Data Bind Classes to XAML 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: Create a XAML Array and Bind to Picker

Open the **App.xaml** file and just after the opening <ResourceDictionary> element, add an array of strings resource to replace the XAML in the <Picker> of phone types in the user detail page. You need to provide a **key** to the array to reference it from another XAML page.

```
<!-- Phone Types Resource -->
<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>
```

Open the **Views\UserDetailView.xaml** file and locate the <Picker> and make it look like the following.

```
<Picker ItemsSource="{StaticResource phoneTypes}" />
```

Try It Out

Run the application and click on **Users | Navigate to Detail** to see the picker loaded with phone type data.

Lab 2: Bind Controls on a View to Properties in a User Class

Stop the application.

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

Delete the **Class1.cs** file.

Right mouse-click on the **AdventureWorks.EntityLayer** project and add a new folder named **EntityClasses**.

Right mouse-click on the **EntityClasses** folder and add a new class named **User**. Replace the entire contents of this new file with the following code.

```
namespace AdventureWorks.EntityLayer;

public class User
{
  public int UserId { get; set; }
  public string LoginId { get; set; } = string.Empty;
  public string FirstName { get; set; } = string.Empty;
  public string LastName { get; set; } = string.Empty;
  public string Email { get; set; } = string.Empty;
  public string Password { get; set; } = string.Empty;
  public string Phone { get; set; } = string.Empty;
  public string PhoneType { get; set; } = string.Empty;
  public bool IsFullTime { 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; } =
DateTime.Now.AddYears(-18);
  public TimeSpan StartTime { get; set; } = new
TimeSpan(6, 0, 0);
}
```

Right mouse-click on the **AdventureWorks.MAUI** project's **Dependencies** folder and add a Project Reference 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"
```

Add an **x:DataType** attribute to the <ContentPage> element to make this page use compiled bindings. Compiled bindings improve the speed of data binding at runtime by resolving binding expressions at compile-time.

```
x:DataType="vm:User"
```

Within the **<ContentPage.Resources>** element declare an instance of the User class within the XAML.

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

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

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

Bind Entry Controls

Add a **{Binding PROPERTY_NAME}** to each <Entry> controls' **Text** property, for example:

```
<Entry Text="{Binding LoginId}" />
```

By adding the **x:DataType** to the <ContentPage>, the background compiler can provide Visual Studio with the list of property names from the User class.

Bind Check Box Controls

Locate each <CheckBox> element and set each ones' **IsChecked** property to the corresponding property in the User class.

```
<CheckBox IsChecked="{Binding IsEnrolledIn401k}" />
```

Bind the Switch Control

Locate the <Switch> control and set it's **IsToggled** property.

```
<Switch Grid.Row="6"
        Grid.Column="1"
        IsToggled="{Binding IsActive}" />
```

Bind Radio Button Controls

Locate the Radio Button under the <Label Text="**Full-Time**" /> element and set it to the following. **Remove** the **x:Name="FullTime"** attribute on this radio button.

```
<RadioButton IsChecked="{Binding IsFullTime}"
             GroupName="EmployeeType" />
```

Locate the Radio Button under the <Label Text="**Part-Time**" /> element and set it to the following.

```
<RadioButton IsChecked="{Binding IsFullTime,
Converter={StaticResource invertedBoolean}}"
             GroupName="EmployeeType" />
```

Bind the Date Picker Control

Locate the <DatePicker> element and bind the **Date** property to the **BirthDate**.

```
<DatePicker Grid.Row="8"
            Grid.Column="1"
            Date="{Binding BirthDate}"
            HorizontalOptions="Start" />
```

Bind the Time Picker Control

Replace the **IsEnabled** binding on the <TimePicker> to use a binding to the **IsFullTime** property in the User class.

Delete the BindingContext.

Set the **Time** property to bind to the **StartTime** property on the User class.

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

Picker

Modify the Picker for the phone types and add the **SelectedItem** attribute.

```
<Picker ItemsSource="{StaticResource phoneTypes}"
        SelectedItem="{Binding PhoneType}" />
```

Try It Out

Run the application and click on **Users | Navigate to Detail** to see the data from the User object appear in each of the controls.

NOTE:	The TimePicker becoming enabled and disabled will NOT work until we add some more code that you will learn about in the next lesson.

Lab 3: 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 ViewModel { get; set; }
```

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

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

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

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

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

Open the **Views\UserDetailView.xaml** file and add a **Clicked** event to the Save button.

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

Try It Out

Run the application and click on the **Users | Navigate to Detail** menu.

Make some changes to some of the data in the controls and click on the **Save** button.

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