

WPF Controls

Overview

Some commonly used WPF controls include:

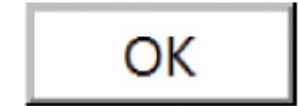
- Buttons
- CheckBox
- RadioButton
- Menu
- Text
 - TextBox
 - RichTextBox
 - PasswordBox
- Image

Buttons

Default theme



High-contrast theme



- The Button class inherits directly from the *System.Windows.Controls.Primitives.ButtonBase* class
- **Click Event:**

The ButtonBase class contains the Click event and the logic that defines what it means to be clicked. As with typical Windows buttons, a click can occur from a mouse's left button being pressed down and then let up or from the keyboard with Enter or spacebar, if the button has focus.
- Its **ClickMode** property can be set to a value of a ClickMode enumeration of Release (the default), Press, and Hover

Buttons Example

Three buttons respond to clicks in three different ways.

- Hover**: the first button changes colors when the user hovers with the mouse over the button.

- Press**: the second button requires that the mouse be pressed while the mouse pointer is over the button.

- Release**: the third does not reset the background color of the buttons until the mouse is pressed and released on the button.

XAML

```
<Button Name="btn1" Background="Pink"
        BorderBrush="Black" BorderThickness="1"
        Click="OnClick1" ClickMode="Hover">
    ClickMe1
</Button>

<Button Name="btn2" Background="LightBlue"
        BorderBrush="Black" BorderThickness="1"
        Click="OnClick2" ClickMode="Press">
    ClickMe2
</Button>

<Button Name="btn3"
        Click="OnClick3" ClickMode="Release">
    Reset
</Button>
```

C#

VB

```
void OnClick1(object sender, RoutedEventArgs e)
{
    btn1.Background = Brushes.LightBlue;
}

void OnClick2(object sender, RoutedEventArgs e)
{
    btn2.Background = Brushes.Pink;
}

void OnClick3(object sender, RoutedEventArgs e)
{
    btn1.Background = Brushes.Pink;
    btn2.Background = Brushes.LightBlue;
}
```

Checkbox

- It has a notion of being clicked by mouse or keyboard.
- It retains a state of being checked or unchecked when clicked.
- It supports a three-state mode, where the state toggles from checked to indeterminate to unchecked.



Checked



Unchecked



Indeterminate

Checkbox Example

A CheckBox and handles the [Checked](#), [Unchecked](#), and [Indeterminate](#) events.

XAML

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

  <TextBlock Text="CheckBox Demonstration" Margin="0,20,10,20"
    FontFamily="Verdana" FontSize="18" FontWeight="Bold"
    Foreground="#FF5C9AC9" Grid.Row="0"/>

  <CheckBox x:Name="cb1" Grid.Row="1" Margin="5,0,0,0"
    Content="Three-state CheckBox" IsThreeState="True"
    Checked="HandleCheck" Unchecked="HandleUnchecked"
    Indeterminate="HandleThirdState" />

  <TextBlock x:Name="text1" Grid.Row="2" Margin="5,0,0,0" />
</Grid>
```

C#

VB

```
private void HandleCheck(object sender, RoutedEventArgs e)
{
    text1.Text = "The CheckBox is checked.";
}


private void HandleUnchecked(object sender, RoutedEventArgs e)
{
    text1.Text = "The CheckBox is unchecked.";
}

private void HandleThirdState(object sender, RoutedEventArgs e)
{
    text1.Text = "The CheckBox is in the indeterminate state.";
}
```

Radio Button

- A RadioButton has two states: true or false. The RadioButton is a control that is usually used as an item in a group of RadioButton controls. However, it is possible to create a single RadioButton. Whether a RadioButton is selected is determined by the state of its IsChecked property.
- When a RadioButton is selected, it cannot be cleared by clicking it.
- When RadioButton elements are grouped, the buttons are mutually exclusive.
A user can select only one item at a time within a RadioButton group. You can group RadioButton controls by placing them inside a parent or by setting the GroupName property on each RadioButton.

```
<StackPanel>
  <RadioButton GroupName="A">Option 1</RadioButton>
  <RadioButton GroupName="A">Option 2</RadioButton>
  <RadioButton GroupName="B">A Different Option 1</RadioButton>
  <RadioButton GroupName="B">A Different Option 2</RadioButton>
</StackPanel>
```



Radio Button Example

Create RadioButton controls, group them inside a container, and handle the [Checked](#) event..

XAML

```
<StackPanel>
  <RadioButton Name="rb1" Checked="WriteText2">Yes</RadioButton>
  <RadioButton Name="rb2" Checked="WriteText2">No</RadioButton>
  <RadioButton Name="rb3" Checked="WriteText2">No opinion</RadioButton>
</StackPanel>
```

C#

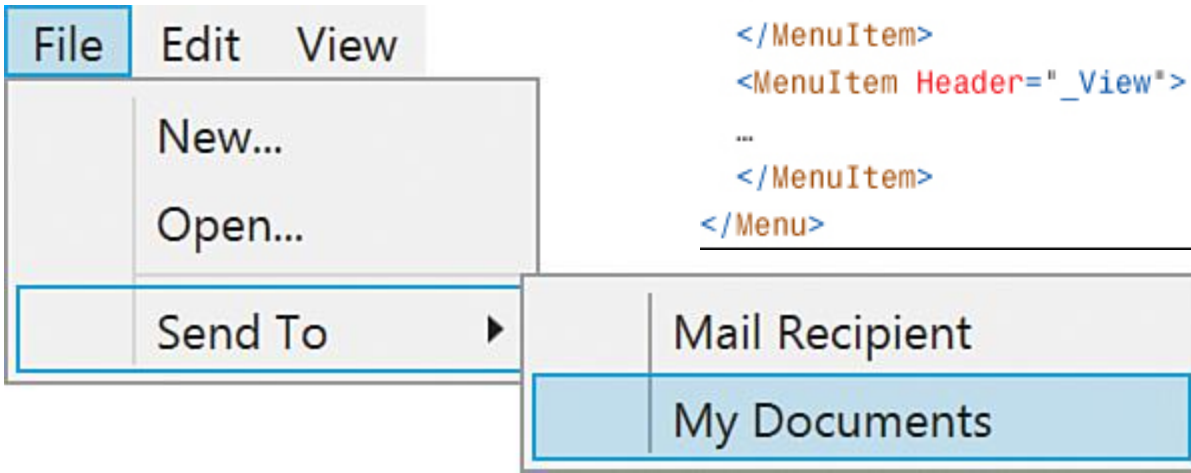
VB

```
void WriteText2(object sender, RoutedEventArgs e)
{
    RadioButton li = (sender as RadioButton);
    txtb.Text = "You clicked " + li.Content.ToString() + ".";
}
```


Menu

Menu Class doc: [https://msdn.microsoft.com/en-us/library/system.windows.controls.menu\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.windows.controls.menu(v=vs.110).aspx)

```
<Menu>
  <MenuItem Header="_File">
    <MenuItem Header="_New..." />
    <MenuItem Header="_Open..." />
    <Separator />
    <MenuItem Header="Sen_d To">
      <MenuItem Header="Mail Recipient" />
      <MenuItem Header="My Documents" />
    </MenuItem>
  </MenuItem>
  <MenuItem Header="_Edit">
    ...
  </MenuItem>
  <MenuItem Header="_View">
    ...
  </MenuItem>
</Menu>
```



TextBox



- Enables users to type one or more lines of text. TextBox stores it in a string property called Text.
- Defines [TextChanged](#) and [SelectionChanged](#) events.
- To enable spell checking in a TextBox (or RichTextBox), you set the attached [SpellCheck.IsEnabled](#) property to true
- Make the text wrap to form additional lines by setting its TextWrapping property to Wrap or WrapWithOverflow.
- **RichTextBox** - a more advanced TextBox that can contain formatted text and arbitrary objects embedded in the text
- **PasswordBox** - a simpler TextBox designed for the entry of a password



Image

- `System.Windows.Controls.Image` enables images (.BMP, .PNG, .GIF, .JPG, and so on) to be rendered in a WPF user interface.