무료 전자 책

배우기

wpf

Free unaffiliated eBook created from
Stack Overflow contributors.

#wpf

1: wpf

WPF (Windows Presentation Foundation) Windows Microsoft . WPF UWP (Universal Windows Platform) .

WPF , . XML XAML (eXtensible Application Markup Language) . XAML WPF .

Windows Forms WPF . , .

WPF Windows Presentation Foundation (Codename Avalon) . Microsoft .NET Framework . WPF Windows Vista, 7, 8 10 Windows XP Server 2003 .

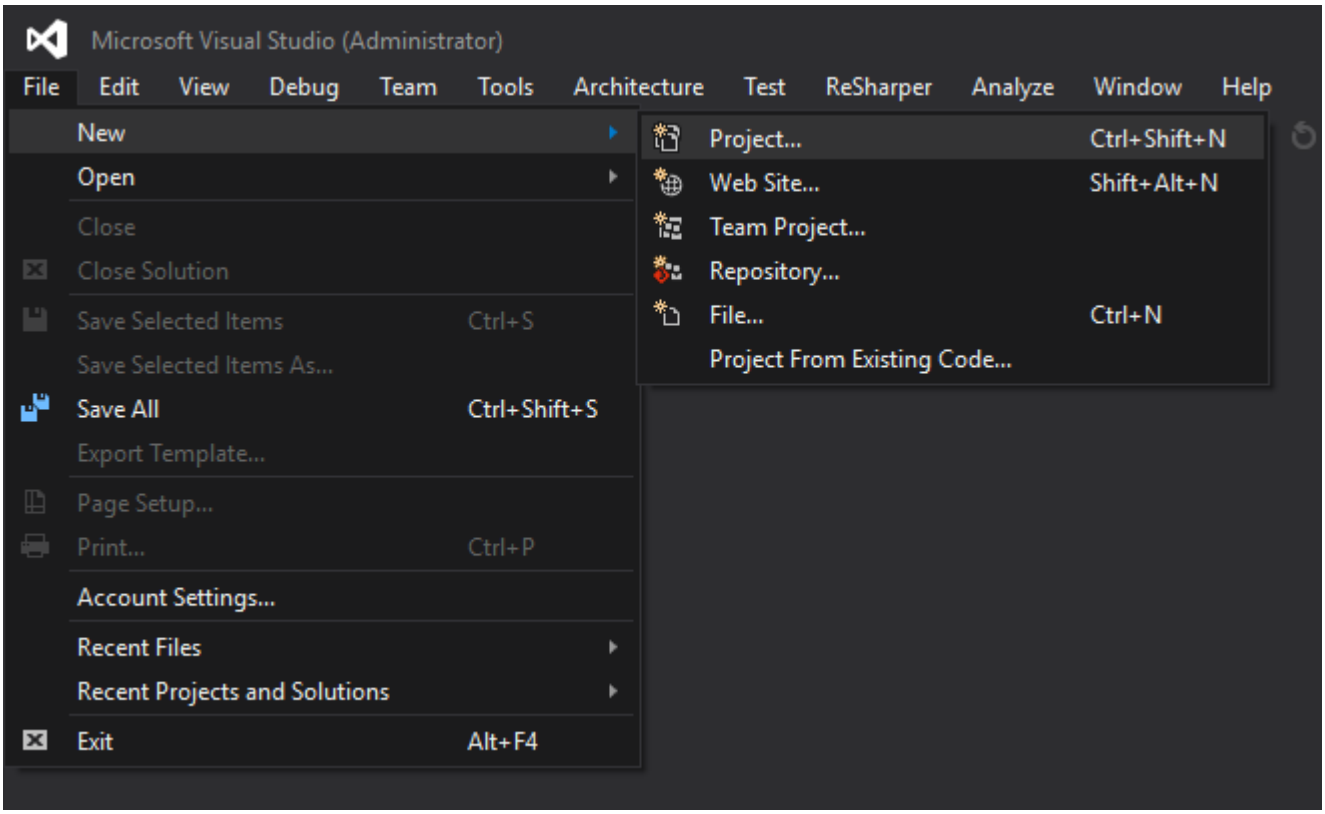
4.6.1 - 2015 12

Examples

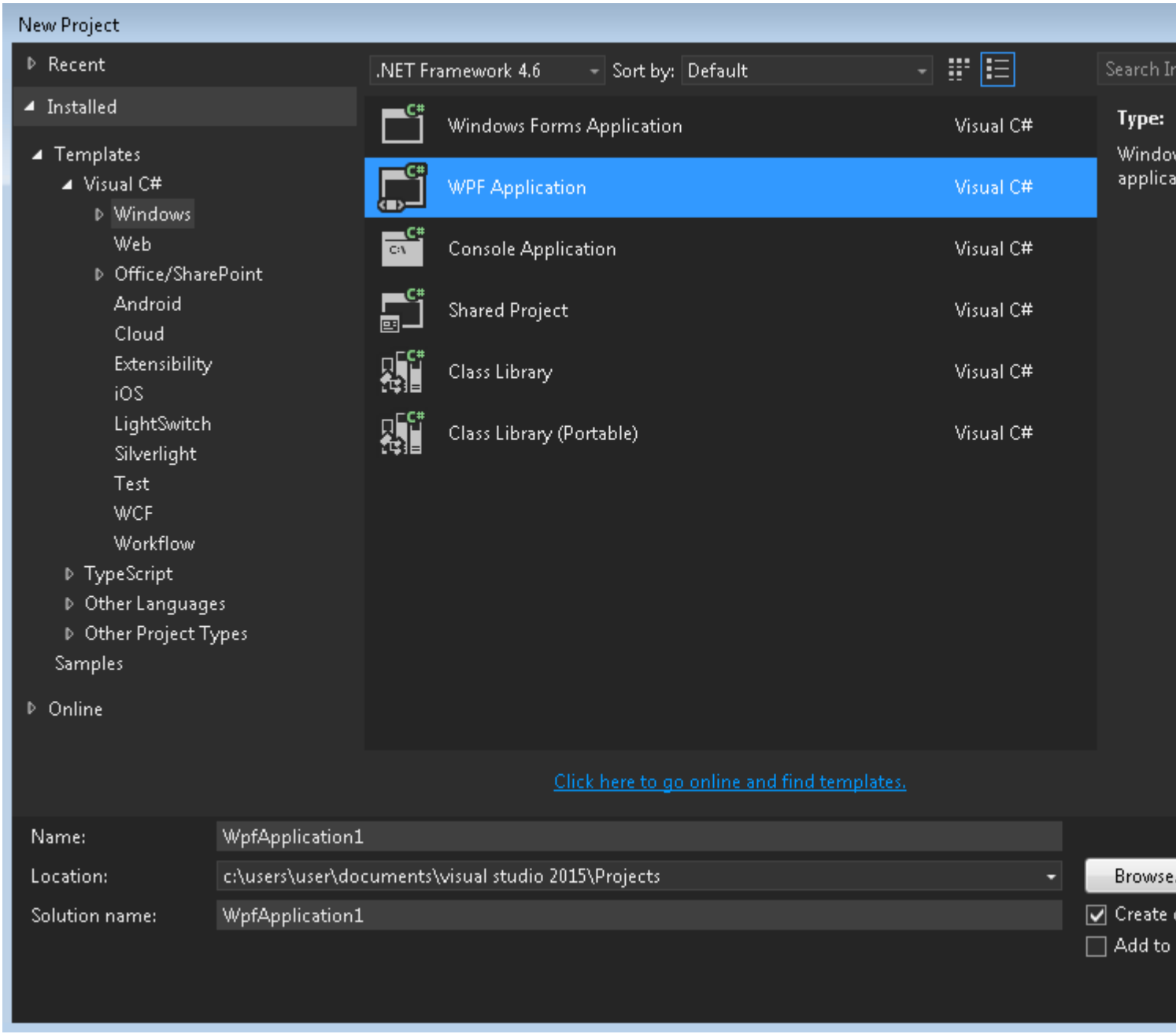
Hello World

Visual Studio WPF

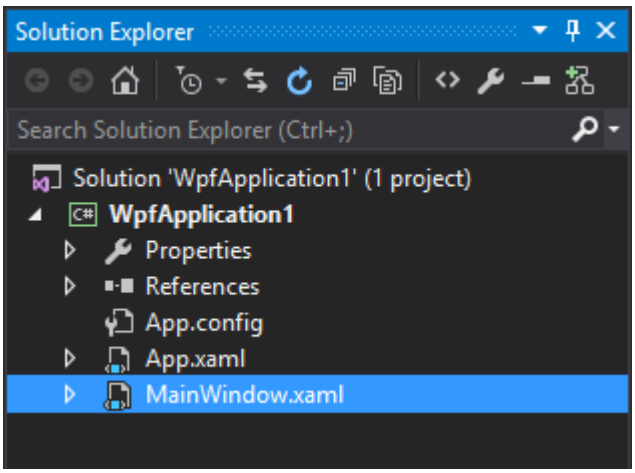
1. → → .



2. → **Visual C # → Windows → WPF** .



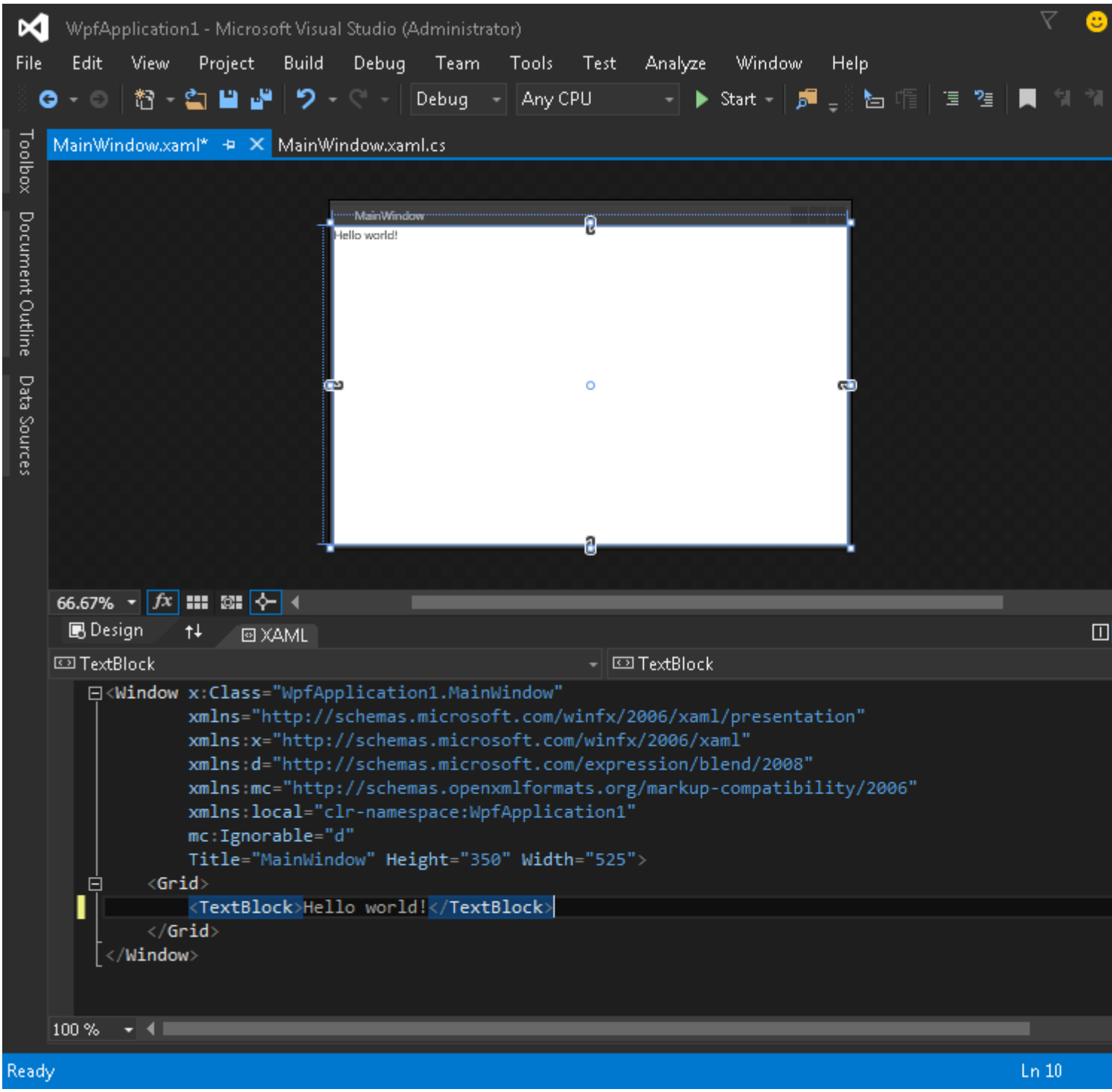
3. **MainWindow.xaml** (→).



4. *XAML* ( ) .

```
<TextBlock>Hello world!</TextBlock>
```

`Grid` :

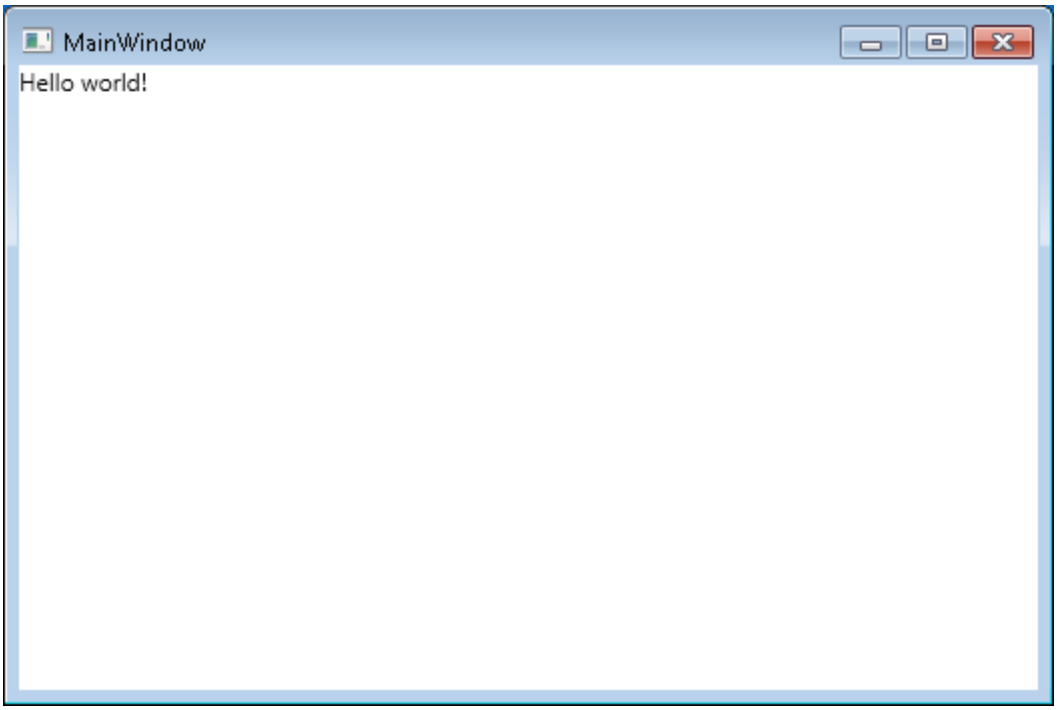


.

```
<Window x:Class="WpfApplication1.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"
        xmlns:local="clr-namespace:WpfApplication1"
        mc:Ignorable="d"
```

```
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <TextBlock>Hello world!</TextBlock>
    </Grid>
</Window>
```

5. F5 □ → . .



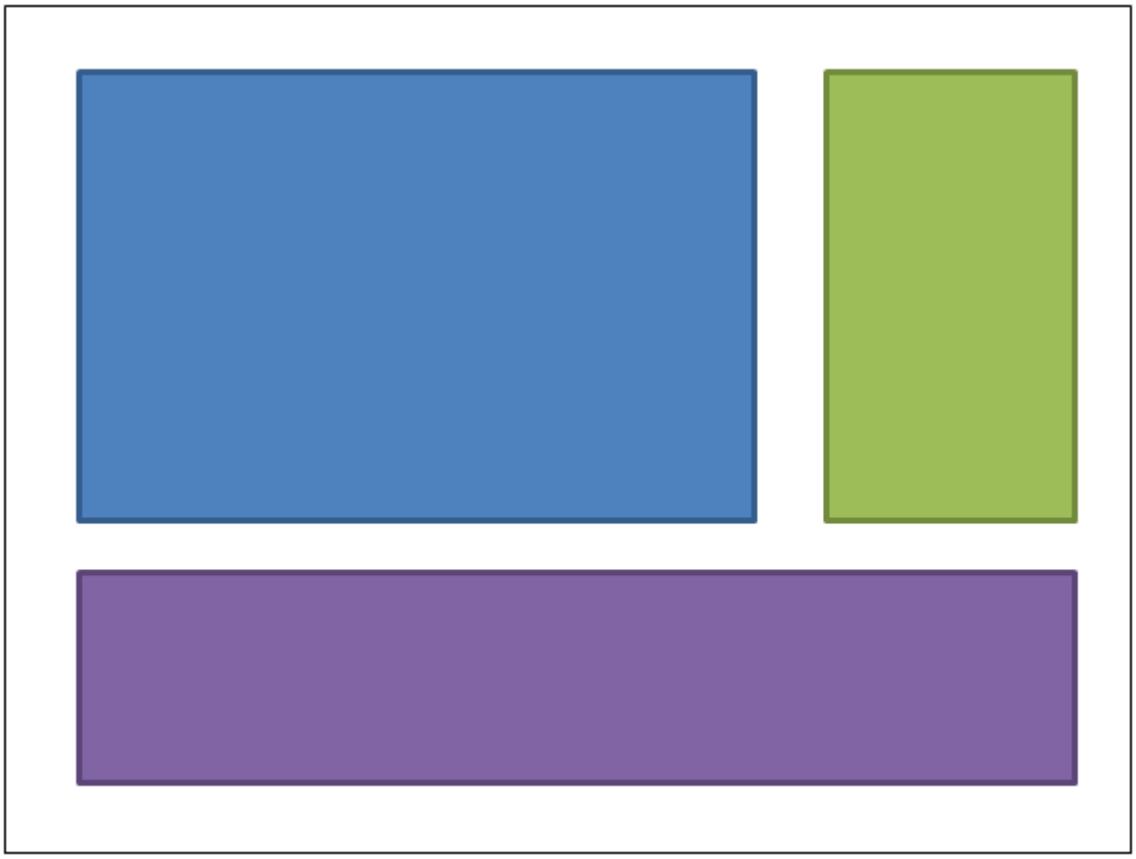
wpf : https://riptutorial.com/ko/wpf/topic/820/wpf-

2: " "

. . .

.

Examples



, 3 .

4,4,0,0 . 4,4,4,0 . 4,4,4,4. XAML ( ).

```
<UserControl x:Class="WpfApplication5.UserControl1HardCoded"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="3*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
```

```
    <Border Grid.Column="0" Grid.Row="0" Margin="4,4,0,0" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
    <Border Grid.Column="1" Grid.Row="0" Margin="4,4,4,0" Background="Green"
BorderBrush="DarkGreen" BorderThickness="5"/>
    <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Margin="4,4,4,4"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
</Grid>
</UserControl>
```

. , ? . 0,4,4,0 . 4,4,4,0, 4,4,0,0 . XAML .

```
<UserControl x:Class="WpfApplication5.UserControl2HardCoded"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="3*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Column="1" Grid.Row="0" Margin="4,4,4,0" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.Row="0" Margin="4,4,0,0" Background="Green"
BorderBrush="DarkGreen" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Margin="4,4,4,4"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
    </Grid>
</UserControl>
```

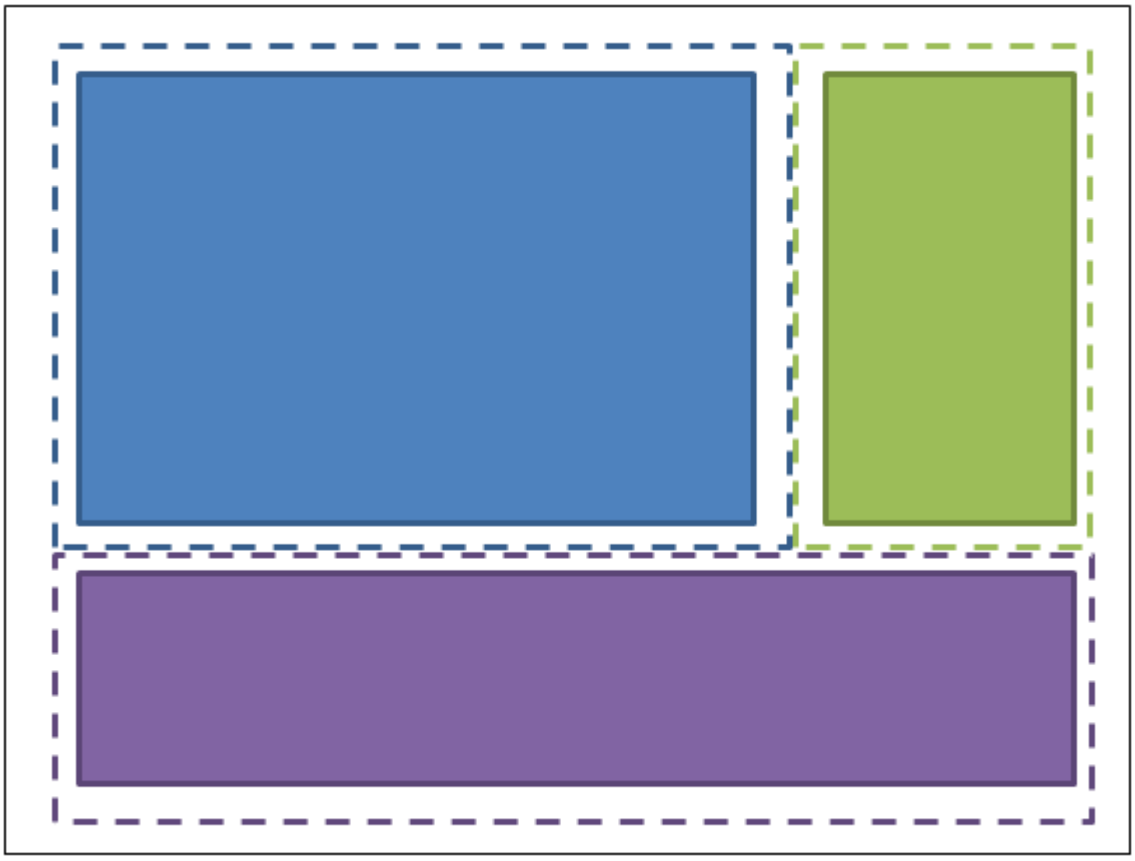
. 4,0,4,4. 4,0,0,4.

```
<UserControl x:Class="WpfApplication5.UserControl3HardCoded"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="3*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="2*"/>
        </Grid.RowDefinitions>

        <Border Grid.Column="1" Grid.Row="1" Margin="4,0,4,4" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
```

```
        <Border Grid.Column="0" Grid.Row="1" Margin="4,0,0,4" Background="Green"
BorderBrush="DarkGreen" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Margin="4,4,4,4"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
    </Grid>
</UserControl>
```

? . (-) .



2,2,2,2. 2,2,2,2. 2,2,2,2. 2,2,2,2 ( ) . XAML .

```
<UserControl x:Class="WpfApplication5.UserControl1HalfTheWhitespace"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300"
             Padding="2,2,2,2">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="3*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Column="0" Grid.Row="0" Margin="2,2,2,2" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
```

```
        <Border Grid.Column="1" Grid.Row="0" Margin="2,2,2,2" Background="Green"
BorderBrush="DarkGreen" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Margin="2,2,2,2"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
    </Grid>
</UserControl>
```

... . . . XAML .

```
<UserControl x:Class="WpfApplication5.UserControl2HalfTheWhitespace"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300"
             Padding="2,2,2,2">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="3*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Column="1" Grid.Row="0" Margin="2,2,2,2" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.Row="0" Margin="2,2,2,2" Background="Green"
BorderBrush="DarkGreen" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Margin="2,2,2,2"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
    </Grid>
</UserControl>
```

. . . XAML .

```
<UserControl x:Class="WpfApplication5.UserControl3HalfTheWhitespace"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300"
             Padding="2,2,2,2">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="3*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="2*"/>
        </Grid.RowDefinitions>

        <Border Grid.Column="1" Grid.Row="1" Margin="2,2,2,2" Background="DodgerBlue"
BorderBrush="DarkBlue" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.Row="1" Margin="2,2,2,2" Background="Green"
```

```
BorderBrush="DarkGreen" BorderThickness="5"/>
        <Border Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Margin="2,2,2,2"
Background="MediumPurple" BorderBrush="Purple" BorderThickness="5"/>
    </Grid>
</UserControl>
```

: " " , " " . . "HalfTheWhiteSpace" .

```
<system:Double x:Key="DefaultMarginSize">2</system:Double>
<Thickness x:Key="HalfTheWhiteSpace" Left="{StaticResource DefaultMarginSize}"
Top="{StaticResource DefaultMarginSize}" Right="{StaticResource DefaultMarginSize}"
Bottom="{StaticResource DefaultMarginSize}"/>
```

(FontFamily, FontSize)

```
<Style x:Key="BaseStyle" TargetType="{x:Type Control}">
    <Setter Property="Margin" Value="{StaticResource HalfTheWhiteSpace}"/>
</Style>
```

TextBox .

```
<Style TargetType="TextBox" BasedOn="{StaticResource BaseStyle}"/>
```

DatePickers, Labels ( ) . TextBlock . . TextBlock . TextBlock ; XAML TextBlock TextBlock .

(: ScrollViewer, Border) .

.

" " : https://riptutorial.com/ko/wpf/topic/9407/----

3: System.Windows.Controls.WebBrowser

WPF .

(https : //) InternetExplorer7 . west-wind.com/posts/2011/may/21/web-browser-control-specifying-the-ie-version).

. HTML (!) . (.)

" "/ " " . .

Examples

BusyIndicator WebBrowser

WebBrowser XAML . , Busy BusyIndicator . WebBrowser BusyIndicator . :

```
    <telerik:RadBusyIndicator IsBusy="{Binding IsBusy}">
        <WebBrowser Visibility="{Binding IsBusy, Converter={StaticResource
InvertBooleanToVisibilityConverter}}"/>
    </telerik:RadBusyIndicator>
```

System.Windows.Controls.WebBrowser : https://riptutorial.com/ko/wpf/topic/9115/system-windows-controls-webbrowser

4: WPF

- {Binding PropertyName} {Binding Path = PropertyName} .
- { = SomeProperty.SomeOtherProperty.YetAnotherProperty}
- { = SomeListProperty [1]}

	. DataContext .
UpdateSourceTrigger	. `LostFocus` . `PropertyChanged` .
	`OneWay` `TwoWay` . , `TwoWay` `OneWay` . `TwoWay` . `TextBox.Text` `OneWay` .
	`StaticResource` DataContext .
	XAML DataContext .
	XAML DataContext .
FallbackValue	.
TargetNullValue	Y `null` , Y .
	`StaticResource` . : `Visibility` .
ConverterParameter	. .
StringFormat	.
	(WPF 4.5) `ViewModel` `BindingSource` `milliseconds`) `ViewModel` . `Mode=TwoWay` `UpdateSourceTrigger=PropertyChanged` .

UpdateSourceTrigger

WPF . ( ) `UpdateSourceTrigger=PropertyChanged` .

`PropertyChanged` .

`LostFocus` . Enter `IsDefault` . .

UWP WPF (4.5) `Delay` . `TextBox` .

Examples

`IValueConverter` () .

: `BooleanToVisibilityConverter` System.Windows.Controls .

XAML :

```
<Window x:Class="StackOverflowDataBindingExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
        <BooleanToVisibilityConverter x:Key="VisibleIfTrueConverter" />
    </Window.Resources>
    <StackPanel>
        <CheckBox x:Name="MyCheckBox"
                  IsChecked="True" />
        <Border Background="Red" Width="20" Height="20"
                Visibility="{Binding Path=IsChecked,ElementName=MyCheckBox,
Converter={StaticResource VisibleIfTrueConverter}}" />
    </StackPanel>
</Window>
```

DataContext

WPF **DataContext** . DataContext .

```
<Window x:Class="StackOverflowDataBindingExample.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"
        xmlns:local="clr-namespace:StackOverflowDataBindingExample"
        xmlns:vm="clr-namespace:StackOverflowDataBindingExample.ViewModels"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Window.DataContext>
        <vm:HelloWorldViewModel />
    </Window.DataContext>
    ...
</Window>
```

DataContext XAML IntelliSense . DataContext XAML IntelliSense .

```
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        DataContext = new HelloWorldViewModel();
    }
}
```

DataContext (: MVVM Light) .

WPF DataContext . DataContext (: ContentPresenter) .

INotifyPropertyChanged

`INotifyPropertyChanged` (, DataContext) . WPF `PropertyChanged` UI . .

C # 6 .

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected void NotifyPropertyChanged([CallerMemberName] string name = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
    }
}
```

`NotifyPropertyChanged` .

1. `NotifyPropertyChanged()` . CallerMemberName setter .
2. `NotifyPropertyChanged(nameof(SomeOtherProperty))` , SomeOtherProperty .

C # 5.0 .NET 4.5 .

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected void NotifyPropertyChanged([CallerMemberName] string name = null)
    {
        var handler = PropertyChanged;
        if (handler != null)
        {
            handler(this, new PropertyChangedEventArgs(name));
        }
    }
}
```

4.5 .NET .

: `INotifyPropertyChanged` " C # "(POCO) . .NET . `Mode OneTime` ( ).

INotifyPropertyChanged ?

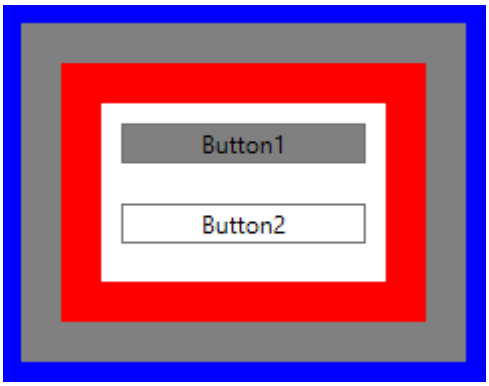
.

```
<StackPanel>
    <CheckBox x:Name="MyCheckBox" IsChecked="True" />
    <TextBlock Text="{Binding IsChecked, ElementName=MyCheckBox}" />
</StackPanel>
```

`RelativeSource` . .

```
<Grid Background="Blue">
    <Grid Background="Gray" Margin="10">
        <Border Background="Red" Margin="20">
            <StackPanel Background="White" Margin="20">
                <Button Margin="10" Content="Button1" Background="{Binding Background,
RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Grid}}}" />
                <Button Margin="10" Content="Button2" Background="{Binding Background,
RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type FrameworkElement}}}" />
            </StackPanel>
        </Border>
    </Grid>
</Grid>
```

`Grid` *Button1* . *Button2* `FrameworkElement` `StackPanel` `StackPanel` .



MultiBinding . Text StringFormat .

```
<TextBlock>
    <TextBlock.Text>
        <MultiBinding StringFormat="{}{0} {1}">
            <Binding Path="User.Forename"/>
            <Binding Path="User.Surname"/>
        </MultiBinding>
    </TextBlock.Text>
</TextBlock>
```

`StringFormat` `IMultiValueConverter` Bindings MultiBinding .

MultiBinding .

WPF : https://riptutorial.com/ko/wpf/topic/2236/wpf---

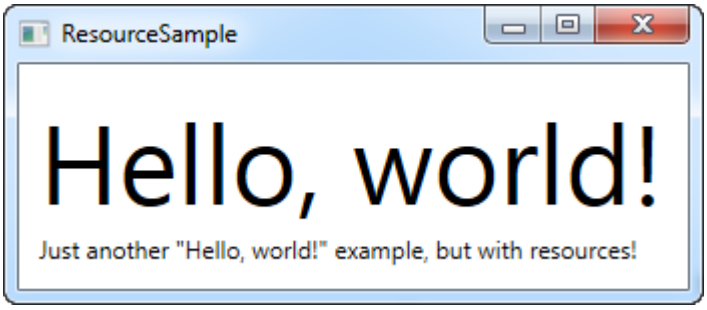
5: WPF

Examples

.

WPF . . , . WPF . . .

```
<Window x:Class="WPFApplication.ResourceSample"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="ResourceSample" Height="150" Width="350">
    <Window.Resources>
        <sys:String x:Key="strHelloWorld">Hello, world!</sys:String>
    </Window.Resources>
    <StackPanel Margin="10">
        <TextBlock Text="{StaticResource strHelloWorld}" FontSize="56" />
        <TextBlock>Just another "<TextBlock Text="{StaticResource strHelloWorld}" />" example,
but with resources!</TextBlock>
    </StackPanel>
</Window>
```



StaticResource x : Key . TextBlock .

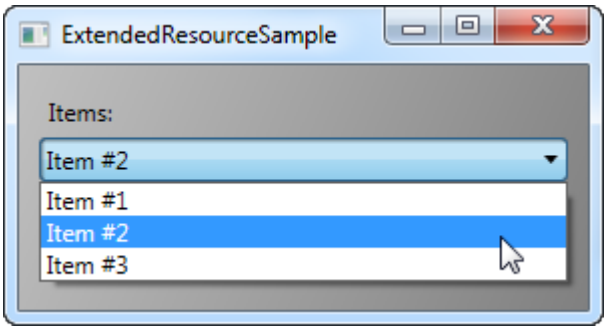
. . .

```
<Window x:Class="WPFApplication.ExtendedResourceSample"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="ExtendedResourceSample" Height="160" Width="300"
        Background="{DynamicResource WindowBackgroundBrush}">
    <Window.Resources>
        <sys:String x:Key="ComboBoxTitle">Items:</sys:String>

        <x:Array x:Key="ComboBoxItems" Type="sys:String">
            <sys:String>Item #1</sys:String>
            <sys:String>Item #2</sys:String>
            <sys:String>Item #3</sys:String>
        </x:Array>

        <LinearGradientBrush x:Key="WindowBackgroundBrush">
            <GradientStop Offset="0" Color="Silver"/>
```

```
            <GradientStop Offset="1" Color="Gray"/>
        </LinearGradientBrush>
    </Window.Resources>
    <StackPanel Margin="10">
        <Label Content="{StaticResource ComboBoxTitle}" />
        <ComboBox ItemsSource="{StaticResource ComboBoxItems}" />
    </StackPanel>
</Window>
```



Window , LinearGradientBrush . ComboBox . .

. . .

```
<StackPanel Margin="10">
    <StackPanel.Resources>
        <sys:String x:Key="ComboBoxTitle">Items:</sys:String>
    </StackPanel.Resources>
    <Label Content="{StaticResource ComboBoxTitle}" />
</StackPanel>
```

StackPanel Label . StackPanel . StackPanel .

. App.xaml WPF App.xaml . .

```
<Application x:Class="WpfSamples.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:sys="clr-namespace:System;assembly=mscorlib"
             StartupUri="WPFApplication/ExtendedResourceSample.xaml">
    <Application.Resources>
        <sys:String x:Key="ComboBoxTitle">Items:</sys:String>
    </Application.Resources>
</Application>
```

WPF App.xaml .

```
<Label Content="{StaticResource ComboBoxTitle}" />
```

Code-behind .

App.xaml :

```
<Application x:Class="WpfSamples.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:sys="clr-namespace:System;assembly=mscorlib"
             StartupUri="WPFApplication/ResourcesFromCodeBehindSample.xaml">
    <Application.Resources>
        <sys:String x:Key="strApp">Hello, Application world!</sys:String>
    </Application.Resources>
</Application>
```

:

```
<Window x:Class="WpfSamples.WPFApplication.ResourcesFromCodeBehindSample"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="ResourcesFromCodeBehindSample" Height="175" Width="250">
    <Window.Resources>
        <sys:String x:Key="strWindow">Hello, Window world!</sys:String>
    </Window.Resources>
    <DockPanel Margin="10" Name="pnlMain">
        <DockPanel.Resources>
            <sys:String x:Key="strPanel">Hello, Panel world!</sys:String>
        </DockPanel.Resources>

        <WrapPanel DockPanel.Dock="Top" HorizontalAlignment="Center" Margin="10">
            <Button Name="btnClickMe" Click="btnClickMe_Click">Click me!</Button>
        </WrapPanel>

        <ListBox Name="lbResult" />
    </DockPanel>
</Window>
```

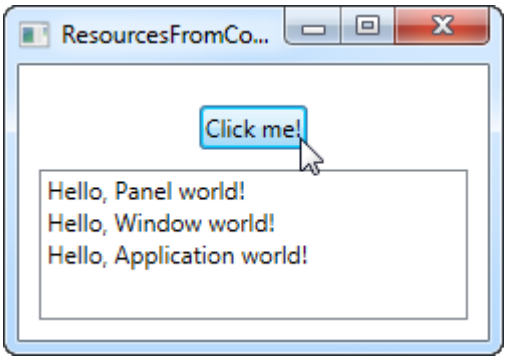
:

```
using System;
using System.Windows;

namespace WpfSamples.WPFApplication
{
        public partial class ResourcesFromCodeBehindSample : Window
        {
                public ResourcesFromCodeBehindSample()
                {
                        InitializeComponent();
                }

                private void btnClickMe_Click(object sender, RoutedEventArgs e)
                {
                        lbResult.Items.Add(pnlMain.FindResource("strPanel").ToString());
                        lbResult.Items.Add(this.FindResource("strWindow").ToString());

lbResult.Items.Add(Application.Current.FindResource("strApp").ToString());
                }
        }
}
```



, "Hello, world!" . : App.xaml , , . ListBox .

. ListBox . FindResource () . ( ) ToString () .

FindResource () . , Application . , FindResource () . .

.

WPF : https://riptutorial.com/ko/wpf/topic/4371/wpf-

6: WPF

WPF WPF . `System.Windows.Interactity Behavior` . Expression Blend SDK [nuget] [1] . [1] : https://www.nuget.org/packages/System.Windows.Interactivity.WPF/

Examples

`ScrollViewer ScrollViewer . ScrollViewer .`

```
public class BubbleMouseWheelEvents : Behavior<UIElement>
{
    protected override void OnAttached()
    {
        base.OnAttached();
        this.AssociatedObject.PreviewMouseWheel += PreviewMouseWheel;
    }

    protected override void OnDetaching()
    {
        this.AssociatedObject.PreviewMouseWheel -= PreviewMouseWheel;
        base.OnDetaching();
    }

    private void PreviewMouseWheel(object sender, MouseWheelEventArgs e)
    {
        var scrollViewer = AssociatedObject.GetChildOf<ScrollViewer>(includeSelf: true);
        var scrollPos = scrollViewer.ContentVerticalOffset;
        if ((scrollPos == scrollViewer.ScrollableHeight && e.Delta < 0) || (scrollPos == 0 &&
e.Delta > 0))
        {
            UIElement rerouteTo = AssociatedObject;
            if (ReferenceEquals(scrollViewer, AssociatedObject))
            {
                rerouteTo = (UIElement) VisualTreeHelper.GetParent(AssociatedObject);
            }

            e.Handled = true;
            var e2 = new MouseWheelEventArgs(e.MouseDevice, e.Timestamp, e.Delta);
            e2.RoutedEvent = UIElement.MouseWheelEvent;
            rerouteTo.RaiseEvent(e2);
        }
    }
}
```

Behaviors `Behavior<T> Behavior<T> . T` (`UIElement` . XAML `Behavior OnAttached . AssociatedControl . OnDetached . .`

`PreviewMouseWheel ScrollViewer . ScrollViewer ScrollViewer . e.Handled true . AssociatedObject MouseWheelEvent . .`

XAML

XAML `interactivity` xml-namespace . XAML .

xmlns : interactivity = " http://schemas.microsoft.com/expression/2010/interactivity "

.

```
<ScrollViewer>
    <!--...Content...-->
    <ScrollViewer>
        <interactivity:Interaction.Behaviors>
            <behaviors:BubbleMouseWheelEvents />
        </interactivity:Interaction.Behaviors>
        <!--...Content...-->
    </ScrollViewer>
    <!--...Content...-->.
</ScrollViewer>
```

BubbleMouseWheelEvents ScrollViewer Behaviors .

GridView ScrollViewer .

WPF : https://riptutorial.com/ko/wpf/topic/8365/wpf-

7: WPF

WPF . .

Examples

.

.

```
<Window x:Class="WPF_Style_Example.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" ResizeMode="NoResize"
    Title="MainWindow"
    Height="150" Width="200">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Button Margin="5" Content="Button 1"/>
    <Button Margin="5" Grid.Row="1" Content="Button 2"/>
</Grid>
```

Visual Studio " " " ..." .

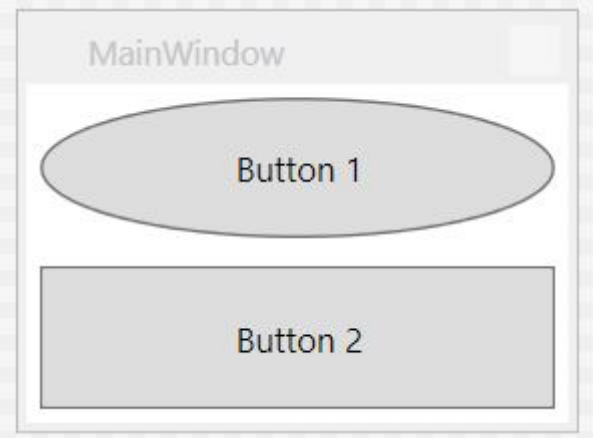
" " .

.

```
<Style x:Key="ButtonStyle1" TargetType="{x:Type Button}">
        <Setter Property="FocusVisualStyle" Value="{StaticResource FocusVisual}"/>
        <Setter Property="Background" Value="{StaticResource Button.Static.Background}"/>
        <Setter Property="BorderBrush" Value="{StaticResource Button.Static.Border}"/>
        <Setter Property="Foreground" Value="{DynamicResource {x:Static
SystemColors.ControlTextBrushKey}}"/>
        <Setter Property="BorderThickness" Value="1"/>
        <Setter Property="HorizontalContentAlignment" Value="Center"/>
        <Setter Property="VerticalContentAlignment" Value="Center"/>
        <Setter Property="Padding" Value="1"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type Button}">
                    <Grid>
                        <Ellipse x:Name="ellipse" StrokeThickness="{TemplateBinding
BorderThickness}" Stroke="{TemplateBinding BorderBrush}" Fill="{TemplateBinding Background}"
SnapsToDevicePixels="true"/>
                        <ContentPresenter x:Name="contentPresenter" Focusable="False"
HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding
Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding
SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
```

```
                    </Grid>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsDefaulted" Value="true">
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{DynamicResource {x:Static SystemColors.HighlightBrushKey}}"/>
                        </Trigger>
                        <Trigger Property="IsMouseOver" Value="true">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.MouseOver.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.MouseOver.Border}"/>
                        </Trigger>
                        <Trigger Property="IsPressed" Value="true">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.Pressed.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.Pressed.Border}"/>
                        </Trigger>
                        <Trigger Property="IsEnabled" Value="false">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.Disabled.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.Disabled.Border}"/>
                            <Setter Property="TextElement.Foreground"
TargetName="contentPresenter" Value="{StaticResource Button.Disabled.Foreground}"/>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

:



x : Key .

```
<Style TargetType="{x:Type Button}">
        <Setter Property="FocusVisualStyle" Value="{StaticResource FocusVisual}"/>
        <Setter Property="Background" Value="{StaticResource Button.Static.Background}"/>
        <Setter Property="BorderBrush" Value="{StaticResource Button.Static.Border}"/>
        <Setter Property="Foreground" Value="{DynamicResource {x:Static
SystemColors.ControlTextBrushKey}}"/>
        <Setter Property="BorderThickness" Value="1"/>
        <Setter Property="HorizontalContentAlignment" Value="Center"/>
        <Setter Property="VerticalContentAlignment" Value="Center"/>
```

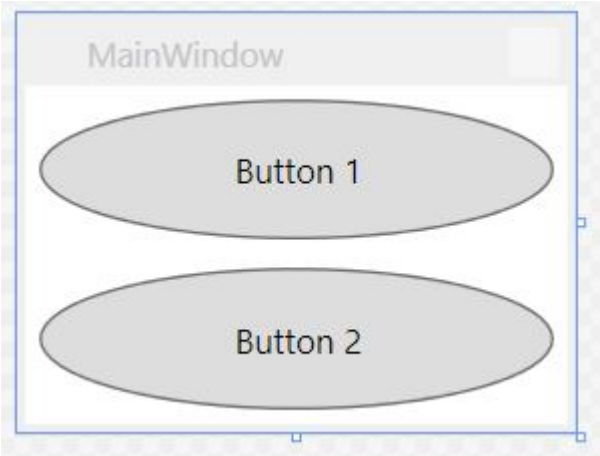
```
        <Setter Property="Padding" Value="1"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type Button}">
                    <Grid>
                        <Ellipse x:Name="ellipse" StrokeThickness="{TemplateBinding
BorderThickness}" Stroke="{TemplateBinding BorderBrush}" Fill="{TemplateBinding Background}"
SnapsToDevicePixels="true"/>
                        <ContentPresenter x:Name="contentPresenter" Focusable="False"
HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding
Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding
SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
                    </Grid>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsDefaulted" Value="true">
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{DynamicResource {x:Static SystemColors.HighlightBrushKey}}"/>
                        </Trigger>
                        <Trigger Property="IsMouseOver" Value="true">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.MouseOver.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.MouseOver.Border}"/>
                        </Trigger>
                        <Trigger Property="IsPressed" Value="true">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.Pressed.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.Pressed.Border}"/>
                        </Trigger>
                        <Trigger Property="IsEnabled" Value="false">
                            <Setter Property="Fill" TargetName="ellipse"
Value="{StaticResource Button.Disabled.Background}"/>
                            <Setter Property="Stroke" TargetName="ellipse"
Value="{StaticResource Button.Disabled.Border}"/>
                            <Setter Property="TextElement.Foreground"
TargetName="contentPresenter" Value="{StaticResource Button.Disabled.Foreground}"/>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

.

```
<Window x:Class="WPF_Style_Example.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" ResizeMode="NoResize"
    Title="MainWindow"
    Height="150" Width="200">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Button Margin="5" Content="Button 1"/>
```

```
    <Button Margin="5" Grid.Row="1" Content="Button 2"/>
</Grid>
```

.



ComboBox

`ComboBox` es :

```
<Window x:Class="WPF_Style_Example.ComboBoxWindow"
    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" ResizeMode="NoResize"
    Title="ComboBoxWindow"
    Height="100" Width="150">
<StackPanel>
    <ComboBox Margin="5" SelectedIndex="0">
        <ComboBoxItem Content="Item A"/>
        <ComboBoxItem Content="Item B"/>
        <ComboBoxItem Content="Item C"/>
    </ComboBox>
    <ComboBox IsEditable="True" Margin="5" SelectedIndex="0">
        <ComboBoxItem Content="Item 1"/>
        <ComboBoxItem Content="Item 2"/>
        <ComboBoxItem Content="Item 3"/>
    </ComboBox>
</StackPanel>
```

`ComboBox` " -> ". .

3 .

```
ComboBoxToggleButton
ComboBoxEditableTextBox
ComboBoxStyle1
```

2 :

```
ComboBoxTemplate
```

```
ComboBoxEditableTemplate
```

ComboBoxToggleButton .

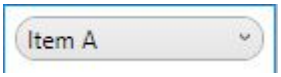
```
<SolidColorBrush x:Key="ComboBox.Static.Border" Color="#FFACACAC"/>
    <SolidColorBrush x:Key="ComboBox.Static.Editable.Background" Color="#FFFFFFFF"/>
    <SolidColorBrush x:Key="ComboBox.Static.Editable.Border" Color="#FFABADB3"/>
    <SolidColorBrush x:Key="ComboBox.Static.Editable.Button.Background" Color="Transparent"/>
    <SolidColorBrush x:Key="ComboBox.Static.Editable.Button.Border" Color="Transparent"/>
    <SolidColorBrush x:Key="ComboBox.MouseOver.Glyph" Color="#FF000000"/>
    <LinearGradientBrush x:Key="ComboBox.MouseOver.Background" EndPoint="0,1"
StartPoint="0,0">
        <GradientStop Color="Orange" Offset="0.0"/>
        <GradientStop Color="OrangeRed" Offset="1.0"/>
    </LinearGradientBrush>
    <SolidColorBrush x:Key="ComboBox.MouseOver.Border" Color="Red"/>
    <SolidColorBrush x:Key="ComboBox.MouseOver.Editable.Background" Color="#FFFFFFFF"/>
    <SolidColorBrush x:Key="ComboBox.MouseOver.Editable.Border" Color="#FF7EB4EA"/>
    <LinearGradientBrush x:Key="ComboBox.MouseOver.Editable.Button.Background" EndPoint="0,1"
StartPoint="0,0">
        <GradientStop Color="#FFEBF4FC" Offset="0.0"/>
        <GradientStop Color="#FFDCECFC" Offset="1.0"/>
    </LinearGradientBrush>
    <SolidColorBrush x:Key="ComboBox.MouseOver.Editable.Button.Border" Color="#FF7EB4EA"/>
    <SolidColorBrush x:Key="ComboBox.Pressed.Glyph" Color="#FF000000"/>
    <LinearGradientBrush x:Key="ComboBox.Pressed.Background" EndPoint="0,1" StartPoint="0,0">
        <GradientStop Color="OrangeRed" Offset="0.0"/>
        <GradientStop Color="Red" Offset="1.0"/>
    </LinearGradientBrush>
    <SolidColorBrush x:Key="ComboBox.Pressed.Border" Color="DarkRed"/>
    <SolidColorBrush x:Key="ComboBox.Pressed.Editable.Background" Color="#FFFFFFFF"/>
    <SolidColorBrush x:Key="ComboBox.Pressed.Editable.Border" Color="#FF569DE5"/>
    <LinearGradientBrush x:Key="ComboBox.Pressed.Editable.Button.Background" EndPoint="0,1"
StartPoint="0,0">
        <GradientStop Color="#FFDAEBFC" Offset="0.0"/>
        <GradientStop Color="#FFC4E0FC" Offset="1.0"/>
    </LinearGradientBrush>
    <SolidColorBrush x:Key="ComboBox.Pressed.Editable.Button.Border" Color="#FF569DE5"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Glyph" Color="#FFBFBFBF"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Background" Color="#FFF0F0F0"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Border" Color="#FFD9D9D9"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Editable.Background" Color="#FFFFFFFF"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Editable.Border" Color="#FFBFBFBF"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Editable.Button.Background"
Color="Transparent"/>
    <SolidColorBrush x:Key="ComboBox.Disabled.Editable.Button.Border" Color="Transparent"/>
    <SolidColorBrush x:Key="ComboBox.Static.Glyph" Color="#FF606060"/>
    <Style x:Key="ComboBoxToggleButton" TargetType="{x:Type ToggleButton}">
        <Setter Property="OverridesDefaultStyle" Value="true"/>
        <Setter Property="IsTabStop" Value="false"/>
        <Setter Property="Focusable" Value="false"/>
        <Setter Property="ClickMode" Value="Press"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type ToggleButton}">
                    <Border x:Name="templateRoot" CornerRadius="10"
BorderBrush="{StaticResource ComboBox.Static.Border}" BorderThickness="{TemplateBinding
BorderThickness}" Background="{StaticResource ComboBox.Static.Background}"
SnapsToDevicePixels="true">
                        <Border x:Name="splitBorder" BorderBrush="Transparent"
```

```
BorderThickness="1" HorizontalAlignment="Right" Margin="0" SnapsToDevicePixels="true"
Width="{DynamicResource {x:Static SystemParameters.VerticalScrollBarWidthKey}}">
                            <Path x:Name="arrow" Data="F1 M 0,0 L 2.667,2.66665 L 5.3334,0 L
5.3334,-1.78168 L 2.6667,0.88501 L0,-1.78168 L0,0 Z" Fill="{StaticResource
ComboBox.Static.Glyph}" HorizontalAlignment="Center" Margin="0" VerticalAlignment="Center"/>
                        </Border>
                    </Border>
                    <ControlTemplate.Triggers>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="true"/>
                                <Condition Binding="{Binding IsMouseOver,
RelativeSource={RelativeSource Self}}" Value="false"/>
                                <Condition Binding="{Binding IsPressed,
RelativeSource={RelativeSource Self}}" Value="false"/>
                                <Condition Binding="{Binding IsEnabled,
RelativeSource={RelativeSource Self}}" Value="true"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.Static.Editable.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.Static.Editable.Border}"/>
                            <Setter Property="Background" TargetName="splitBorder"
Value="{StaticResource ComboBox.Static.Editable.Button.Background}"/>
                            <Setter Property="BorderBrush" TargetName="splitBorder"
Value="{StaticResource ComboBox.Static.Editable.Button.Border}"/>
                        </MultiDataTrigger>
                        <Trigger Property="IsMouseOver" Value="true">
                            <Setter Property="BorderThickness" TargetName="templateRoot"
Value="2"/>
                        </Trigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsMouseOver,
RelativeSource={RelativeSource Self}}" Value="true"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="false"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.MouseOver.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.MouseOver.Border}"/>
                        </MultiDataTrigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsMouseOver,
RelativeSource={RelativeSource Self}}" Value="true"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="true"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.MouseOver.Editable.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.MouseOver.Editable.Border}"/>
                            <Setter Property="Background" TargetName="splitBorder"
Value="{StaticResource ComboBox.MouseOver.Editable.Button.Background}"/>
                            <Setter Property="BorderBrush" TargetName="splitBorder"
Value="{StaticResource ComboBox.MouseOver.Editable.Button.Border}"/>
                        </MultiDataTrigger>
                        <Trigger Property="IsPressed" Value="true">
```

```
                            <Setter Property="Fill" TargetName="arrow" Value="{StaticResource
ComboBox.Pressed.Glyph}"/>
                        </Trigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsPressed,
RelativeSource={RelativeSource Self}}" Value="true"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="false"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.Pressed.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.Pressed.Border}"/>
                        </MultiDataTrigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsPressed,
RelativeSource={RelativeSource Self}}" Value="true"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="true"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.Pressed.Editable.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.Pressed.Editable.Border}"/>
                            <Setter Property="Background" TargetName="splitBorder"
Value="{StaticResource ComboBox.Pressed.Editable.Button.Background}"/>
                            <Setter Property="BorderBrush" TargetName="splitBorder"
Value="{StaticResource ComboBox.Pressed.Editable.Button.Border}"/>
                        </MultiDataTrigger>
                        <Trigger Property="IsEnabled" Value="false">
                            <Setter Property="Fill" TargetName="arrow" Value="{StaticResource
ComboBox.Disabled.Glyph}"/>
                        </Trigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsEnabled,
RelativeSource={RelativeSource Self}}" Value="false"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="false"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.Disabled.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.Disabled.Border}"/>
                        </MultiDataTrigger>
                        <MultiDataTrigger>
                            <MultiDataTrigger.Conditions>
                                <Condition Binding="{Binding IsEnabled,
RelativeSource={RelativeSource Self}}" Value="false"/>
                                <Condition Binding="{Binding IsEditable,
RelativeSource={RelativeSource AncestorType={x:Type ComboBox}}}" Value="true"/>
                            </MultiDataTrigger.Conditions>
                            <Setter Property="Background" TargetName="templateRoot"
Value="{StaticResource ComboBox.Disabled.Editable.Background}"/>
                            <Setter Property="BorderBrush" TargetName="templateRoot"
Value="{StaticResource ComboBox.Disabled.Editable.Border}"/>
                            <Setter Property="Background" TargetName="splitBorder"
Value="{StaticResource ComboBox.Disabled.Editable.Button.Background}"/>
                            <Setter Property="BorderBrush" TargetName="splitBorder"
```

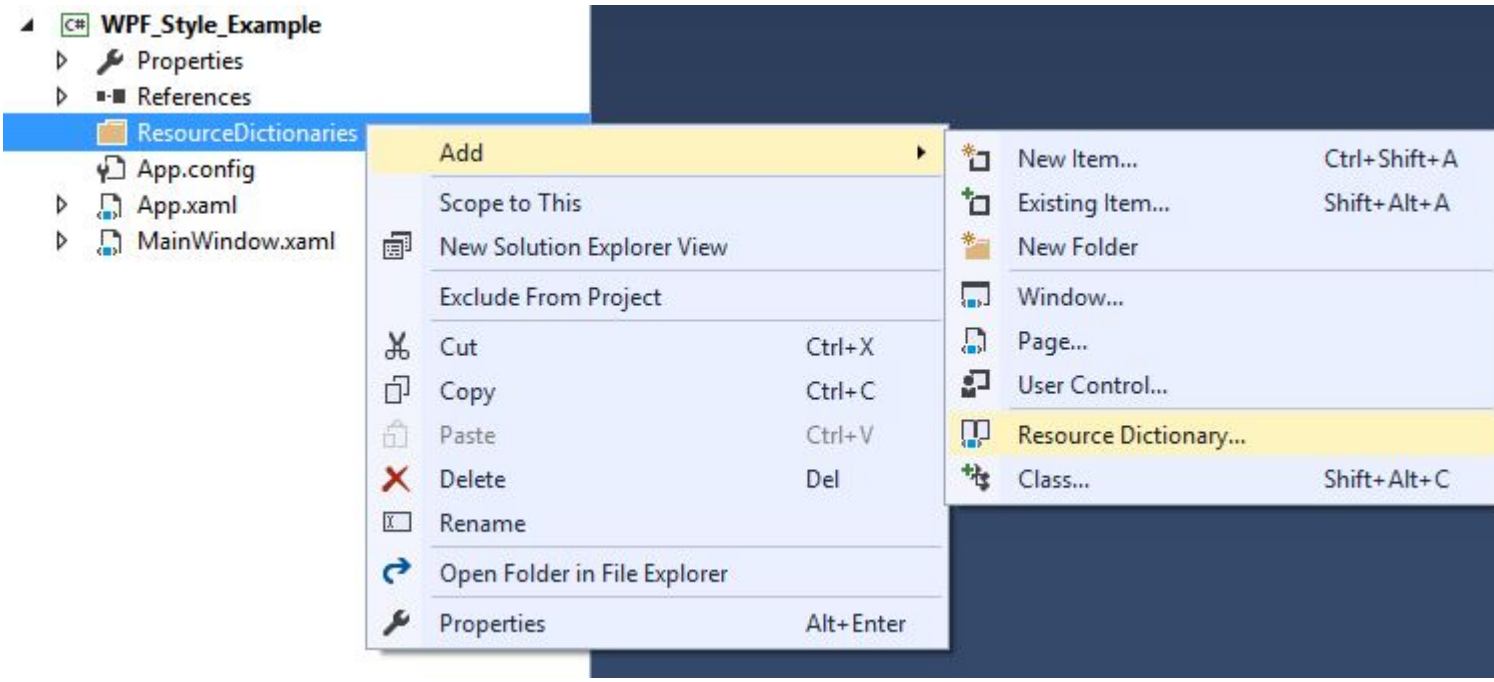
```
Value="{StaticResource ComboBox.Disabled.Editable.Button.Border}"/>
                        </MultiDataTrigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

ComboBox .



. ComboBoxEditableTextBox ComboBoxEditableTemplate .

App.xaml .



App.xaml . App.xaml .

```
<Application
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         x:Class="WPF_Style_Example.App"
         StartupUri="MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="ResourceDictionaries/Dictionary1.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

Dictionary1.xaml App.xaml . Visual Studio .

DoubleAnimation

Window .

```
<Window x:Class="WPF_Style_Example.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" ResizeMode="NoResize"
    Title="MainWindow"
    Height="150" Width="250">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Button Margin="5" Content="Button 1" Width="200"/>
    <Button Margin="5" Grid.Row="1" Content="Button 2" Width="200"/>
</Grid>
```

App.xaml 200 100, 100 200 .

```
<Style TargetType="{x:Type Button}">
    <Setter Property="FocusVisualStyle" Value="{StaticResource FocusVisual}"/>
    <Setter Property="Background" Value="{StaticResource Button.Static.Background}"/>
    <Setter Property="BorderBrush" Value="{StaticResource Button.Static.Border}"/>
    <Setter Property="Foreground" Value="{DynamicResource {x:Static
SystemColors.ControlTextBrushKey}}"/>
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="HorizontalContentAlignment" Value="Center"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
    <Setter Property="Padding" Value="1"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Grid Background="White">
                    <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}"
BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}"
SnapsToDevicePixels="true">
                        <ContentPresenter x:Name="contentPresenter" Focusable="False"
HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding
Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding
SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
                    </Border>
                </Grid>
                <ControlTemplate.Triggers>
                    <EventTrigger RoutedEvent="MouseEnter">
                        <BeginStoryboard>
                            <Storyboard>
                                <DoubleAnimation To="100" From="200"
Storyboard.TargetProperty="Width" Storyboard.TargetName="border" Duration="0:0:0.25"/>
                            </Storyboard>
                        </BeginStoryboard>
                    </EventTrigger>
                    <EventTrigger RoutedEvent="MouseLeave">
                        <BeginStoryboard>
                            <Storyboard>
```

```
                                <DoubleAnimation To="200" From="100"
Storyboard.TargetProperty="Width" Storyboard.TargetName="border" Duration="0:0:0.25"/>
                            </Storyboard>
                        </BeginStoryboard>
                    </EventTrigger>
                    <Trigger Property="IsDefaulted" Value="true">
                        <Setter Property="BorderBrush" TargetName="border"
Value="{DynamicResource {x:Static SystemColors.HighlightBrushKey}}"/>
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="true">
                        <Setter Property="Background" TargetName="border"
Value="{StaticResource Button.MouseOver.Background}"/>
                        <Setter Property="BorderBrush" TargetName="border"
Value="{StaticResource Button.MouseOver.Border}"/>
                    </Trigger>
                    <Trigger Property="IsPressed" Value="true">
                        <Setter Property="Background" TargetName="border"
Value="{StaticResource Button.Pressed.Background}"/>
                        <Setter Property="BorderBrush" TargetName="border"
Value="{StaticResource Button.Pressed.Border}"/>
                    </Trigger>
                    <Trigger Property="IsEnabled" Value="false">
                        <Setter Property="Background" TargetName="border"
Value="{StaticResource Button.Disabled.Background}"/>
                        <Setter Property="BorderBrush" TargetName="border"
Value="{StaticResource Button.Disabled.Border}"/>
                        <Setter Property="TextElement.Foreground"
TargetName="contentPresenter" Value="{StaticResource Button.Disabled.Foreground}"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

WPF : https://riptutorial.com/ko/wpf/topic/9670/wpf--

8: WPF

Examples

DispatcherObject

`Object`

```
public Dispatcher Dispatcher { get; }
```

WPF `DispatcherObject` . Dispatcher .

Dispatcher DispatcherObject . DispatcherObject DispatcherObject Dispatcher `Invoke` `BeginInvoke` `Invoke` .

DependencyObject

`DispatcherObject`

```
public object GetValue(DependencyProperty dp);
public void   SetValue(DependencyProperty dp, object value);
```

`DependencyObject` . WPF WPF `DependencyObject` .

WPF : https://riptutorial.com/ko/wpf/topic/3571/wpf-

9: WPF

. XAML C # VB .

.

- WPF : public internal.
- C # WPF XAML .
- VB WPF XAML Custom Tool `PublicVbMyResourcesResXFileCodeGenerator` .
- VB WPF Resources.resx
  - .
  - ' ' .
  -

Examples

VB XAML

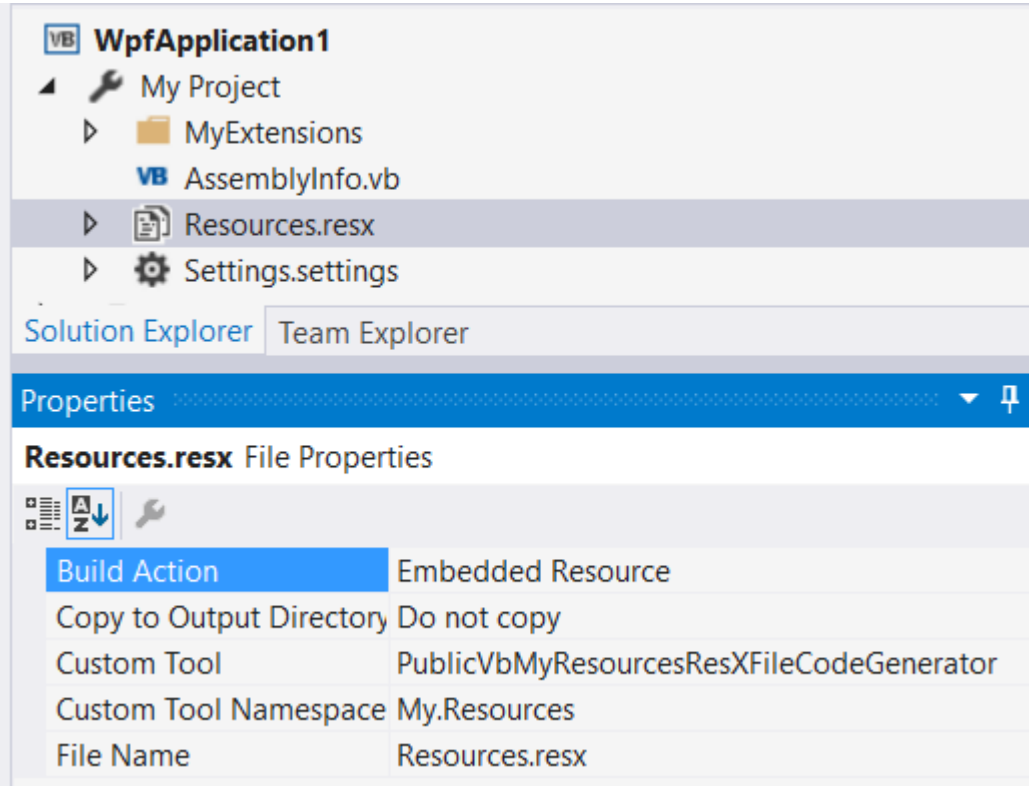
```
<Window x:Class="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:WpfApplication1"
    xmlns:my="clr-namespace:WpfApplication1.My.Resources"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
<Grid>
    <StackPanel>
        <Label Content="{Binding Source={x:Static my:Resources.MainWindow_Label_Country}}" />
    </StackPanel>
</Grid>
```

VB

VB `VbMyResourcesResXFileCodeGenerator` . (XAML) . Custom Tool `Public` .

VB WPF Resources.resx

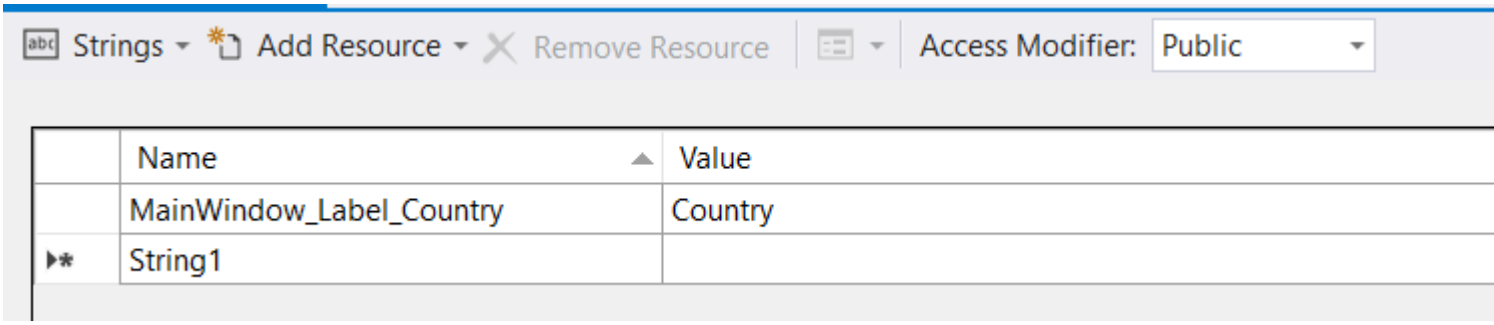
- .
- ' ' .
- " "



C # XAML

```
<Window x:Class="WpfApplication2.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"
    xmlns:local="clr-namespace:WpfApplication2"
    xmlns:resx="clr-namespace:WpfApplication2.Properties"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
<Grid>
    <StackPanel>
        <Label Content="{Binding Source={x:Static resx:Resources.MainWindow_Label_Country}}"/>
    </StackPanel>
</Grid>
```

. "" .



WPF : https://riptutorial.com/ko/wpf/topic/3905/wpf-

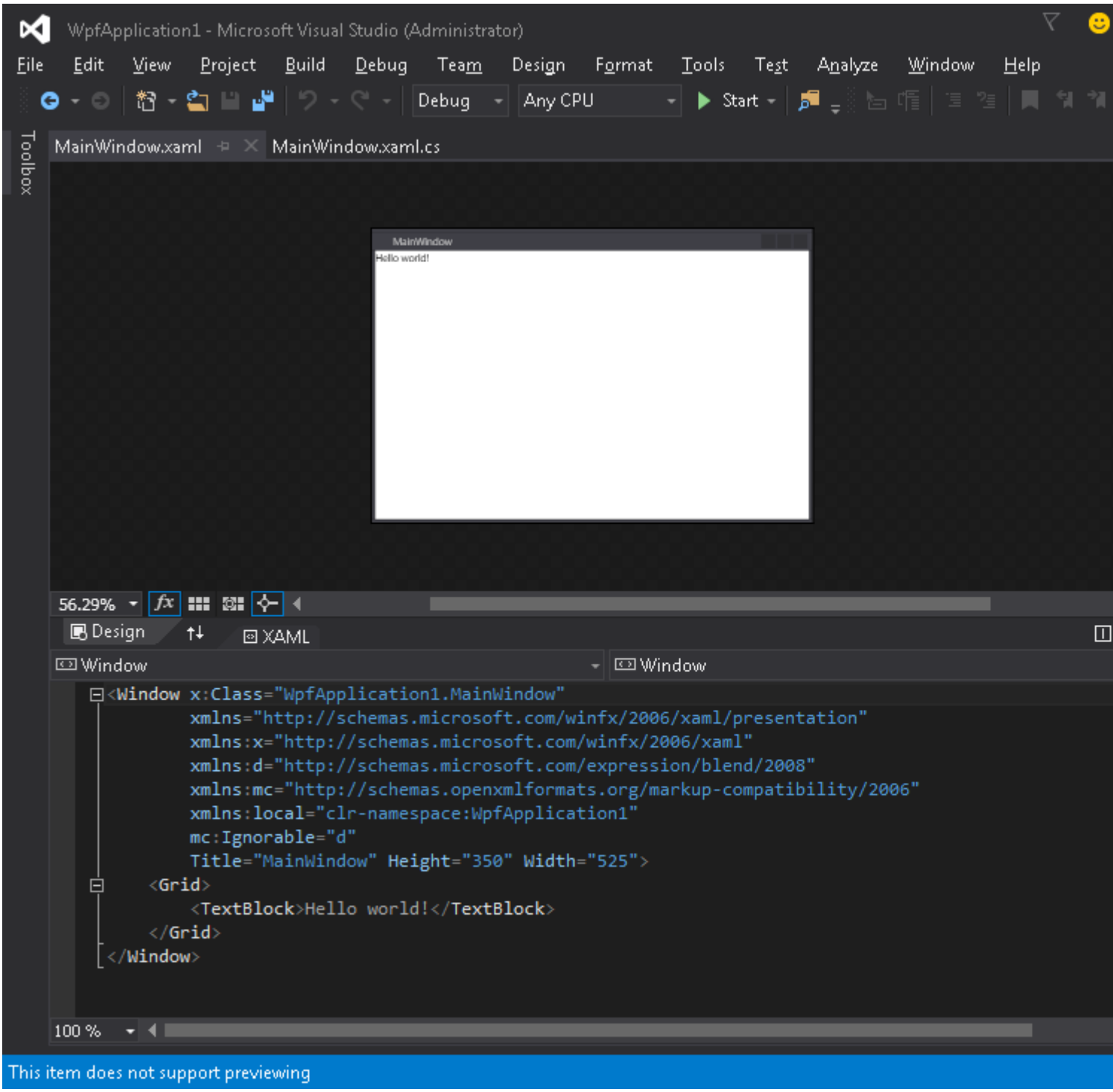
10: WPF

WPF (CLR) .NET Framework . . WPF .

Examples

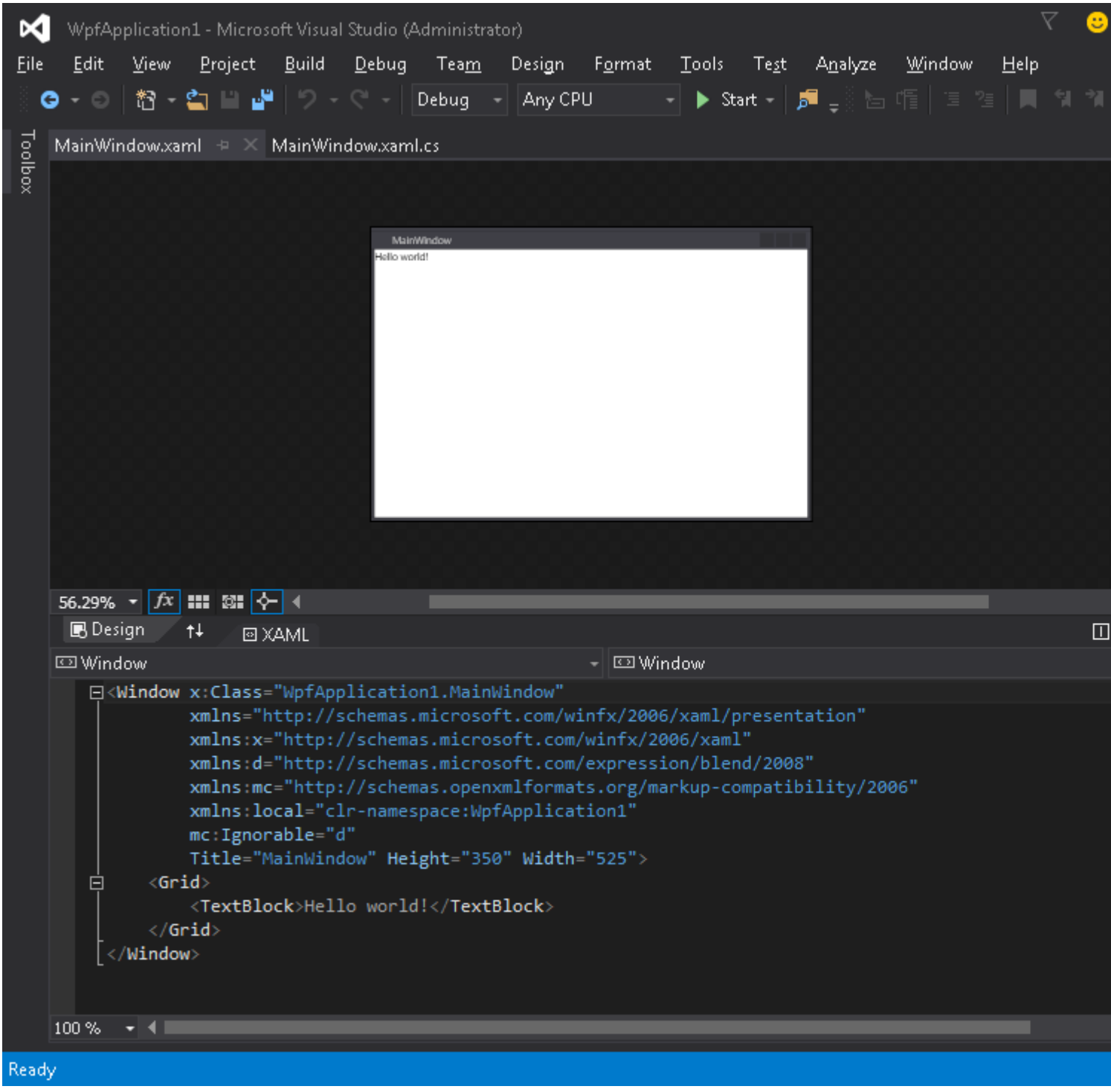
Visual Studio WPF .

1. (: *Images*).



2.

(→) **SplashScreen** :

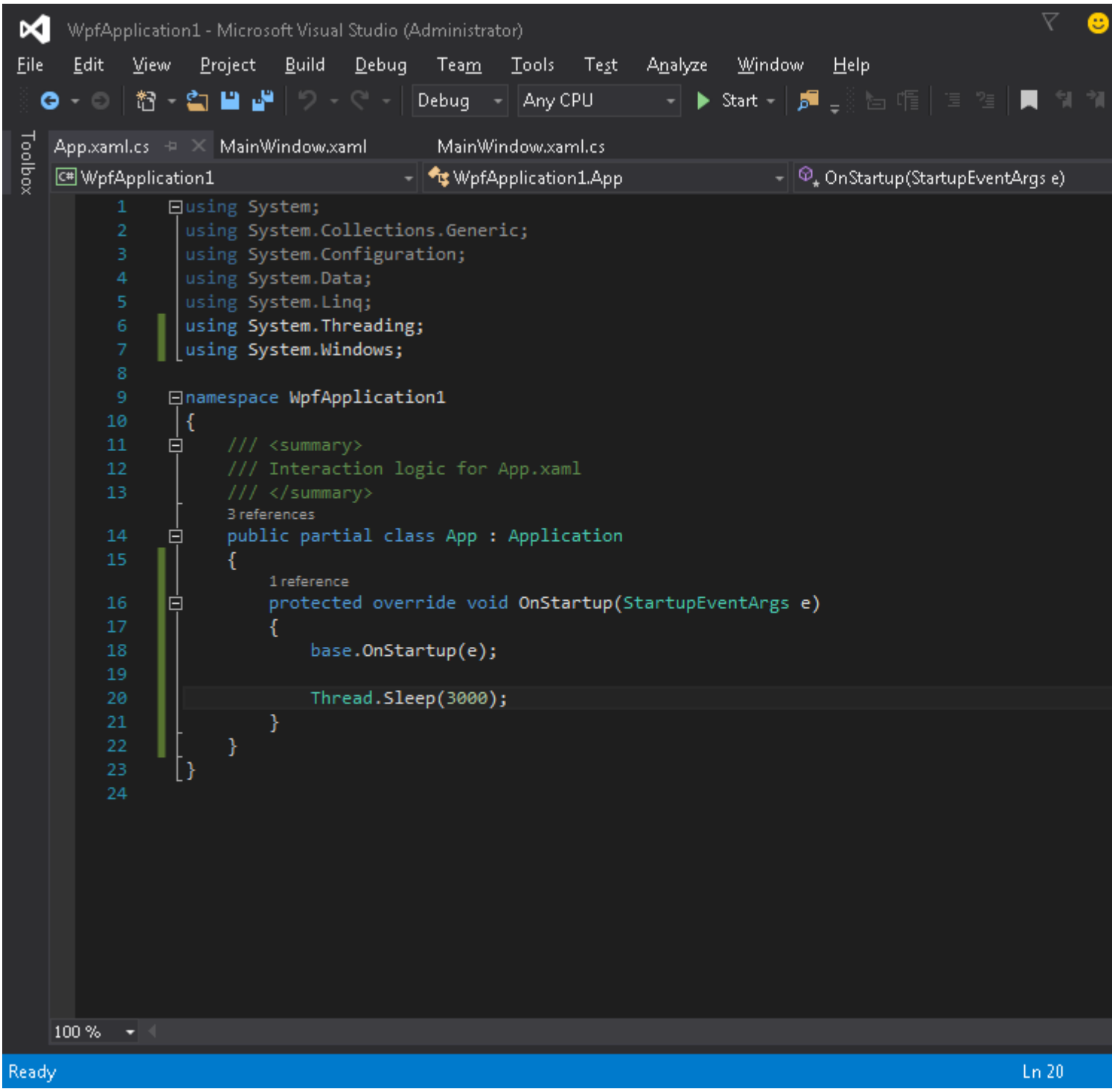


3. . (300).

.

`Application.Startup` .

1. **App.xaml.cs**
2. `using System.Threading;` `using System.Threading;` `using System.Threading;`
3. `OnStartup` `OnStartup` `Thread.Sleep(3000);` `Thread.Sleep(3000);` :



.

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading;
using System.Windows;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for App.xaml
```

```
    /// </summary>
    public partial class App : Application
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);

            Thread.Sleep(3000);
        }
    }
}
```

4. . 3 .

WPF Window . MainWindow . .

SplashScreenWindow .

```
<Window x:Class="SplashScreenExample.SplashScreenWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        Height="30"
        Width="200">
    <Grid>
        <ProgressBar IsIndeterminate="True" />
        <TextBlock HorizontalAlignment="Center"
                   VerticalAlignment="Center">Loading...</TextBlock>
    </Grid>
</Window>
```

Application.OnStartup (***App.xaml.cs***) .

```
public partial class App
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);

        //initialize the splash screen and set it as the application main window
        var splashScreen = new SplashScreenWindow();
        this.MainWindow = splashScreen;
        splashScreen.Show();

        //in order to ensure the UI stays responsive, we need to
        //do the work on a different thread
        Task.Factory.StartNew(() =>
        {
            //simulate some work being done
            System.Threading.Thread.Sleep(3000);

            //since we're not on the UI thread
            //once we're done we need to use the Dispatcher
            //to create and show the main window
            this.Dispatcher.Invoke(() =>
            {
```

```
                //initialize the main window, set it as the application main window
                //and close the splash screen
                var mainWindow = new MainWindow();
                this.MainWindow = mainWindow;
                mainWindow.Show();
                splashScreen.Close();
            });
        });
    }
}
```

`MainWindow` . ***App.xaml*** `Application StartupUri="MainWindow.xaml"` `StartupUri="MainWindow.xaml"` .

WPF `Window . MainWindow` . .

`SplashScreenWindow` .

```
<Window x:Class="SplashScreenExample.SplashScreenWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        Height="30"
        Width="200">
    <Grid>
        <ProgressBar x:Name="progressBar" />
        <TextBlock HorizontalAlignment="Center"
                   VerticalAlignment="Center">Loading...</TextBlock>
    </Grid>
</Window>
```

`SplashScreenWindow` (***SplashScreenWindow.xaml.cs***) .

```
public partial class SplashScreenWindow : Window
{
    public SplashScreenWindow()
    {
        InitializeComponent();
    }

    public double Progress
    {
        get { return progressBar.Value; }
        set { progressBar.Value = value; }
    }
}
```

`Application.OnStartup` , (***App.xaml.cs***) .

```
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
```

```
        //initialize the splash screen and set it as the application main window
        var splashScreen = new SplashScreenWindow();
        this.MainWindow = splashScreen;
        splashScreen.Show();

        //in order to ensure the UI stays responsive, we need to
        //do the work on a different thread
        Task.Factory.StartNew(() =>
        {
            //we need to do the work in batches so that we can report progress
            for (int i = 1; i <= 100; i++)
            {
                //simulate a part of work being done
                System.Threading.Thread.Sleep(30);

                //because we're not on the UI thread, we need to use the Dispatcher
                //associated with the splash screen to update the progress bar
                splashScreen.Dispatcher.Invoke(() => splashScreen.Progress = i);
            }

            //once we're done we need to use the Dispatcher
            //to create and show the main window
            this.Dispatcher.Invoke(() =>
            {
                //initialize the main window, set it as the application main window
                //and close the splash screen
                var mainWindow = new MainWindow();
                this.MainWindow = mainWindow;
                mainWindow.Show();
                splashScreen.Close();
            });
        });
    }
}
```

`MainWindow` . ***App.xaml*** `Application StartupUri="MainWindow.xaml"` `StartupUri="MainWindow.xaml"` .

WPF : https://riptutorial.com/ko/wpf/topic/3948/wpf---

11: WPF MVVM

. "" `INotifyPropertyChanged` , . , . .

MVVM , . (VM) () MVVM VM . VM . VM `TextBox` , `Button` UI .

UI . " " . UI . VM . UI .

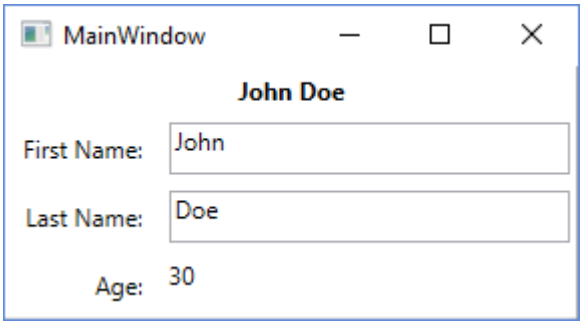
`RelayCommand` `RelayCommand` WPF WPF . .

`RelayCommand` `Microsoft.Expression.Interactivity.Core` `ActionCommand` .

Examples

WPF C # MVVM

WPF C # Windows MVVM . " " .



XAML

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

    <TextBlock Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2" Margin="4" Text="{Binding
FullName}" HorizontalAlignment="Center" FontWeight="Bold"/>

    <Label Grid.Column="0" Grid.Row="1" Margin="4" Content="First Name:"
HorizontalAlignment="Right"/>
    <!-- UpdateSourceTrigger=PropertyChanged makes sure that changes in the TextBoxes are
immediately applied to the model. -->
    <TextBox Grid.Column="1" Grid.Row="1" Margin="4" Text="{Binding FirstName,
UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Left" Width="200"/>

    <Label Grid.Column="0" Grid.Row="2" Margin="4" Content="Last Name:"
HorizontalAlignment="Right"/>
```

```
    <TextBox Grid.Column="1" Grid.Row="2" Margin="4" Text="{Binding LastName,
UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Left"  Width="200"/>

    <Label Grid.Column="0" Grid.Row="3" Margin="4" Content="Age:"
HorizontalAlignment="Right"/>
    <TextBlock Grid.Column="1" Grid.Row="3" Margin="4" Text="{Binding Age}"
HorizontalAlignment="Left"/>

</Grid>
```

```
public partial class MainWindow : Window
{
    private readonly MyViewModel _viewModel;

    public MainWindow() {
        InitializeComponent();
        _viewModel = new MyViewModel();
        // The DataContext serves as the starting point of Binding Paths
        DataContext = _viewModel;
    }
}
```

```
// INotifyPropertyChanged notifies the View of property changes, so that Bindings are updated.
sealed class MyViewModel : INotifyPropertyChanged
{
    private User user;

    public string FirstName {
        get {return user.FirstName;}
        set {
            if(user.FirstName != value) {
                user.FirstName = value;
                OnPropertyChange("FirstName");
                // If the first name has changed, the FullName property needs to be udpated as
well.
                OnPropertyChange("FullName");
            }
        }
    }

    public string LastName {
        get { return user.LastName; }
        set {
            if (user.LastName != value) {
                user.LastName = value;
                OnPropertyChange("LastName");
                // If the first name has changed, the FullName property needs to be udpated as
well.
                OnPropertyChange("FullName");
            }
        }
    }

    // This property is an example of how model properties can be presented differently to the
View.
    // In this case, we transform the birth date to the user's age, which is read only.
    public int Age {
        get {
            DateTime today = DateTime.Today;
```

```
            int age = today.Year - user.BirthDate.Year;
            if (user.BirthDate > today.AddYears(-age)) age--;
            return age;
        }
    }

    // This property is just for display purposes and is a composition of existing data.
    public string FullName {
        get { return FirstName + " " + LastName; }
    }

    public MyViewModel() {
        user = new User {
            FirstName = "John",
            LastName = "Doe",
            BirthDate = DateTime.Now.AddYears(-30)
        };
    }

    public event PropertyChangedEventHandler PropertyChanged;

    protected void OnPropertyChange(string propertyName) {
        if(PropertyChanged != null) {
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

```
sealed class User
{
    public string FirstName { get; set; }

    public string LastName { get; set; }

    public DateTime BirthDate { get; set; }
}
```

MV **VM** "VM". () . .

```
public class CustomerEditViewModel
{
    /// <summary>
    /// The customer to edit.
    /// </summary>
    public Customer CustomerToEdit { get; set; }

    /// <summary>
    /// The "apply changes" command
    /// </summary>
    public ICommand ApplyChangesCommand { get; private set; }

    /// <summary>
    /// Constructor
    /// </summary>
    public CustomerEditViewModel()
    {
        CustomerToEdit = new Customer
                         {
                             Forename = "John",
```

```
                            Surname = "Smith"
                        };

        ApplyChangesCommand = new RelayCommand(
            o => ExecuteApplyChangesCommand(),
            o => CustomerToEdit.IsValid);
    }

    /// <summary>
    /// Executes the "apply changes" command.
    /// </summary>
    private void ExecuteApplyChangesCommand()
    {
        // E.g. save your customer to database
    }
}
```

`Customer` `CustomerToEdit` .

`RelayCommand` `ApplyChangesCommand` . WPF .

`RelayCommand` . (:). . `IsValid` . false . . .

`TextBox` (`Customer`). `TextBox` Tab "" .

`INotifyPropertyChanged` (INPC) . , `Customer` `CustomerToEdit` . `TextBox` forename .

`Customer` , `DataContext` ( ). . VM INPC `CustomerToEdit` " "getter setter setter `PropertyChanged` .

`ApplyChangesCommand` INPC . , `Initialize()` .

INPC . INPC . .

M VVM "M". .

.

```
public class Customer : INotifyPropertyChanged
{
    private string _forename;
    private string _surname;
    private bool _isValid;

    public event PropertyChangedEventHandler PropertyChanged;

    /// <summary>
    /// Customer forename.
    /// </summary>
    public string Forename
    {
        get
        {
            return _forename;
        }
        set
```

```
        {
            if (_forename != value)
            {
                _forename = value;
                OnPropertyChanged();
                SetIsValid();
            }
        }
    }

    /// <summary>
    /// Customer surname.
    /// </summary>
    public string Surname
    {
        get
        {
            return _surname;
        }
        set
        {
            if (_surname != value)
            {
                _surname = value;
                OnPropertyChanged();
                SetIsValid();
            }
        }
    }

    /// <summary>
    /// Indicates whether the model is in a valid state or not.
    /// </summary>
    public bool IsValid
    {
        get
        {
            return _isValid;
        }
        set
        {
            if (_isValid != value)
            {
                _isValid = value;
                OnPropertyChanged();
            }
        }
    }

    /// <summary>
    /// Sets the value of the IsValid property.
    /// </summary>
    private void SetIsValid()
    {
        IsValid = !string.IsNullOrEmpty(Forename) && !string.IsNullOrEmpty(Surname);
    }

    /// <summary>
    /// Raises the PropertyChanged event.
    /// </summary>
    /// <param name="propertyName">Name of the property.</param>
```

```
    private void OnPropertyChanged([CallerMemberName] string propertyName = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

`PropertyChanged` `INotifyPropertyChanged` . . . `PropertyChanged` WPF . WPF .

. public . WPF "" . *WPF* .

M **V** VM "V". . Visual Studio HTML XAML .

`Customer` XAML. WPF `MainWindow.xaml` `<Window ...> </Window>` .

```
<StackPanel Orientation="Vertical"
            VerticalAlignment="Top"
            Margin="20">
    <Label Content="Forename"/>
    <TextBox Text="{Binding CustomerToEdit.Forename}"/>

    <Label Content="Surname"/>
    <TextBox Text="{Binding CustomerToEdit.Surname}"/>

    <Button Content="Apply Changes"
            Command="{Binding ApplyChangesCommand}" />
</StackPanel>
```

`TextBox` es . forename . `TextBox` `Label` "Apply" `Button` .

`TextBox` `Text` .

```
Text="{Binding CustomerToEdit.Forename}"
```

`TextBox` "path" `CustomerToEdit.Forename` . ? " "- . `CustomerToEdit` . `Customer` `Forename` " " .

`Button` XAML View-Model `ApplyChangesCommand` `Command` . VM .

DataContext

? " " . F7 `DataContext` . :

```
    public MainWindow()
    {
        InitializeComponent();

        // Our new line:-
        DataContext = new CustomerEditViewModel();
    }
```

MVVM .

MVVM

MVVM WPF `Events` .

`EventHandler` (`Code-Behind`).

```
public MainWindow()
{
    _dataGrid.CollectionChanged += DataGrid_CollectionChanged;
}

private void DataGrid_CollectionChanged(object sender,
System.Collections.Specialized.NotifyCollectionChangedEventArgs e)
{
        //Do what ever
}
```

MVVM `Commands` :

```
 <Button Command="{Binding Path=CmdStartExecution}" Content="Start" />
```

(`Cmd`) . xaml - .

MVVM `ViewModel` Command (`Button` "eq" `Button_Click`) `ViewModel` .

.

1. System.Windows.Input.Command
2. RelayCommand (: .

.

```
private RelayCommand _commandStart;
public ICommand CmdStartExecution
{
   get
   {
      if(_commandStart == null)
      {
          _commandStart = new RelayCommand(param => Start(), param => CanStart());
      }
      return _commandStart;
   }
}

public void Start()
{
   //Do what ever
}

public bool CanStart()
{
    return (DateTime.Now.DayOfWeek == DayOfWeek.Monday); //Can only click that button on
mondays.
}
```

:

ICommand xaml Control . RelayCommand Action (, Method) . Null-Check Command . RelayCommand RelayCommand . (Action<object> execute) (Action<object> execute, Predicate<object> canExecute) .

, () bool Method "" Control Control .

Method false Button Enabled="false" .

CommandParameters

```
<DataGrid x:Name="TicketsDataGrid">
    <DataGrid.InputBindings>
        <MouseBinding Gesture="LeftDoubleClick"
                      Command="{Binding CmdTicketClick}"
                      CommandParameter="{Binding ElementName=TicketsDataGrid,
                                                 Path=SelectedItem}" />
    </DataGrid.InputBindings>
<DataGrid />
```

DataGrid.SelectedItem ViewModel Click_Command .

ICommand .

```
private RelayCommand _commandTicketClick;

public ICommand CmdTicketClick
{
   get
   {
       if(_commandTicketClick == null)
       {
           _commandTicketClick = new RelayCommand(param => HandleUserClick(param));
       }
       return _commandTicketClick;
   }
}

private void HandleUserClick(object item)
{
    MyModelClass selectedItem = item as MyModelClass;
    if (selectedItem != null)
    {
        //Do sth. with that item
    }
}
```

WPF MVVM : https://riptutorial.com/ko/wpf/topic/2134/wpf-mvvm

12: WPF

WPF . (:) (:).

(: *App.xaml*) `ResourceDictionary` . `StackPanel` `TextBlock` .

```
<StackPanel>
    <StackPanel.Resources>
        <Style TargetType="TextBlock">
            <Setter Property="Margin" Value="5,5,5,0"/>
            <Setter Property="Background" Value="#FFF0F0F0"/>
            <Setter Property="Padding" Value="5"/>
        </Style>
    </StackPanel.Resources>

    <TextBlock Text="First Child"/>
    <TextBlock Text="Second Child"/>
    <TextBlock Text="Third Child"/>
</StackPanel>
```

- .
- `StaticResource` . , , / .
- `DynamicResource` `StaticResource` .

MSDN .

-
-
-

Examples

`x:Key` .

```
<StackPanel>
    <StackPanel.Resources>
        <Style x:Key="MyTextBlockStyle" TargetType="TextBlock">
            <Setter Property="Background" Value="Yellow"/>
            <Setter Property="FontWeight" Value="Bold"/>
        </Style>
    </StackPanel.Resources>

    <TextBlock Text="Yellow and bold!" Style="{StaticResource MyTextBlockStyle}" />
    <TextBlock Text="Also yellow and bold!" Style="{DynamicResource MyTextBlockStyle}" />
    <TextBlock Text="Plain text." />
</StackPanel>
```

. `TargetType` `x:Key` .

```
<StackPanel>
    <StackPanel.Resources>
        <Style TargetType="TextBlock">
            <Setter Property="Background" Value="Yellow"/>
            <Setter Property="FontWeight" Value="Bold"/>
        </Style>
    </StackPanel.Resources>

    <TextBlock Text="Yellow and bold!"  />
    <TextBlock Text="Also yellow and bold!" />
    <TextBlock Style="{x:Null}" Text="I'm not yellow or bold; I'm the control's default
style!" />
</StackPanel>
```

`TextBlock` / . `BasedOn` . .

```
<Style x:Key="BaseTextBlockStyle" TargetType="TextBlock">
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Foreground" Value="#FFBBBBBB" />
    <Setter Property="FontFamily" Value="Arial" />
</Style>

<Style x:Key="WarningTextBlockStyle"
       TargetType="TextBlock"
       BasedOn="{StaticResource BaseTextBlockStyle">
    <Setter Property="Foreground" Value="Red"/>
    <Setter Property="FontWeight" Value="Bold" />
</Style>
```

`WarningTextBlockStyle` `TextBlock` 12px Arial .

`TargetType` `x:Key` .

```
<!-- Implicit -->
<Style TargetType="TextBlock">
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Foreground" Value="#FFBBBBBB" />
    <Setter Property="FontFamily" Value="Arial" />
</Style>

<Style x:Key="WarningTextBlockStyle"
       TargetType="TextBlock"
       BasedOn="{StaticResource {x:Type TextBlock}}">
    <Setter Property="Foreground" Value="Red"/>
    <Setter Property="FontWeight" Value="Bold" />
</Style>
```

WPF : https://riptutorial.com/ko/wpf/topic/4090/wpf-

13:

	.
	values .
targetType	.
	.
	.

IValueConverter IMultiValueConverter ?

IValueConverter IMultiValueConverter - .

()

1. 0 .
2. .
3. .
4. .

.

. IValueConverter IMultiValueConverter . .

Examples

- BooleanToVisibilityConverter [IValueConverter]

. bool Visibility .

: **System.Windows.Controls .**

```
public sealed class BooleanToVisibilityConverter : IValueConverter
{
    /// <summary>
    /// Convert bool or Nullable bool to Visibility
    /// </summary>
    /// <param name="value">bool or Nullable bool</param>
    /// <param name="targetType">Visibility</param>
    /// <param name="parameter">null</param>
    /// <param name="culture">null</param>
    /// <returns>Visible or Collapsed</returns>
    public object Convert(object value, Type targetType, object parameter, CultureInfo
```

```
culture)
    {
        bool bValue = false;
        if (value is bool)
        {
            bValue = (bool)value;
        }
        else if (value is Nullable<bool>)
        {
            Nullable<bool> tmp = (Nullable<bool>)value;
            bValue = tmp.HasValue ? tmp.Value : false;
        }
        return (bValue) ? Visibility.Visible : Visibility.Collapsed;
    }

    /// <summary>
    /// Convert Visibility to boolean
    /// </summary>
    /// <param name="value"></param>
    /// <param name="targetType"></param>
    /// <param name="parameter"></param>
    /// <param name="culture"></param>
    /// <returns></returns>
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        if (value is Visibility)
        {
            return (Visibility)value == Visibility.Visible;
        }
        else
        {
            return false;
        }
    }
}
```

1.

```
<BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
```

3. .

```
<Button Visibility="{Binding AllowEditing,
                             Converter={StaticResource BooleanToVisibilityConverter}}"/>
```

[IValueConverter]

. bool Visibility . Inverted True .

```
public class BooleanToVisibilityConverter : IValueConverter
{
    public bool Inverted { get; set; }

    /// <summary>
    /// Convert bool or Nullable bool to Visibility
```

```
    /// </summary>
    /// <param name="value">bool or Nullable bool</param>
    /// <param name="targetType">Visibility</param>
    /// <param name="parameter">null</param>
    /// <param name="culture">null</param>
    /// <returns>Visible or Collapsed</returns>
    public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        bool bValue = false;
        if (value is bool)
        {
            bValue = (bool)value;
        }
        else if (value is Nullable<bool>)
        {
            Nullable<bool> tmp = (Nullable<bool>)value;
            bValue = tmp ?? false;
        }

        if (Inverted)
            bValue = !bValue;
        return (bValue) ? Visibility.Visible : Visibility.Collapsed;
    }

    /// <summary>
    /// Convert Visibility to boolean
    /// </summary>
    /// <param name="value"></param>
    /// <param name="targetType"></param>
    /// <param name="parameter"></param>
    /// <param name="culture"></param>
    /// <returns>True or False</returns>
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        if (value is Visibility)
        {
            return ((Visibility) value == Visibility.Visible) && !Inverted;
        }

        return false;
    }
}
```

1.

```
xmlns:converters="clr-namespace:MyProject.Converters;assembly=MyProject"
```

2.

```
<converters:BooleanToVisibilityConverter x:Key="BoolToVisibilityInvertedConverter"
                                         Inverted="False"/>
```

3. .

```
<Button Visibility="{Binding AllowEditing, Converter={StaticResource
BoolToVisibilityConverter}}"/>
```

[IMultiValueConverter]

`IMultiValueConverter` xaml `MultiBinding` `.values` .

```
public class AddConverter : IMultiValueConverter
{
    public object Convert(object[] values, Type targetType, object parameter, CultureInfo
culture)
    {
        decimal sum = 0M;

        foreach (string value in values)
        {
            decimal parseResult;
            if (decimal.TryParse(value, out parseResult))
            {
                sum += parseResult;
            }
        }

        return sum.ToString(culture);
    }

    public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
CultureInfo culture)
    {
        throw new NotSupportedException();
    }
}
```

1.

```
xmlns:converters="clr-namespace:MyProject.Converters;assembly=MyProject"
```

2.

```
<converters:AddConverter x:Key="AddConverter"/>
```

3. .

```
<StackPanel Orientation="Vertical">
    <TextBox x:Name="TextBox" />
    <TextBox x:Name="TextBox1" />
    <TextBlock >
        <TextBlock.Text>
            <MultiBinding Converter="{StaticResource AddConverter}">
                <Binding Path="Text" ElementName="TextBox"/>
                <Binding Path="Text" ElementName="TextBox1"/>
            </MultiBinding>
         </TextBlock.Text>
    </TextBlock>
</StackPanel>
```

ConverterParameter

`ConverterParameter` `ConverterParameter` . ConverterParameter .

```
public class MultiplyConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        if (value == null)
            return 0;

        if (parameter == null)
            parameter = 1;

        double number;
        double coefficient;

        if (double.TryParse(value.ToString(), out number) &&
double.TryParse(parameter.ToString(), out coefficient))
        {
            return number * coefficient;
        }

        return 0;
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        throw new NotSupportedException();
    }
}
```

1.

```
xmlns:converters="clr-namespace:MyProject.Converters;assembly=MyProject"
```

2.

```
<converters:MultiplyConverter x:Key="MultiplyConverter"/>
```

3. .

```
<StackPanel Orientation="Vertical">
    <TextBox x:Name="TextBox" />
    <TextBlock Text="{Binding Path=Text,
                             ElementName=TextBox,
                             Converter={StaticResource MultiplyConverter},
                             ConverterParameter=10}"/>
</StackPanel>
```

[IValueConverter]

.

```
public class ValueConverterGroup : List<IValueConverter>, IValueConverter
```

```
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        return this.Aggregate(value, (current, converter) => converter.Convert(current,
targetType, parameter, culture));
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        throw new NotSupportedException();
    }
}
```

EnumToBooleanConverter BooleanToVisibilityConverter .

```
<local:ValueConverterGroup x:Key="EnumToVisibilityConverter">
    <local:EnumToBooleanConverter/>
    <local:BooleanToVisibilityConverter/>
</local:ValueConverterGroup>
```

CurrentMode Ready .

```
<Button Content="Ok" Visibility="{Binding Path=CurrentMode, Converter={StaticResource
EnumToVisibilityConverter}, ConverterParameter={x:Static local:Mode.Ready}"/>
```

MarkupExtension .

.

```
<converters:SomeConverter x:Key="SomeConverter"/>
```

MarkupExtension ProvideValue . .

```
namespace MyProject.Converters
{
public class Converter_Negative : MarkupExtension, IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        return this.ReturnNegative(value);
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        return this.ReturnNegative(value);
    }

    private object ReturnNegative(object value)
    {
        object result = null;
        var @switch = new Dictionary<Type, Action> {
```

```
            { typeof(bool), () => result=!(bool)value },
            { typeof(byte), () => result=-1*(byte)value },
            { typeof(short), () => result=-1*(short)value },
            { typeof(int), () => result=-1*(int)value },
            { typeof(long), () => result=-1*(long)value },
            { typeof(float), () => result=-1f*(float)value },
            { typeof(double), () => result=-1d*(double)value },
            { typeof(decimal), () => result=-1m*(decimal)value }
        };

        @switch[value.GetType()]();
        if (result == null) throw new NotImplementedException();
        return result;
    }

    public Converter_Negative()
        : base()
    {
    }

    private static Converter_Negative _converter = null;

    public override object ProvideValue(IServiceProvider serviceProvider)
    {
        if (_converter == null) _converter = new Converter_Negative();
        return _converter;
    }
}
}
```

:

1. xmlns : converters = "clr-namespace : MyProject.Converters; assembly = MyProject"

2. ```
   <RichTextBox IsReadOnly="{Binding Path=IsChecked, ElementName=toggleIsEnabled,
   Converter={converters:Converter_Negative}}"/>
   ```

IMultiValueConverter

`IMultiValueConverter MultiBinding  CommandParameter     .`

```
namespace MyProject.Converters
{
    public class Converter_MultipleCommandParameters : MarkupExtension, IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType, object parameter, CultureInfo
culture)
        {
            return values.ToArray();
        }
        public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
CultureInfo culture)
        {
            throw new NotSupportedException();
        }
```
```

```
        private static Converter_MultipleCommandParameters _converter = null;

        public override object ProvideValue(IServiceProvider serviceProvider)
        {
            if (_converter == null) _converter = new Converter_MultipleCommandParameters();
            return _converter;
        }

        public Converter_MultipleCommandParameters()
            : base()
        {
        }
    }
}
```

:

1. - SomeCommand SomeCommand (: *DelegateCommand* *ICommand*).

```
 private ICommand _SomeCommand;
 public ICommand SomeCommand
 {
     get { return _SomeCommand ?? (_SomeCommand = new DelegateCommand(a =>
OnSomeCommand(a))); }
 }

 private void OnSomeCommand(object item)
 {
     object[] parameters = item as object[];

     MessageBox.Show(
         string.Format("Execute command: {0}\nParameter 1: {1}\nParamter 2: {2}\nParamter
3: {3}",
         "SomeCommand", parameters[0], parameters[1], parameters[2]));
 }
```

2. xmlns : converters = "clr-namespace : MyProject.Converters; assembly = MyProject"

3.
```
<Button Width="150" Height="23" Content="Execute some command" Name="btnTestSomeCommand"
     Command="{Binding Path=SomeCommand}" >
     <Button.CommandParameter>
         <MultiBinding Converter="{converters:Converter_MultipleCommandParameters}">
             <Binding RelativeSource="{RelativeSource Self}" Path="IsFocused"/>
             <Binding RelativeSource="{RelativeSource Self}" Path="Name"/>
             <Binding RelativeSource="{RelativeSource Self}" Path="ActualWidth"/>
         </MultiBinding>
     </Button.CommandParameter>
 </Button>
```

: https://riptutorial.com/ko/wpf/topic/3950/----

14:

Examples

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Column="1" Text="abc"/>
    <TextBlock Grid.Row="1" Grid.Column="1" Text="def"/>
</Grid>
```

RowDefinition ColumnDefinition .

Grid . Grid.Row Grid.Column 0. 0.

. .

/

Grid.RowSpan Grid.ColumnSpan Grid . TextBlock Grid .

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Column="2" Text="abc"/>
    <TextBlock Grid.Column="1" Grid.ColumnSpan="2" Text="def"/>
</Grid>
```

Grid SharedSizeGroup . Grid Grid.IsSharedSizeScope True .

```
<StackPanel Grid.IsSharedSizeScope="True">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" SharedSizeGroup="MyGroup"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        [...]
    </Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" SharedSizeGroup="MyGroup"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        [...]
```

```
    </Grid>
</StackPanel>
```

Grid .

: https://riptutorial.com/ko/wpf/topic/6483/-

15: UserControls

UserControl Control . UserControl XAML . . . UserControl .

Examples

ComboBox

UserControl ComboBox .

UserControl . ComboBox Label .

CustomComboBox.xaml

```
<UserControl x:Class="UserControlDemo.CustomComboBox"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:cnvrt="clr-namespace:UserControlDemo"
             x:Name="customComboBox">
    <UserControl.Resources>
        <cnvrt:InverseNullVisibilityConverter x:Key="invNullVisibleConverter" />
    </UserControl.Resources>
    <Grid>
        <ComboBox x:Name="comboBox"
                  ItemsSource="{Binding ElementName=customComboBox, Path=MyItemsSource}"
                  SelectedItem="{Binding ElementName=customComboBox, Path=MySelectedItem}"
                  HorizontalContentAlignment="Left" VerticalContentAlignment="Center"/>

        <Label HorizontalAlignment="Left" VerticalAlignment="Center"
               Margin="0,2,20,2" IsHitTestVisible="False"
               Content="{Binding ElementName=customComboBox, Path=DefaultText}"
               Visibility="{Binding ElementName=comboBox, Path=SelectedItem,
Converter={StaticResource invNullVisibleConverter}}"/>
    </Grid>
</UserControl>
```

UserControl . ComboBox .

.

- UserControl `x:Name` . . , .
- ComboBox `ElementName` UserControl . UserControl .
- . Label ComboBox . `IsHitTestVisible=false` `IsHitTestVisible=false` . ComboBox .
- Label `InverseNullVisibility` . .

CustomComboBox.xaml.cs

```
public partial class CustomComboBox : UserControl
{
    public CustomComboBox()
    {
```

```
        InitializeComponent();
    }

    public static DependencyProperty DefaultTextProperty =
        DependencyProperty.Register("DefaultText", typeof(string), typeof(CustomComboBox));

    public static DependencyProperty MyItemsSourceProperty =
        DependencyProperty.Register("MyItemsSource", typeof(IEnumerable),
typeof(CustomComboBox));

    public static DependencyProperty SelectedItemProperty =
        DependencyProperty.Register("SelectedItem", typeof(object), typeof(CustomComboBox));

    public string DefaultText
    {
        get { return (string)GetValue(DefaultTextProperty); }
        set { SetValue(DefaultTextProperty, value); }
    }

    public IEnumerable MyItemsSource
    {
        get { return (IEnumerable)GetValue(MyItemsSourceProperty); }
        set { SetValue(MyItemsSourceProperty, value); }
    }

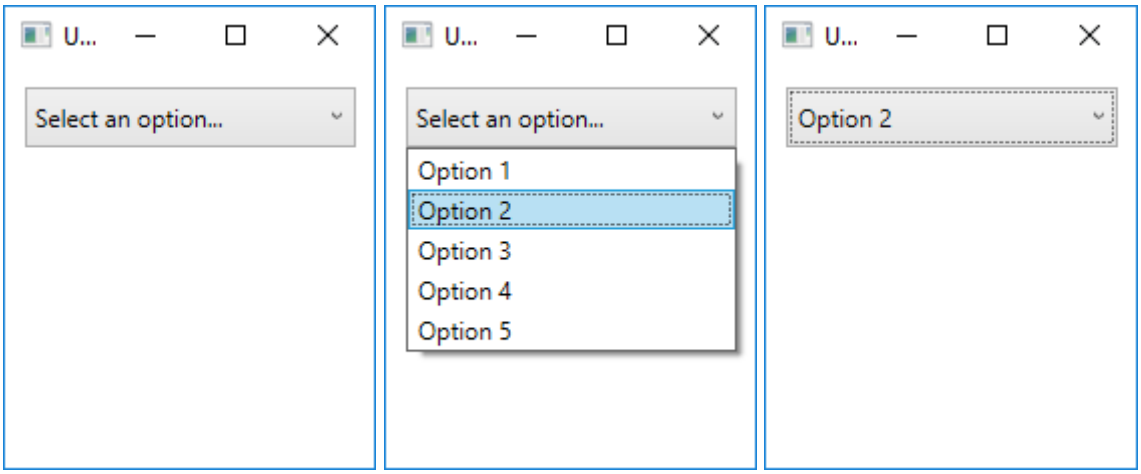
    public object MySelectedItem
    {
        get { return GetValue(MySelectedItemProperty); }
        set { SetValue(MySelectedItemProperty, value); }
    }
}
```

UserControl . ComboBox (: ComboBox `ItemsSource` `MyItemsSource` . .

`CustomComboBox` UserControl .

```
<Window x:Class="UserControlDemo.UserControlDemo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:cntrls="clr-namespace:UserControlDemo"
        Title="UserControlDemo" Height="240" Width=200>
    <Grid>
        <cntrls:CustomComboBox HorizontalAlignment="Left" Margin="10,10,0,0"
VerticalAlignment="Top" Width="165"
                               MyItemsSource="{Binding Options}"
                               MySelectedItem="{Binding SelectedOption, Mode=TwoWay}"
                               DefaultText="Select an option..."/>
    <Grid>
</Window>
```

:



InverNullVisibilityConverter UserControl Label . III .

```
public class InverseNullVisibilityConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        return value == null ? Visibility.Visible : Visibility.Hidden;
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
    {
        throw new NotImplementedException();
    }
}
```

UserControls : https://riptutorial.com/ko/wpf/topic/6508/-----usercontrols-

16:

ProvideValue	MarkupExtension XAML .

, .

XAML ({}) . .

. (<>) XAML .

https://msdn.microsoft.com/en-us/library/ms747254(v=vs.110).aspx .

Examples

IValueConverter

IValueConverter . BoolToVisibilityConverter . XAML .

```
Visibility="{Binding [BoolProperty], Converter={[xmlns]:BoolToVisibilityConverter}}"
```

bool .

```
public class BoolToVisibilityConverter : MarkupExtension, IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
        {
            if (value is bool)
                return (bool)value ? Visibility.Visible : Visibility.Collapsed;
            else
                return Visibility.Collapsed;
        }

        public object ConvertBack(object value, Type targetType, object parameter, CultureInfo
culture)
        {
            if (value is Visibility)
            {
                if ((Visibility)value == Visibility.Visible)
                    return true;
                else
                    return false;
            }
            else
                return false;
        }

        public override object ProvideValue(IServiceProvider serviceProvider)
        {
            return this;
        }
```

```
    }
```

XAML

XAML .

`x:Type` Type . .

```
<object property="{x:Type prefix:typeNameValue}" .../>
```

`x:Static` . .

```
<object property="{x:Static prefix:typeName.staticMemberName}" .../>
```

`x:Null` null .

```
<object property="{x:Null}" .../>
```

`x:Array` WPF XAML .

```
<x:Array Type="typeName">
  arrayContents
</x:Array>
```

: https://riptutorial.com/ko/wpf/topic/6619/--

17:

Trigger , DataTrigger , MultiTrigger , MultiDataTrigger EventTrigger WPF .

FrameworkElement FrameworkContentElement . .

- EventTrigger <Style> . EventTrigger <Style> Triggers .
- <Trigger> <Setter> . <Trigger> .
- <Setter> . <TextBlock Text="Sample"> . Text .
- .

Examples

Trigger .

```
<TextBlock>
    <TextBlock.Style>
        <Style TargetType="{x:Type TextBlock}">
            <Style.Triggers>
                <Trigger Property="Text" Value="Pass">
                    <Setter Property="Foreground" Value="Green"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </TextBlock.Style>
</TextBlock>
```

Text "Pass" TextBlock .

MultiTrigger

MultiTrigger Trigger . MultiTrigger . <Condition> .

```
<TextBlock x:Name="_txtBlock" IsEnabled="False">
    <TextBlock.Style>
        <Style TargetType="{x:Type TextBlock}">
            <Style.Triggers>
                <MultiTrigger>
                    <MultiTrigger.Conditions>
                        <Condition Property="Text" Value="Pass"/>
                        <Condition Property="IsEnabled" Value="True"/>
                    </MultiTrigger.Conditions>
                    <Setter Property="Foreground" Value="Green"/>
                </MultiTrigger>
            </Style.Triggers>
        </Style>
    </TextBlock.Style>
</TextBlock>
```

MultiTrigger .

DataTrigger

`DataTrigger` , UI . .

```
public class Cheese
{
    public string Name { get; set; }
    public double Age { get; set; }
    public int StinkLevel { get; set; }
}
```

`TextBlock` `DataContext` .

```
<TextBlock Text="{Binding Name}">
    <TextBlock.DataContext>
        <local:Cheese Age="12" StinkLevel="100" Name="Limburger"/>
    </TextBlock.DataContext>
    <TextBlock.Style>
        <Style TargetType="{x:Type TextBlock}">
            <Style.Triggers>
                <DataTrigger Binding="{Binding StinkLevel}" Value="100">
                    <Setter Property="Foreground" Value="Green"/>
                </DataTrigger>
            </Style.Triggers>
        </Style>
    </TextBlock.Style>
</TextBlock>
```

`TextBlock.Foreground` . `StinkLevel` 100 XAML `Text.Foreground` .

: https://riptutorial.com/ko/wpf/topic/9624/

18: UI

Examples

UI

UI . .net . UI . .

.

. . . :

```
//Prepare the action
Action taskAction = new Action( () => {
    int progress = 0;
    Action invokeAction = new Action( () => { progressBar.Value = progress; });
    while (progress <= 100) {
        progress = CalculateSomething();
        progressBar.Dispatcher.Invoke( invokeAction );
    }
} );

//After .net 4.5
Task.Run( taskAction );

//Before .net 4.5
Task.Factory.StartNew( taskAction ,
    CancellationToken.None,
    TaskCreationOptions.DenyChildAttach,
    TaskScheduler.Default);
```

UI `DispatcherObject` (`System.Windows.Threading`) Dispatcher . Dispatcher Dispatcher . . dispatcher delegate `int progress` .

UI `invokeAction` .

```
//Prepare the action
Action taskAction = new Action( () => {
    int progress = 0;
    Action<int> invokeAction = new Action<int>( (i) => { progressBar.Value = i; } )
    while (progress <= 100) {
        progress = CalculateSomething();
        progressBar.Dispatcher.BeginInvoke(
            invokeAction,
            progress );
    }
} );

//After .net 4.5
Task.Run( taskAction );

//Before .net 4.5
Task.Factory.StartNew( taskAction ,
    CancellationToken.None,
```

```
    TaskCreationOptions.DenyChildAttach,
    TaskScheduler.Default);
```

int progress   .

UI   : https://riptutorial.com/ko/wpf/topic/6128/--ui--

19: :

(float)	Dependency .

- XAML *xmlns : i* *System.Windows.Interactivity* .
- *xmlns : b* .
- XAML .

Examples

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Interactivity;

namespace MyBehaviorAssembly
{

public class SliderDragEndValueBehavior : Behavior<Slider>
{

    public static readonly DependencyProperty ValueProperty = DependencyProperty.Register(
        "Value", typeof (float), typeof (SliderDragEndValueBehavior), new
PropertyMetadata(default(float)));

    public float Value
    {
        get { return (float) GetValue(ValueProperty); }
        set { SetValue(ValueProperty, value); }
    }

    protected override void OnAttached()
    {
        RoutedEventHandler handler = AssociatedObject_DragCompleted;
        AssociatedObject.AddHandler(Thumb.DragCompletedEvent, handler);
    }

    private void AssociatedObject_DragCompleted(object sender, RoutedEventArgs e)
    {
        Value = (float) AssociatedObject.Value;
    }

    protected override void OnDetaching()
    {
        RoutedEventHandler handler = AssociatedObject_DragCompleted;
        AssociatedObject.RemoveHandler(Thumb.DragCompletedEvent, handler);
    }
}
}
```

XAML

```
<UserControl x:Class="Example.View"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
         xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

         xmlns:b="MyBehaviorAssembly;assembly=MyBehaviorAssembly"

         mc:Ignorable="d"
         d:DesignHeight="200" d:DesignWidth="200"

         >
           <Slider>
            <i:Interaction.Behaviors>
                <b:SliderDragEndValueBehavior

                        Value="{Binding Value, Mode=OneWayToSource,
UpdateSourceTrigger=PropertyChanged}"

                        />
            </i:Interaction.Behaviors>
          </Slider>

</UserControl>
```

: : https://riptutorial.com/ko/wpf/topic/6339/------

20:

`System.Speech` Microsoft .

1. SpeechSynthesizer speechSynthesizerObject = SpeechSynthesizer ();

   speechSynthesizerObject.Speak ("Text to Speak");

Examples

- Hello World

```
using System;
using System.Speech.Synthesis;
using System.Windows;

namespace Stackoverflow.SpeechSynthesisExample
{
        public partial class SpeechSynthesisSample : Window
        {
                public SpeechSynthesisSample()
                {
                        InitializeComponent();
                        SpeechSynthesizer speechSynthesizer = new SpeechSynthesizer();
                        speechSynthesizer.Speak("Hello, world!");
                }
        }
}
```

: https://riptutorial.com/ko/wpf/topic/8368/-

21:

PixelHeight (System.Int32)	
PixelWidth (System.Int32)	
PixelFormat (System.Windows.Media.PixelFormat)	
	IList <T> - C #
DpiX	Dpi - .
DpiY	Dpi - .

- XAML *xmlns : i* *System.Windows.Interactivity* .
- *xmlns : b* .
- XAML .
- *IList* . C # *IList* .
  .
- .
- -
- Pixel, PixelWidth, PixelHeight PixelFormat .
- .

Examples

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Interactivity;
using System.Windows.Media;
using System.Windows.Media.Imaging;

namespace MyBehaviorAssembly
{
    public class PixelSupportBehavior : Behavior<Image>
    {

    public static readonly DependencyProperty PixelsProperty = DependencyProperty.Register(
        "Pixels", typeof (IList<byte>), typeof (PixelSupportBehavior), new
PropertyMetadata(default(IList<byte>),OnPixelsChanged));

    private static void OnPixelsChanged(DependencyObject d, DependencyPropertyChangedEventArgs
e)
    {
        var b = (PixelSupportBehavior) d;
        var pixels = (IList<byte>) e.NewValue;
```

```
        b.RenderPixels(pixels);
    }

    public IList<byte> Pixels
    {
        get { return (IList<byte>) GetValue(PixelsProperty); }
        set { SetValue(PixelsProperty, value); }
    }

    public static readonly DependencyProperty PixelFormatProperty =
DependencyProperty.Register(
        "PixelFormat", typeof (PixelFormat), typeof (PixelSupportBehavior), new
PropertyMetadata(PixelFormats.Default,OnPixelFormatChanged));


    private static void OnPixelFormatChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
    {
        var b = (PixelSupportBehavior) d;
        var pixelFormat = (PixelFormat) e.NewValue;

        if(pixelFormat==PixelFormats.Default)
            return;

        b._pixelFormat = pixelFormat;

        b.InitializeBufferIfAttached();
    }

    public PixelFormat PixelFormat
    {
        get { return (PixelFormat) GetValue(PixelFormatProperty); }
        set { SetValue(PixelFormatProperty, value); }
    }

    public static readonly DependencyProperty PixelWidthProperty =
DependencyProperty.Register(
        "PixelWidth", typeof (int), typeof (PixelSupportBehavior), new
PropertyMetadata(default(int),OnPixelWidthChanged));

    private static void OnPixelWidthChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
    {
        var b = (PixelSupportBehavior)d;
        var value = (int)e.NewValue;

        if(value<=0)
            return;

        b._pixelWidth = value;

        b.InitializeBufferIfAttached();
    }

    public int PixelWidth
    {
        get { return (int) GetValue(PixelWidthProperty); }
        set { SetValue(PixelWidthProperty, value); }
    }
```

```
    public static readonly DependencyProperty PixelHeightProperty =
DependencyProperty.Register(
        "PixelHeight", typeof (int), typeof (PixelSupportBehavior), new
PropertyMetadata(default(int),OnPixelHeightChanged));

    private static void OnPixelHeightChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
    {
        var b = (PixelSupportBehavior)d;
        var value = (int)e.NewValue;

        if (value <= 0)
            return;

        b._pixelHeight = value;

        b.InitializeBufferIfAttached();
    }

    public int PixelHeight
    {
        get { return (int) GetValue(PixelHeightProperty); }
        set { SetValue(PixelHeightProperty, value); }
    }

    public static readonly DependencyProperty DpiXProperty = DependencyProperty.Register(
        "DpiX", typeof (int), typeof (PixelSupportBehavior), new
PropertyMetadata(96,OnDpiXChanged));

    private static void OnDpiXChanged(DependencyObject d, DependencyPropertyChangedEventArgs
e)
    {
        var b = (PixelSupportBehavior)d;
        var value = (int)e.NewValue;

        if (value <= 0)
            return;

        b._dpiX = value;

        b.InitializeBufferIfAttached();
    }

    public int DpiX
    {
        get { return (int) GetValue(DpiXProperty); }
        set { SetValue(DpiXProperty, value); }
    }

    public static readonly DependencyProperty DpiYProperty = DependencyProperty.Register(
        "DpiY", typeof (int), typeof (PixelSupportBehavior), new
PropertyMetadata(96,OnDpiYChanged));

    private static void OnDpiYChanged(DependencyObject d, DependencyPropertyChangedEventArgs
e)
    {
        var b = (PixelSupportBehavior)d;
        var value = (int)e.NewValue;

        if (value <= 0)
```

```
            return;

        b._dpiY = value;

        b.InitializeBufferIfAttached();
    }

    public int DpiY
    {
        get { return (int) GetValue(DpiYProperty); }
        set { SetValue(DpiYProperty, value); }
    }

    private IntPtr _backBuffer = IntPtr.Zero;
    private int _bytesPerPixel;
    private const int BitsPerByte = 8;
    private int _pixelWidth;
    private int _pixelHeight;
    private int _dpiX;
    private int _dpiY;
    private Int32Rect _imageRectangle;
    private readonly GCHandle _defaultGcHandle = new GCHandle();
    private PixelFormat _pixelFormat;

    private int _byteArraySize;
    private WriteableBitmap _bitMap;

    private bool _attached;

    protected override void OnAttached()
    {
        _attached = true;
        InitializeBufferIfAttached();
    }

    private void InitializeBufferIfAttached()
    {
        if(_attached==false)
            return;

        ReevaluateBitsPerPixel();

        RecomputeByteArraySize();

        ReinitializeImageSource();
    }

    private void ReevaluateBitsPerPixel()
    {
        var f = _pixelFormat;

        if (f == PixelFormats.Default)
        {
            _bytesPerPixel = 0;
            return;
        };

        _bytesPerPixel = f.BitsPerPixel/BitsPerByte;
    }

    private void ReinitializeImageSource()
```

```
    {
        var f = _pixelFormat;
        var h = _pixelHeight;
        var w = _pixelWidth;

        if (w<=0|| h<=0 || f== PixelFormats.Default)
            return;

        _imageRectangle = new Int32Rect(0, 0, w, h);
        _bitMap = new WriteableBitmap(w, h, _dpiX, _dpiY, f, null);
        _backBuffer = _bitMap.BackBuffer;
        AssociatedObject.Source = _bitMap;
    }

    private void RenderPixels(IList<byte> pixels)
    {
        if (pixels == null)
        {
            return;
        }

        var buffer = _backBuffer;
        if (buffer == IntPtr.Zero)
            return;

        var size = _byteArraySize;

        var gcHandle = _defaultGcHandle;
        var allocated = false;
        var bitMap = _bitMap;
        var rect = _imageRectangle;
        var w = _pixelWidth;
        var locked = false;
        try
        {
            gcHandle = GCHandle.Alloc(pixels, GCHandleType.Pinned);
            allocated = true;

            bitMap.Lock();
            locked = true;
            var ptr = gcHandle.AddrOfPinnedObject();
            _bitMap.WritePixels(rect, ptr, size,w);
        }
        finally
        {
            if(locked)
                bitMap.Unlock();

            if (allocated)
                gcHandle.Free();
        }
    }

    private void RecomputeByteArraySize()
    {
        var h = _pixelHeight;
        var w = _pixelWidth;
        var bpp = _bytesPerPixel;

        if (w<=0|| h<=0 || bpp<=0)
            return;
```

```
        _byteArraySize = (w * h * bpp);
    }

    public PixelSupportBehavior()
    {
        _pixelFormat = PixelFormats.Default;
    }
}
}
```

XAML

```
<UserControl x:Class="Example.View"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
         xmlns:b="clr-namespace:MyBehaviorAssembly;assembly=MyBehaviorAssembly"
         xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
         mc:Ignorable="d"
         d:DesignHeight="200" d:DesignWidth="200"
         >

                <Image Stretch="Uniform">

                <i:Interaction.Behaviors>

                    <b:PixelSupportBehavior
                            PixelHeight="{Binding PixelHeight}"
                            PixelWidth="{Binding PixelWidth}"
                            PixelFormat="{Binding PixelFormat}"
                            Pixels="{Binding Pixels}"
                            />

                </i:Interaction.Behaviors>

            </Image>

</UserControl>
```

: https://riptutorial.com/ko/wpf/topic/6435/--------

22:

CLR . CLR DependencyObject GetValue () .

.

- DependencyProperty.Register (, Type , ownerType)
- DependencyProperty.Register (, Type , ownerType, PropertyMetadata typeMetadata)
- DependencyProperty.Register (, propertyType, ownerType, PropertyMetadata typeMetadata, ValidateValueCallback validateValueCallback)
- DependencyProperty.RegisterAttached (, Type , ownerType)
- DependencyProperty.RegisterAttached (, Type , ownerType, PropertyMetadata typeMetadata)
- DependencyProperty.RegisterAttached (, propertyType, ownerType, PropertyMetadata typeMetadata, ValidateValueCallback validateValueCallback)
- DependencyProperty.RegisterReadOnly (, Type , ownerType, PropertyMetadata typeMetadata)
- DependencyProperty.RegisterReadOnly (, propertyType, ownerType, PropertyMetadata typeMetadata, ValidateValueCallback validateValueCallback)
- DependencyProperty.RegisterAttachedReadOnly (, Type , ownerType, PropertyMetadata typeMetadata)
- DependencyProperty.RegisterAttachedReadOnly (, propertyType, ownerType, PropertyMetadata typeMetadata, ValidateValueCallback validateValueCallback)

	`String`
propertyType	`Type typeof(int)`
	`Type typeof(MyControl)` : `typeof(MyControl)` `typeof(MyAttachedProperties)` .
typeMetadata	, `System.Windows.PropertyMetadata` ( ) `FrameworkPropertyMetadata` `System.Windows.Data.BindingMode.TwoWay` .
validateValueCallback	true , false.

Examples

?

WPF . , , , CLR WPF .

`TextBox.Text` . `Text` CLR .

```
<TextBox Text="{Binding FirstName}" />
```

FrameworkElement , Control DependencyObject .

"propdp" propdp . .

```
public class MyControl : Control
{
    public int MyProperty
    {
        get { return (int)GetValue(MyPropertyProperty); }
        set { SetValue(MyPropertyProperty, value); }
    }

    // Using a DependencyProperty as the backing store for MyProperty.
    // This enables animation, styling, binding, etc...
    public static readonly DependencyProperty MyPropertyProperty =
        DependencyProperty.Register("MyProperty", typeof(int), typeof(MyControl),
            new PropertyMetadata(0));
}
```

, , .

/ .

1. **CLR .** . GetValue SetValue .
2. . DependencyProperty public static readonly . CLR "Property" (: Text TextProperty .
3. **CLR .** (XAML) CLR . PropertyMetadata .

```
public static readonly DependencyProperty MyPropertyProperty =
    DependencyProperty.Register("MyProperty", typeof(int), typeof(MyControl),
        new PropertyMetadata(0, MyPropertyChangedHandler));

private static void MyPropertyChangedHandler(DependencyObject sender,
DependencyPropertyChangedEventArgs args)
{
    // Use args.OldValue and args.NewValue here as needed.
    // sender is the object whose property changed.
    // Some unboxing required.
}
```

Mode=TwoWay (TextBox.Text) PropertyMetadata FrameworkPropertyMetadata .

```
public static readonly DependencyProperty MyPropertyProperty =
    DependencyProperty.Register("MyProperty", typeof(int), typeof(MyControl),
        new FrameworkPropertyMetadata(0,
FrameworkPropertyMetadataOptions.BindsTwoWayByDefault));
```

?

DependencyObject .

.

1. . , Grid Grid.Row , Grid.Column , Grid.RowSpan Grid.ColumnSpan .
2. . TextBox .
3. (ToolTipService FocusManager .
4. , DataContext .

Grid .

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>

    <Label Grid.Column="0" Content="Your Name:" />
    <TextBox Grid.Column="1" Text="{Binding FirstName}" />
</Grid>
```

Grid.Column Label TextBox . Grid .

Grid . Grid.Column DependencyPropertyChangedEventHandler .

```
public static readonly DependencyProperty RowProperty =
    DependencyProperty.RegisterAttached("Row", typeof(int), typeof(Grid),
        new FrameworkPropertyMetadata(0, ...));

public static void SetRow(UIElement element, int value)
{
    if (element == null)
        throw new ArgumentNullException("element");

    element.SetValue(RowProperty, value);
}

public static int GetRow(UIElement element)
{
    if (element == null)
        throw new ArgumentNullException("element");

    return ((int)element.GetValue(RowProperty));
}
```

CLR . .

DependencyObject .

RowProperty GetRow SetRow .

MSDN :

. RegisterAttached .

?

. IsMouseOver IsKeyboardFocusWithin .

DependencyObject .

```
public class MyControl : Control
{
    private static readonly DependencyPropertyKey MyPropertyPropertyKey =
        DependencyProperty.RegisterReadOnly("MyProperty", typeof(int), typeof(MyControl),
            new FrameworkPropertyMetadata(0));

    public static readonly DependencyProperty MyPropertyProperty =
MyPropertyPropertyKey.DependencyProperty;

    public int MyProperty
    {
        get { return (int)GetValue(MyPropertyProperty); }
        private set { SetValue(MyPropertyPropertyKey, value); }
    }
}
```

.

1. DependencyProperty private DependencyPropertyKey .
2. CLR public protected private .

Setter MyPropertyPropertyKey MyPropertyProperty SetValue . SetValue DependencyPropertyKey . , InvalidOperationException Throw.

: https://riptutorial.com/ko/wpf/topic/2914/--

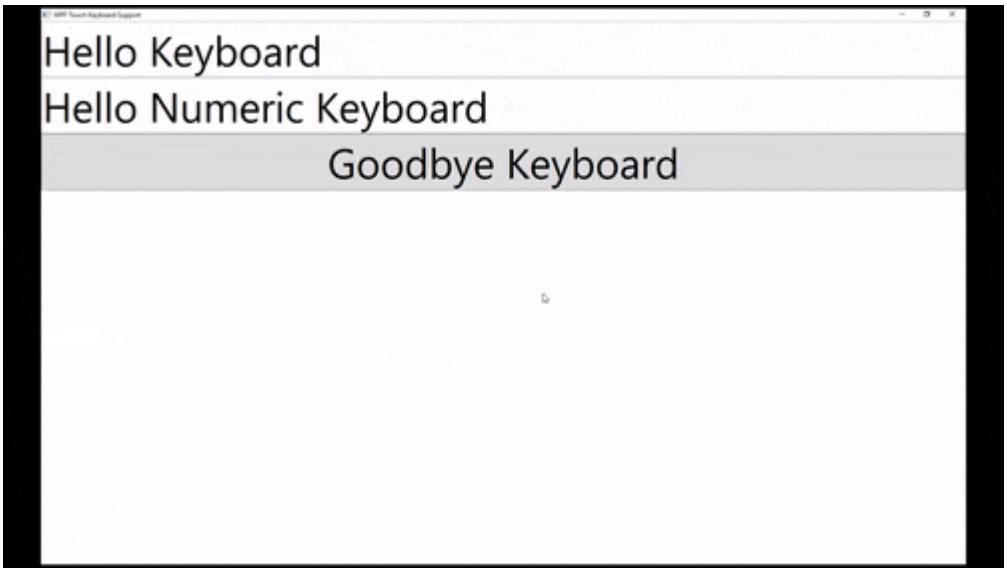
23:

Examples

Windows 8 Windows 10

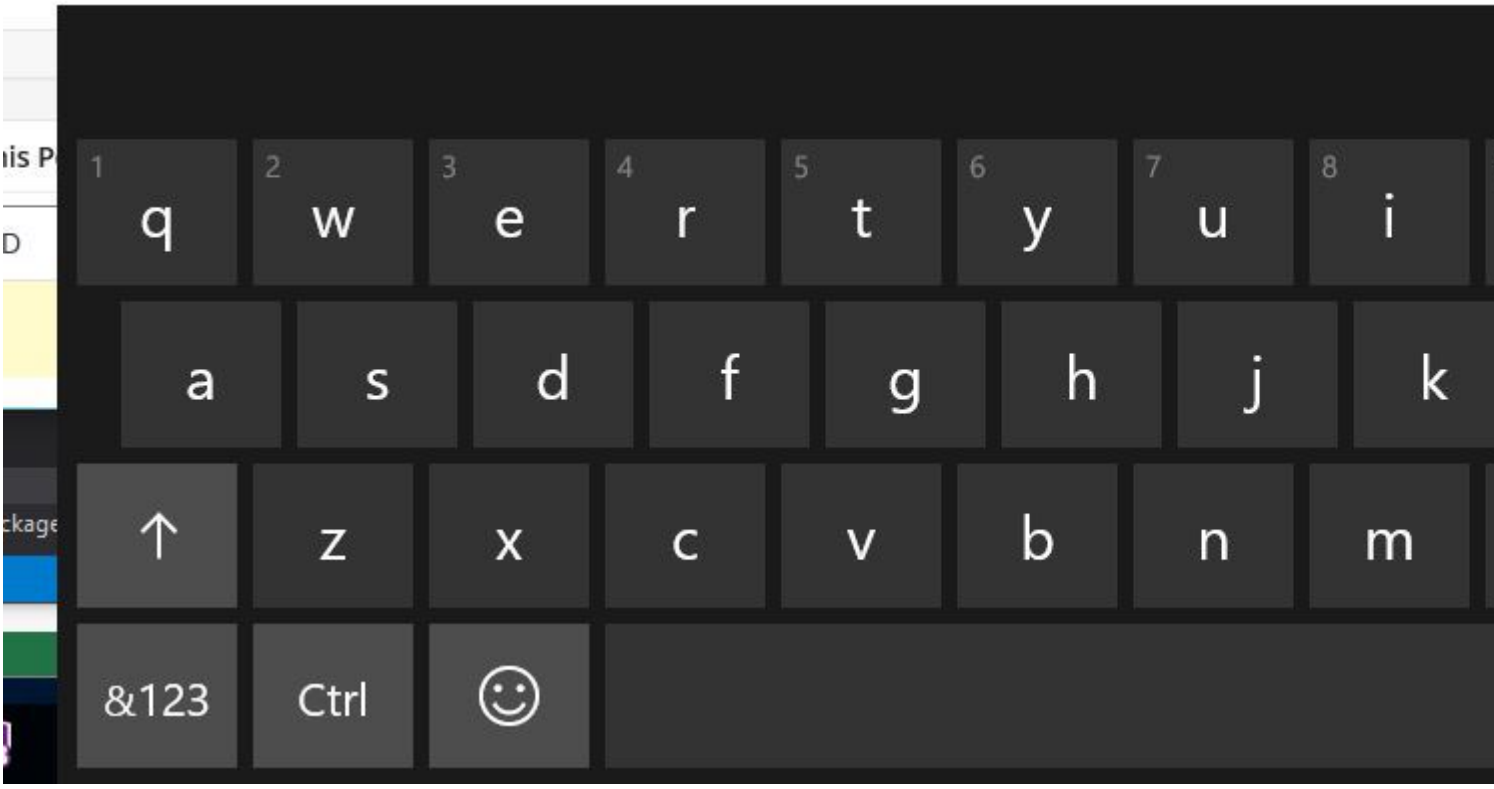
.NET Framework 4.6.2 WPF

.NET Framework 4.6.2 () WPF .



.NET Framework 4.6.1 WPF

WPF . , WPF `TextBox` . .



Windows 8 Windows 10 PC `C:\Program Files\Common Files\Microsoft Shared\Ink\TabTip.exe` exe `C:\Program Files\Common Files\Microsoft Shared\Ink\TabTip.exe` .

`GotTouchCapture` `TextBox` ( ). .

```
public class TouchEnabledTextBox : TextBox
{
    public TouchEnabledTextBox()
    {
        this.GotTouchCapture += TouchEnabledTextBox_GotTouchCapture;
    }

    private void TouchEnabledTextBox_GotTouchCapture(
       object sender,
       System.Windows.Input.TouchEventArgs e )
    {
        string touchKeyboardPath =
           @"C:\Program Files\Common Files\Microsoft Shared\Ink\TabTip.exe";
        Process.Start( touchKeyboardPath );
    }
}
```

.

```
//added field
private Process _touchKeyboardProcess = null;

//replace Process.Start line from the previous listing with
_touchKeyboardProcess = Process.Start( touchKeyboardPath );
```

`LostFocus` .

```
//add this at the end of TouchEnabledTextBox's constructor
this.LostFocus += TouchEnabledTextBox_LostFocus;

//add this method as a member method of the class
private void TouchEnabledTextBox_LostFocus( object sender, RoutedEventArgs eventArgs ){
   if ( _touchKeyboardProcess != null )
   {
      _touchKeyboardProcess.Kill();
      //nullify the instance pointing to the now-invalid process
      _touchKeyboardProcess = null;
   }
}
```

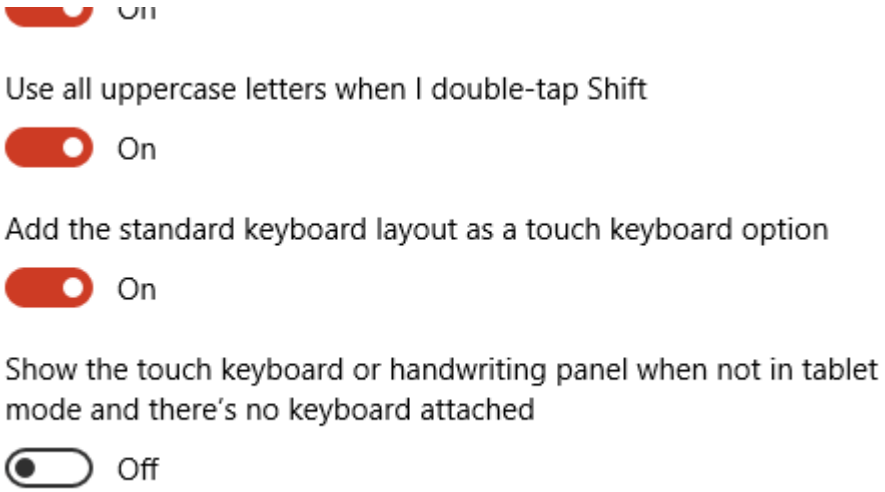
Windows 10

Windows 10 PC . WPF .

Windows 10

Windows 10 . .

. " " .



.

: https://riptutorial.com/ko/wpf/topic/6799/---

S. No		Contributors
1	wpf	Community, Derpcode, Gusdor, Matthew Cargille, Nasreddine, Sam, Stephen Wilson
2	" "	Richardissimo
3	System.Windows.Controls.WebBrowser	Richardissimo
4	WPF	Adi Lester, Arie, Gabor Barat, Guttsy, Ian Wold, Jirajha, vkluge, wkl
5	WPF	Elangovan, John Strit, SUB-HDR
6	WPF	Bradley Uffner
7	WPF	J R
8	WPF	Adi Lester
9	WPF	Dabblernl
10	WPF	Grx70, Sam
11	WPF MVVM	Andrew Stephens, Athafoud, Dutts, Felix D., Felix Too, H.B., James LaPenn, Kcvin, kowsky, Matt Klein, RamenChef, STiLeTT, TrBBoI, vkluge
12	WPF	Guttsy, Jakub Lokša
13		Adi Lester, Arie, Dalstroem, galakt, ltiveron
14		Alexander Mandt, vkluge
15	UserControls	ltiveron, Mage Xy
16		Alexander Pacha, Emad
17		John Strit, Maxim
18	UI	Mert Gülsoy
19	:	Eyal Perry
20		BKO
21		Eyal Perry

22		Adi Lester, auticus, Clemens, Guttsy
23		Martin Zikmund