Displaying Informational and Error Messages 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: Add Two New Properties to View Model Base Class

Open the **BaseClasses\ViewModelBase.cs** file in the Common.Library project and add two new private variables.

```
private bool _IsDataProcessing;
private bool _ExceptionHappened;
```

Add two new public properties.

```
/// <summary>
/// Get/Set if data is being processed
/// </summary>
public bool IsDataProcessing
{
  get { return _IsDataProcessing; }
  set
  {
    _IsDataProcessing = value;
    RaisePropertyChanged(nameof(IsDataProcessing));
  }
}

/// <summary>
/// Get/Set whether or not the page had exception occur
/// </summary>
public bool ExceptionHappened
{
  get { return _ExceptionHappened; }
  set
  {
    _ExceptionHappened = value;
    RaisePropertyChanged(nameof(ExceptionHappened));
  }
}
```

Add two new methods.

```
#region BeginProcessing Method
protected virtual void BeginProcessing()
{
  InfoMessage = string.Empty;
  LastErrorMessage = string.Empty;
  LastException = null;
  RowsAffected = 0;
  IsDataProcessing = true;
  ExceptionHappened = false;
}
#endregion

#region EndProcessing Methods
protected virtual void EndProcessing()
{
  IsDataProcessing = false;
  if (!string.IsNullOrEmpty(LastErrorMessage) ||
      LastException != null) {
    ExceptionHappened = true;
  }
}
#endregion
```

Lab 2: Modify View Model Classes

Open the **ViewModelClasses\UserViewModel.cs** file.

In the **GetAsync()** method, remove the line of code **RowsAffected = 0;** at the beginning of the method.

Call the **BeginProcessing()** method just before the **try** block.

Call the **EndProcessing()** method just after the closing brace for the **catch** block.

In the **GetAsync(id)** method, call the **BeginProcessing()** method just before the **try** block.

Call the **EndProcessing()** method just after the closing brace for the **catch** block.

Within the **catch** block, remove the **RowsAffected = 0;** line of code.

Modify Product View Model

Open the **ViewModelClasses\ProductViewModel.cs** file.

In the **GetAsync()** method, remove the line of code **RowsAffected = 0;** at the beginning of the method.

Call the **BeginProcessing()** method just before the **try** block.

Call the **EndProcessing()** method just after the closing brace for the **catch** block.

In the **GetAsync(id)** method, call the **BeginProcessing()** method just before the **try** block.

Call the **EndProcessing()** method just after the closing brace for the **catch** block.

Within the **catch** block, remove the **RowsAffected = 0;** line of code.

Lab 3: Modify the List Views to Display Informational Messages

Open the **Views\UserListView.xaml** file and add a new row definition as shown below.

```
<Grid.RowDefinitions>
  <RowDefinition Height="Auto" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
</Grid.RowDefinitions>
```

In between the <partial:HeaderView …> and the <ListView> add a new <VerticalStackLayout>.

```
<VerticalStackLayout Grid.Row="1">
  <!-- ************************** -->
  <!-- Informational Message Area -->
  <!-- ************************** -->
  <Label Margin="20,4" />
  <Label Text="{Binding Path=InfoMessage}"
         Style="{StaticResource InfoMessageArea}" />
</VerticalStackLayout>
```

Set the **Grid.Row** property on the <ListView> to "2".

```
<ListView Grid.Row="2" … >
```

Open the **Resources\Styles\AppStyles.xaml** file and add a new **keyed** style.

```
<Style TargetType="Label"
       x:Key="InfoMessageArea">
  <Setter Property="FontSize"
          Value="18" />
  <Setter Property="Margin"
          Value="4,4" />
</Style>
```

Modify the Product List View

Open the **Views\ProductListView.xaml** file and add a new row definition as shown below.

```
<Grid.RowDefinitions>
  <RowDefinition Height="Auto" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
</Grid.RowDefinitions>
```

In between the <partial:HeaderView …> and the <CollectionView> add a new <VerticalStackLayout>.

```
<VerticalStackLayout Grid.Row="1">
  <!-- ************************** -->
  <!-- Informational Message Area -->
  <!-- ************************** -->
  <Label Margin="20,4" />
  <Label Text="{Binding Path=InfoMessage}"
         Style="{StaticResource InfoMessageArea}" />
</VerticalStackLayout>
```

Set the **Grid.Row** property on the <CollectionView> to "2".

```
<CollectionView Grid.Row="2" … >
```

Try It Out

Run the application, click on the **Users** menu and you should see informational message "Found *31* Users" displayed. You can click on the **Products** menu and see the same informational message with a different number of products.

Lab 4: Display Processing Messages

Open the **ViewModelClasses\UserViewModel.cs** file and add in the GetAsync() method, just before the call to **await Repository.GetAsync()**, add the following informational message.

```
InfoMessage = "Please wait while loading users...";
```

To simulate this process taking a long time, temporarily add an additional line just **below** this message.

```
await Task.Run(() => Thread.Sleep(2000));
```

Open the **ViewModelClasses\ProductViewModel.cs** file and add in the GetAsync() method, just before the await Repository.GetAsync() call, add the following informational message.

```
InfoMessage = "Please wait while loading products...";
```

Try It Out

Run the application, click on the **Users** menu and you should see informational message "Please wait while loading users…" displayed.

Remove the **Thread.Sleep()** call.

Lab 5: Modify List Views to Display Error Messages

Open the **Views\UserListView.xaml** file and add a new row definition as shown below.

```
<Grid.RowDefinitions>
  <RowDefinition Height="Auto" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
  <RowDefinition Height="Auto" />
</Grid.RowDefinitions>
```

Scroll down to the closing </ListView> element and insert the following XAML before the closing </Grid> element.

```
<!-- ****************** -->
<!-- Error Message Area -->
<!-- ****************** -->
<Label Grid.Row="3"
       IsVisible="{Binding ExceptionHappened}"
       Text="{Binding Path=LastErrorMessage}"
       Style="{StaticResource ErrorMessage}" />
```

Open the **Resources\Styles\AppStyles.xaml** file and add a new keyed style.

```
<Style TargetType="Label"
       x:Key="ErrorMessage">
  <Setter Property="FontAttributes"
          Value="Bold" />
  <Setter Property="TextColor"
          Value="Red" />
</Style>
```

Modify the Product List View

Open the **Views\ProductListView.xaml** file and add a new row definition as shown below.

```
<Grid.RowDefinitions>
  <RowDefinition Height="Auto" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
  <RowDefinition Height="Auto" />
</Grid.RowDefinitions>
```

Scroll down to the closing </ListView> element and insert the following XAML before the closing </Grid> element.

```
<!-- ***************** -->
<!-- Error Message Area -->
<!-- ***************** -->
<Label Grid.Row="3"
       IsVisible="{Binding ExceptionHappened}"
       Text="{Binding Path=LastErrorMessage}"
       Style="{StaticResource ErrorMessage}" />
```

Try It Out

Open the **MauiProgram.cs** file and comment out the following line.

```
//builder.Services.AddScoped<IRepository<User>,
UserRepository>();
```

Run the application and click on the **Users** menu and you should see the **error message** that the repository object is not set.

Open the **MauiProgram.cs** file and **UNCOMMENT** the line you previously commented.

Lab 6: Display Error Messages on Detail Views

Open the **Views\UserDetailView.xaml** file and add a new **Auto** to the RowDefinition attribute.

Scroll down to the bottom of the file and locate the last closing </HorizontalStackLayout> and insert the following XAML in between this element and the closing </Grid> element.

```
<!-- ***************** -->
<!-- Error Message Area -->
<!-- ***************** -->
<Label Grid.Row="12"
       IsVisible="{Binding ExceptionHappened}"
       Text="{Binding Path=LastErrorMessage}"
       Style="{StaticResource ErrorMessage}" />
```

Modify Product Detail View

Open the **Views\ProductDetailView.xaml** file and add a new **Auto** to the RowDefinition attribute.

Scroll down to the bottom of the file and locate the last closing </HorizontalStackLayout> and insert the following XAML in between this element and the closing </Grid> element.

```
<!-- ***************** -->
<!-- Error Message Area -->
<!-- ***************** -->
<Label Grid.Row="14"
       IsVisible="{Binding ExceptionHappened}"
       Text="{Binding Path=LastErrorMessage}"
       Style="{StaticResource ErrorMessage}" />
```