"/>
  <TextBlock Text="ヘルプ" VerticalAlignment="Center"/>
 </StackPanel>
</Button>
```

Data Binding

データバインディング

データバインディングの進化

- 基本思想はWindowsフォームと同様

- 柔軟性や機能面で大幅向上

- データバインディングを広く活用可

誤解

- UIとロジックの分離

XAML

コードビハインド

MainWindow.xaml

MainWindow.xaml.vb

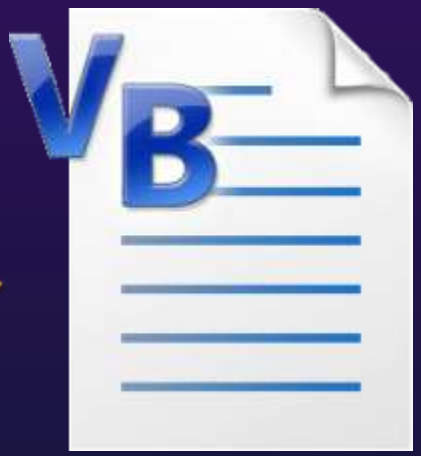
データバインディングの目的

- UIとロジックの分離

XAML +
コードビハインド

ロジック

データ
バインディング



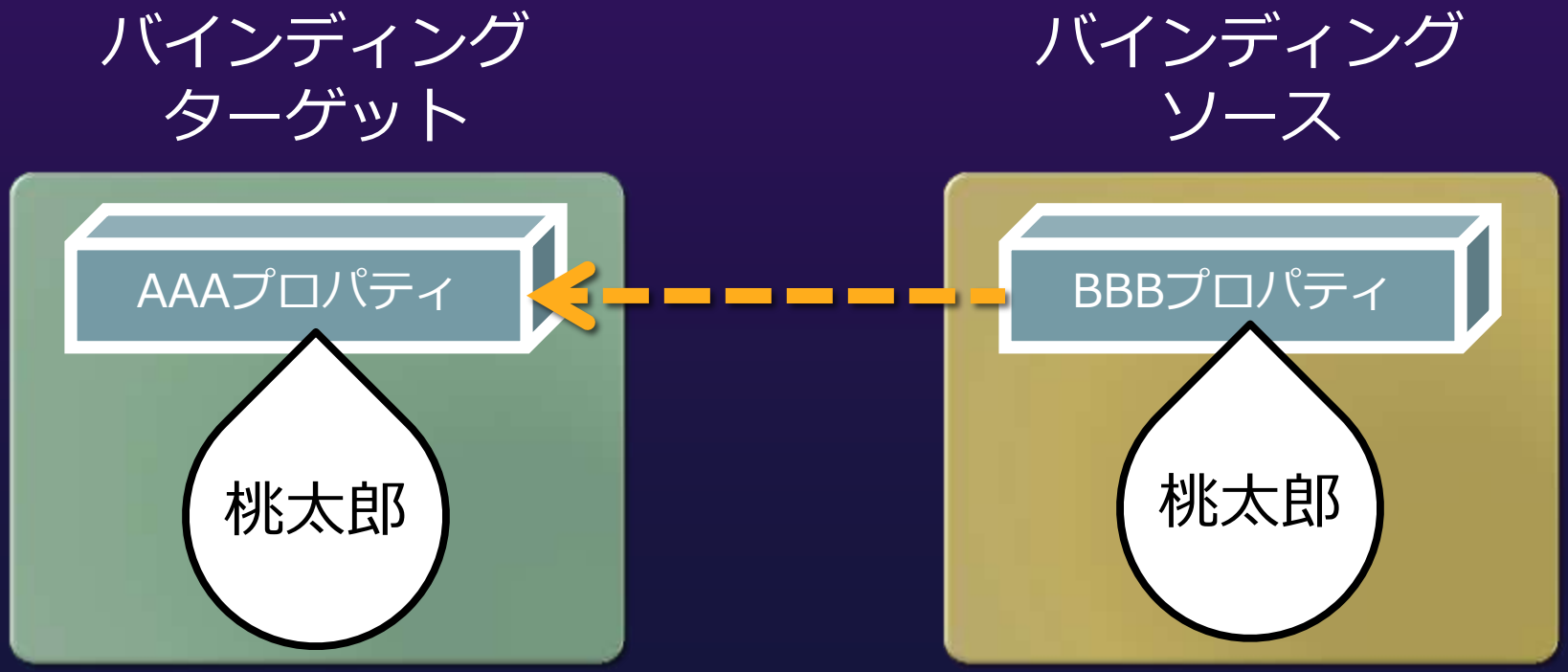
MainWindow.xaml.vb

MainWindow.xaml

Logic.vb

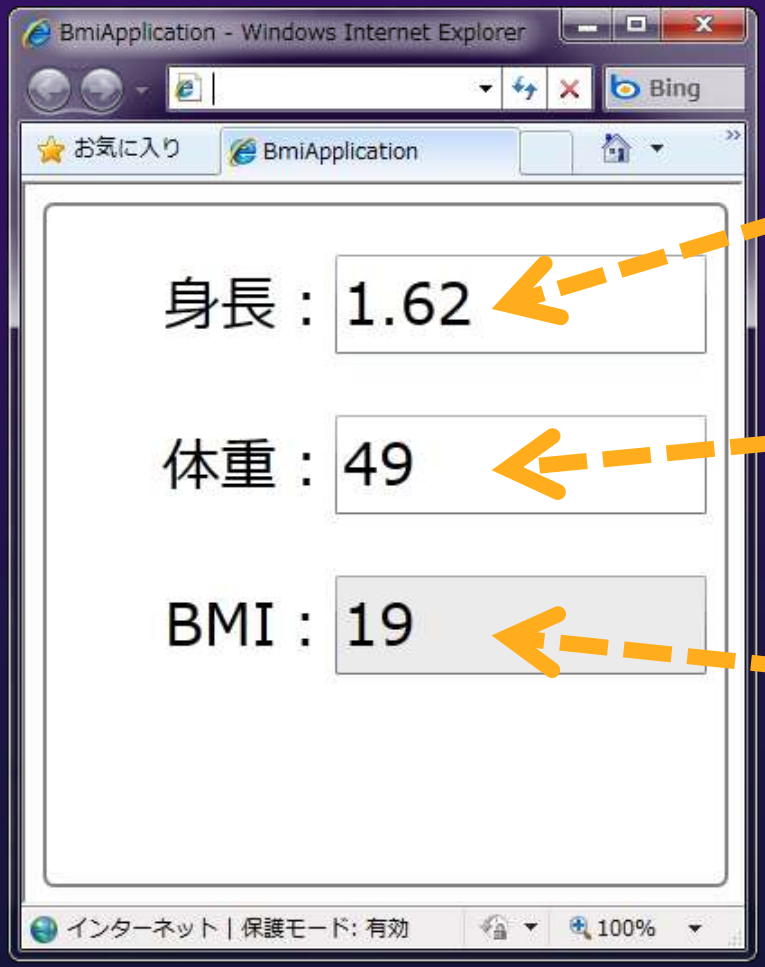
データバインディング

- ２つのオブジェクトのプロパティに対する単方向、または双方向の値の同期処理



BMI測定アプリの例

UI（ビュー）

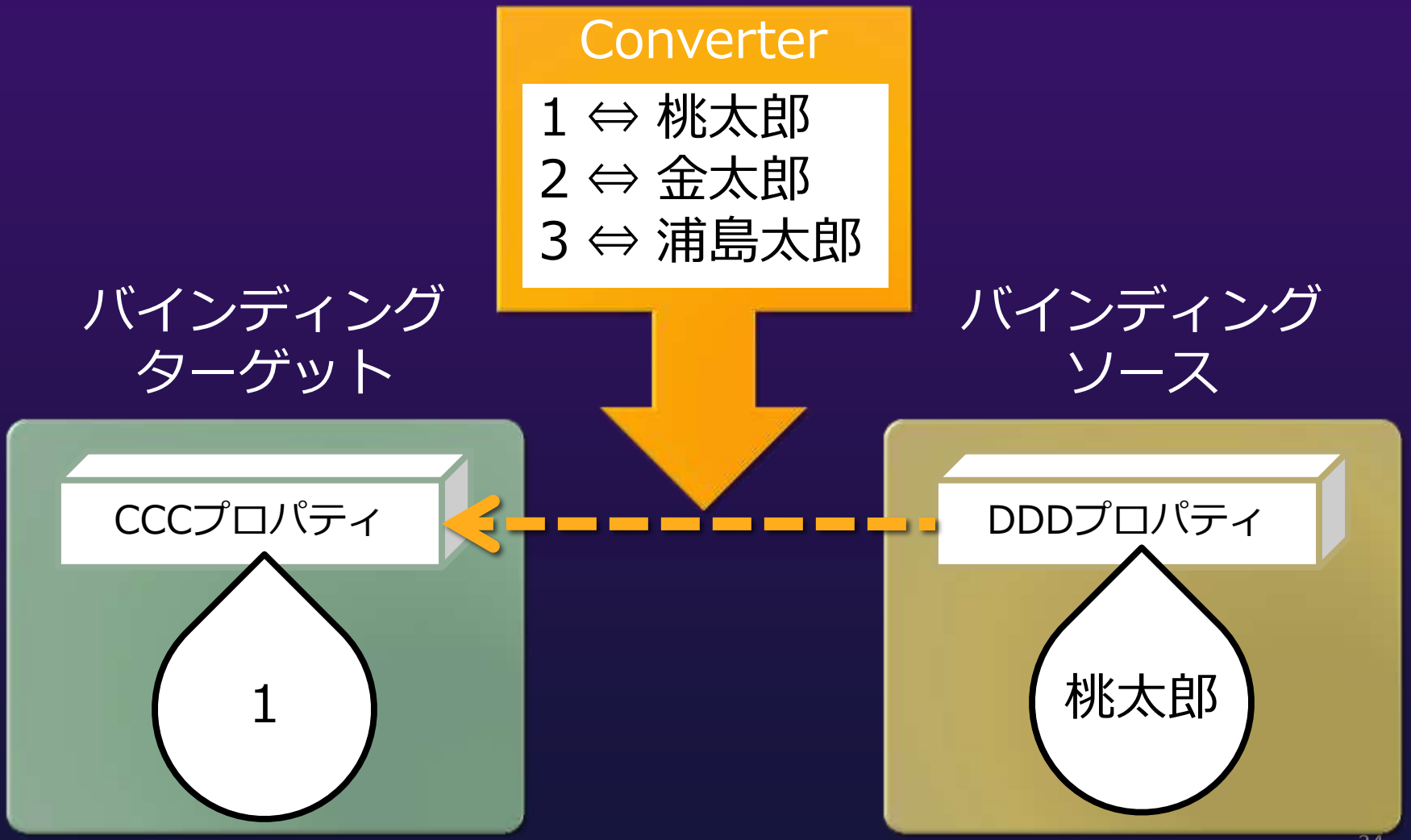


Personオブジェクト

コンバーター

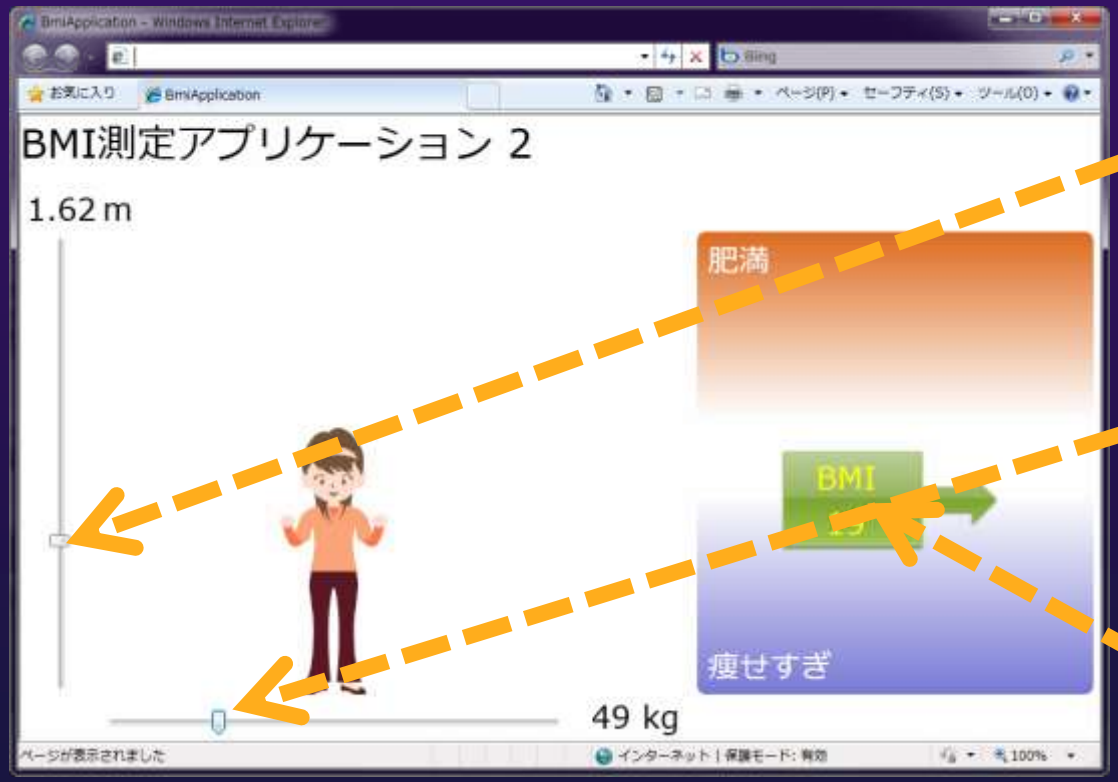
- コンバーターを使用することで、データ変換を伴った値の同期処理が可能となる
- IValueConverterインターフェイス
  - Convertメソッド
    - ソースからターゲット方向への値を変換
  - ConvertBackメソッド
    - ターゲットからソース方向への値を変換

コンバーター


Converter
1 ⇔ 桃太郎
2 ⇔ 金太郎
3 ⇔ 浦島太郎
バインディング
ターゲット
バインディング
ソース
CCCプロパティ
DDDプロパティ
1
桃太郎

BMI測定アプリの例

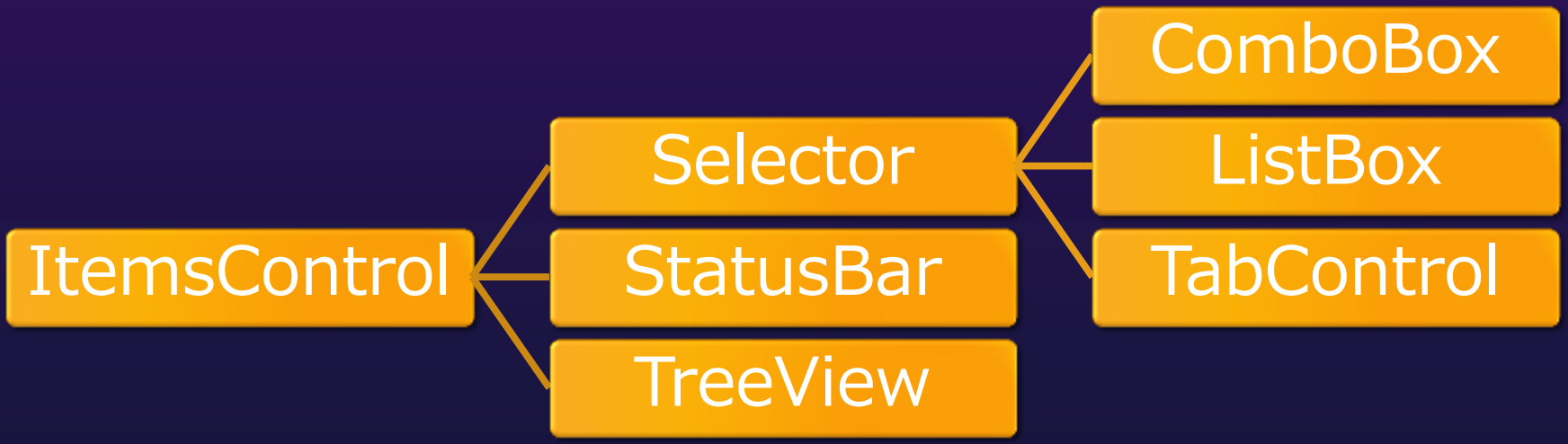
UI（ビュー）



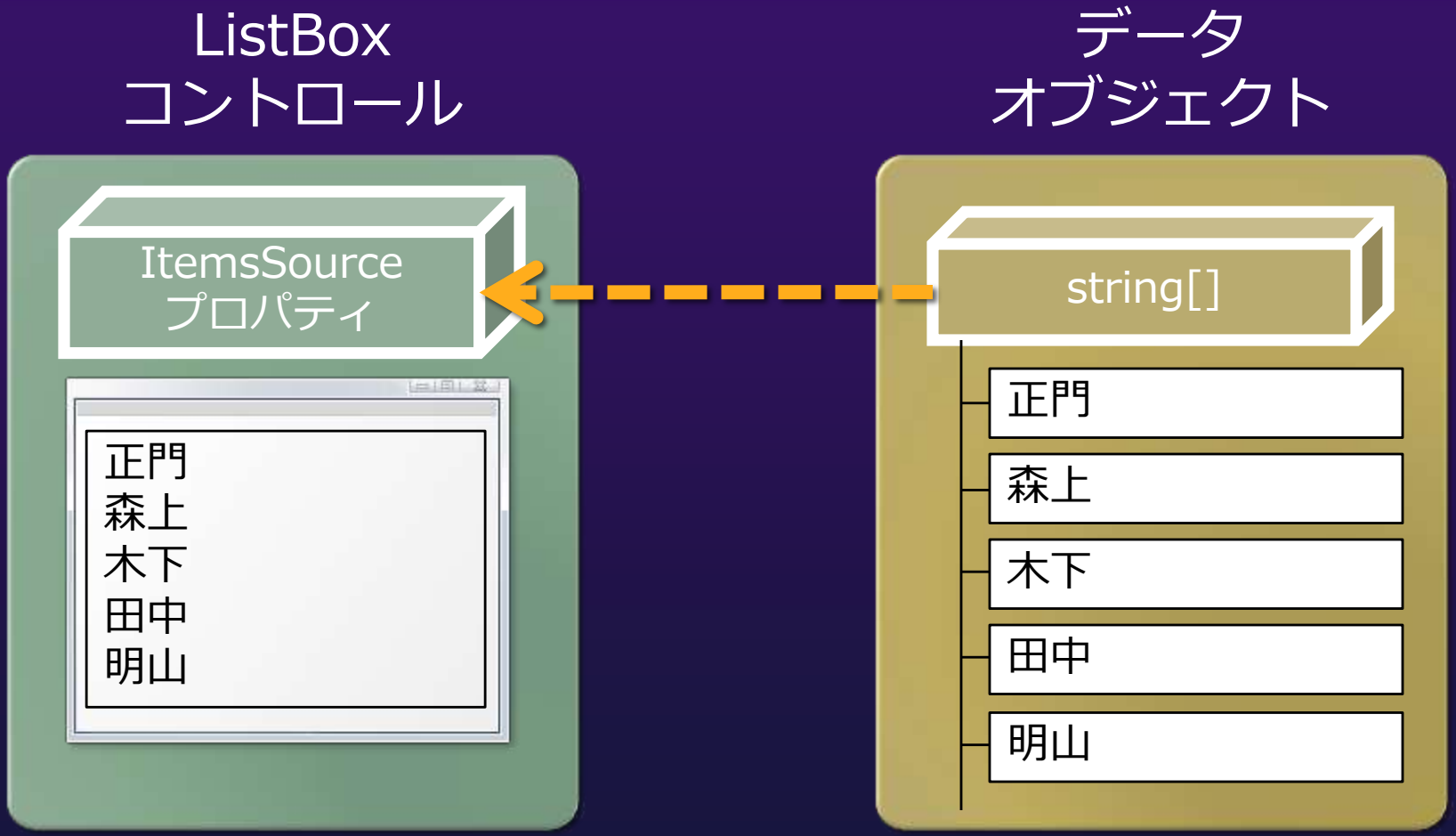
Personオブジェクト

コレクションデータとの連結

コレクションデータとバインディングし、
視覚化する役割を担うのが
ItemsControl



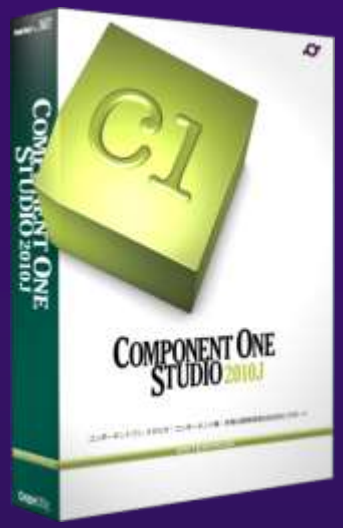
コレクションデータの例



２つのテンプレート

- ItemTemplateプロパティ（DataTemplate型）
  - 1項目分のデータをどのように視覚化するかをDataTemplateを使って定義
- ItemsPanelプロパティ（ItemsPanelTemplate型）
  - 各項目のレイアウトに使用されるパネルをItemsPanelTemplateを使って定義
  - ListBoxの既定値はVirtualizingStackPanel

COMPONENT ONE® STUDIO 2010J



http://www.grapecity.com/japan/C1/

エディション	概要
for Windows Forms 2010J	Windowsフォーム開発用の12コンポーネント
for ASP.NET 2010J	ASP.NET開発用、Ajaxに対応した28のコンポーネント
for Silverlight 2010J	Silverlight 3開発用の39のコンポーネント
Enterprise 2010J	上記のすべてのコンポーネントとWPF用のコンポーネント、合計80種を収録した最上位エディション

まとめ

- 20年来のWin16/32から初めて刷新されたのがWPF UIフレームワーク
- 高い柔軟性とUI分離のしやすさは、業務アプリにも大きく貢献
  - レイアウトとスケーリング
  - 限りなく自由なGUI部品
  - 強力なデータバインディング


GrapeCity®