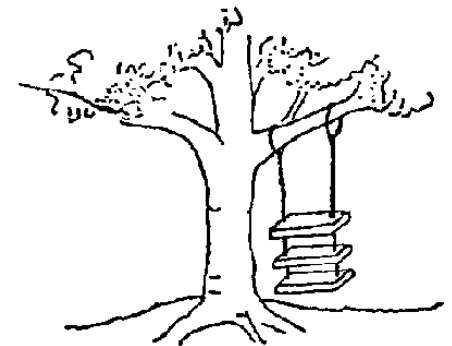


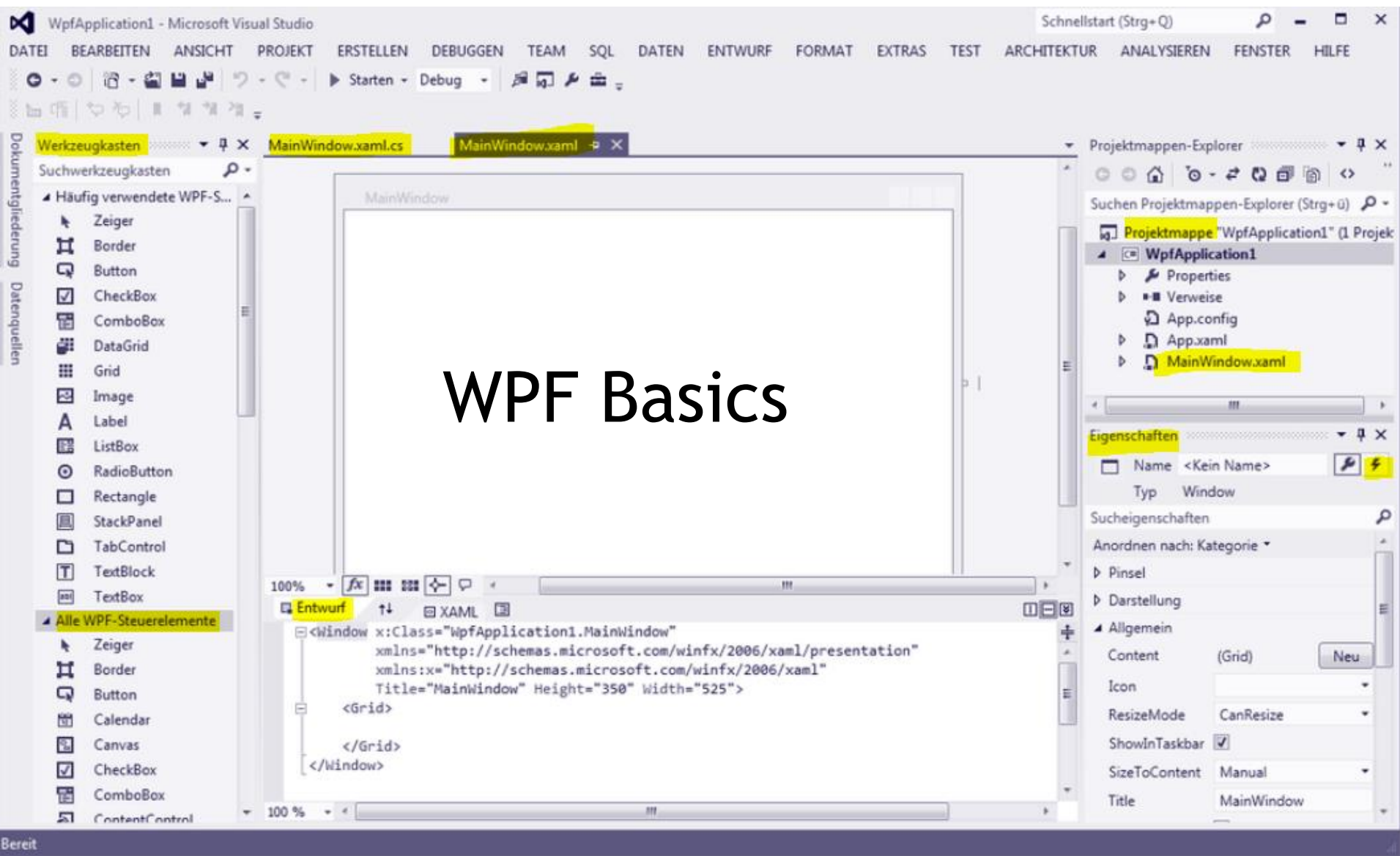
WPF Controls

Software Entwicklung



Overview

- WPF Basics
 - New Window
 - MessageBox
- WPF Controls
- WPF Panels
- WPF Data Binding
 - <https://www.codeproject.com/Articles/1112919/MVVM-for-beginners>
- WPF MVVM
 - <https://www.codeproject.com/Articles/1052346/ICommand-Interface-in-WPF>
- WPF ICommand
 - <https://www.codeproject.com/Articles/1052346/ICommand-Interface-in-WPF>
 - <https://msdn.microsoft.com/en-us/magazine/dd419663.aspx#id0090030>



Create a new Window(WPF)

The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays the code for `WindowTextBlock.xaml.cs`. The code is as follows:

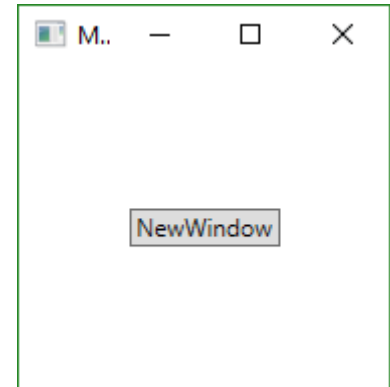
```
17  /// <summary>
18  /// Interaktionslogik für Window1.xaml
19  /// </summary>
20  4-Verweise
21  public partial class WindowTextBlock : Window
22  {
23      1-Verweis
24      public WindowTextBlock()
25      {
26          InitializeComponent();
27          //this.Show();
28      }
29  }
```
- Projektmappen-Explorer:** Shows the project structure with `WPF_Controls` selected. A yellow box highlights the `WPF_Controls` folder.
- Context Menu:** A right-click context menu is open over the `WPF_Controls` folder. The `Hinzufügen` (Add) option is selected, which has opened a sub-menu.
- Sub-menu:** The sub-menu for `Hinzufügen` is open, showing options like `Neues Element...`, `Vorhandenes Element...`, `Neuer Ordner`, `REST-API-Client...`, `Verweis...`, `Webverweis...`, `Dienstverweis...`, `Verbundener Dienst`, `Analysetool...`, `Fenster...` (highlighted with a yellow box), `Seite...`, `Benutzersteuerelement...`, `Ressourcenwörterbuch...`, and `Klasse...`.
- Other UI Elements:** The `Fehlerliste` (Error List) is visible at the bottom left, showing `Gesamte Projektmappe`. The `100 %` zoom level is indicated at the bottom left.

New Window

- Instantiate the Window and Show it:

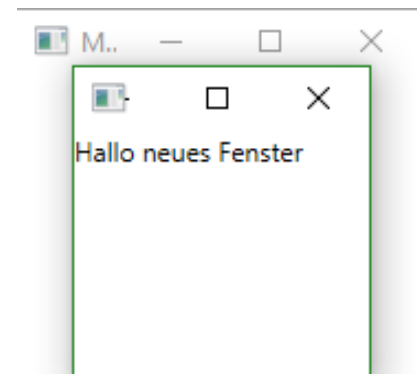
```
U-Verweise
public MainWindow()
{
    InitializeComponent();
}

1-Verweis
private void Button_Click(object sender, RoutedEventArgs e)
{
    WindowTextBlock w = new WindowTextBlock();
    w.Show();
}
```



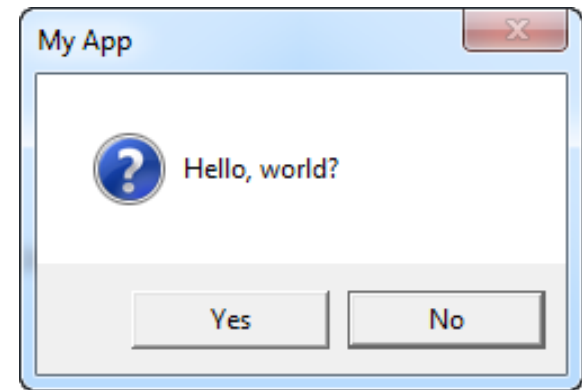
- Add a new Window: WindowTextBlock

```
<Window x:Class="WPF_Controls.WindowTextBlock"
        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"
        xmlns:local="clr-namespace:WPF_Controls"
        mc:Ignorable="d"
        Title="NewWindow" Height="150" Width="150">
    <Grid>
        <TextBlock Text="Hallo neues Fenster"/>
    </Grid>
</Window>
```



Message Box

- purpose is:
 - show a message to the user
 - offer one or several ways for the user to respond to the message
- MessageBox is used
 - by calling the static Show() method
 - `MessageBox.Show("Hello, world!");`
- Try the complete example:

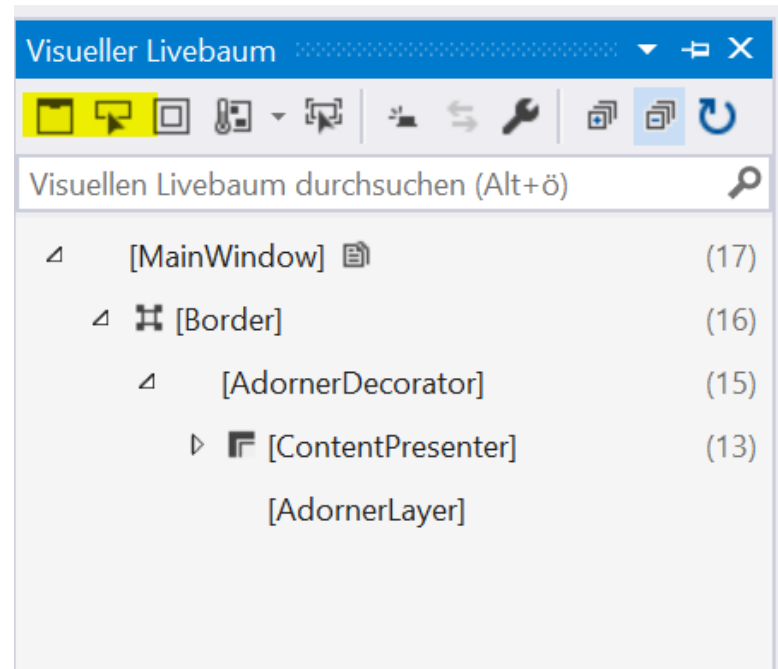
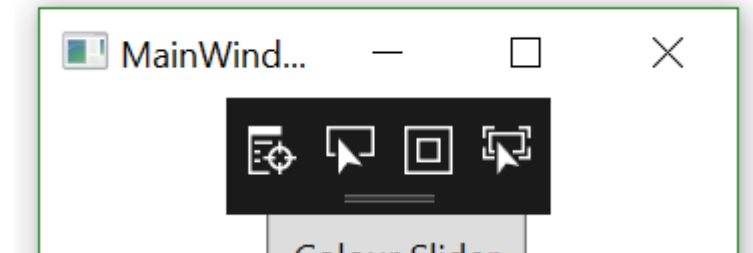


<https://www.wpf-tutorial.com/dialogs/the-messagebox/>

Remove Black Bar

- Get rid of the black bar
- Start your App
- Menu:
 - Debug
 - Window
 - Visual LiveTree

Laufzeittools in Anwendung
wegklicken

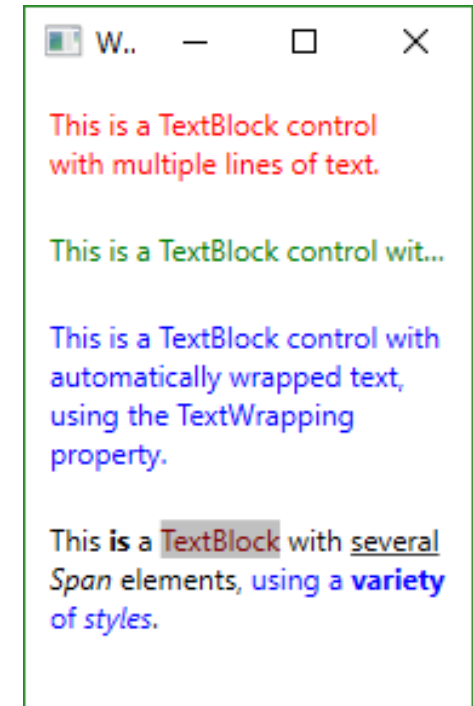


TextBlock

Show a Text in a TextBlock

Read from a file and show the content in the TextBlock

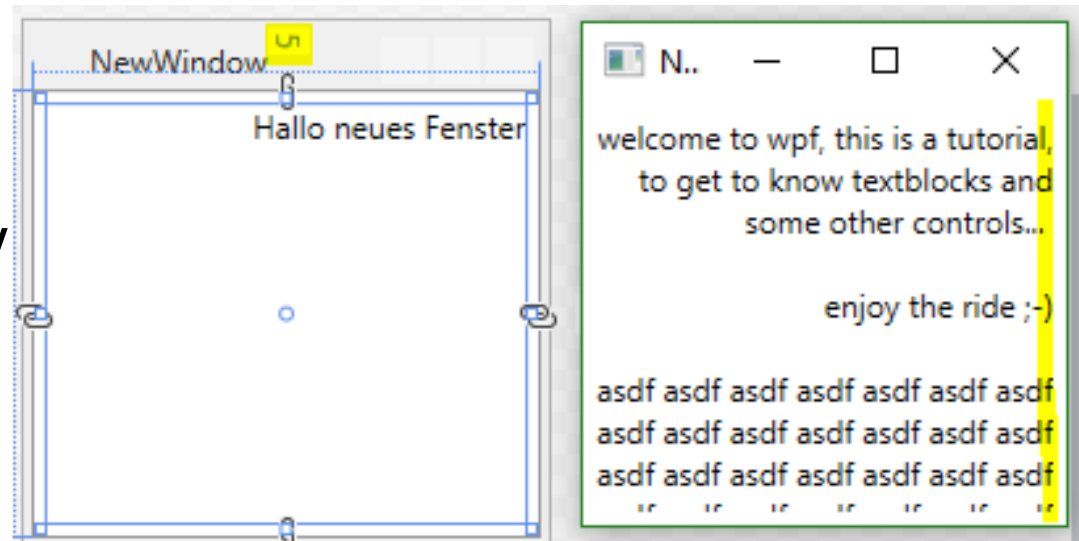
Format the text colourful and stylish



<https://wpf-tutorial.com/basic-controls/the-textblock-control/>

TextBlock

- Add
 - Textblock to a new window
- Set
 - Margin
 - TextWrapping
 - TextAlignment
- Fill the text with the content of a file

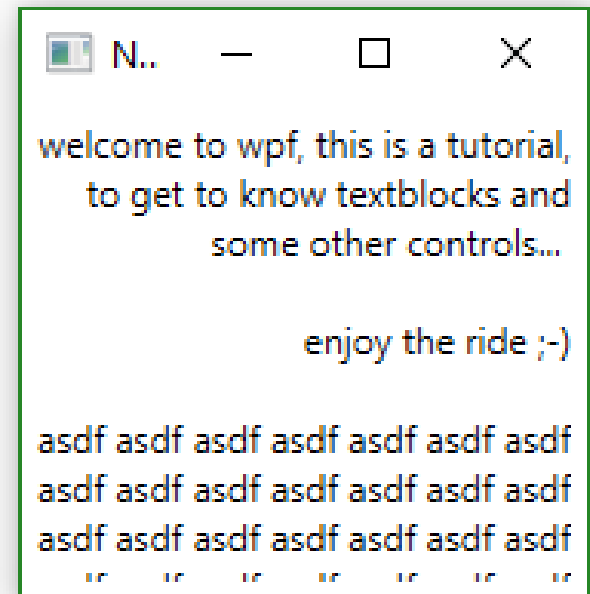


```
<Grid>  
    <TextBlock Name="TblFileContent" Margin="5" TextAlignment="Right"  
        TextWrapping="Wrap" Text="Hallo neues Fenster"/>  
</Grid>
```

Read from File

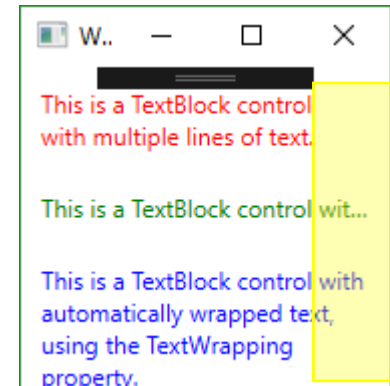
```
public partial class WindowTextBlock : Window
{
    1-Verweis
    public WindowTextBlock()
    {
        InitializeComponent();
        ReadFile(@"../../welcome.txt");
        this.Show();
    }

    1-Verweis
    private void ReadFile(string filename)
    {
        string s = File.ReadAllText(filename);
        TblFileContent.Text = s;
    }
}
```



Color and wrap the Text

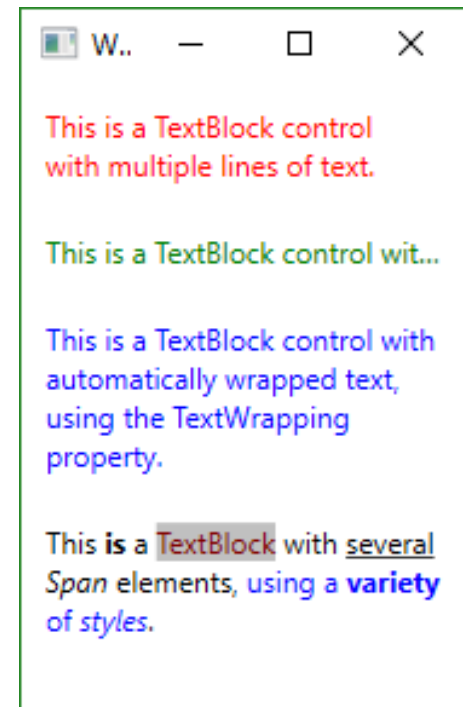
```
Title="WindowTextBlockColor" Height="200" Width="200">
<StackPanel>
  <TextBlock Margin="10" Foreground="Red">
    This is a TextBlock control<LineBreak />
    with multiple lines of text.
  </TextBlock>
  <TextBlock Margin="10" TextTrimming="CharacterEllipsis" Foreground="Green">
    This is a TextBlock control with text that may not be
    rendered completely, which will be indicated with an ellipsis.
  </TextBlock>
  <TextBlock Margin="10" TextWrapping="Wrap" Foreground="Blue">
    This is a TextBlock control with automatically wrapped text,
    using the TextWrapping property.
  </TextBlock>
</StackPanel>
```



Formatting the Text in the Textblock

- Using Span
- Background & Foreground
- TextDecorations
- FontStyle
- Bold, Italic, Underline

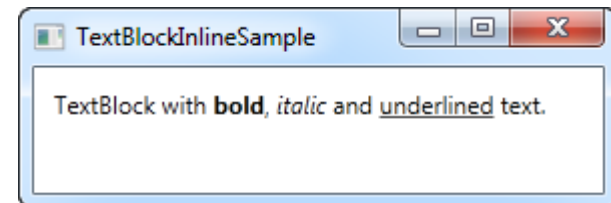
```
<TextBlock Margin="10" TextWrapping="Wrap">  
    This <Span FontWeight="Bold">is</Span> a  
    <Span Background="Silver" Foreground="Maroon">TextBlock</Span>  
    with <Span TextDecorations="Underline">several</Span>  
    <Span FontStyle="Italic">Span</Span> elements,  
    <Span Foreground="Blue">  
        using a <Bold>variety</Bold> of <Italic>styles</Italic>  
    </Span>.  
</TextBlock>
```



TextBlock - Inline Formatting

- Advanced formatting:

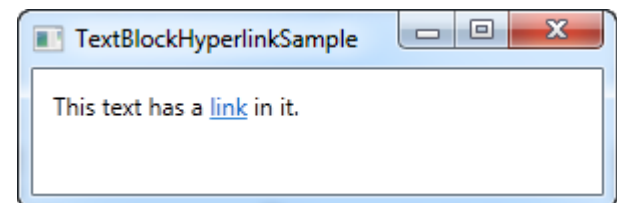
```
<TextBlock Margin="10" TextWrapping="Wrap">  
    TextBlock with <Bold>bold</Bold>,  
    <Italic>italic</Italic> and  
    <Underline>underlined</Underline> text.  
</TextBlock>
```



- Hyperlink

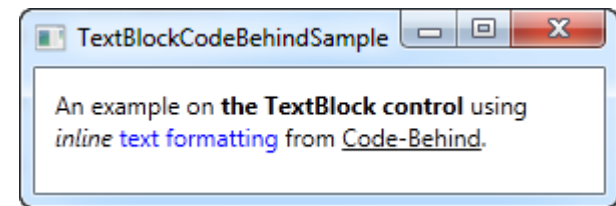
```
<TextBlock Margin="10" TextWrapping="Wrap">  
    This text has a <Hyperlink  
        RequestNavigate="Hyperlink_RequestNavigate"  
        NavigateUri="https://www.google.com">link</Hyperlink> in it.  
</TextBlock>
```

```
private void Hyperlink_RequestNavigate(object sender, System.Windows.Navigation.RequestNavigateEventArgs e)  
{  
    System.Diagnostics.Process.Start(e.Uri.AbsoluteUri);  
}
```



TextBlock - Inline Formatting (Run)

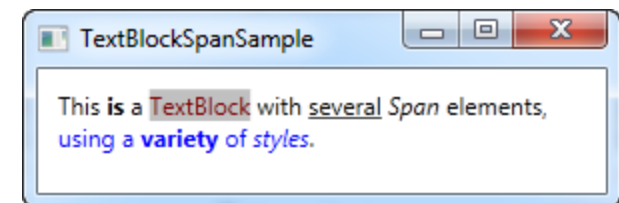
- Run vs Span
 - Span is more flexible



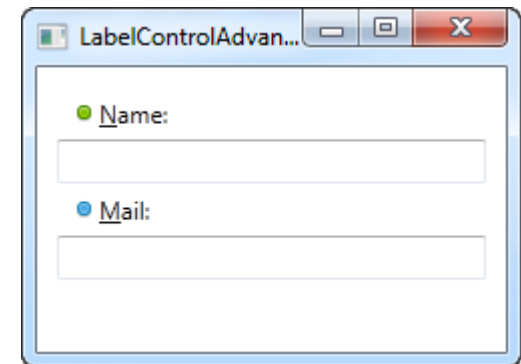
- Run element allows
 - to style a string using all the available properties of the Span element
- Span element may contain
 - other inline elements,
a Run element may only contain plain text

TextBlock - Inline Formatting (Span)

- doesn't have any specific rendering by default
- allows to set almost any kind of specific rendering
 - including font size, style and weight, background and foreground colors and so on



```
<TextBlock Margin="10" TextWrapping="Wrap">  
  This <Span FontWeight="Bold">is</Span> a  
  <Span Background="Silver" Foreground="Maroon">TextBlock</Span>  
  with <Span TextDecorations="Underline">several</Span>  
  <Span FontStyle="Italic">Span</Span> elements,  
  <Span Foreground="Blue">  
    using a <Bold>variety</Bold> of <Italic>styles</Italic>  
  </Span>.  
</TextBlock>
```

Labels

Content instead of Text Property

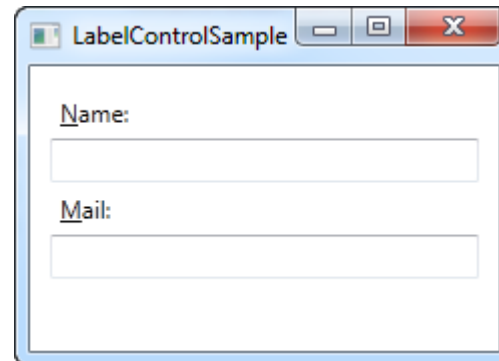
<https://wpf-tutorial.com/basic-controls/the-label-control/>

Labels

- use the **Target** property to connect the Label and the designated control

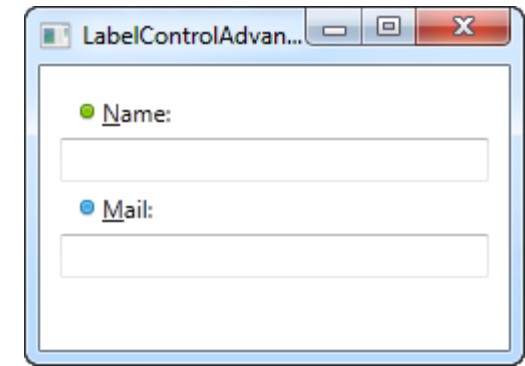
```
<StackPanel Margin="10">  
    <Label Content="_Name:" Target="{Binding ElementName=txtName}" />  
    <TextBox Name="txtName" />  
    <Label Content="_Mail:" Target="{Binding ElementName=txtMail}" />  
    <TextBox Name="txtMail" />  
</StackPanel>
```

- Label a Textbox:
 - Name
 - Mail



Advanced Using of Labels

- Labels have a Content Property
 - Add a StackPanel as Content



```
Title="WindowLabel" Height="180" Width="250">
<StackPanel Margin="10">
  <Label Target="{Binding ElementName=txtName}">
    <StackPanel Orientation="Horizontal">
      <Image Source="http://cdn1.iconfinder.com/data/icons/fatcow/16/bullet_green.png" />
      <AccessText Text="_Name:" />
    </StackPanel>
  </Label>
  <TextBox Name="txtName" />
  <Label Target="{Binding ElementName=txtMail}">
    <StackPanel Orientation="Horizontal">
      <Image Source="http://cdn1.iconfinder.com/data/icons/fatcow/16/bullet_blue.png" />
      <AccessText Text="_Mail:" />
    </StackPanel>
  </Label>
  <TextBox Name="txtMail" />
</StackPanel>
..
```



Textbox

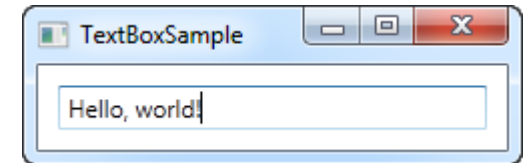
Text-input control in WPF

write plain text, on a single line, for dialog input, or in multiple lines, like an editor

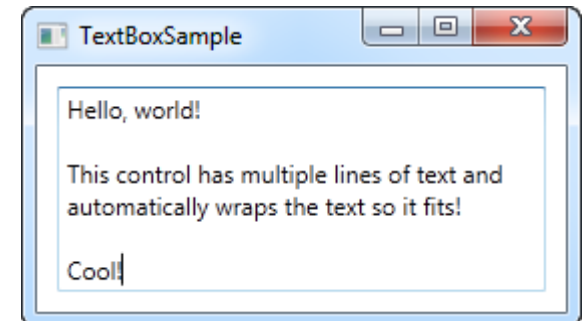
<https://wpf-tutorial.com/basic-controls/the-textbox-control/>

Textbox

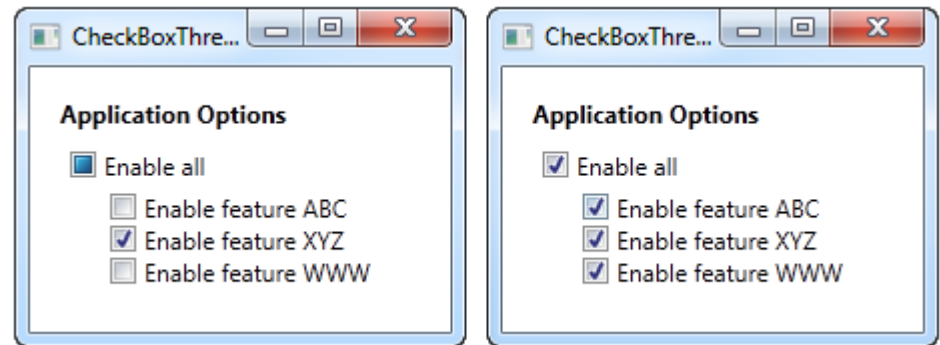
- Single Line
 - Text
- Multiple Lines
 - AcceptsReturn
 - TextWrapping



```
<TextBox Text="Hello, world!" />
```



```
<Grid Margin="10">  
    <TextBox AcceptsReturn="True" TextWrapping="Wrap" />  
</Grid>
```



CheckBox

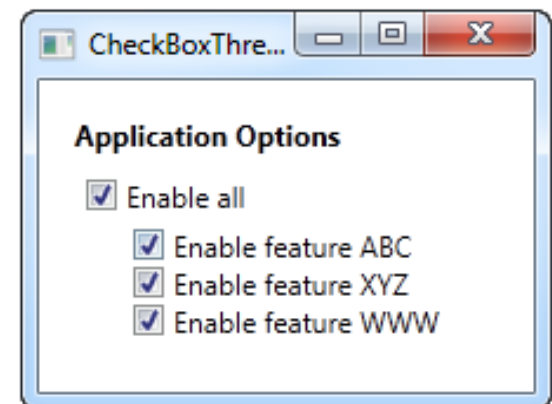
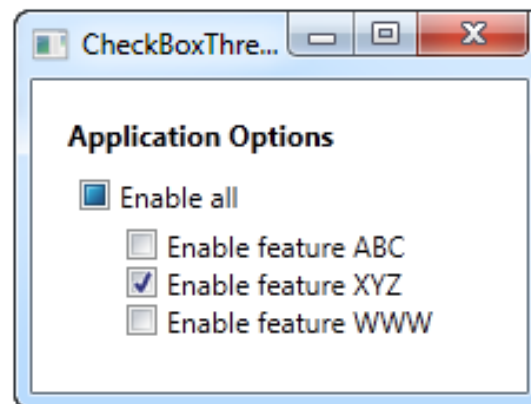
Select one or multiple choices

<https://wpf-tutorial.com/basic-controls/the-checkbox-control/>

Checkbox

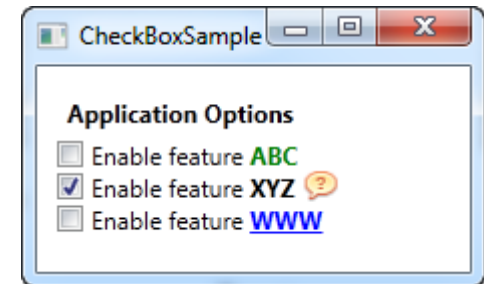
- Simple Style

```
Title="WindowCheckBox" Height="140" Width="250">  
<StackPanel Margin="10">  
    <Label FontWeight="Bold">Application Options</Label>  
    <CheckBox>Enable feature ABC</CheckBox>  
    <CheckBox IsChecked="True">Enable feature XYZ</CheckBox>  
    <CheckBox>Enable feature WWW</CheckBox>  
</StackPanel>
```



Checkbox Advanced

- With Custom content



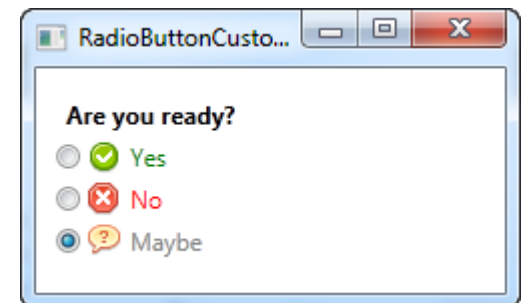
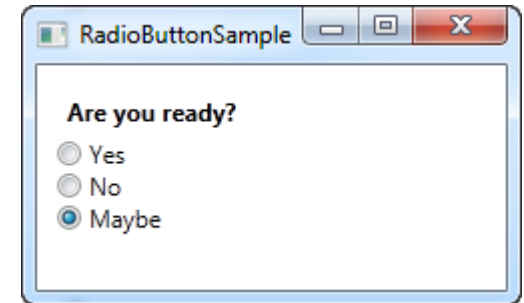
```
<StackPanel Margin="10">
  <Label FontWeight="Bold">Application Options</Label>
  <CheckBox>
    <TextBlock>
      Enable feature <Run Foreground="Green"
                        FontWeight="Bold">ABC</Run>
    </TextBlock>
  </CheckBox>
  <CheckBox IsChecked="True">
    <WrapPanel>
      <TextBlock>
        Enable feature <Run FontWeight="Bold">XYZ</Run>
      </TextBlock>
      <Image Source="https://upload.wikimedia.org/wikipedia/commons/f/f6/Lol_question_mark.png"
              Width="16" Height="16" Margin="5,0" />
    </WrapPanel>
  </CheckBox>
  <CheckBox>
    <TextBlock>
      Enable feature <Run Foreground="Blue"
                        TextDecorations="Underline"
                        FontWeight="Bold">WWW</Run>
    </TextBlock>
  </CheckBox>
</StackPanel>
```

You typically use the Run element only when you want to format a discrete section of text within the TextBlock.

RadioButton

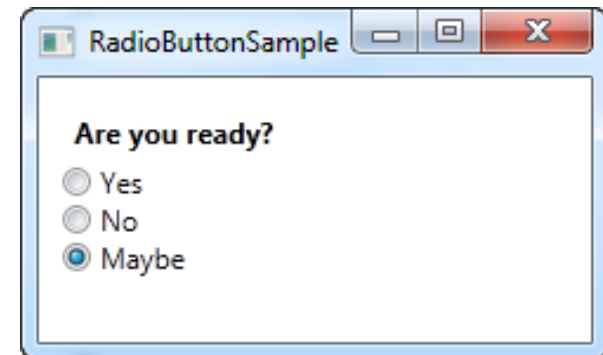
allows you to give your user a list of possible options

achieve the same effect, using less space,
with the ComboBox control



RadioButton

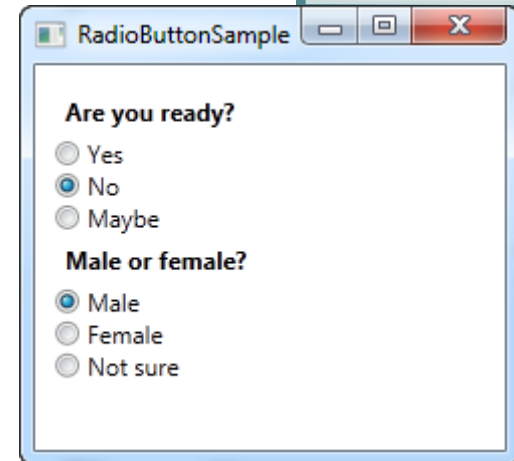
- Make your choice, only choose one...



```
Title="WindowRadioButton" Height="150" Width="250">
<StackPanel Margin="10">
    <Label FontWeight="Bold">Are you ready?</Label>
    <RadioButton>Yes</RadioButton>
    <RadioButton>No</RadioButton>
    <RadioButton IsChecked="True">Maybe</RadioButton>
</StackPanel>
```

RadioButton Groups

- Group your RadioButtons



RadioButtonSample

Are you ready?

☐ Yes

☒ No

☐ Maybe

Male or female?

☒ Male

☐ Female

☐ Not sure

```
<StackPanel Margin="10">
    <Label FontWeight="Bold">Are you ready?</Label>
    <RadioButton GroupName="ready">Yes</RadioButton>
    <RadioButton GroupName="ready">No</RadioButton>
    <RadioButton GroupName="ready" IsChecked="True">Maybe</RadioButton>

    <Label FontWeight="Bold">Male or female?</Label>
    <RadioButton GroupName="sex">Male</RadioButton>
    <RadioButton GroupName="sex">Female</RadioButton>
    <RadioButton GroupName="sex" IsChecked="True">Not sure</RadioButton>
</StackPanel>
```

WPF Panels

- control the rendering of elements
 - their size and dimensions
 - their position and the arrangement of their child content

