Lesson Two
XAML Styles and Resource Dictionaries

Like good web site architecture, XAML separates design from programming. An important XAML skill to learn is creating UI styles. Like CSS for websites, XAML styles change the UI look and feel quickly.

Styles drive the look of your interface while Resource Dictionaries are repositories for common items used by your application. You may have one dictionary for labels and titles; another could store a collection of colors and fonts, and a third contains configuration information like device IP addresses, encrypted user names and passwords.

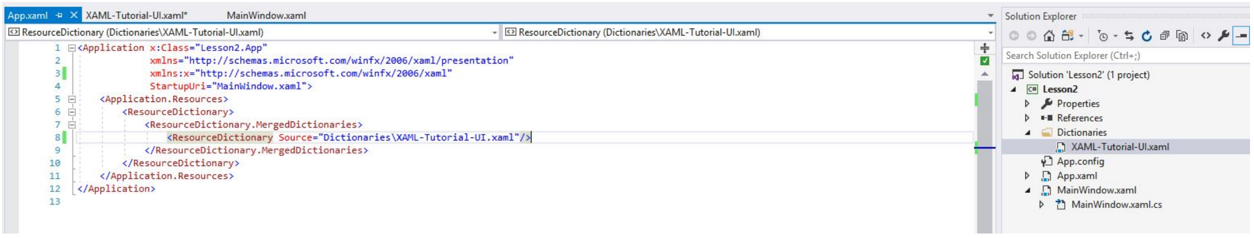
A resource dictionary can also hold UX styles for buttons, toggle switches, layout elements and more.

A good practice is to put dictionaries of similar items into their own folder. Even if you start with one dictionary, you'll find yourself adding more before the project is complete.

Adding a Resource Dictionary

Add a new folder by right-clicking on the Application Title (found below the Solution). Select Add > New Folder, then name your folder Dictionaries. Right-click on your Dictionaries folder and select Add > Resource Dictionary and name it: XAML-Tutorial-UI.xaml.

We'll be adding styles to the UI resource dictionary as we go, but first the application needs to be told to use the dictionary and where to find it. Open App.xaml in the Solution Explorer and insert the ResourceDictionary lines in the Application.Resources section. If you want to include more Dictionaries, just add a new <ResourceDictionary Source="…..xaml"/> tag.


App.xaml
XAML-Tutorial-UI.xaml*
MainWindow.xaml
ResourceDictionary (Dictionaries\XAML-Tutorial-UI.xaml)
ResourceDictionary (Dictionaries\XAML-Tutorial-UI.xaml)
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Lesson2' (1 project)
Lesson2
Properties
References
Dictionaries
XAML-Tutorial-UI.xaml
App.config
App.xaml
MainWindow.xaml
MainWindow.xaml.cs

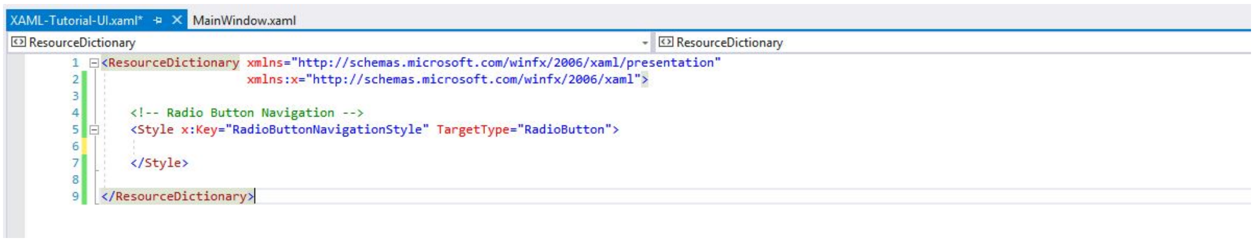
```
<Application x:Class="Lesson2.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Dictionaries\XAML-Tutorial-UI.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

When done adding the Resource Dictionary definition, save and close App.xaml.

Creating a Style

This radio button has three major elements laid out horizontally: a round button, which will change color based on whether it's turned off or on, a button icon and a button title.

Start by adding the Style tag and give the style a key which will be used later. The target type is the kind of control this style will define, in this case a radio button.



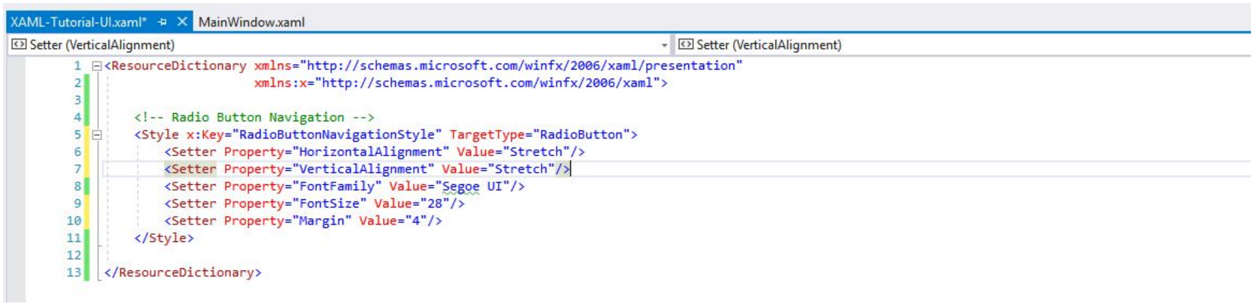
```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!-- Radio Button Navigation -->
    <Style x:Key="RadioButtonNavigationStyle" TargetType="RadioButton">

    </Style>

</ResourceDictionary>
```

Next, add the Setter tags to set certain properties such as: alignment, font size, and margins to our radio buttons.



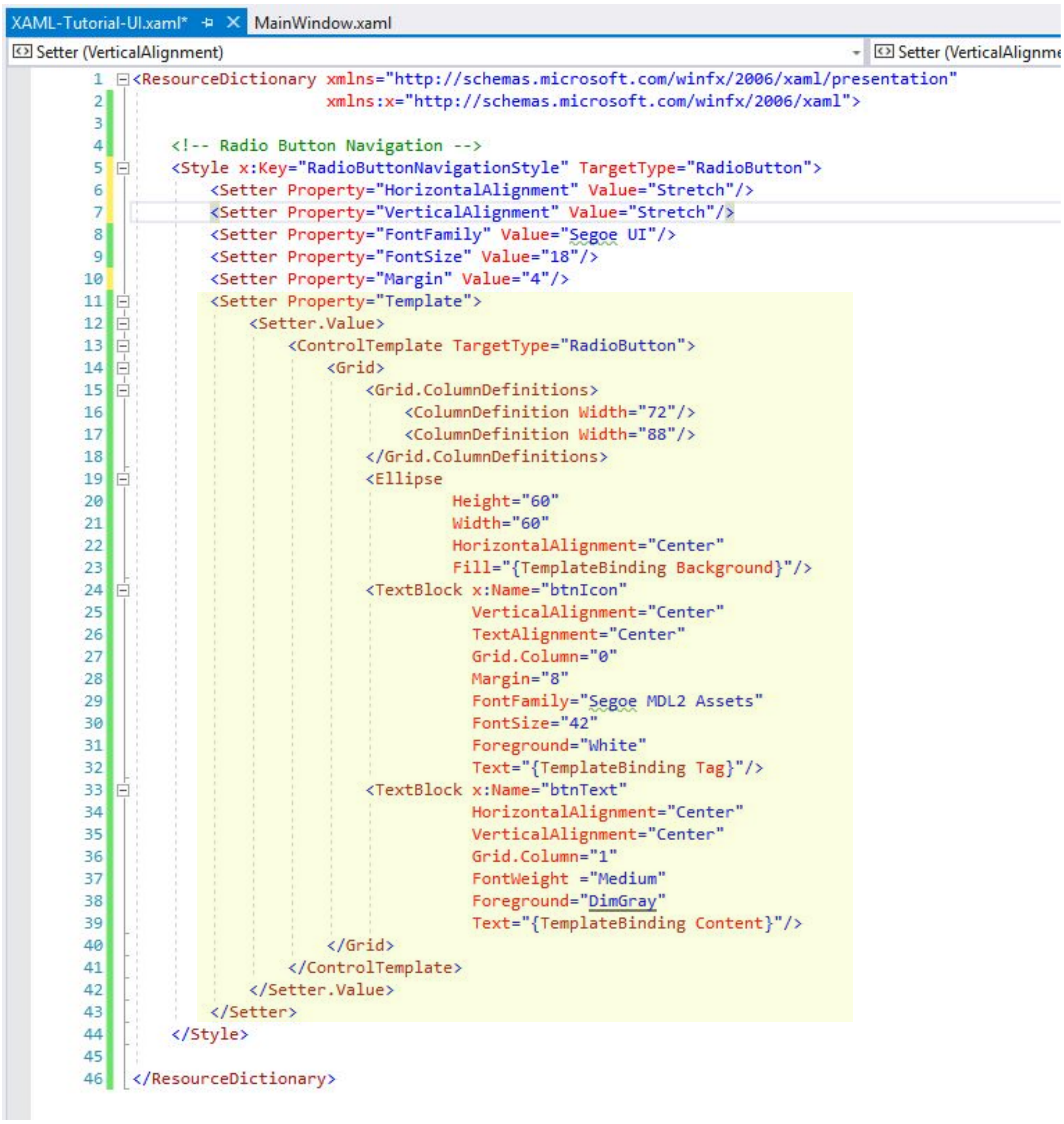
```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!-- Radio Button Navigation -->
    <Style x:Key="RadioButtonNavigationStyle" TargetType="RadioButton">
        <Setter Property="HorizontalAlignment" Value="Stretch"/>
        <Setter Property="VerticalAlignment" Value="Stretch"/>
        <Setter Property="FontFamily" Value="Segoe UI"/>
        <Setter Property="FontSize" Value="28"/>
        <Setter Property="Margin" Value="4"/>
    </Style>

</ResourceDictionary>
```

Using Template and Control Template

One powerful Setter tag is Template. Its Control Template value drives the radio button's layout. This Control Template defines a two-column, single row grid, an ellipse, a text block for the icon, and a text block for the button title.



```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!-- Radio Button Navigation -->
    <Style x:Key="RadioButtonNavigationStyle" TargetType="RadioButton">
        <Setter Property="HorizontalAlignment" Value="Stretch"/>
        <Setter Property="VerticalAlignment" Value="Stretch"/>
        <Setter Property="FontFamily" Value="Segoe UI"/>
        <Setter Property="FontSize" Value="18"/>
        <Setter Property="Margin" Value="4"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="RadioButton">
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="72"/>
                            <ColumnDefinition Width="88"/>
                        </Grid.ColumnDefinitions>
                        <Ellipse
                                 Height="60"
                                 Width="60"
                                 HorizontalAlignment="Center"
                                 Fill="{TemplateBinding Background}"/>
                        <TextBlock x:Name="btnIcon"
                                   VerticalAlignment="Center"
                                   TextAlignment="Center"
                                   Grid.Column="0"
                                   Margin="8"
                                   FontFamily="Segoe MDL2 Assets"
                                   FontSize="42"
                                   Foreground="White"
                                   Text="{TemplateBinding Tag}"/>
                        <TextBlock x:Name="btnText"
                                   HorizontalAlignment="Center"
                                   VerticalAlignment="Center"
                                   Grid.Column="1"
                                   FontWeight ="Medium"
                                   Foreground="DimGray"
                                   Text="{TemplateBinding Content}"/>
                    </Grid>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

</ResourceDictionary>
```

Looking closer, the Control Template defines RadioButton as the TargetType; in other words the new Style will have all the attributes of a RadioButton.

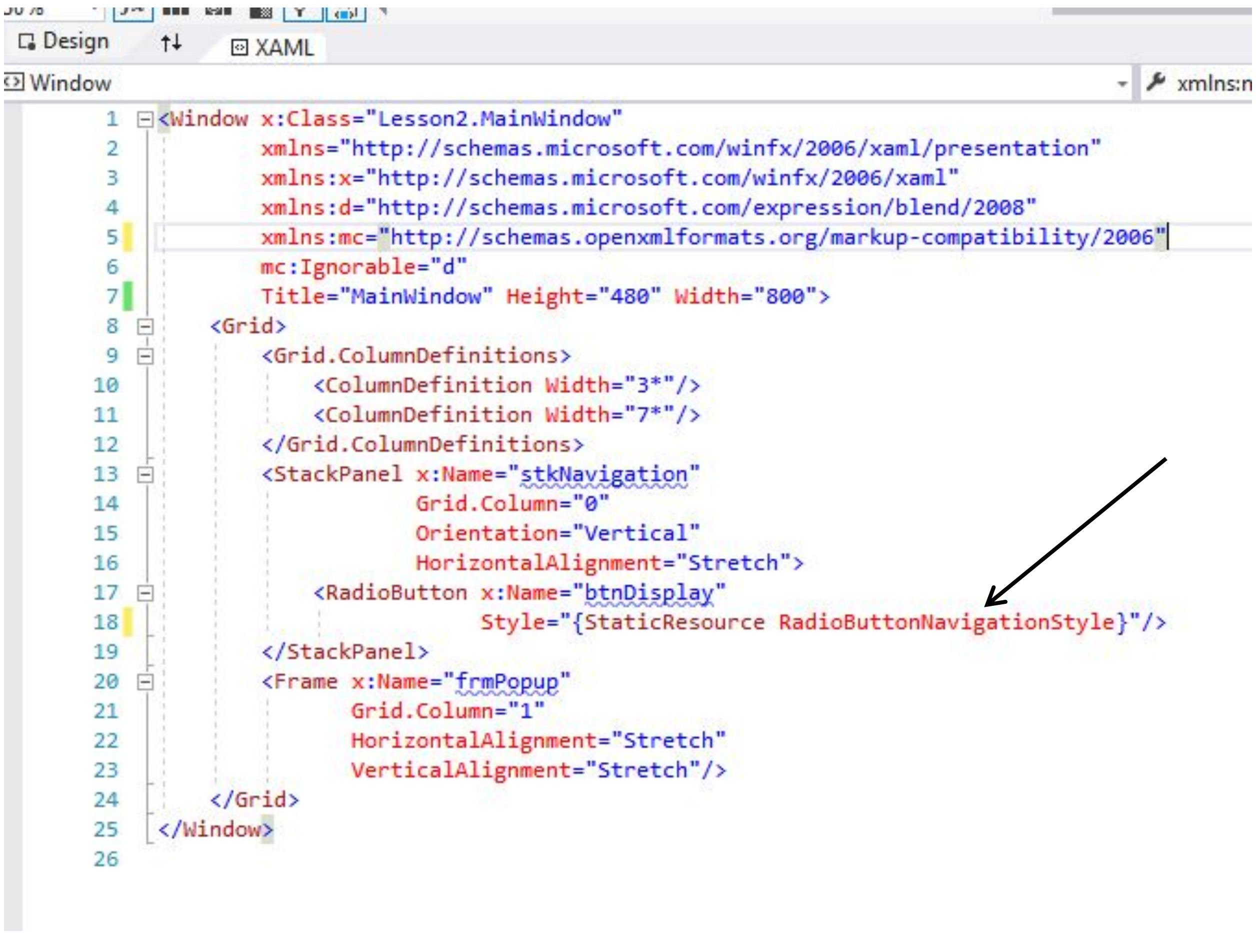
The Grid is the layout container for all the components in the button style. This two-column grid is split in quarters. The ellipse uses 25% of the control and the button title used the remaining 75%. The ellipse is assigned a fixed height and width and placed in grid column zero.

The first text block uses the Segoe MDL2 Assets font shipped with Windows 10. This font contains numerous icons. It is available as a download for Windows 8 and Mac. This TextBlock is laid over the ellipse to add the icon to the button. Note the Text property uses a special value named {TemplateBinding Tag}. The use of this value will be clearer when the XAML is completed for the radio buttons.

The second text block is placed in Grid.Column 1. Template Binding is used again for the value of the content property. The next section demonstrates how template binding works.

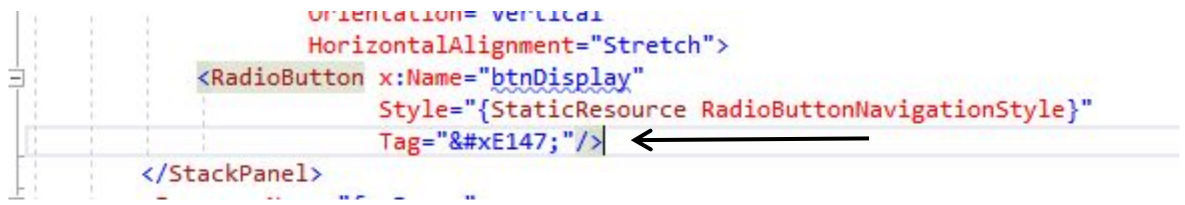
Adding the Style to the Radio Button

To complete this lesson, open MainPage.xaml and add the style to each button, using a special value named StaticResource, which connects to your Resource Dictionary.



```
<Window x:Class="Lesson2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="MainWindow" Height="480" Width="800">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="3*"/>
            <ColumnDefinition Width="7*"/>
        </Grid.ColumnDefinitions>
        <StackPanel x:Name="stkNavigation"
                    Grid.Column="0"
                    Orientation="Vertical"
                    HorizontalAlignment="Stretch">
            <RadioButton x:Name="btnDisplay"
                         Style="{StaticResource RadioButtonNavigationStyle}"/>
        </StackPanel>
        <Frame x:Name="frmPopup"
               Grid.Column="1"
               HorizontalAlignment="Stretch"
               VerticalAlignment="Stretch"/>
    </Grid>
</Window>
```

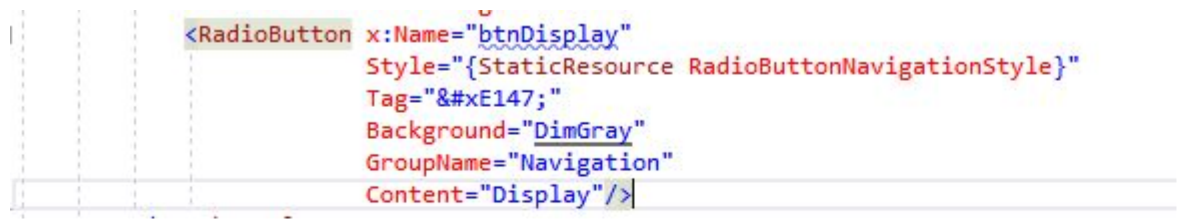
In the previous section the Template Binding value was discussed. Add the Tag property with the numeric value for the Display icon. Note the value uses: &#x as a prefix and ends with a semicolon.



```
                    HorizontalAlignment="Stretch">
               <RadioButton x:Name="btnDisplay"
                            Style="{StaticResource RadioButtonNavigationStyle}"
                            Tag="&#xE147;"/>
          </StackPanel>
```

Give the button a background color of Dim Gray and a Content value of Display.

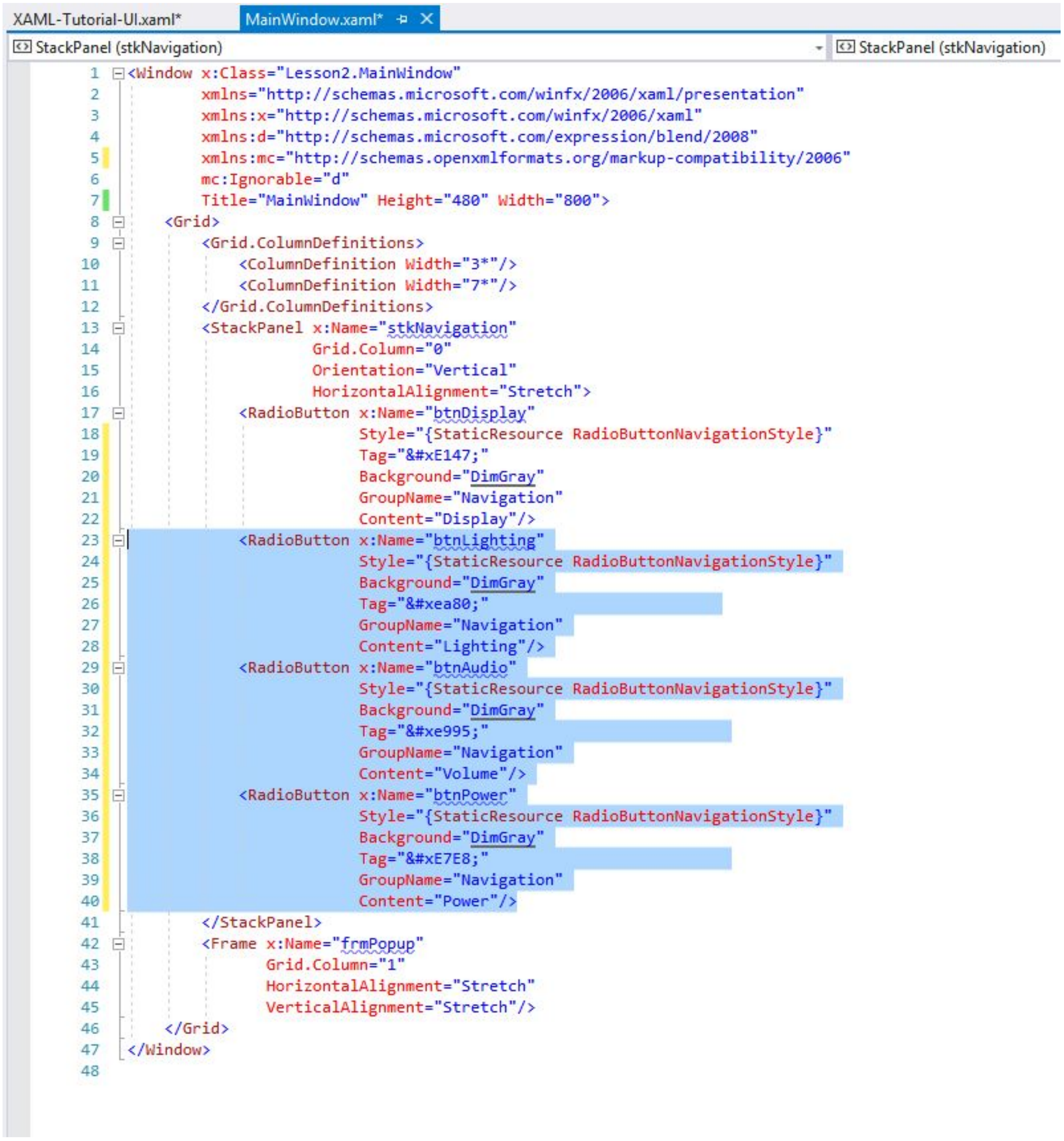
The radio button's GroupName property creates the group so only one button may be selected at a time. Since these buttons control the popup windows, let’s name the group: Navigation



```
               <RadioButton x:Name="btnDisplay"
                            Style="{StaticResource RadioButtonNavigationStyle}"
                            Tag="&#xE147;"
                            Background="DimGray"
                            GroupName="Navigation"
                            Content="Display"/>
```

Lesson 2 XAML View

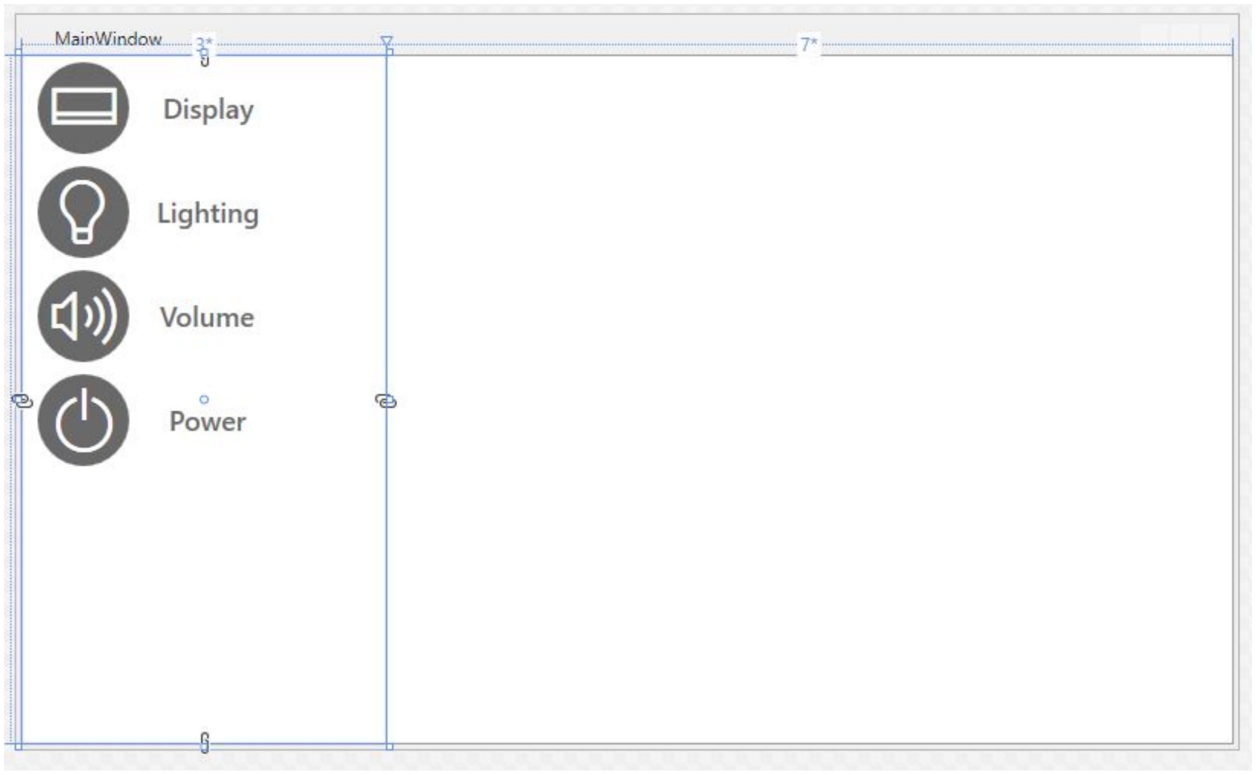
Main Page in less than 50 lines of code



```
<Window x:Class="Lesson2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="MainWindow" Height="480" Width="800">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="3*"/>
            <ColumnDefinition Width="7*"/>
        </Grid.ColumnDefinitions>
        <StackPanel x:Name="stkNavigation"
                    Grid.Column="0"
                    Orientation="Vertical"
                    HorizontalAlignment="Stretch">
            <RadioButton x:Name="btnDisplay"
                         Style="{StaticResource RadioButtonNavigationStyle}"
                         Tag="&#xE147;"
                         Background="DimGray"
                         GroupName="Navigation"
                         Content="Display"/>
            <RadioButton x:Name="btnLighting"
                         Style="{StaticResource RadioButtonNavigationStyle}"
                         Background="DimGray"
                         Tag="&#xea80;"
                         GroupName="Navigation"
                         Content="Lighting"/>
            <RadioButton x:Name="btnAudio"
                         Style="{StaticResource RadioButtonNavigationStyle}"
                         Background="DimGray"
                         Tag="&#xe995;"
                         GroupName="Navigation"
                         Content="Volume"/>
            <RadioButton x:Name="btnPower"
                         Style="{StaticResource RadioButtonNavigationStyle}"
                         Background="DimGray"
                         Tag="&#xE7E8;"
                         GroupName="Navigation"
                         Content="Power"/>
        </StackPanel>
        <Frame x:Name="frmPopup"
               Grid.Column="1"
               HorizontalAlignment="Stretch"
               VerticalAlignment="Stretch"/>
    </Grid>
</Window>
```

Note the Lighting, Audio and Power radio buttons were added to complete the Navigation sidebar.

Lesson 2 Design View


MainWindow
3*
7*
Display
Lighting
Volume
Power