XAML Binding Lab - WPF

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Bind Text Box to Label

Create a new WPF application named **SimpleDataBindingSamples**.

Add the following code to the MainWindow.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <TextBox x:Name="DataEntry" />
    <Label Content="{Binding ElementName=DataEntry,
Path=Text}" />
  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 2: Bind Combo Box to Text Block

Add the following code to the MainWindow.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <ComboBox x:Name="Language">
      <ComboBoxItem Content="C#" />
      <ComboBoxItem Content="Visual Basic" />
      <ComboBoxItem Content="C++" />
      <ComboBoxItem Content="F#" />
    </ComboBox>
    <Label Content="{Binding ElementName=Language,
Path=SelectionBoxItem}" />
    <Label Content="{Binding ElementName=Language,
Path=SelectedItem.Content}" />
  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 3: Bind to Font Family

Add the following code to the MainWindow.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <ComboBox Name="Fonts"
              ItemsSource="{x:Static
Fonts.SystemFontFamilies}" />

    <Label Content="This is some Text."
           FontSize="16"
           FontFamily="{Binding ElementName=Fonts,
Path=SelectedItem}" />
  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 4: Bind IsEnabled Property

Add the following code to the MainWindow.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <CheckBox Content="Has Benefits?"
              Name="HasBenefits" />
    <CheckBox Content="401k"
              IsEnabled="{Binding Path=IsChecked,
ElementName=HasBenefits}" />
    <CheckBox Content="Health Care"
              IsEnabled="{Binding Path=IsChecked,
ElementName=HasBenefits}" />

  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 5: Boolean to Visibility Converter

Add the following code to the MainWindow.Resources section.

```
<Window.Resources>
  <BooleanToVisibilityConverter
     x:Key="BoolToVisibility" />
</Window.Resources>
```

Add the following to main window.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <CheckBox Content="Has Benefits?"
              Name="HasBenefits" />
    <CheckBox Content="401k"
              Visibility="{Binding Path=IsChecked,
ElementName=HasBenefits, Converter={StaticResource
BoolToVisibility}}" />
    <CheckBox Content="Health Care"
              Visibility="{Binding Path=IsChecked,
ElementName=HasBenefits, Converter={StaticResource
BoolToVisibility}}" />
  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 6: Not Boolean to Visibility Converter

Right mouse-click on the Project and add a new folder named **Converters**.

Right mouse-click on the **Converters** folder and add a new class named **NotBooleanToVisibilityConverter.**

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace SimpleDataBindingSamples.Converters
{
  public class NotBooleanToVisiblityConverter :
IValueConverter
  {
    public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
    {
      if (!(bool)value) {
        return Visibility.Visible;
      }
      else {
        return Visibility.Collapsed;
      }
    }

    public object ConvertBack(object value, Type
targetType, object parameter, CultureInfo culture)
    {
      throw new NotImplementedException();
    }
  }
}
```

Add an XML namespace to the window.

```
xmlns:convert="clr-
namespace:SimpleDataBindingSamples.Converters"
```

Add the following code to the MainWindow.Resources section.

```
<Window.Resources>
  <convert:NotBooleanToVisiblityConverter
x:Key="NotBoolToVisibility" />
</Window.Resources>
```

Add the following to main window.

```
<Border BorderBrush="Black"
        BorderThickness="2"
        Margin="10">
  <StackPanel Margin="10">
    <CheckBox Content="Has Benefits?"
              Name="HasBenefits" />
    <CheckBox Content="401k"
              Visibility="{Binding Path=IsChecked,
ElementName=HasBenefits, Converter={StaticResource
NotBoolToVisibility}}" />
    <CheckBox Content="Health Care"
              Visibility="{Binding Path=IsChecked,
ElementName=HasBenefits, Converter={StaticResource
NotBoolToVisibility}}" />
  </StackPanel>
</Border>
```

Try It Out

Run the application to view the results.

Lab 7: Binding to Radio Buttons

Right mouse-click on the **Converters** folder and add a new class named **InvertBooleanConverter**.

```
using System;
using System.Globalization;
using System.Windows.Data;

namespace SimpleDataBindingSamples.Converters {
  /// <summary>
  /// Call this converter to change a True value to a
False and False to a True
  /// </summary>
  public class InvertBooleanConverter : IValueConverter
{
    public object Convert(object value, Type targetType,
                          object parameter, CultureInfo
culture) {
      return !(bool)value;
    }

    public object ConvertBack(object value, Type
targetType, object parameter, CultureInfo culture) {
      return (bool)value;
    }
  }
}
```

Open the **MainWindow.xaml** file and add a new namespace to the MainWindow (if not already there).

```
xmlns:convert="clr-
namespace:SimpleDataBindingSamples.Converters"
```

Add the following code to the MainWindow.Resources section.

```
<Window.Resources>
  <convert:InvertBooleanConverter x:Key="invertBoolean"
/>
</Window.Resources>
```

Add the following to main window.

```
<StackPanel>
  <RadioButton Content="Yes"
               IsChecked="{Binding Path=IsActive}" />
  <RadioButton Content="No"
               IsChecked="{Binding Path=IsActive,
Converter={StaticResource invertBoolean}}" />
</StackPanel>
```

Open the **MainWindow.xaml.cs** file and add a new public property.

```
public bool CanSellProduct { get; set; }
```

Modify the constructor to look like the following.

```
public RadioButtonBinding() {
  InitializeComponent();

  CanSellProduct = true;
  this.DataContext = this;
}
```

Try It Out

Run the application to view the results.

Now change the **"true"** to a **"false"** and run the view again to see the No radio button is now selected.