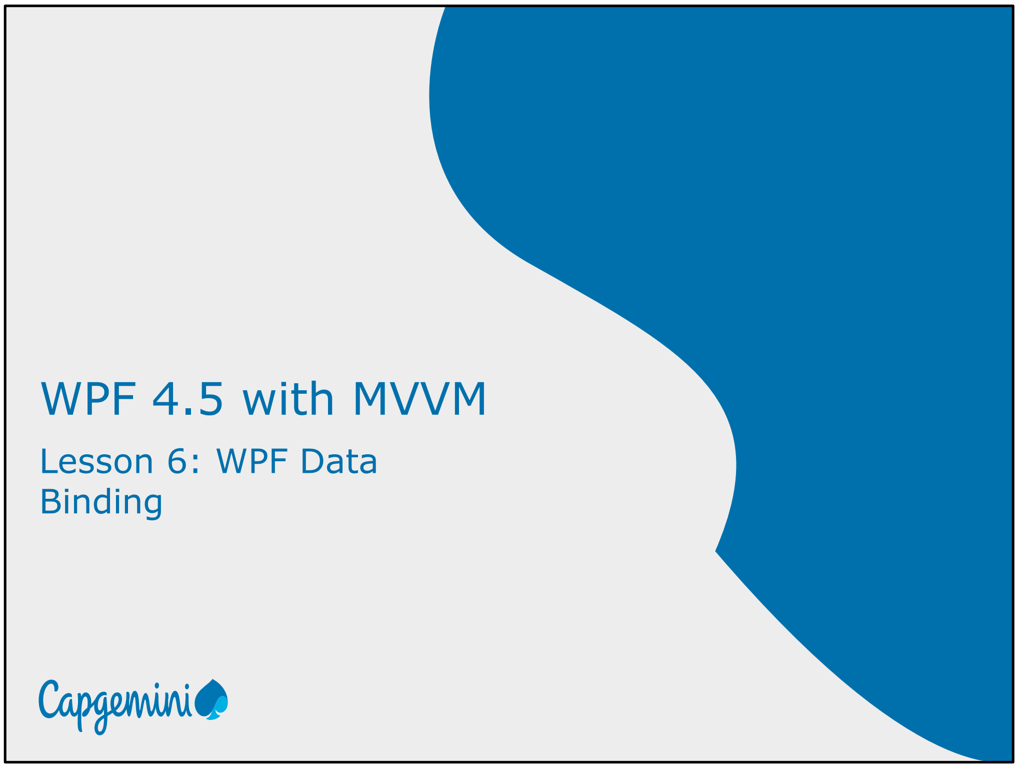
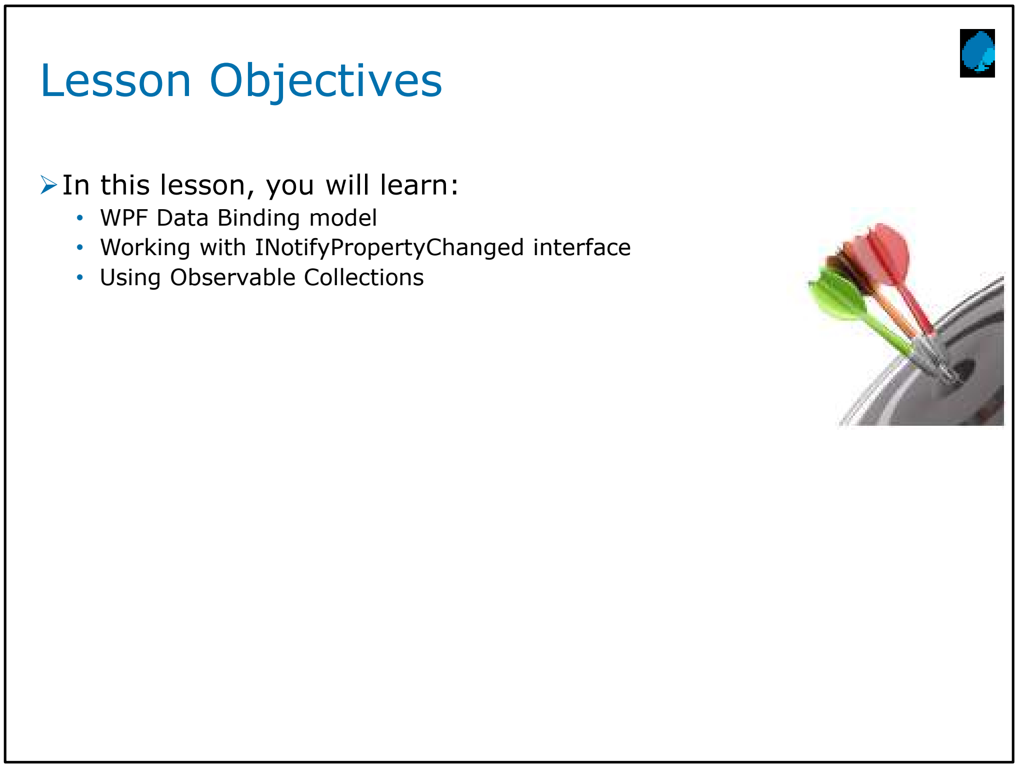


WPF 4.5 with MVVM

Lesson 6: WPF Data Binding



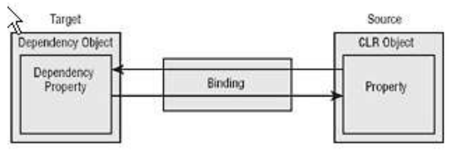
Lesson Objectives

- In this lesson, you will learn:
  - WPF Data Binding model
  - Working with INotifyPropertyChanged interface
  - Using Observable Collections

6.1: Introduction

Concept Of Data Binding

- Data binding is a relationship that conveys to WPF to extract some information from a source object and use it to set a property in a target object
- The target property is always a dependency property, and it is usually in a WPF element
- The ultimate goal of WPF data binding is to display some information in the user interface



Data Binding:
Concept :
At its simplest, data binding is a relationship that conveys to WPF to extract some information from a source object and use it to set a property in a target object. The target property is always a dependency property, and it is usually in a WPF element — after all, the ultimate goal of WPF data binding is to display some information in your user interface. However, the source object can be just about anything, ranging from another WPF element to an ADO.NET data object (like the DataTable and DataRow) or a data-only object of your own creation.

6.1: Introduction

Concept Of Data Binding

- The source object can be just about anything, ranging from another WPF element to an ADO.NET data object (like the DataTable and DataRow) or a data-only object of your own creation
- The simplest data binding scenario occurs when your source object is a WPF element and your source property is a dependency property

Data Binding:
Concept (contd.):
The simplest data binding scenario occurs when your source object is a WPF element and your source property is a dependency property.
That is because dependency properties have built-in support for change notification. As a result, when you change the value of the dependency property in the source object, the bound property in the target object is immediately updated. This is exactly what you want, and it happens without requiring you to build any additional infrastructure.

6.1: Introduction

Concept Of Data Binding

- This is because dependency properties have built-in support for change notification
- As a result, when you change the value of the dependency property in the source object, the bound property in the target object is updated immediately

6.1: Introduction

Element To Element Data Binding

```
<Slider Name="sliderFontSize" Margin="3"
Minimum="1" Maximum="40" Value="10"
TickFrequency="1" TickPlacement="TopLeft">
</Slider>
```

Data binding expressions use a XAML markup extension (and hence have curly braces). You begin with the word Binding, because you're creating an instance of the System.Windows.Data.Binding class. Although you can configure a Binding object in several ways, in this situation you need to set just two properties: the ElementName that indicates the source element and a Path that indicates the property in the source element.

6.1: Introduction

Element To Element Data Binding

```
<TextBlock Margin="10" Text="Simple Text"
Name="lblSampleText"
FontSize="{Binding ElementName=sliderFontSize, Path=Value}" >
</TextBlock>
```

6.1: Introduction

Automatically Updating the Source of Data Binding

- WPF 4.5 has added new property named 'Delay' to Binding markup extension.
- This 'Delay' property can be used to specify an amount of time to pass after the property changes on the target before the source updates.

Automatically Updating the Source of Data Binding:

A binding is created between two objects namely the source (from where the data comes) and the target (where the data is propagated).
A Binding is nothing but synchronization between two properties. This synchronization takes place immediately i.e. each change of the target value even a small one will update the source.
WPF 4.5 has added new property named 'Delay' to Binding markup extension.
This 'Delay' property can be used to specify an amount of time to pass after the property changes on the target before the source updates.

This feature can be very useful:

> When working with controls like the Slider for which there is no need to update a source value for each pixel that the Slider moves.
> To display the typed text in TextBox by "block" instead displaying one letter at a time.

6.1: Introduction

Element To Element Data Binding

```
<Slider Name="sliderFontSize" Margin="3"
Minimum="1" Maximum="40" Value="10"
TickFrequency="1" TickPlacement="TopLeft">
</Slider>
```

Data binding expressions use a XAML markup extension (and hence have curly braces). You begin with the word Binding, because you're creating an instance of the System.Windows.Data.Binding class. Although you can configure a Binding object in several ways, in this situation you need to set just two properties: the ElementName that indicates the source element and a Path that indicates the property in the source element.

6.1: Introduction

Element To Element Data Binding

```
<TextBlock Margin="10" Text="Simple Text"
Name="lblSampleText"
FontSize="{Binding ElementName=sliderFontSize,
Delay=2000,Path=Value}" >
</TextBlock>
```

Concept Of Data Binding

- Databinding and DataTemplates are the most powerful features of WPF
- Support for Databinding is built into WPF from its core
- Almost every graphics/UI object that you will work with in WPF inherits from DependencyObject
- The functionality supported by this base is what powers animation, styling, and databinding

6.1: Introduction

Concept Of Data Binding

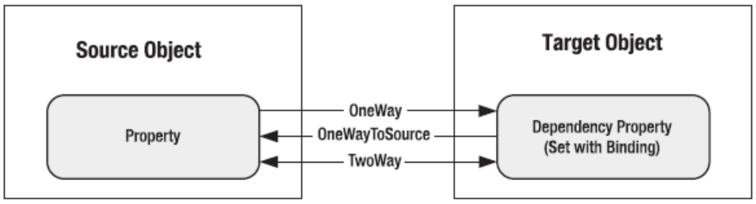
- Objects that inherit from DependencyObject support a special type of property called a DependencyProperty
- Most of the properties you will work with are DependencyProperties
- Example: Text, Content, Width, Height
- Any DependencyProperty can be animated, styled, and databound

6.2: Data Binding

Binding Directions

➢ Following are the Binding Directions:

- OneWay: The target property is updated when the source property changes
- TwoWay: The target property is updated when the source property changes, and the source property is updated when the target property changes
- OneTime: The target property is set initially based on the source property value. However, changes are ignored from that point onwards
- OneWayToSource: It is similar to OneWay but in reverse. The source property is updated when the target property changes



Data Binding:
Binding Directions:
Following are the binding directions:

One-time:

Binding goes from the source to the target and occurs only once when the application is started or the data context changes. Here, you get a snapshot of the data.

One-way:

Binding goes from the source to the target. This is useful for read-only data, as it is not possible to change the data from the user interface. To get updates to the user interface, the source must implement the interface INotifyPropertyChanged.

Two-way:

With a two-way binding, the user can make changes to the data from the UI. Binding occurs in both directions - from the source to the target and from the target to the source.

One-way-to-source:

With one-way-to-source binding, if the target property changes, then the source object gets updated.

6.2: Data Binding

Binding To Objects

- When binding to an object that is not an element, you need to give up the Binding.Element-Name property and use one of the following properties instead:
- Source. This is a reference that points to the source object,in other words, the object that’s supplying the data
- DataContext. If you do not specify a source using the Source property, WPF searches up the element tree starting at the current element

6.2: Data Binding

Binding To Objects

- It examines the DataContext property of each element and uses the first one that is not null
- The DataContext property is extremely useful if you need to bind several properties of the same object to different elements, because you can set the DataContext property of a higher-level container object rather than directly on the target element

6.2: Data Binding

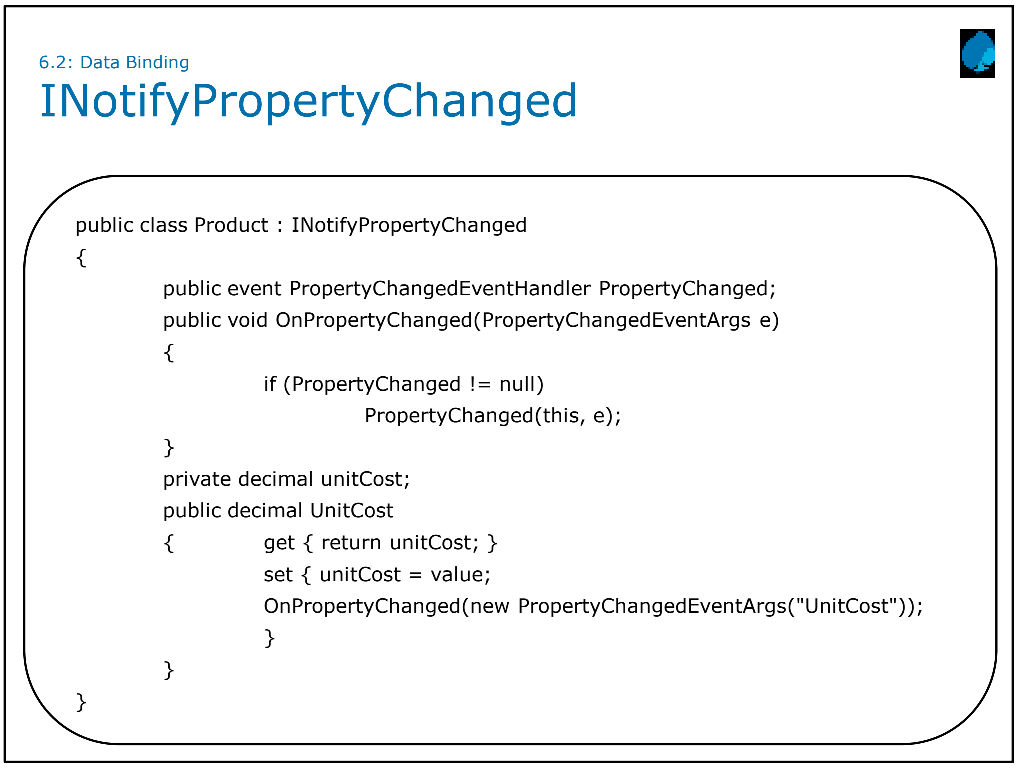
Demo

- Demo on Data Binding custom objects

6.2: Data Binding

Change Notification

- When the WPF element property changes, it notifies the target about the change
- This will not happen for custom Objects, as they will not provide Change Notification
- You can use three approaches to solve this problem
  - You can make each property in the Custom class a dependency property
  - You can raise an event for each property
  - You can implement the System.ComponentModel.INotifyPropertyChanged interface, which requires a single event named PropertyChanged
- The first approach relies on the WPF dependency property infrastructure, while both the second and the third rely on events
- Usually, when creating a data object, you will use the third approach



6.2: Data Binding

INotifyPropertyChanged

```
public class Product : INotifyPropertyChanged
{
        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged(PropertyChangedEventArgs e)
        {
                if (PropertyChanged != null)
                        PropertyChanged(this, e);
        }
        private decimal unitCost;
        public decimal UnitCost
        {       get { return unitCost; }
                set { unitCost = value;
                OnPropertyChanged(new PropertyChangedEventArgs("UnitCost"));
                }
        }
}
```

6.2: Data Binding

List Binding

- Binding to a list is more frequently done than binding to simple objects
- Binding to a list is very similar to binding to a simple object
- You can assign the complete list to the DataContext from code behind, or you can use an ObjectDataProvider that accesses an object factory that returns a list

6.2: Data Binding

List Binding

- With elements that support binding to a list (for example, a ListBox), the complete list is bound
- With elements that just support binding to one object (for example, a TextBox), the current item is bound

6.2: Data Binding

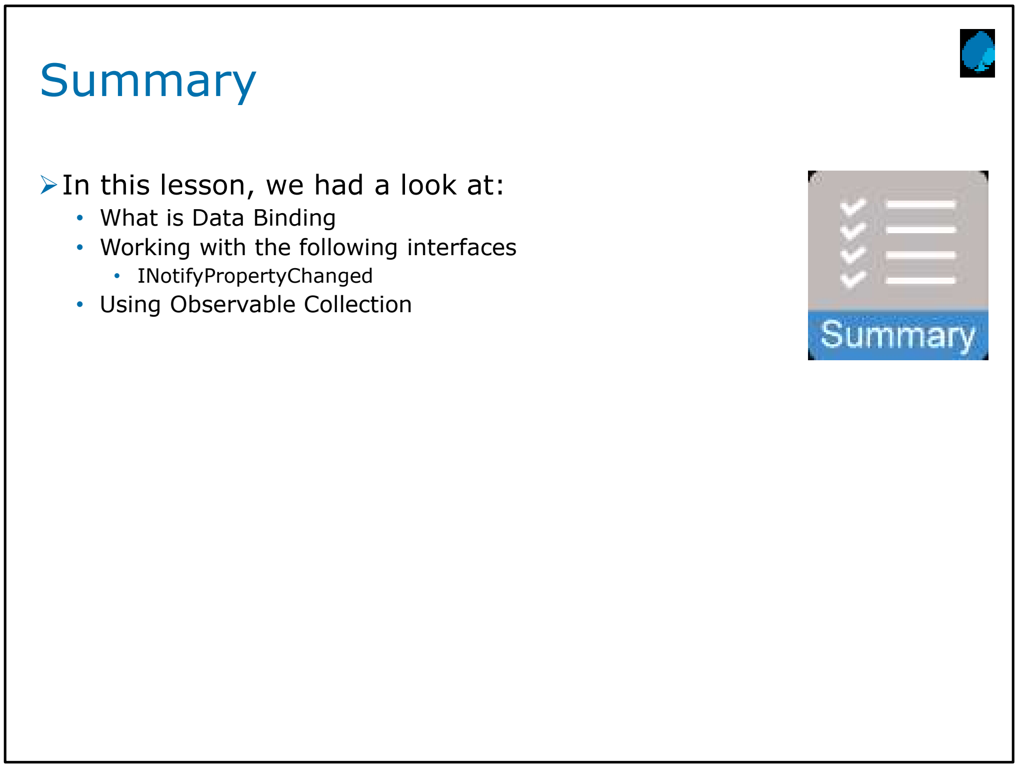
Binding To XML

- WPF data binding has special support for binding to XML data
- You can use XmlDataProvider as a data source and bind the elements by using XPath expressions
- For a hierarchical display, you can use the TreeView control and create the view for the items by using the HierarchicalDataTemplate

6.2: Data Binding

Demo

- Demo on Data Binding and List Binding



Summary

- In this lesson, we had a look at:
  - What is Data Binding
  - Working with the following interfaces
    - INotifyPropertyChanged
  - Using Observable Collection