XAML Object 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 to User Class

Re-open the **AdventureWorks.MAUI** solution you created earlier.

Right mouse-click on the Solution and add a **new Class Library** project named **AdventureWorks.EntityLayer**.

Rename Class1.cs to **User.cs** and replace the code with the following.

```
namespace AdventureWorks.EntityLayer;

public class User {
  public User() {
    LoginId = string.Empty;
    FirstName = string.Empty;
    LastName = string.Empty;
    Email = string.Empty;
    Password = string.Empty;
    Phone = string.Empty;
    PhoneType = string.Empty;
  }

  public int UserId { get; set; }
  public string LoginId { get; set; }
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public string Email { get; set; }
  public string Password { get; set; }
  public string Phone { get; set; }
  public string PhoneType { get; set; }
  public bool IsEnrolledIn401k { get; set; }
  public bool IsEnrolledInHealthCare { get; set; }
  public bool IsEnrolledInHSA { get; set; }
  public bool IsEnrolledInFlexTime { get; set; }
  public bool IsActive { get; set; }
  public DateTime BirthDate { get; set; }
}
```

Right mouse-click on the **AdventureWorks.MAUI** project's **Dependencies** folder and add a project dependency to the **AdventureWorks.EntityLayer** project.

Create User Class Using XAML

Open the **Views\UserDetailView.xaml** file and add a new XML namespace.

```
xmlns:vm="clr-
namespace:AdventureWorks.EntityLayer;assembly=AdventureW
orks.EntityLayer"
```

Create a <ContentPage.Resources> element and declare an instance of the User class.

```
<ContentPage.Resources>
  <vm:User x:Key="viewModel"
           LoginId="JohnSmith123"
           FirstName="John"
           LastName="Smith"
           Email="John@smith.com"
           Phone="615.222.2333"
           PhoneType="Mobile"
           IsEnrolledIn401k="True"
           IsEnrolledInFlexTime="True"
           IsEnrolledInHealthCare="True"
           IsEnrolledInHSA="False"
           IsActive="True"
           BirthDate="10-03-1975" />
</ContentPage.Resources>
```

Modify the <Border> element to have a BindingContext that points to the **viewModel** resource.

```
<Border Style="{StaticResource Screen.Border}"
        BindingContext="{StaticResource viewModel}">
```

Add {Binding Path=[property]} to each control.

NOTE:	Disregard the PasswordBox, RadioButton, and Picker controls for now.

Try It Out

Run the application and click on the **Users** menu to view the results.

Lab 2: Connect to User Object in Code Behind

Open the **Views\UserDetailView.xaml.cs** file and add a using statement.

```
using AdventureWorks.EntityLayer;
```

Add a public property of the data type User to the UserDetailView class.

```
public User UserObject { get; set; }
```

Modify the constructor to retrieve the object created by XAML and assign it to this new property.

```
public UserDetailView() {
  InitializeComponent();

  UserObject = (User)this.Resources["viewModel"];
}
```

Add a Click event procedure that you can hook up to the Save button on the UserDetailView view.

```
private void SaveButton_Click(object sender, EventArgs
e)
{
  System.Diagnostics.Debugger.Break();
}
```

Open the **Views\UserDetailView.xaml** file and modify the Save button.

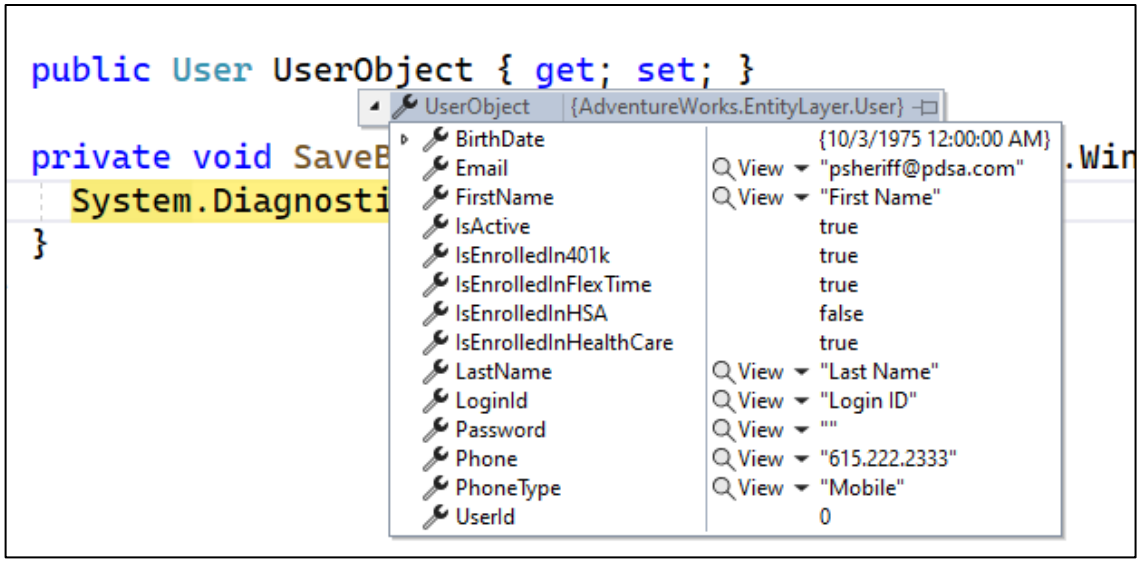
```
<Button Content="Save"
        Clicked="SaveButton_Click" />
```

Try It Out

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

Make some changes to some of the data in the User Information view and click on the **Save** button.

Hover over the **UserObject** variable and you should see the changes you made.



Lab 3: Try to Change Data in User Object

Open the **Views\UserDetailView.xaml.cs** file and try to change the LoginId property in the constructor. Add the code shown in bold below to the constructor.

```
public UserDetailView()
{
  InitializeComponent();

  UserObject = (User)this.Resources["viewModel"];
  UserObject.LoginId = "asdfasdfasfd";
}
```

Try It Out

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

Notice the LoginId **DID NOT** change.

Lab 4: Implement OnPropertyChanged

Right mouse-click on the Solution and select **Add | New Project...** from the menu.

Select **Class Library** from the project template list.

Set the Project Name to **Common.Library**.

Set the .NET version to **.NET 7**.

Rename Class1.cs to **CommonBase.cs**.

Replace the code in this file with the following.

```
using System.ComponentModel;

namespace Common.Library;

public abstract class CommonBase :
INotifyPropertyChanged {
  #region Constructor
  /// <summary>
  /// Constructor for CommonBase class
  /// </summary>
  public CommonBase() {
    Init();
  }
  #endregion

  #region Init Method
  /// <summary>
  /// Initialize any properties of this class
  /// </summary>
  public virtual void Init() {
  }
  #endregion

  #region RaisePropertyChanged Method
  /// <summary>
  /// Event used to raise changes to any bound UI
objects
  /// </summary>
  public event PropertyChangedEventHandler?
PropertyChanged;

  public virtual void RaisePropertyChanged(string
propertyName) {
    this.PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
  }
  #endregion
}
```

Right mouse-click on the **Common.Library** project and add a new class named **EntityBase**.

```
namespace Common.Library;

public class EntityBase : CommonBase
{
}
```

Set Dependencies

Right mouse-click on the **Dependencies** folder in the **AdventureWorks.EntityLayer** and add a project reference to the **Common.Library** project.

Right mouse-click on the **AdventureWorks.MAUI** project's **Dependencies** folder and add a project dependency to the **Common.Library** project.

Inherit from the EntityBase Class

Open the **User.cs** file and change it to inherit from the **EntityBase** class.

```
public class User : EntityBase
```

Replace the **LoginId** property definition with the following:

```
private string _LoginId;

public string LoginId
{
  get { return _LoginId; }
  set {
    _LoginId = value;
    RaisePropertyChanged(nameof(LoginId));
  }
}
```

In the constructor, replace ***LoginId = string.Empty;*** with the following.

```
_LoginId = string.Empty;
```

Try It Out

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

You should now see the **LoginId** value HAS changed.

Update the User Class

You should now make all the properties in the User class follow the same design pattern as the one you just created for the LoginId property.

Replace all the code in the **User.cs** file with the following code.

```
using Common.Library;

namespace AdventureWorks.EntityLayer;

public class User : EntityBase {
  #region Constructor
  public User() {
    _LoginId = string.Empty;
    _FirstName = string.Empty;
    _LastName = string.Empty;
    _Email = string.Empty;
    _Password = string.Empty;
    _Phone = string.Empty;
    _PhoneType = string.Empty;
  }
  #endregion

  #region Private Variables
  private int _UserId { get; set; }
  private string _LoginId;
  private string _FirstName { get; set; }
  private string _LastName { get; set; }
  private string _Email { get; set; }
  private string _Password { get; set; }
  private string _Phone { get; set; }
  private string _PhoneType { get; set; }
  private bool _IsEnrolledIn401k { get; set; }
  private bool _IsEnrolledInHealthCare { get; set; }
  private bool _IsEnrolledInHSA { get; set; }
  private bool _IsEnrolledInFlexTime { get; set; }
  private bool _IsActive { get; set; }
  private DateTime _BirthDate { get; set; }
  #endregion

  #region Public Properties
  public int UserId
  {
    get { return _UserId; }
    set {
      _UserId = value;
      RaisePropertyChanged(nameof(UserId));
    }
  }

  public string LoginId
  {
    get { return _LoginId; }
```

```
    set {
      _LoginId = value;
      RaisePropertyChanged(nameof(LoginId));
    }
  }

  public string FirstName
  {
    get { return _FirstName; }
    set {
      _FirstName = value;
      RaisePropertyChanged(nameof(FirstName));
    }
  }

  public string LastName
  {
    get { return _LastName; }
    set {
      _LastName = value;
      RaisePropertyChanged(nameof(LastName));
    }
  }

  public string Email
  {
    get { return _Email; }
    set {
      _Email = value;
      RaisePropertyChanged(nameof(Email));
    }
  }

  public string Password
  {
    get { return _Password; }
    set {
      _Password = value;
      RaisePropertyChanged(nameof(Password));
    }
  }

  public string Phone
  {
    get { return _Phone; }
    set {
      _Phone = value;
```

```
      RaisePropertyChanged(nameof(Phone));
    }
  }

  public string PhoneType
  {
    get { return _PhoneType; }
    set {
      _PhoneType = value;
      RaisePropertyChanged(nameof(PhoneType));
    }
  }

  public bool IsEnrolledIn401k
  {
    get { return _IsEnrolledIn401k; }
    set {
      _IsEnrolledIn401k = value;
      RaisePropertyChanged(nameof(IsEnrolledIn401k));
    }
  }

  public bool IsEnrolledInHealthCare
  {
    get { return _IsEnrolledInHealthCare; }
    set {
      _IsEnrolledInHealthCare = value;

RaisePropertyChanged(nameof(IsEnrolledInHealthCare));
    }
  }

  public bool IsEnrolledInHSA
  {
    get { return _IsEnrolledInHSA; }
    set {
      _IsEnrolledInHSA = value;
      RaisePropertyChanged(nameof(IsEnrolledInHSA));
    }
  }

  public bool IsEnrolledInFlexTime
  {
    get { return _IsEnrolledInFlexTime; }
    set {
      _IsEnrolledInFlexTime = value;
```

```
RaisePropertyChanged(nameof(IsEnrolledInFlexTime));
    }
  }

  public bool IsActive
  {
    get { return _IsActive; }
    set {
      _IsActive = value;
      RaisePropertyChanged(nameof(IsActive));
    }
  }

  public DateTime BirthDate
  {
    get { return _BirthDate; }
    set {
      _BirthDate = value;
      RaisePropertyChanged(nameof(BirthDate));
    }
  }
  #endregion
}
```

Lab 5: Bindable Properties

Remember the HeaderView user control you built? Let's replace the hard-coded strings with properties that you can bind to.

Open the **PartialViews\HeaderView.xaml.cs** file and add two bindable properties.

```
public string ViewTitle
{
 get { return (string)GetValue(ViewTitleProperty); }
 set { SetValue(ViewTitleProperty, value); }
}

public static readonly BindableProperty
ViewTitleProperty =
BindableProperty.Create("ViewTitle", typeof(string),
typeof(HeaderView), string.Empty);

public string ViewSubTitle
{
  get { return (string)GetValue(ViewSubTitleProperty); }
  set { SetValue(ViewSubTitleProperty, value); }
}

public static readonly BindableProperty
ViewSubTitleProperty =
BindableProperty.Create("ViewSubTitle", typeof(string),
typeof(HeaderView), string.Empty);
```

Modify the constructor to set the BindingContext of this ContentView to itself and to initialize these two bindable properties to default string values.

```
public HeaderView()
{
  InitializeComponent();

  this.BindingContext = this;
  ViewTitle = "View Title";
  ViewSubTitle = "View Sub Title";
}
```

Open the **PartialViews\HeaderView.xaml** file and replace the two **Text** properties in the two Label controls with the following.

```
<Label Text="{Binding Path=ViewTitle}"
        FontSize="24"
        Grid.Row="0" />
<Label Text="{Binding Path=ViewSubTitle}"
        FontSize="16"
        Grid.Row="1" />
```

Open the **Views\UserDetailView.xaml** file, locate the <partialViews:HeaderView> control, and add the **ViewTitle** and **ViewSubTitle** properties to this control.

```
<partialViews:HeaderView Grid.ColumnSpan="2"
      ViewTitle="User Detail"
      ViewSubTitle="Modify the user information." />
```

Open the **Views\LoginView.xaml** file, locate the <partialViews:HeaderView> control, and add the **ViewTitle** and **ViewSubTitle** properties to this control.

```
<partialViews:HeaderView Grid.ColumnSpan="2"
      ViewTitle="Login"
      ViewSubTitle="Enter your login credentials." />
```

Open the **Views\ProductDetailView.xaml** file, locate the <partialViews:HeaderView> control, and add the **ViewTitle** and **ViewSubTitle** properties to this control.

```
<partialViews:HeaderView
      ViewTitle="Product Detail"
      ViewSubTitle="Modify the product information." />
```

Try It Out

Run the application and click on each menu to see each page and ensure you can see the Title and Sub-title you set.

Lab 6: Binding to Picker

Open the **Views\UserDetailView.xaml** file and within the <ContentPage.Resources> element, add an array of strings that you are going to use to replace the hard-coded values in the <Picker> of phone types.

```
<x:Array x:Key="phoneTypes"
          Type="{x:Type x:String}">
  <x:String>Home</x:String>
  <x:String>Mobile</x:String>
  <x:String>Other</x:String>
</x:Array>
```

Scroll down and locate the <Picker> and make it look like the following.

```
<Picker Grid.Column="1"
        ItemsSource="{StaticResource phoneTypes}"
        SelectedItem="{Binding Path=PhoneType}" />
```

Try It Out

Run the application and click on the **User** menu to see the picker positioned to the correct value set in the User object above.

Lab 7: Bind Radio Buttons

Right mouse-click on the Solution and add a new **.NET MAUI Class Library** project named **Common.Library.MAUI**. Be sure to set the .NET version to **.NET 7**.

Delete the **Class1.cs** file.

Copy the **Converters** folder from the data binding project you created and place it into the **Common.Library.MAUI** project.

Open the **Converters\InvertBooleanConverter.cs** file and change the namespace from "XamlBindingMAUI.Converters" to "Common.Library.MAUI".

Modify User Detail View

Right mouse-click on the **AdventureWorks.MAUI** project and add a new project dependency to the **Common.Library.MAUI** project.

Open the **Views\UserDetailView.xaml** file and add a new XML namespace.

```
xmlns:commonMAUI="clr-
namespace:Common.Library.MAUI;assembly=Common.Library.MA
UI"
```

Add a new resource in the <ContentPage.Resources> element.

```
<commonMAUI:InvertBooleanConverter
   x:Key="invertBoolean" />
```

Locate the two radio buttons.

Modify them to look like the following.

```
<RadioButton GroupName="StillEmployed"
  IsChecked="{Binding Path=IsActive}" />

<RadioButton GroupName="StillEmployed"
  IsChecked="{Binding Path=IsActive,
Converter={StaticResource invertBoolean}}" />
```

Try It Out

Run the application and click on the **User** menu and see which radio button is checked.

You can modify the IsActive property in the <vm:User …> element to "False" to see the radio button check change.