XAML Control Binding Lab - MAUI

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** .NET MAUI application named **XamlBindingMAUI**.

Open the **MainPage.xaml** file and replace the <VerticalStackLayout> with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        HorizontalOptions="Start"
        VerticalOptions="Start"
        MinimumHeightRequest="200"
        MinimumWidthRequest="300">
  <VerticalStackLayout Margin="10">
    <Label Text="Enter some Text Below" />
    <Entry x:Name="DataEntry" />
    <Label BindingContext="{x:Reference Name=DataEntry}"
           Text="{Binding Path=Text}" />
  </VerticalStackLayout>
</Border>
```

Try It Out

Run the application and type a value into the Entry control. What you type should be displayed in the Label.

Lab 2: Bind Slider to Label

The slider is used to illustrate a numeric scale.

> **NOTE:** Place the **Maximum** attribute first, then the minimum attribute after to avoid an error.

The **Value** property is a double data type, so if you want to increment by whole numbers, you must change the value in the code-behind.

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        HorizontalOptions="Start"
        VerticalOptions="Start"
        MinimumHeightRequest="200"
        MinimumWidthRequest="300">
  <VerticalStackLayout Margin="10">
    <!-- NOTE: Set the Maximum property before the
Minimum Property -->
    <Slider x:Name="slider"
            ValueChanged="slider_ValueChanged"
            Maximum="100"
            Minimum="1" />
    <Label BindingContext="{x:Reference Name=slider}"
            Text="{Binding Path=Value}" />
  </VerticalStackLayout>
</Border>
```

Write the ValueChanged event procedure.

```
private void slider_ValueChanged(object sender,
ValueChangedEventArgs e)
{
  // Only increment by whole numbers
  if (slider != null) {
    slider.Value = Convert.ToInt32(e.NewValue);
  }
}
```

Try It Out

Run the application and try moving the Slide thumb to the right and left to see the value displayed in the Label.

Lab 3: Bind Picker to Label

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        HorizontalOptions="Start"
        VerticalOptions="Start"
        MinimumHeightRequest="200"
        MinimumWidthRequest="300"
        Margin="10">
  <VerticalStackLayout Margin="10">
    <Picker x:Name="Language">
      <Picker.ItemsSource>
        <x:Array Type="{x:Type x:String}">
          <x:String>C#</x:String>
          <x:String>Visual Basic</x:String>
          <x:String>C++</x:String>
          <x:String>F#</x:String>
        </x:Array>
      </Picker.ItemsSource>
    </Picker>
    <Label Text="{Binding Path=SelectedItem}"
            BindingContext="{x:Reference Language}" />
  </VerticalStackLayout>
</Border>
```

Try It Out

Run the application and select a value from the Picker and you should see the value selected appear in the label.

Lab 4: Bind CheckBox IsEnabled Property

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        HorizontalOptions="Start"
        VerticalOptions="Start"
        MinimumHeightRequest="200"
        MinimumWidthRequest="300"
        Margin="10">
  <VerticalStackLayout Margin="10">
    <HorizontalStackLayout>
      <Label Text="Has Benefits?" />
      <CheckBox x:Name="hasBenefits" />
    </HorizontalStackLayout>
    <HorizontalStackLayout>
      <Label Text="401k" />
      <CheckBox IsEnabled="{Binding Path=IsChecked}"
                BindingContext="{x:Reference
hasBenefits}" />
    </HorizontalStackLayout>
    <HorizontalStackLayout>
      <Label Text="Health Care" />
      <CheckBox IsEnabled="{Binding Path=IsChecked}"
                BindingContext="{x:Reference
hasBenefits}" />
    </HorizontalStackLayout>
  </VerticalStackLayout>
</Border>
```

Try It Out

Run the application check and uncheck the Has Benefits? checkbox to see the other two checkboxes become enabled and disabled respectively.

Lab 5: Bind IsVisible Property

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        Margin="10">
  <Grid Margin="10"
        RowDefinitions="Auto,Auto,Auto"
        ColumnDefinitions="Auto,*">
    <Label Grid.Row="0"
           Text="Has Benefits?" />
    <CheckBox Grid.Row="0"
              Grid.Column="1"
              x:Name="HasBenefits" />
    <Label Grid.Row="1"
           Grid.Column="0"
           IsVisible="{Binding Path=IsChecked}"
           BindingContext="{x:Reference HasBenefits}"
           Text="401k" />
    <CheckBox Grid.Row="1"
              Grid.Column="1"
              IsVisible="{Binding Path=IsChecked}"
              BindingContext="{x:Reference HasBenefits}"
/>
    <Label Grid.Row="2"
           Grid.Column="0"
           IsVisible="{Binding Path=IsChecked}"
           BindingContext="{x:Reference HasBenefits}"
           Text="Health Care" />
    <CheckBox Grid.Row="2"
              Grid.Column="1"
              IsVisible="{Binding Path=IsChecked}"
              BindingContext="{x:Reference HasBenefits}"
/>
  </Grid>
</Border>
```

Try It Out

Run the application check and uncheck the Has Benefits? checkbox to see the other two checkboxes become visible and invisible respectively.

Lab 6: Stepper

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<HorizontalStackLayout VerticalOptions="Start">
  <Label Text="Your Age"
         VerticalTextAlignment="Center" />
  <Entry BindingContext="{x:Reference theStepper}"
         Text="{Binding Path=Value}" />
  <Stepper x:Name="theStepper"
           Minimum="18"
           Maximum="99"
           Increment="1" />
</HorizontalStackLayout>
```

Try It Out

Run the application and click the up and down arrows in the Stepper to see the value in the Entry control change.

Lab 7: Binding to Radio Buttons

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 **InvertBooleanConverter**. Replace all the template code in this file with the following code.

```
using System.Globalization;

namespace XamlBindingMAUI.Converters
{
  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 null;
    }
  }
}
```

Open the **MainPage.xaml.cs** file and add a property named IsActive.

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

Modify the constructor to set the BindingContext for the whole ContentPage to itself. Set the **IsActive** property to a true value.

```
public InvertBooleanPage()
{
  InitializeComponent();

  this.BindingContext = this;
  IsActive = true;
}
```

Open the **MainPage.xaml** file and add a new namespace.

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

Add the following code to the <ContentPage.Resources> section.

```
<ContentPage.Resources>
  <converters:InvertBooleanConverter
x:Key="invertBoolean" />
</ContentPage.Resources>
```

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<Border Stroke="Black"
        StrokeThickness="2"
        HorizontalOptions="Start"
        VerticalOptions="Start"
        MinimumHeightRequest="200"
        MinimumWidthRequest="300"
        Margin="10">
  <Grid Margin="10"
        RowDefinitions="Auto,Auto,Auto"
        ColumnDefinitions="Auto,*">
    <Label Grid.Row="0"
           Grid.Column="0"
           Text="Still Employed?" />
    <Label Grid.Row="1"
           Grid.Column="0"
           Text="Yes" />
    <RadioButton Grid.Row="1"
                 Grid.Column="1"
                 IsChecked="{Binding Path=IsActive}" />
    <Label Grid.Row="2"
           Grid.Column="0"
           Text="No" />
    <RadioButton Grid.Row="2"
                 Grid.Column="1"
                 IsChecked="{Binding Path=IsActive,
Converter={StaticResource invertBoolean}}" />
  </Grid>
</Border>
```

Try It Out

Run the application and you should see the Yes radio button is selected. This is because you set the IsActive property on the ContentPage to a true value. Stop the application and change the IsActive property to a **false** and run the view again to see the **No** radio button is now selected.

Lab 8: Switch

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<HorizontalStackLayout HorizontalOptions="Start"
                       VerticalOptions="Start">
  <Label Text="Employee Status"
         VerticalOptions="Center" />
  <Switch />
</HorizontalStackLayout>
```

Try It Out

Run the application on Windows and see that the <Switch> element has hard-coded labels **On** and **Off**.

Run the application on the Android Emulator and see that the <Switch> element does NOT have the hard-coded labels.

Add a Converter Class

Right mouse-click on the **Converters** folder and add a new class named **SwitchTextConverter**. Replace all the template code in this file with the following code.

```
using System.Globalization;

namespace XamlBindingMAUI.Converters
{
  public class SwitchTextConverter : IValueConverter
  {
    public string TrueText { get; set; } = "Yes";
    public string FalseText { get; set; } = "No";

    public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
    {
      if ((bool)value) {
        return TrueText;
      }
      else {
        return FalseText;
      }
    }

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

Replace the previous code you wrote in the **MainPage.xaml** file with the following XAML code.

```
<HorizontalStackLayout HorizontalOptions="Start"
                       VerticalOptions="Start">
  <Label Text="Employee Status"
         VerticalOptions="Center" />
  <Switch Margin="0,0,-5,0"
          x:Name="activeEmployee" />
  <Label Margin="-100,10,10,10"
         BackgroundColor="White">
    <Label.Text>
      <Binding Source="{x:Reference activeEmployee}"
               Path="IsToggled">
        <Binding.Converter>
          <converters:SwitchTextConverter
TrueText="Active"

FalseText="Inactive" />
        </Binding.Converter>
      </Binding>
    </Label.Text>
  </Label>
</HorizontalStackLayout>
```

Try It Out

Run the application and show the new text based on what you have entered in to the SwitchTextConverter **TrueText** and **FalseText** property values. Try different values in here to get a feel for how this converter works.