XAML MVVM and DI 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: MVVM

Let's now create a view model class to retrieve data and feed it to the view.

Create a Base ViewModel Class

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

```
namespace Common.Library;

public class ViewModelBase : CommonBase
{
}
```

Create a View Model Layer Project

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

Add Dependencies to View Model Layer

Right mouse-click on the **Dependencies** folder in this new **AdventureWorks.ViewModelLayer** project and add a Project Reference to the **Common.Library** project and the **AdventureWorks.EntityLayer** project.

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

```
using AdventureWorks.EntityLayer;
using Common.Library;

namespace AdventureWorks.ViewModelLayer;

public class UserViewModel : ViewModelBase
{
  #region Private Variables
  private User? _UserObject = new();
  #endregion

  #region Public Properties
  public User? UserObject
  {
    get { return _UserObject; }
    set
    {
      _UserObject = value;
      RaisePropertyChanged(nameof(UserObject));
    }
  }
  #endregion

  #region Get(id) Method
  /// <summary>
  /// Get a single user object
  /// </summary>
  /// <param name="id">The UserId to locate</param>
  /// <returns>An instance of a User object</returns>
  public User? Get(int id)
  {
    try {
      // TODO: Get a User from a data store

      // MOCK Data
      UserObject = new User {
        UserId = id,
        LoginId = "SallyJones",
        FirstName = "Sally",
        LastName = "Jones",
        Email = "Sallyj@jones.com",
        Phone = "615.987.3456",
        PhoneType = "Mobile",
        IsEnrolledIn401k = true,
        IsEnrolledInFlexTime = false,
        IsEnrolledInHealthCare = true,
        IsEnrolledInHSA = false,
        IsActive = true,
        BirthDate = Convert.ToDateTime("08-13-1989")
      };
    }
    catch (Exception ex) {
      System.Diagnostics.Debug.WriteLine(ex.ToString());
    }

    return UserObject;
  }
  #endregion
}
```

Lab 2: Dependency Injection

Right mouse-click on the **Dependencies** folder in the **AdventureWorks.MAUI** project and add a Project Reference to the **AdventureWorks.ViewModelLayer** project.

Open the **MauiProgram.cs** file and just below the builder.UseMauiApp<App>() add the following code.

```
// Add services to be used in Dependency Injection
builder.Services.AddTransient<UserViewModel>();
builder.Services.AddTransient<UserDetailView>();
```

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

```
using AdventureWorks.ViewModelLayer;
```

Replace the public UserObject property with the following.

```
private readonly UserViewModel ViewModel;
```

Modify the constructor to look like the following.

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

  ViewModel = viewModel;
}
```

Add an override for the OnAppearing() event procedure.

```
protected override void OnAppearing()
{
  base.OnAppearing();

    BindingContext = ViewModel;

    ViewModel.Get(8);
}
```

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

Replace the **xmlns:vm** with the following.

```
xmlns:vm="clr-
namespace:AdventureWorks.ViewModelLayer;assembly=Adventu
reWorks.ViewModelLayer
```

Delete the **<vm:User …>** object declared in the <ContentPage.Resources> element.

Remove the **BindingContext="…"** from the <Border>

For each {Binding Path=FIELDNAME} you created, add a "UserObject." before each field name. For example.

```
{Binding Path=UserObject.LoginId}
```

Try It Out

Run the application and click on the **Users** menu to see the data appear.

Demo 3: Commanding

Open the **UserViewModel.cs** file and add a new Save() method.

```
#region Save Method
public virtual bool Save()
{
  // TODO: Write code to save data
  System.Diagnostics.Debugger.Break();

  return true;
}
#endregion
```

Add a Commanding View Model Class

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

Right mouse-click on the **ViewModelsCommanding** folder and add a new class named **UserViewModelCommanding**.

Replace the entire contents of this new file with the following code.

```
using AdventureWorks.ViewModelLayer;
using System.Windows.Input;

namespace AdventureWorks.MAUI.ViewModelsCommanding;

public class UserViewModelCommanding : UserViewModel
{
  public UserViewModelCommanding()
  {
    SaveCommand = new Command(() => Save(), () => true);
  }

  public ICommand SaveCommand { get; private set; }
}
```

Open the **Views\UserDetailView.xaml** file and change the "vm" XML namespace to reference the new namespace.

```
xmlns:vm="clr-
namespace:AdventureWorks.MAUI.ViewModelsCommanding"
```

Modify the Save button to look like the following. You are removing the **Clicked** event call and replacing it with the **Command** attribute.

```
<Button Text="Save"
        Command="{Binding Path=SaveCommand}" />
```

Change the View Model Reference

Open the **MauiProgram.cs** file and change the dependency injection to use this new view model.

```
builder.Services.AddTransient<UserViewModelCommanding>()
;
```

Open the **Views\UserDetailView.xaml.cs** file and **remove** the **SaveButton_Click**() event procedure.

Change all references to the **UserViewModel** class to the **UserViewModelCommanding** class.

Try It Out

Run the application, click on the **Users** menu, then click on the **Save** button and you should end up on the break in the Save() method.