# Data Binding Basics in .NET MAUI Labs

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 Stepper to an Entry Control

Open the **Views\ProductDetailView.xaml** file.

Locate the <Entry> element below the <Label Text="**Cost**"> and replace it with the following XAML.

```
<HorizontalStackLayout Grid.Row="4"
                       Grid.Column="1">
  <Entry Text="{Binding Value}"
         BindingContext="{x:Reference CostStepper}" />
  <Stepper x:Name="CostStepper"
           Minimum="1"
           Maximum="9999"
           Increment="1" />
</HorizontalStackLayout>
```

Locate the <Entry> element below the <Label Text="**Price**"> and replace it with the following XAML.

```
<HorizontalStackLayout Grid.Row="5"
                       Grid.Column="1">
  <Entry Text="{Binding Value}"
         BindingContext="{x:Reference PriceStepper}" />
  <Stepper x:Name="PriceStepper"
           Minimum="1"
           Maximum="9999"
           Increment="1" />
</HorizontalStackLayout>
```

## Try It Out

Run the application and click on the **Products** menu.

Increment and decrement each stepper to see the value change in the corresponding Entry controls.

**Notice** that you are allowed to set the **Price less than the Cost**. This SHOULD NOT be allowed.

# Lab 2: Bind One Stepper to the Other

Open the **Views\ProductDetailView.xaml** file.

Locate the <Stepper x:Name="CostStepper" …> and modify it to look like the following.

**NOTES:**

Be sure to add the **Value** attribute so there is a valid starting value.

Make sure the lines in bold are in the exact order shown or you will receive an error.

```
<Stepper x:Name="CostStepper"
         Value="10"
         Minimum="1"
         Maximum="{Binding Value}"
         BindingContext="{x:Reference PriceStepper}"
         Increment="1" />
```

**NOTES:**

Be sure to add the **Value** attribute so there is a valid starting value.

Make sure the lines in **bold** are in the exact order shown or you will receive an error.

Locate the <Stepper x:Name="PriceStepper" …> and modify it to look like the following.

```
<Stepper x:Name="PriceStepper"
         Value="20"
         Minimum="{Binding Value}"
         Maximum="9999"
         BindingContext="{x:Reference CostStepper}"
         Increment="1" />
```

## Try It Out

Run the application and click on the **Products** menu.

Increment and decrement each stepper to see the value change in the corresponding Entry controls.

**Notice** that you can **no longer** make the Price less than the Cost, nor Cost greater than the Price.

# Lab 3: Disable One Control by Binding to IsEnabled on a CheckBox

Open the **Views\UserDetailView.xaml** file.

Locate the CheckBox under the <Label Text="**Flex Time?**"> and add an **x:Name** attribute.

```
<CheckBox x:Name="FlexTime" />
```

Locate the TimePicker and add an IsEnabled to bind to if the **IsChecked** property is true. You need to set the BindingContext attribute to refer to the FlexTime check box you just set the x:Name upon.

```
<TimePicker Grid.Row="9"
            Grid.Column="1"
            BindingContext="{x:Reference FlexTime}"
            IsEnabled="{Binding IsChecked}"
            Time="06:00:00" />
```

## Try It Out

Run the application and click on **Users | Navigate to Detail** and **check** the **Flex Time** check box to see the Start Time <TimePicker> become enabled.

# Lab 4: Use a ValueConverter to Change a Boolean Value

**Stop** the application.

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 **InvertedBoolConverter**.

```
using System.Globalization;

namespace AdventureWorks.MAUI.Converters;

public class InvertedBoolConverter : IValueConverter
{
  public object? Convert(object? value, Type targetType,
object? parameter, CultureInfo culture)
  {
    if (value != null) {
      return !(bool)value;
    }
    else {
      return null;
    }
  }

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

# Use the Converter Class

Open the **Views\UserDetailView.xaml** file.

Add an XML namespace to bring in the Converter class.

```
xmlns:converters="clr-
namespace:AdventureWorks.MAUI.Converters"
```

Add a **<ContentPage.Resources>** section and create an instance of the InvertedBoolConverter class.

```
<ContentPage.Resources>
  <converters:InvertedBoolConverter
x:Key="invertedBoolean" />
</ContentPage.Resources>
```

**Remove** the **x:Name="FlexTime"** from the <CheckBox>.

Locate the RadioButton under the <Label Text="**Full-Time**" /> and add an **x:Name** attribute.

```
<RadioButton x:Name="FullTime"
             IsChecked="True"
             GroupName="EmployeeType" />
```

Modify the <TimePicker> to use the converter.

```
<TimePicker Grid.Row="9"
            Grid.Column="1"
            BindingContext="{x:Reference FullTime}"
            IsEnabled="{Binding IsChecked,
Converter={StaticResource invertedBoolean}}"
            Time="06:00:00" />
```

# Try It Out

Run the application and click on **Users | Navigate to Detail** and notice the **Start Time** is disabled but check the **Part-Time** radio button and it becomes enabled.