XAML - DATA BINDING

Data binding is a mechanism in XAML applications that provides a simple and easy way for Windows Runtime Apps using partial classes to display and interact with data. The management of data is entirely separated from the way the data is displayed in this mechanism.

Data binding allows the flow of data between UI elements and data object on user interface. When a binding is established and the data or your business model changes, then it will reflect the updates automatically to the UI elements and vice versa. It is also possible to bind, not to a standard data source, but rather to another element on the page. Data binding can be of two types −

- One-way data binding
- Two-way data binding

One-Way Data Binding

In one-way binding, data is bound from its source *thatistheobjectthatholdsthedata* to its target *thatistheobjectthatdisplaysthedata*.

Let's have a look at a simple example of one-way data binding. The following XAML code creates four text blocks with some properties.

```
<Window x:Class = "DataBindingOneWay.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   Title = "MainWindow" Height = "350" Width = "604">

   <Grid>

      <StackPanel Name = "Display">

         <StackPanel Orientation = "Horizontal" Margin = "50, 50, 0, 0">
            <TextBlock Text = "Name: " Margin = "10" Width = "100" />
            <TextBlock Margin = "10" Width = "100" Text = "{Binding Name}" />
         </StackPanel>

         <StackPanel Orientation = "Horizontal" Margin = "50,0,50,0">
            <TextBlock Text = "Title: " Margin = "10" Width = "100" />
            <TextBlock Margin = "10" Width = "100" Text = "{Binding Title}" />
         </StackPanel>

      </StackPanel>

   </Grid>

</Window>
```

Text properties of two text blocks are set to "Name" and "Title" statically, while the other two text blocks Text properties are bound to "Name" and "Title" which are class variables of Employee class which is shown below.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataBindingOneWay {
   public class Employee {
      public string Name { get; set; }
      public string Title { get; set; }
```

```
      public static Employee GetEmployee() {
         var emp = new Employee() {
            Name = "Ali Ahmed", Title = "Developer"
         };
         return emp;
      }
   }
}
```

In this class, we have just two variables, **Name** and **Title**, and one static method in which the Employee object is initialized which will return that employee object. So we are binding to a property, Name and Title, but we have not selected what object that property belongs to. The easiest way is to assign an object to DataContext whose properties we are binding in the following C# code −

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace DataBindingOneWay {
   /// <summary>
      /// Interaction logic for MainWindow.xaml
   /// </summary>

   public partial class MainWindow : Window {

      public MainWindow() {
         InitializeComponent();
         DataContext = Employee.GetEmployee();
      }
   }
}
```

Let's run this application and you can see immediately in our MainWindow that we have successfully bound to the Name and Title of that Employee object.

Two-Way Data Binding

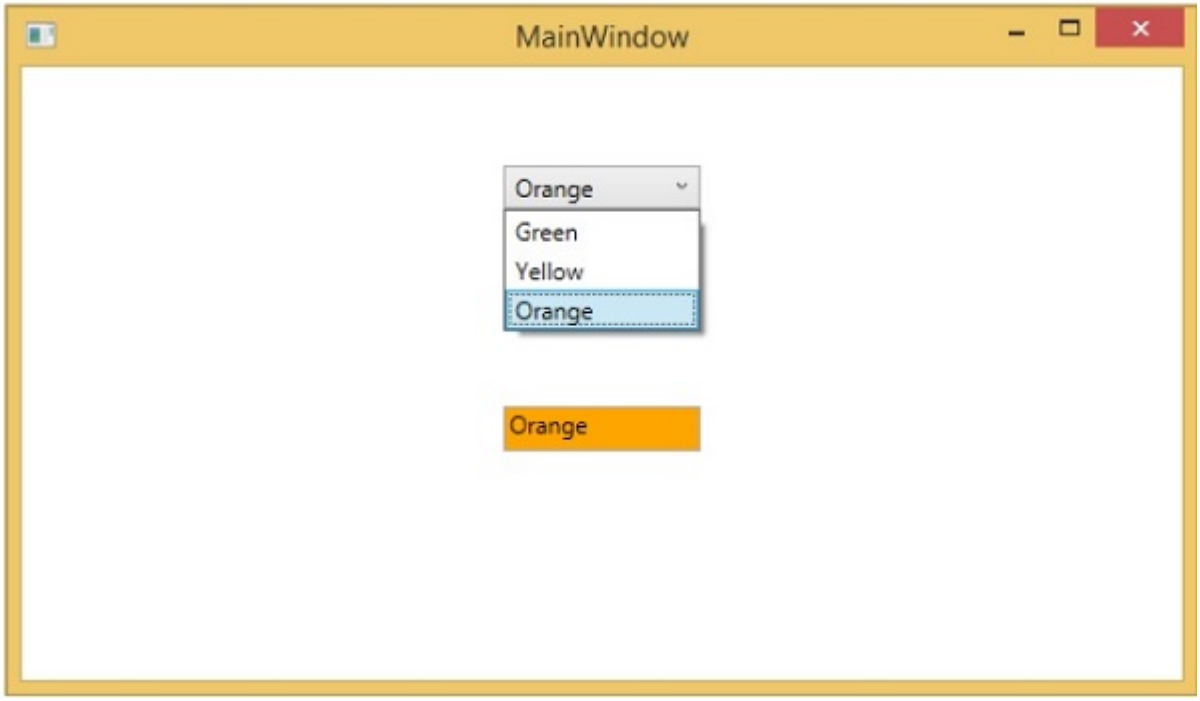
In two-way binding, the user can modify the data through the user interface and have that data updated in the source. If the source changes while the user is looking at the view, you would want to update the view.

Example

Let’s have a look at the following example in which one combobox with three combobox items and one textbox are created with some properties. In this example, we don’t have any standard data source, but the UI elements are bound to other UI elements.

```
<Window x:Class = "XAMLTestBinding.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   Title = "MainWindow" Height = "350" Width = "604">

   <StackPanel>

      <ComboBox Name = "comboBox"  Margin = "50" Width = "100">
         <ComboBoxItem Content = "Green" />
         <ComboBoxItem  Content = "Yellow" IsSelected = "True" />
         <ComboBoxItem Content = "Orange" />
      </ComboBox>

      <TextBox  Name = "textBox" Margin = "50"
         Width = "100" Height = "23" VerticalAlignment = "Top"
         Text  = "{Binding ElementName = comboBox, Path = SelectedItem.Content,
         Mode = TwoWay, UpdateSourceTrigger = PropertyChanged}"
         Background = "{Binding ElementName = comboBox, Path = SelectedItem.Content}">
      </TextBox>

   </StackPanel>

</Window>
```

When you compile and execute the above code, it will produce the following output. When the user selects an item from the combobox, the textbox text and the background color will be updated accordingly.



Similarly, when the user types a valid color name in the textbox, then the combobox and the textbox background color will also be updated.