XAML Styles Lab - WPF

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: Hard Coded Styles

Use the **AdventureWorks.WPF** sample solution.

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

Add **Margin="5"** to all controls including the stack panel.

Try it Out

Run the application and click on the **Login** menu to see the results.

While the application is still running, go back and remove the margin from the stack panel.

Lab 2: Move to UserControl.Resources

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

Remove all the **Margin** attributes you added.

Just before the <Grid> element, add the following code.

```
<UserControl.Resources>
  <Style TargetType="Label">
    <Setter Property="Margin"
            Value="5" />
  </Style>
  <Style TargetType="TextBox">
    <Setter Property="Margin"
            Value="5" />
  </Style>
  <Style TargetType="PasswordBox">
    <Setter Property="Margin"
            Value="5" />
  </Style>
  <Style TargetType="Button">
    <Setter Property="Margin"
            Value="5" />
  </Style>
</UserControl.Resources>
```

Try it Out

Run the application and click on the **Login** menu and notice the spacing is still there.

Click on the **Users** or **Products** menu and notice there is no spacing on these.

Lab 3: Move to Application.Resources

Open the **App.xaml** file.

Cut the <Style> elements from the <UserControl.Resources> on the LoginView view and move them within the <Application.Resources> element.

Add two more styles.

```
<Style TargetType="TextBlock">
  <Setter Property="Margin"
          Value="5" />
</Style>
<Style TargetType="Image">
  <Setter Property="Margin"
          Value="5" />
</Style>
```

Try it Out

Run the application and any of the menus and notice the spacing is now applied to all controls.

Lab 4: Closest Style Wins

Open the **Views\LoginView.xaml** file and add the following code within the <UserControl.Resources> element.

```
<Style TargetType="Button">
  <Setter Property="Margin"
          Value="15" />
</Style>
```

Try it Out

Run the application and click on the **Login** menu and notice these buttons now have a larger spacing around them than on the other views.

Lab 5: Using Keyed Styles

Open the **App.xaml** file and add a new keyed style.

```
<Style TargetType="Button"
        x:Key="DefaultButtonStyle">
  <Setter Property="Margin"
          Value="5" />
  <Setter Property="Padding"
          Value="5" />
  <Setter Property="FontWeight"
          Value="Bold" />
</Style>
```

Open the **Views\LoginView.xaml** file and remove the complete <UserControl.Resources> element.

Add a **Style** attribute to the Login button.

```
<Button Content="Login"
   Style="{StaticResource DefaultButtonStyle}" />
```

Try it Out

Run the application and click on the **Login** menu to see the results.

Lab 6: Using BasedOn

Open the **App.xaml** file and add two new keyed styles before all the other styles.

```
<Style TargetType="Control"
       x:Key="BaseControl">
  <Setter Property="Margin"
          Value="5" />
</Style>
<Style TargetType="FrameworkElement"
       x:Key="BaseFrameworkElement">
  <Setter Property="Margin"
          Value="5" />
</Style>
```

Modify all the other <Style> elements below these to look like the following.

```
<Style TargetType="Label"
        BasedOn="{StaticResource BaseControl}">
</Style>
<Style TargetType="TextBox"
        BasedOn="{StaticResource BaseControl}">
</Style>
<Style TargetType="PasswordBox"
        BasedOn="{StaticResource BaseControl}">
</Style>
<Style TargetType="Button"
        BasedOn="{StaticResource BaseControl}">
</Style>
<Style TargetType="TextBlock"
        BasedOn="{StaticResource BaseFrameworkElement}">
</Style>
<Style TargetType="Image"
        BasedOn="{StaticResource BaseFrameworkElement}">
</Style>
<Style TargetType="Button"
        BasedOn="{StaticResource BaseControl}"
        x:Key="DefaultButtonStyle">
  <Setter Property="Padding"
          Value="5" />
  <Setter Property="FontWeight"
          Value="Bold" />
</Style>
```

This reduces the number of duplicate <Setter> elements and provides consistency throughout your application.

Try it Out

Run the application to ensure everything still looks the same.

Lab 7: Resource Dictionaries

You might want to have multiple sets of styles that you can swap in and out.

Best to move the resources into a ResourceDictionary in a separate folder.

Right mouse-click on the project and add a new folder named **Resources**.

Right mouse-click on the **Resources** folder and add a new folder named **Styles**.

Right mouse-click on the **Resources\Styles** folder and select **Add | Resource Dictionary (WPF) …** from the menu.

Set the name to **StandardStyles.xaml**.

Open the **App.xaml** file and cut all the styles out of the <Application.Resources> element and paste them into the new <ResourceDictionary> element in the StandardStyles file.

Back in the **App.xaml** file, make the <Application.Resources> element look like the following.

```
<Application.Resources>
  <ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
      <ResourceDictionary
Source="/Resources/Styles/StandardStyles.xaml" />
    </ResourceDictionary.MergedDictionaries>
  </ResourceDictionary>
</Application.Resources>
```

Try it Out

Run the application to ensure everything still looks the same.

Lab 8: Add Standard Colors

Try to keep colors out of your main style dictionaries.

Colors should be put into a different dictionary.

Right mouse-click on the **Resources\Styles** folder and select **Add | Resource Dictionary (WPF) …** from the menu.

Set the name to **StandardColors.xaml**.

Add the following colors into this new resource dictionary.

```
<Color x:Key="Primary">#512BD4</Color>
<Color x:Key="Secondary">#DFD8F7</Color>
<Color x:Key="Tertiary">#2B0B98</Color>
<Color x:Key="Success">Blue</Color>
<Color x:Key="Danger">Red</Color>
<Color x:Key="Info">Yellow</Color>

<SolidColorBrush x:Key="PrimaryBrush"
                  Color="{StaticResource Primary}" />
<SolidColorBrush x:Key="SecondaryBrush"
                  Color="{StaticResource Secondary}" />
<SolidColorBrush x:Key="TertiaryBrush"
                  Color="{StaticResource Tertiary}" />
<SolidColorBrush x:Key="SuccessBrush"
                  Color="{StaticResource Success}" />
<SolidColorBrush x:Key="DangerBrush"
                  Color="{StaticResource Danger}" />
<SolidColorBrush x:Key="InfoBrush"
                  Color="{StaticResource Info}" />
```

Copy the **Resources\Styles\StandardStyles.xaml** file to a new file within the same folder named **StandardStylesWithColors.xaml**.

Open the **StandardStylesWithColors.xaml** file.

Modify the Label style.

```
<Style TargetType="Label"
       BasedOn="{StaticResource BaseControl}">
  <Setter Property="Foreground"
          Value="{StaticResource PrimaryBrush}" />
</Style>
```

Modify the Button style.

```
<Style TargetType="Button"
        BasedOn="{StaticResource BaseControl}">
  <Setter Property="Background"
          Value="{StaticResource PrimaryBrush}" />
  <Setter Property="Foreground"
          Value="{StaticResource InfoBrush}" />
</Style>
```

Modify keyed style named **DefaultButtonStyle**.

```
<Style TargetType="Button"
        BasedOn="{StaticResource BaseControl}"
        x:Key="DefaultButtonStyle">
  <Setter Property="Background"
          Value="{StaticResource PrimaryBrush}" />
  <Setter Property="Foreground"
          Value="{StaticResource InfoBrush}" />
  <Setter Property="Padding"
          Value="5" />
  <Setter Property="FontWeight"
          Value="Bold" />
</Style>
```

Open the **App.xaml** file and modify the entries

```
<ResourceDictionary>
  <ResourceDictionary.MergedDictionaries>
    <ResourceDictionary
Source="/Resources/Styles/StandardColors.xaml" />
    <ResourceDictionary
Source="/Resources/Styles/StandardStylesWithColors.xaml"
/>
  </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
```

Try it Out

Run the application and you should see the new colors appear on all label and button controls.

Lab 9: Add Alternate Colors

Right mouse-click on the **Resources\Styles** folder and select **Add | Resource Dictionary (WPF) …** from the menu.

Set the name to **AlternateColors.xaml**.

Add the following colors into this new resource dictionary.

```
<Color x:Key="Primary">Green</Color>
<Color x:Key="Secondary">Red</Color>
<Color x:Key="Tertiary">Black</Color>
<Color x:Key="Success">Blue</Color>
<Color x:Key="Danger">Red</Color>
<Color x:Key="Info">Yellow</Color>

<SolidColorBrush x:Key="PrimaryBrush"
                 Color="{StaticResource Primary}" />
<SolidColorBrush x:Key="SecondaryBrush"
                 Color="{StaticResource Secondary}" />
<SolidColorBrush x:Key="TertiaryBrush"
                 Color="{StaticResource Tertiary}" />
<SolidColorBrush x:Key="SuccessBrush"
                 Color="{StaticResource Success}" />
<SolidColorBrush x:Key="DangerBrush"
                 Color="{StaticResource Danger}" />
<SolidColorBrush x:Key="InfoBrush"
                 Color="{StaticResource Info}" />
```

Open the **App.xaml** file and change the <ResouceDictionary> that references the **StandardColors.xaml** file to the **AlternateColors.xaml** file.

Try it Out

Run the application and you should see the alternate colors appear on all label and button controls.

Lab 10: Other Kinds of Resources

You saw colors as a resource.

You can also have strings and numbers as a resource.

Open the **Resources\Styles\StandardStyles.xaml** file and add a new XML namespace to the <ResourceDictionary>.

```
xmlns:sys="clr-namespace:System;assembly=mscorlib"
```

Before all other styles, add the following resource.

```
<sys:Double x:Key="DefaultFontSize">16</sys:Double>
```

Add the following <Setter> to the keyed style named **BaseControl**.

```
<Setter Property="FontSize"
        Value="{StaticResource DefaultFontSize}" />
```

Change the **App.xaml** file back to just use the **StandardStyles.xaml** resource dictionary.

Try it Out

Run the application and you should see all labels are now much larger.

Add a String Resource

Just below the DefaultFontSize resource you just added, add a string resource.

```
<sys:String x:Key="ApplicationTitle">
  Adventure Works
</sys:String>
```

Open the **MainWindow.xaml** file and modify the Title property.

```
Title="{StaticResource ApplicationTitle}"
```

Open the **Views\HomeView.xaml** file and modify the <Label>.

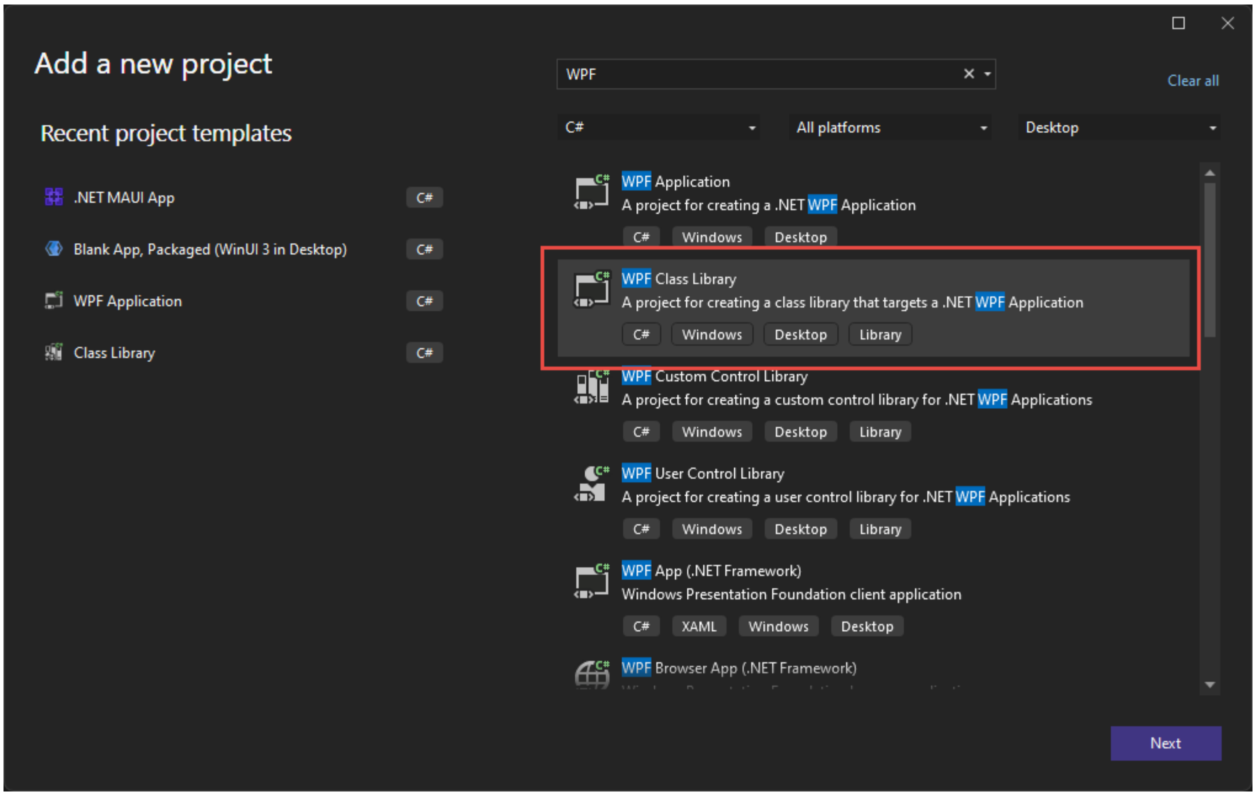
```
<Label FontSize="48"
       HorizontalAlignment="Center"
       Content="{StaticResource ApplicationTitle}" />
```

Try it Out

Run the application and you should see Adventure Works where you did before.

Lab 11: Resource Dictionaries from Library

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


Add a new project
Recent project templates
.NET MAUI App C#
Blank App, Packaged (WinUI 3 in Desktop) C#
WPF Application C#
Class Library C#
WPF
Clear all
C#
All platforms
Desktop
WPF Application
A project for creating a .NET WPF Application
C# Windows Desktop
WPF Class Library
A project for creating a class library that targets a .NET WPF Application
C# Windows Desktop Library
WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
C# Windows Desktop Library
WPF User Control Library
A project for creating a user control library for .NET WPF Applications
C# Windows Desktop Library
WPF App (.NET Framework)
Windows Presentation Foundation client application
C# XAML Windows Desktop
WPF Browser App (.NET Framework)
Next

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

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

Remove the **Class1.cs** file.

Move the **Resources** folder from the **AdventureWorks.WPF** project to the new class library project.

Right mouse-click on the **Dependencies** folder in the **AdventureWorksWPF** project and add a reference to the new class library project.

Open the **App.xaml** file and modify the Source attribute of the <ResourceDictionary> to the following.

```
<ResourceDictionary
Source="pack://application:,,,/Common.Library.WPF;compon
ent/Resources/Styles/StandardStyles.xaml" />
```

Try it Out

Run the application to ensure everything still looks the same.