Telerik Academy

Data-binding

In Windows Universal Applications

Apps for Windows Phone & Windows Store

Telerik Software Academy
http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

Why We Need Data Binding?

Telerik Academy

Why We Need Data Binding?

- The purpose of most applications is to:
  - Display data to users
  - Let them edit that data
- Developers' job is:
  - Bring the data from a variety of sources
  - Expose the data in object, hierarchical, or relational format
- With the XAML data binding engine, you get more features with less code

Why We Need Data Binding? (2)

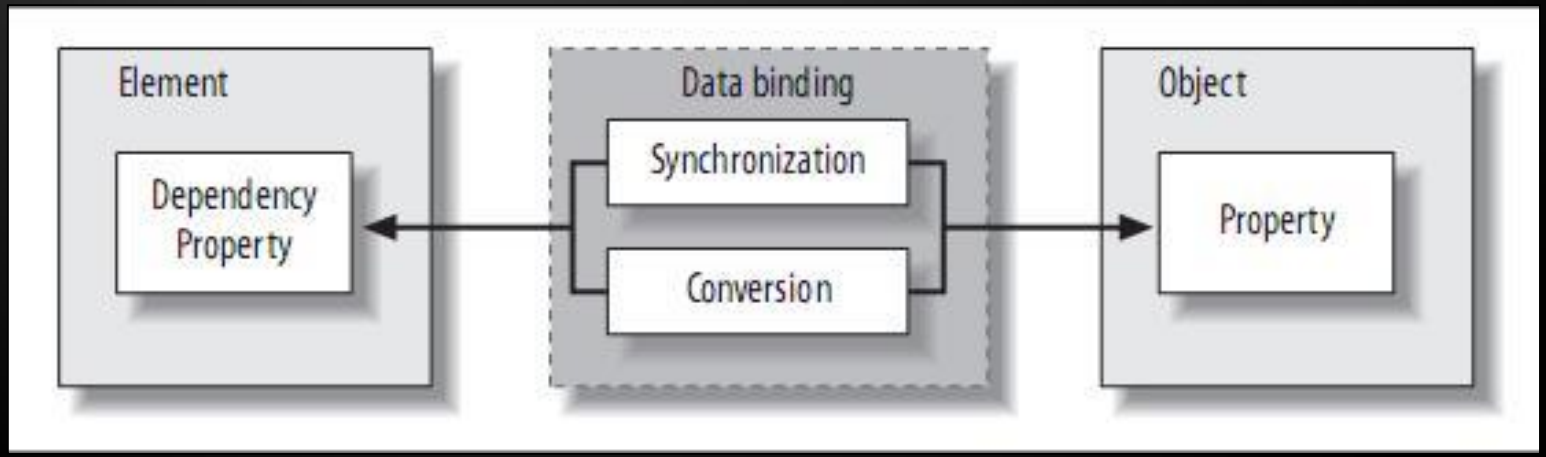
- Data binding is pulling information out of an object into another object or property
  - Data binding means automatically change the value of a property when the value of another property is changed
- Many Windows applications are all about data
- Data binding is a top concern in a user interface technologies like Windows Phone and Windows Store
- Windows Universal and XAML provide very powerful data binding mechanisms

Telerik Academy

Simple Binding

Simple Binding

- Binding in XAML is the act of registering two properties with the data binding engine
  - Letting the engine keep them synchronized
- The synchronization and conversion are duties of the data binding engine



Simple Binding (2)

- Binding a property to a data source property:

```
<TextBox ...>
  <TextBox.Text>
    <Binding Path="SomeName" />
  </TextBox.Text>
</TextBox>
```

- The shorthand binding syntax:

```
<TextBox Text="{Binding Path=SomeName}" />
```

- Binding between the `Text` property of the `TextBox` and an object called `SomeName`
  - `SomeName` is a property of some object to be named later

Telerik Academy

Data Contexts

Data Contexts

- In XAML every `FrameworkElement` and every `FrameworkContentElement` has a `DataContext` property
  - `DataContext` is an object used as data source during the binding, addressed by binding `Path`
- If you don't specify a `Source` property
  - The XAML binding engine goes up the element tree in searching of a DataContext

Data Contexts (2)

- Two controls with the same logical parent can bind to the same data source

```
<!-- DataContextWindow.xaml -->
<Grid Name="GridMain">
  …
  <TextBlock …>Name: </TextBlock>
  <TextBox Text="{Binding Path=Name}" … />
  <TextBlock …>Age:</TextBlock>
  <TextBox Text="{Binding Path=Age}" … />
  <Button Name="ButtonBirthday Content="Birthday!" … />
</Grid>
```

- Providing a `DataContext` value for both of the text box controls

Data Contexts (3)

- Setting an object as a value of the grid's `DataContext` property in the `MainWindow` constructor:

```
public partial class MainWindow : Window
{
  Person person = new Person("Tom", 11);

  public MainWindow()
  {
    InitializeComponent();
    GridMain.DataContext = person;
  }
  …
}
```

Telerik Academy

Data Contexts

Live Demo

Telerik Academy

Binding to Other Controls

Binding to Other Controls

- XAML provides binding of one element's property to another element's property

```
<TextBox Name="ageTextBox" Foreground="Red" … />
<Button …
  Foreground="{Binding ElementName=ageTextBox,
  Path=Foreground}" Content="Birthday" />
```

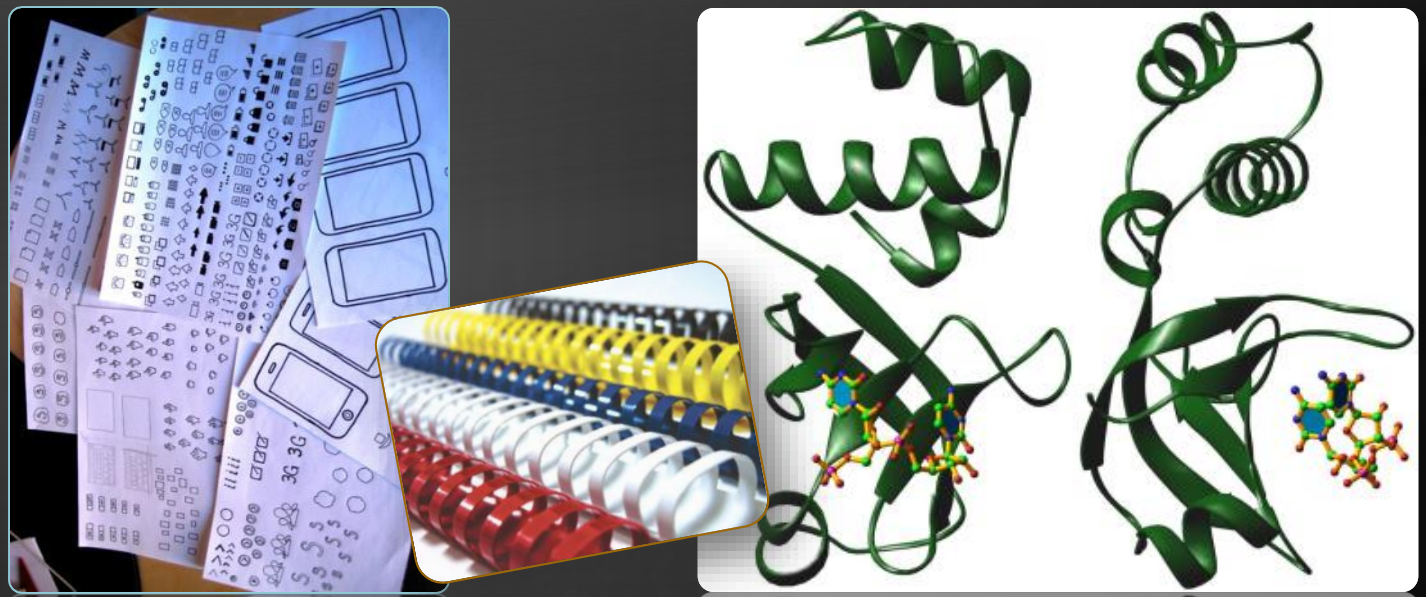
- The button's foreground brush will always follow foreground brush's color of the age TextBox

Telerik Academy

Binding to Other Controls

Live Demo

Telerik Academy



The Binding Class and Its Properties

Binding Class

- A more full-featured binding example

```
<TextBox x:Name="TextBoxPerson"
    Foreground="{Binding Path=Age, Mode=OneWay,
    Source={StaticResource Tom},
    Converter={StaticResource ageConverter}}" />
```

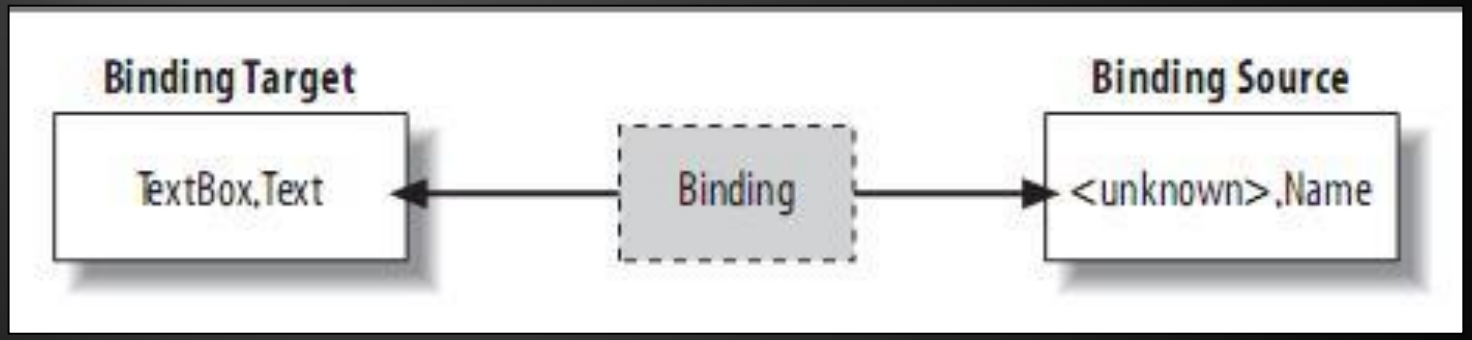
- This features are represent in `Binding` class
  - `Converter` – convert values back and forth from the data source
  - `ConverterParameter` – parameter passed to the `IValueConverter` methods during the conversion

Binding Class (2)

- More Binding class properties
  - ElementName – used when the source of the data is a UI element as well as the target
  - Mode – one of the BindingMode values TwoWay, OneWay, OneTime, OneWayToSource, or Default
  - Path – path to the data in the data source object
  - Source – a reference to the data source

Binding Class (3)

- The binding target can be any XAML element
  - Only allowed to bind to the element's dependency properties



- The TextBox control is the binding target
- Object that provides the data is the binding source

Telerik Academy

Value Conversion

Value Conversion

- In the previous example we might decide that anyone over age 25 is hot
  - Should be marked in the UI as red
- Binding to a non-Text property

```
<TextBox
  Text="{Binding Path=Age}"
  Foreground="{Binding Path=Age, …}" … />
```

- Bound the age text box's `Text` property to the `Person` object's `Age` property

Value Conversion (2)

- How to bind the `Foreground` property of the text box to `Age` property on the `Person` object?
  - The `Age` is of type `Int32` and `Foreground` is of type `Brush`
  - Mapping from `Int32` to `Brush` needs to be applied to the data binding from `Age` to `Foreground`
  - That's the job of a `ValueConverter`

Value Conversion (3)

- A value converter is an implementation of the `IValueConverter` interface
  - `Convert()` – converting from the source data to the target UI data
  - `ConvertBack()` – convert back from the UI data to the source data

Value Conversion (4)

- Converter Int32 → Brush

```
public class AgeToForegroundConverter : IValueConverter
{
  public object Convert(object value,
    Type targetType, …)
  {
    if (targetType != typeof(Brush))
    {
      return null;
    }
    int age = int.Parse(value.ToString());
    return (age > 25 ? Brushes.Red : Brushes.Black);
  }
  …
}
```

Value Conversion (4)

- Creating an instance of the converter class in the XAML:

```
<Window … xmlns:local="clr-namespace:WithBinding">
  <Window.Resources>
    <local:Person x:Key="Tom" … />
    <local:AgeToForegroundConverter x:Key="ageConverter"/>
  </Window.Resources>
  <Grid DataContext="{StaticResource Tom}"> …
    <TextBox
      Text="{Binding Path=Age}"
      Foreground="{Binding Path=Age,
      Converter={StaticResource ageConverter}}" … /> …
    <Button … Foreground="{Binding Path=Foreground,
      ElementName=ageTextBox}">Birthday</Button>
  </Grid>
</Window>
```

Telerik Academy

Value Conversion

Live Demo

Telerik Academy

Binding Path Syntax

Binding Path Syntax

- When you use `Path=Something` in a `Binding` statement, the `Something` can be in a number of formats
  - `Path=Property` – bind to the property of the current object (`Path=Age`)
  - `Path=(OwnerType.AttachedProperty)` – bind to an attached dependency property (`Path=(Validation.HasError)`)
  - `Path=Property.SubProperty` – bind to a subproperty (`Path=Name.Length`)

Binding Path Syntax (2)

- Other formats
  - `Path=Property[n]` – bind to an indexer (`Path=Names[0]`)
  - `Path=Property/Property` – master-detail binding (`Path=Customers/Orders`)
  - `Path=(OwnerType.AttachedProperty)[n].SubProperty` – bind to a mixture of properties, subproperties, and indexers
    - `Path=(Validation.Errors)[0].ErrorContent`)

Update Source Triggers

Telerik Academy

Update Source Triggers

- Binding can happen immediately when the control state changes
  - Using the `UpdateSourceTrigger` property on the `Binding` object

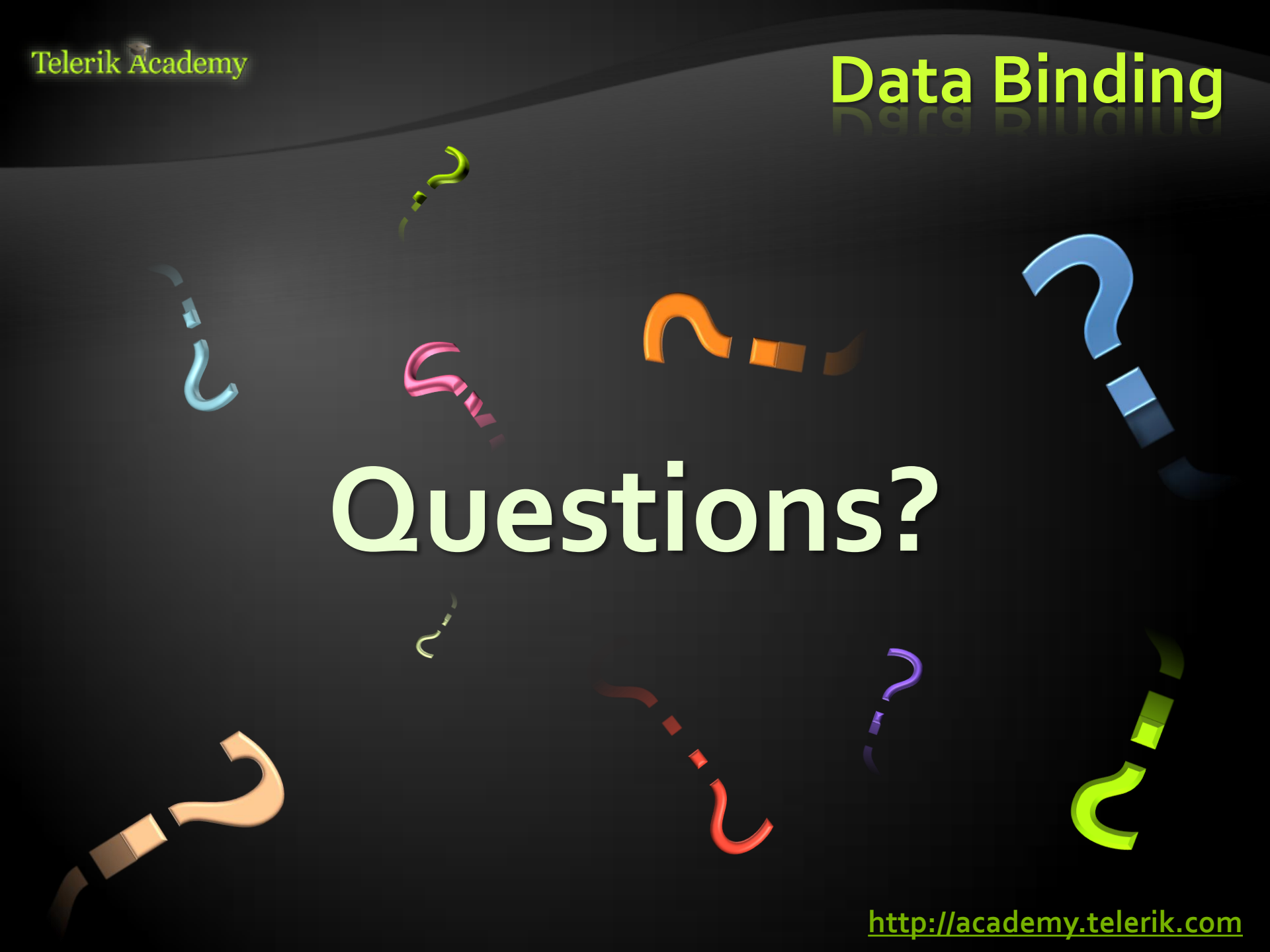
```
<TextBox …>
  <TextBox.Text>
    <Binding Path="Age"
      UpdateSourceTrigger="PropertyChanged"> …
    </Binding>
  </TextBox.Text>
</TextBox>
```

Update Source Triggers

- `UpdateSourceTrigger` values
  - `Default` – updates "naturally" based on the target control
  - `PropertyChanged` – updates the source immediately
  - `Explicit` – when `BindingExpression.UpdateSource()` is explicitly called

Telerik Academy

Data Binding



Questions?

http://academy.telerik.com

Exercises

1. Write a program that show a simple window, it contains two controls a `Slider` and a `TextBlock` with a single line of text. If you pull the thumb in the slider to the right, the font size of the text is increased immediately. Add a label that shows the current font size. Use data binding.
2. Add to the previous exercise few buttons each of which applies a preset value to the slider. When you click the "Set to Large" button the code in `Click` event sets the value of the slider, which in turn forces a change to the font size of the text through data binding.

Exercises (2)

3. Write a program that shows a simple window, which contains a `TextBlock` and setup the `TextBlock` to draw its text from a `TextBox` and its current foreground and background colors from separate lists of colors.