Module 5
Základní Data Binding a validace

MGR. TOMÁŠ HAVETTA - MCT

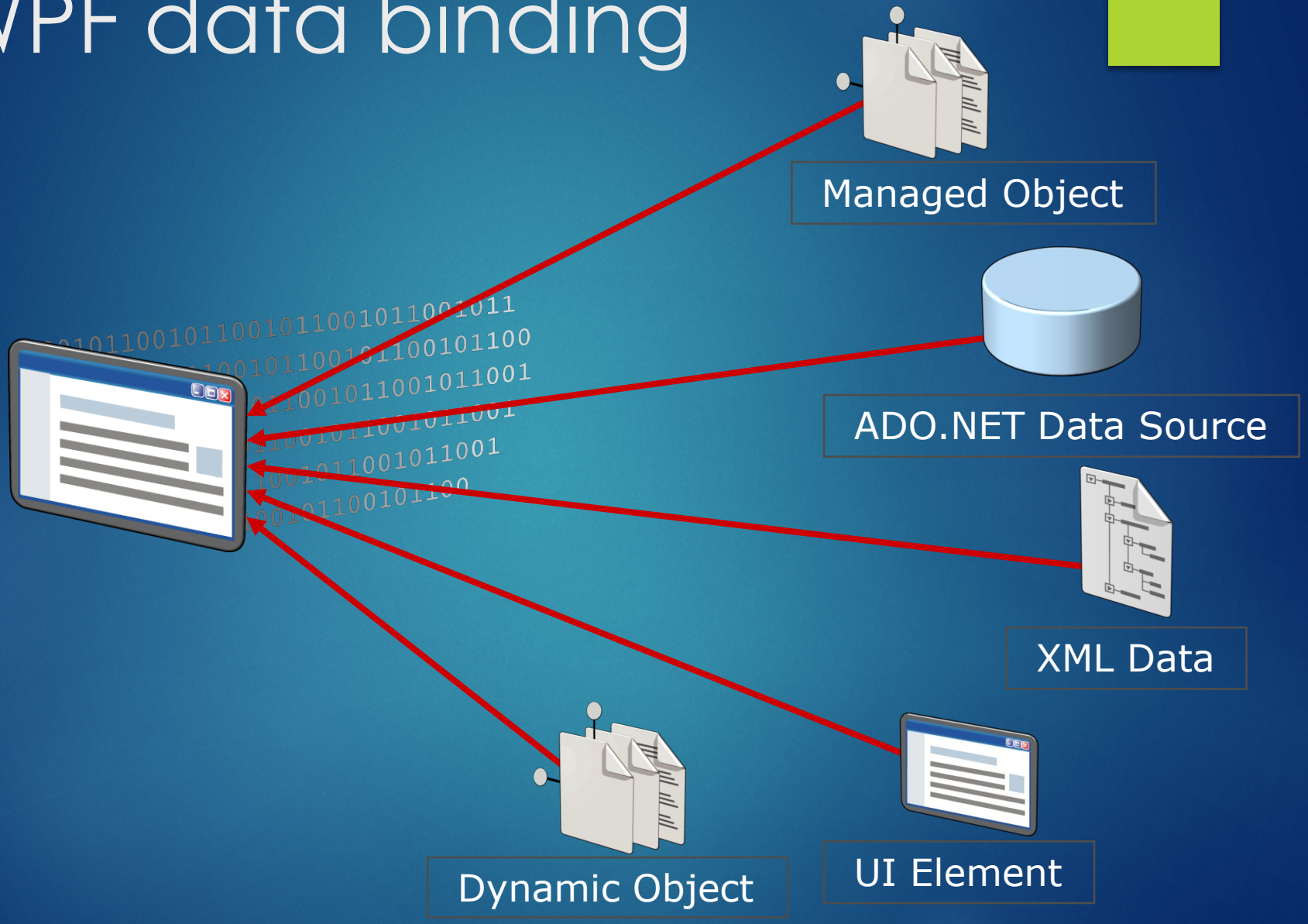
Obsah

- Princip Data Bindingu
- Vytvoření Data Bindingu v XAML
- Propagace změn
- Konverze dat
- Validace dat
- Data v Design režimu

Blok 1: Princip Data Bindingu

- WPF data binding
- Propojení zdroje a cíle
- Režimy data bindingu

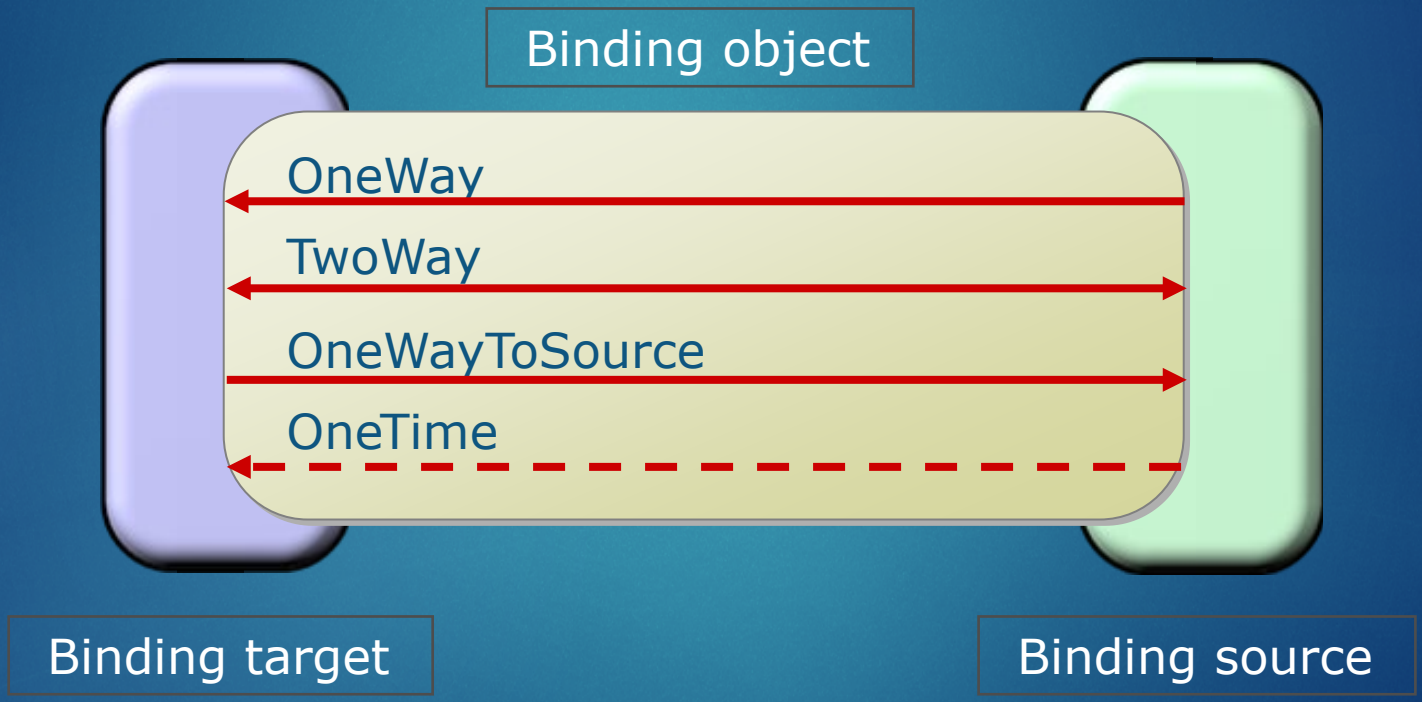
WPF data binding



Propojení zdroje a cíle

- Binding vyžaduje Dependency vlastnost na straně WPF elementu
- U zdroje musíme znát cestu k požadované vlastnosti

Režimy data bindingu



Blok 2: Vytvoření Data Bindingu v XAML

- Vlastnost třídy
- Více elementů na jeden zdroj
- Celý objekt
- XML data
- UI element

Vlastnost třídy

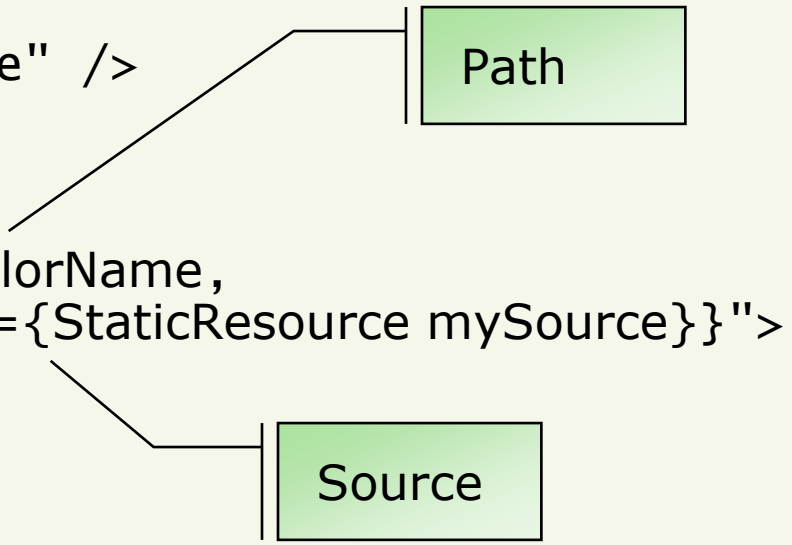
- V Resources připravit instanci třídy
- Path nastavit na vybranou vlastnost

```
<DockPanel xmlns:c="clr-namespace:MyNamespace">

  <DockPanel.Resources>
    <c:MyClass x:Key="mySource" />
  </DockPanel.Resources>

  <Button Background=
            "{Binding Path=ColorName,
                      Source={StaticResource mySource}}">
  ...
  </Button>

</DockPanel>
```



Více elementů na jeden zdroj

- Využít DataContext nadřízeného elementu
- V Bindingu nezadávat Source

```
<DockPanel xmlns:c="clr-namespace:MyNamespace">
  <DockPanel.Resources>...</DockPanel.Resources>
  <DockPanel.DataContext>
    <Binding Source="{StaticResource mySource}"/>
  </DockPanel.DataContext>
  <Button Background="{Binding Path=BackColorName}">
    ...
  </Button>
  <TextBox Foreground="{Binding Path=ForeColorName}">
    ...
  </TextBox>
</DockPanel>
```

Source

Path

Celý objekt

- Binding nemá Path

```
<DockPanel
  xmlns:sys="clr-namespace:System;assembly=mscorlib">
  <DockPanel.Resources>
    <sys:String x:Key="myData">Hello World!</sys:String>
  </DockPanel.Resources>
  <DockPanel.DataContext>
    <Binding Source="{StaticResource myData}"/>
  </DockPanel.DataContext>
  <Label Content="{Binding}">
</DockPanel>
```

Source

Empty binding syntax

XML data

- XmlDataProvider => v Resources, definuje vazbu na XML soubor
- Nutná znalost XPath

```
<ListBox>
    <ListBox.ItemsSource>
      <Binding Source="{StaticResource inventoryData}"
               XPath="*[@Stock='out'] | *[@Number>=8 or
@Number=3]"/>
    </ListBox.ItemsSource>
    ...
        <TextBlock Text="{Binding XPath=Title}">
          <TextBlock.Tooltip>
            <TextBlock
              Text="{Binding Path=Attributes[0].Value}" />
          </TextxBlock.Tooltip>
        </TextBlock>
      ...
  </ListBox>
```

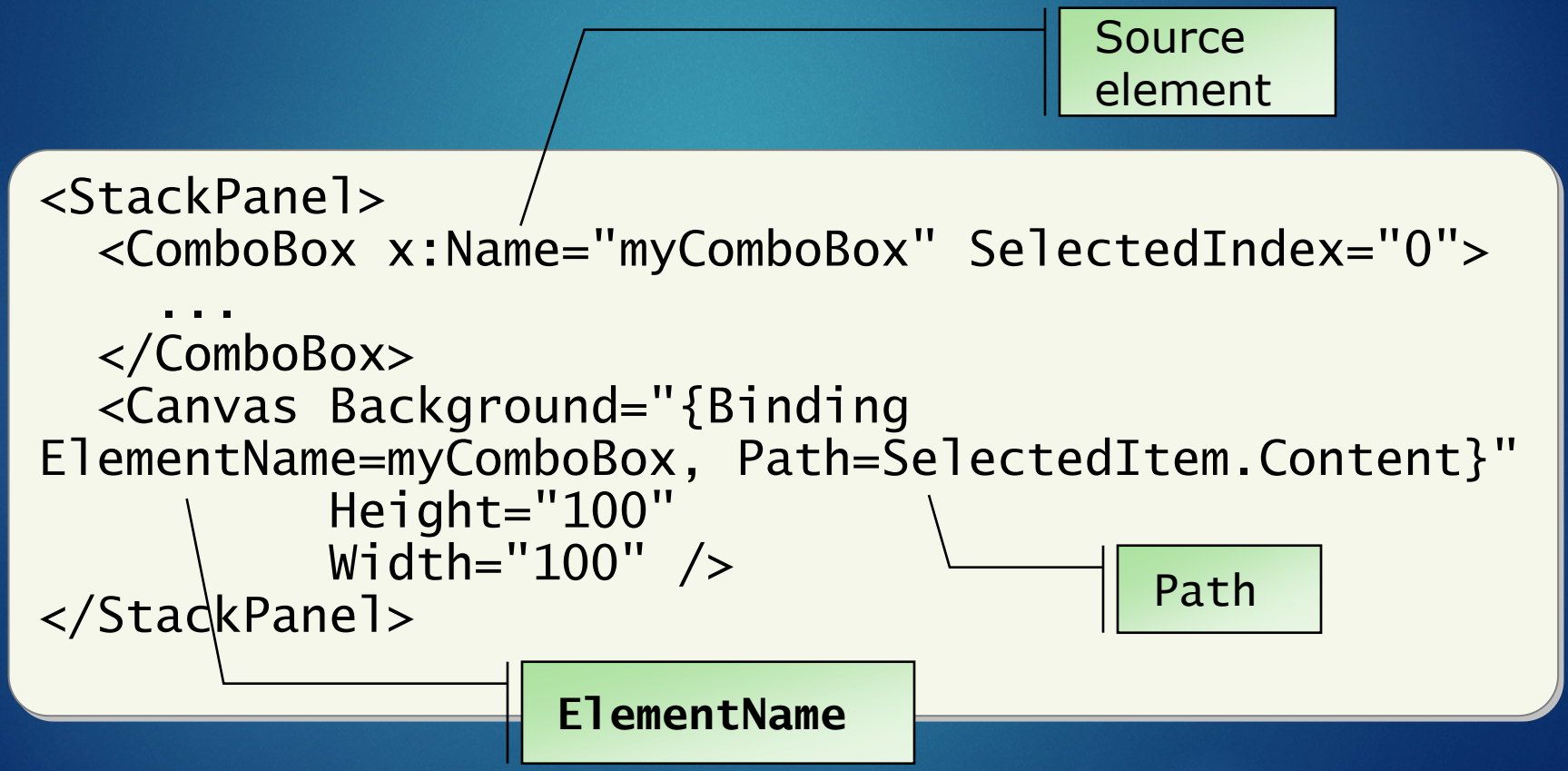
Source

XPath query

Node binding

UI element

- Místo Source se definuje ElementName

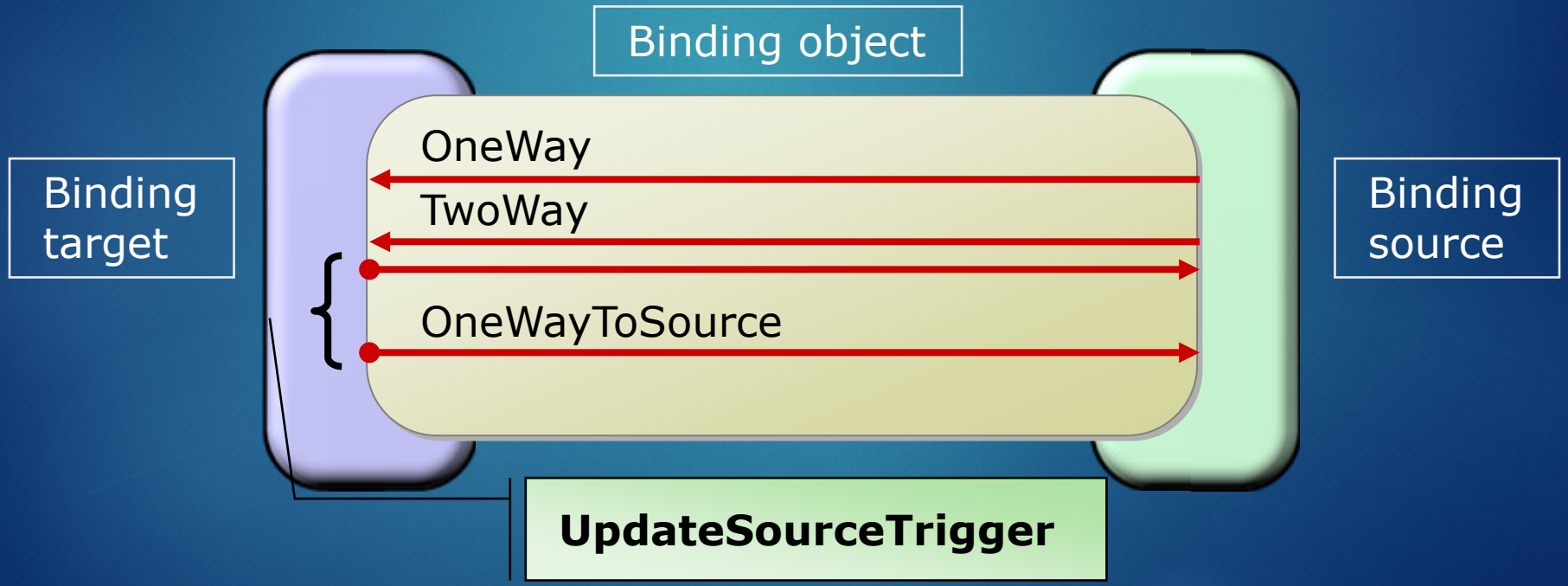


Blok 3: Propagace změn

- Princip notifikace
- Notifikace změny zdroje
- Aktualizace zdroje změnou UI

Princip notifikace

- Binding rozhoduje, kterým směrem předá změny mezi UI a zdrojem a kdy to provede



Notifikace změny zdroje

- INotifyPropertyChanged interface
  - Implementovat interface
  - Vytvořit metodu OnPropertyChanged
  - Volat ji v setrech vlastností třídy

```
public class Person : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;

  protected virtual void OnPropertyChanged(string name)
  {
     PropertyChanged?.Invoke(this,new PropertyChangedEventArgs(name));
  }
}
```

Aktualizace zdroje změnou UI

- UpdateSourceTrigger – vlastnost Bindingu
  - Default
  - Explicit
  - LostFocus
  - PropertyChanged

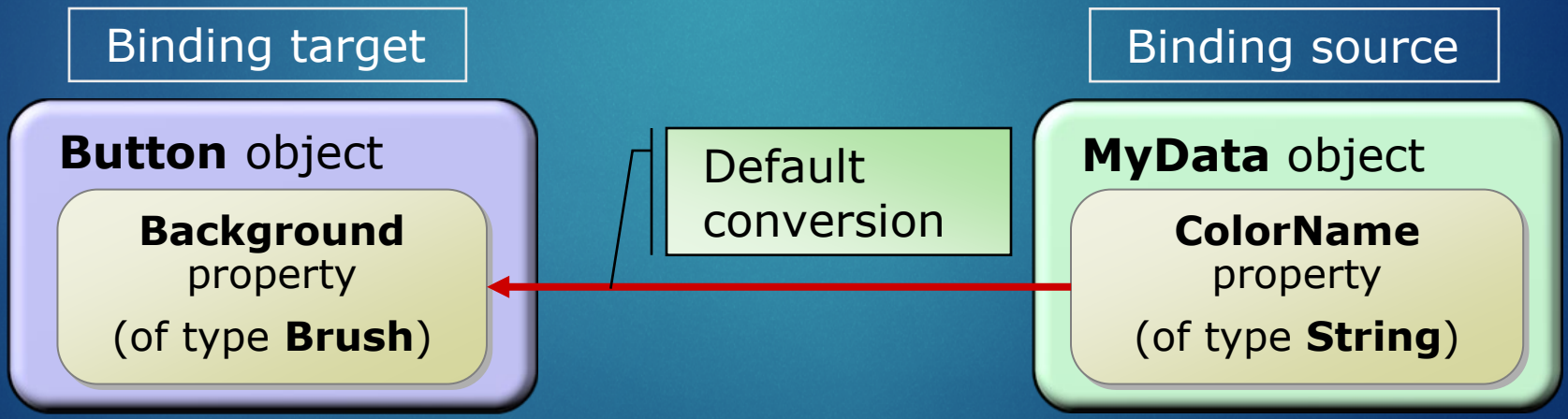
```
<TextBox Width="100">
  <TextBox.Text>
    <Binding Source="{StaticResource myData}"
             Path="ColorName"
             UpdateSourceTrigger="PropertyChanged" />
  </TextBox.Text>
</TextBox>
```

Blok 4: Konverze dat

- Defaultní konverze
- Implementace vlastního convertoru

Defaultní konverze

- Pokud to jde, WPF sama zajistí konverzi
  - String Red na štětec správné barvy
- Pokud neexistuje defaultní konvertor, dojde k výjimce



Implementace vlastního convertoru

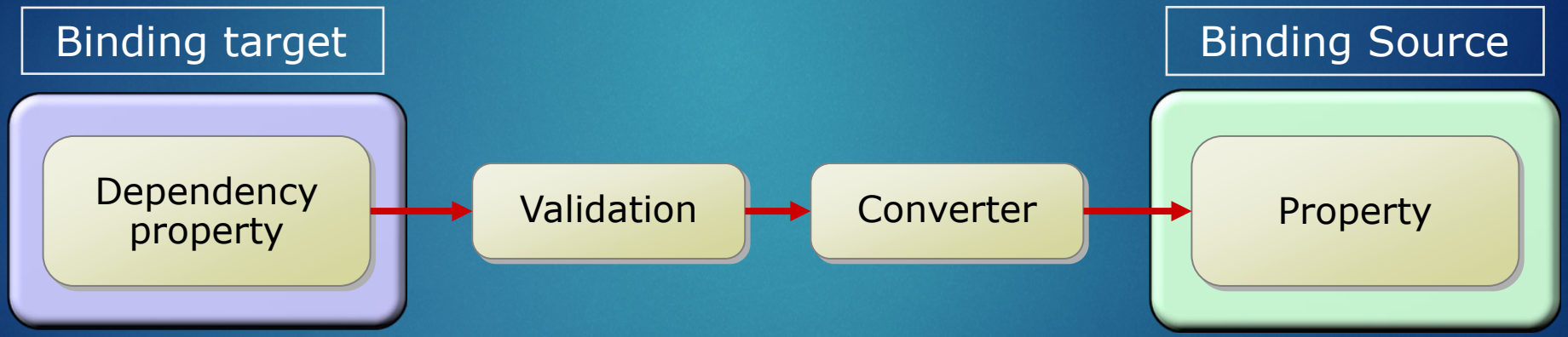
- Vytvořit třídu implementující **IValueConverter**
- Použít správně **ValueConversion** atribut
- Implementovat metody **Convert** a **ConvertBack** podle potřeby
- Vytvořit její instanci v Resources a použít v Bindingu

Blok 5: Validace dat

- Defaultní validace
- Vizualizace chyby
- Vytvoření vlastních pravidel
- Specifikace pravidel v XAML
- Validace pomocí zdrojové třídy

Základy validace

- WPF podporuje několik metod validací
  - Pomocí tříd odvozených z ValidationRule

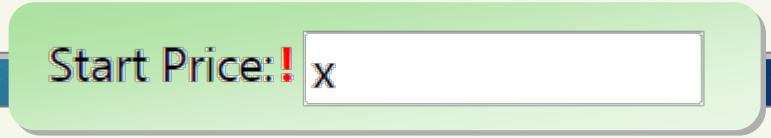



```
<Binding.ValidationRules>
  <ExceptionValidationRule />
</Binding.ValidationRules>
```

Vizualizace chyby

- Použít defaultní režim => Doporučeno
- Vytvořit vlastní ErrorTemplate
- AdornedElementPlaceholder – reprezentuje element v chybovém stavu

```
<ControlTemplate x:Key="errorTemplate">
  <DockPanel>
    <TextBlock Foreground="Red">!</TextBlock>
    <AdornedElementPlaceholder/>
  </DockPanel>
</ControlTemplate>
```




```
<TextBox Validation.ErrorTemplate =
         "{StaticResource errorTemplate}" ...>
```

Vytvoření vlastních pravidel

- Vytvořit třídu odvozenou z ValidationRule
- Implementovat metodu Validate a vrátit ValidationResult

```
public class FutureDateRule : ValidationRule
{
  public override ValidationResult Validate(
    object value, CultureInfo ci)
  {...
      return new ValidationResult(false, "Value is
not a valid date.");
    ...
      return ValidationResult.ValidResult;
  }
}
```

Specifikace pravidel v XAML

- Nastavit Bindingu vlastnost ValidationRules
- Zadat všechny požadované pravidla

```
<TextBox xmlns:src="clr-namespace:MySample">
  <TextBox.Text>
    <Binding ...>
      <Binding.ValidationRules>
        <src:FutureDateRule />
      </Binding.ValidationRules>
    </Binding>
  </TextBox.Text>
</TextBox>
```

ValidationRules

Custom validation class

Validace pomocí zdrojové třídy

- Business pravidla často vyžadují kombinaci několika vlastností a nejde je jednoduše vyhodnotit v UI
- Validace může
  - Odchytávat výjimky z UI
  - Využívat interface **IDataErrorInfo**
  - Využít novější interface **INotifyDataErrorInfo**
- **Kombinace DataAnnotationAttributes a INotifyDataErrorInfo**

Blok 6: Data v Design režimu

- Problémy design režimu
- Design time attributy

Problémy design režimu

- Autor aplikace má bez dat problém korektně vidět UI bez potřeby spustit aplikaci
- Data pro design režim
  - Pomocná třída pro binding zadaná v Resources
  - Speciální d: atributy
  - Využít Blend a jeho podporu dat při tvorbě UI

Design time attributy

- Nastavit vlastnosti z d: namespace
- Podporuje většinu vlastností, ne všechny

```
<Window
  mc:Ignorable="d"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  d:DesignHeight="350"
  d:DesignWidth="500"
  SizeToContent="WidthAndHeight">
  ...
</Window>
```

Ignorable attribute

Required XML namespaces

Use of the attributes

Lab: Základní binding

- Předělejte aplikaci na jeden TextBox, jeden ListBox pro výběr konverze a jeden Label pro výsledek
- Zajistěte přepočítávání ihned při psaní do TextBoxu
- Vytvořte Convertor, který zajistí že barva písma výsledku bude
  - Černá pro hodnoty do 100
  - Zelená pro hodnoty od 100 do 500
  - Červená pro hodnoty nad 500
- Zajistěte validaci vstupu