Implement INotifyPropertyChanged to Update UI 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: 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();

  ViewModel = (User)this.Resources["viewModel"];
  ViewModel.LoginId = "ANewLoginId123";
}
```

Try It Out

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

Notice the LoginId **DOES NOT** have the value you put into the constructor.

Lab 2: Implement OnPropertyChanged

Stop the application.

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**.

Delete the **Class1.cs** file.

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

Right mouse-click on the **BaseClasses** folder and create a class named **CommonBase**. Replace the entire code in this new file with the following code.

```
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 **BaseClasses** folder in the **Common.Library** project and add a new class named **EntityBase**. Replace the entire contents of the new file with the following code.

```
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 **EntityClasses\User.cs** file and add a using statement.

```
using Common.Library;
```

Change the class declaration so the User class inherits from the **EntityBase** class.

```
public class User : EntityBase
```

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

```
private string _LoginId = string.Empty;

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

Try It Out

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

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

Lab 3: Update the User Class to Implement INotifyPropertyChanged

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.

Open the **EntityClasses\User.cs** file and replace the entire contents with the following code.

```
using Common.Library;

namespace AdventureWorks.EntityLayer;

public class User : EntityBase
{
  #region Private Variables
  private int _UserId;
  private string _LoginId = string.Empty;
  private string _FirstName = string.Empty;
  private string _LastName = string.Empty;
  private string _Email = string.Empty;
  private string _Password = string.Empty;
  private string _Phone = string.Empty;
  private string _PhoneType = string.Empty;
  private bool _IsFullTime;
  private bool _IsEnrolledIn401k;
  private bool _IsEnrolledInHealthCare;
  private bool _IsEnrolledInHSA;
  private bool _IsEnrolledInFlexTime;
  private bool _IsActive;
  private DateTime _BirthDate = DateTime.Now.AddYears(-
18);
  private TimeSpan? _StartTime = new(6, 0, 0);
  #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 IsFullTime
  {
    get { return _IsFullTime; }
    set
    {
      _IsFullTime = value;
      RaisePropertyChanged(nameof(IsFullTime));
    }
  }

  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));
    }
  }

  public TimeSpan? StartTime
  {
    get { return _StartTime; }
    set
    {
      _StartTime = value;
      RaisePropertyChanged(nameof(StartTime));
    }
  }
```

```
  public string FullName
  {
    get { return FirstName + " " + LastName; }
  }

  public string LastNameFirstName
  {
    get { return LastName + ", " + FirstName; }
  }
  #endregion
}
```

Use the OnAppearing() Method

Open the **Views\UserDetailView.xaml.cs** file and remove the line of code that changes the **LoginId** in the constructor.

Add the following code to override the OnAppearing() method.

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

  ViewModel.LoginId = "PeterPiper384";
  ViewModel.FirstName = "Peter";
  ViewModel.LastName = "Piper";
  ViewModel.Email = "Peter@pipering.com";
}
```

Try It Out

Run the application and click on **Users | Navigate to Detail** to see the changes on the UI that you made to the properties in the OnAppearing() event.