

# **Windows Presentation Foundation (WPF)**

Yrd. Doç. Dr. Özgü Can

# Windows Presentation Foundation (WPF)

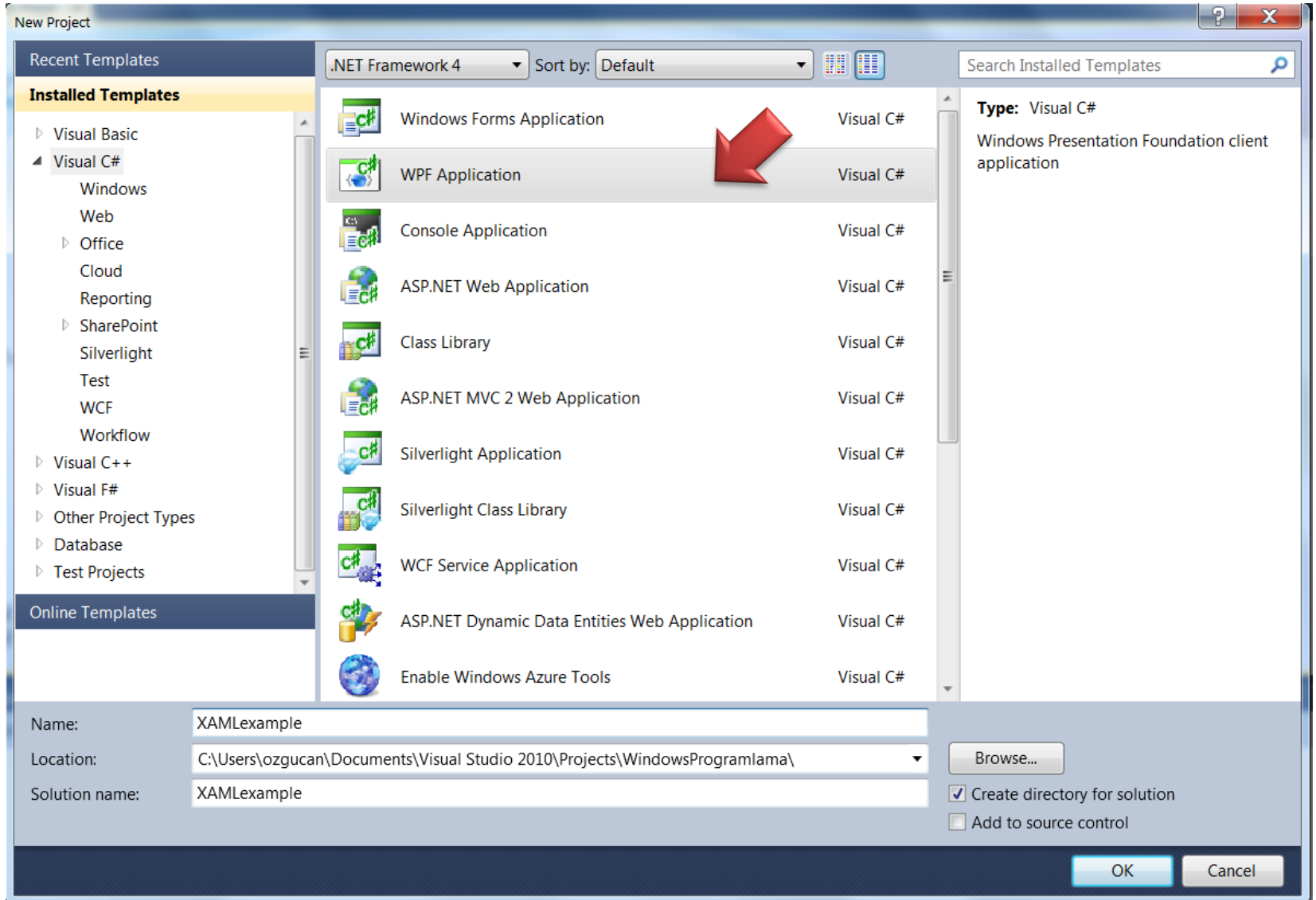
- Microsoft'un; GUI, grafik, animasyon ve çoklu ortam (multimedia) çatısıdır.
- Tek bir teknoloji kullanarak; GUI, resim, animasyon, 2D ya da 3D grafikler, ses ve video yeteneklerine sahip uygulamalar geliştirilmesine izin verir.

# XAML kullanarak GUI

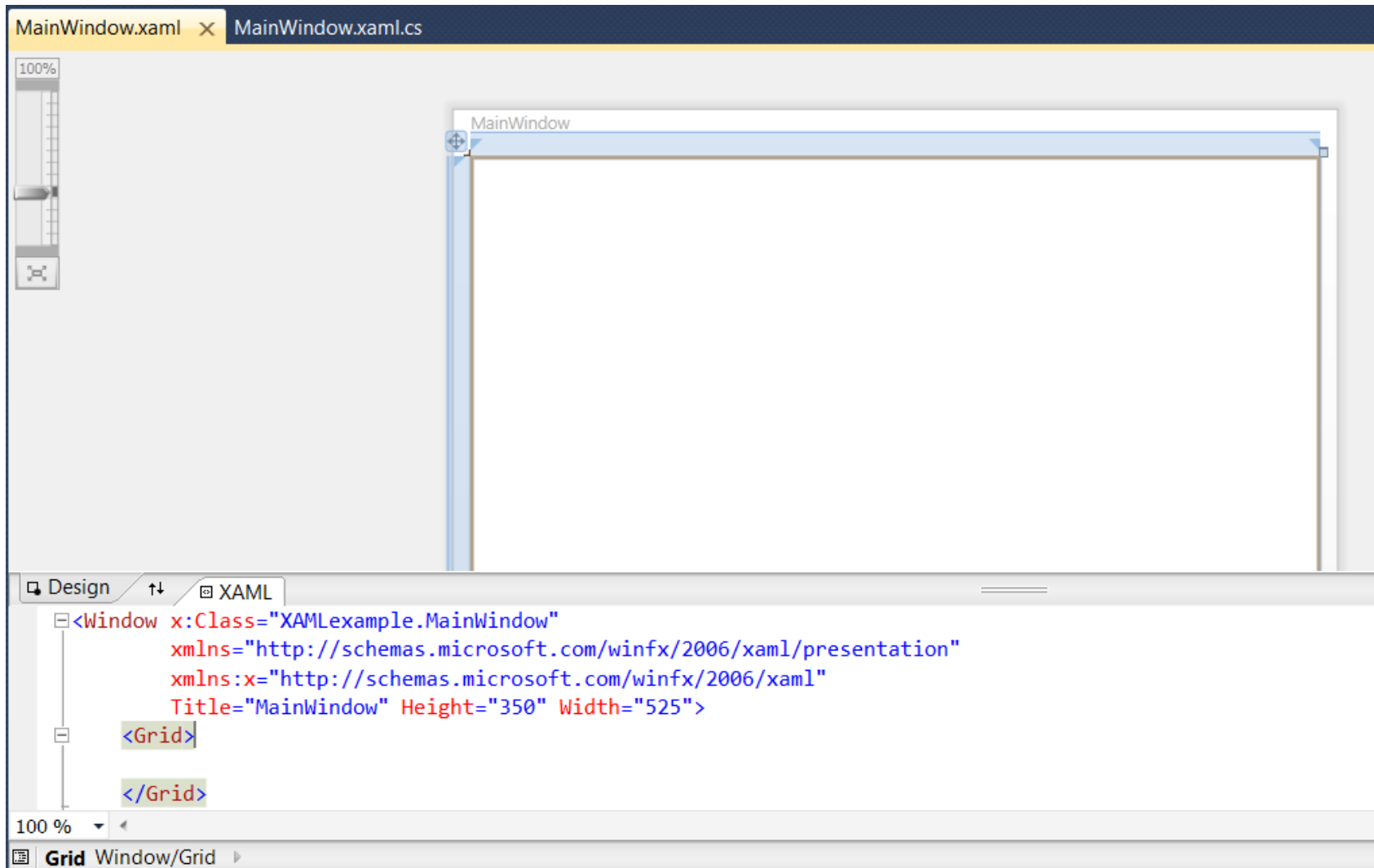
- E**X**tensible **A**pplication **M**arkup **L**anguage\*
- XAML, bir WPF uygulamasının **görünümünü** tanımlar.
- XAML belgeleri **XML belgeleridir**.
  - XML belgesinde olduğu gibi tek bir kök (root) öge vardır.

\* [http://msdn.microsoft.com/tr-tr/library/ms752059\(v=vs.110\).aspx](http://msdn.microsoft.com/tr-tr/library/ms752059(v=vs.110).aspx)

# Uygulama



# Uygulama



MainWindow.xaml X MainWindow.xaml.cs

```
<Window x:Class="XAMLExample.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">
    <Grid>

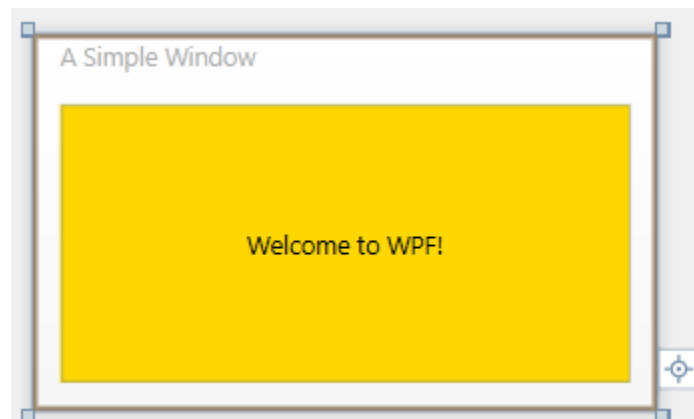
    </Grid>
</Window>
```

MainWindow.xaml\* X MainWindow.xaml.cs

```
<Window x:Class="XAMLExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="A Simple Window" Height="150" Width="250">

    <!-- a layout container -->
    <Grid Background="Gold">

        <!-- a Label control -->
        <Label HorizontalAlignment="Center" VerticalAlignment="Center">
            Welcome to WPF!
        </Label>
    </Grid>
</Window>
```



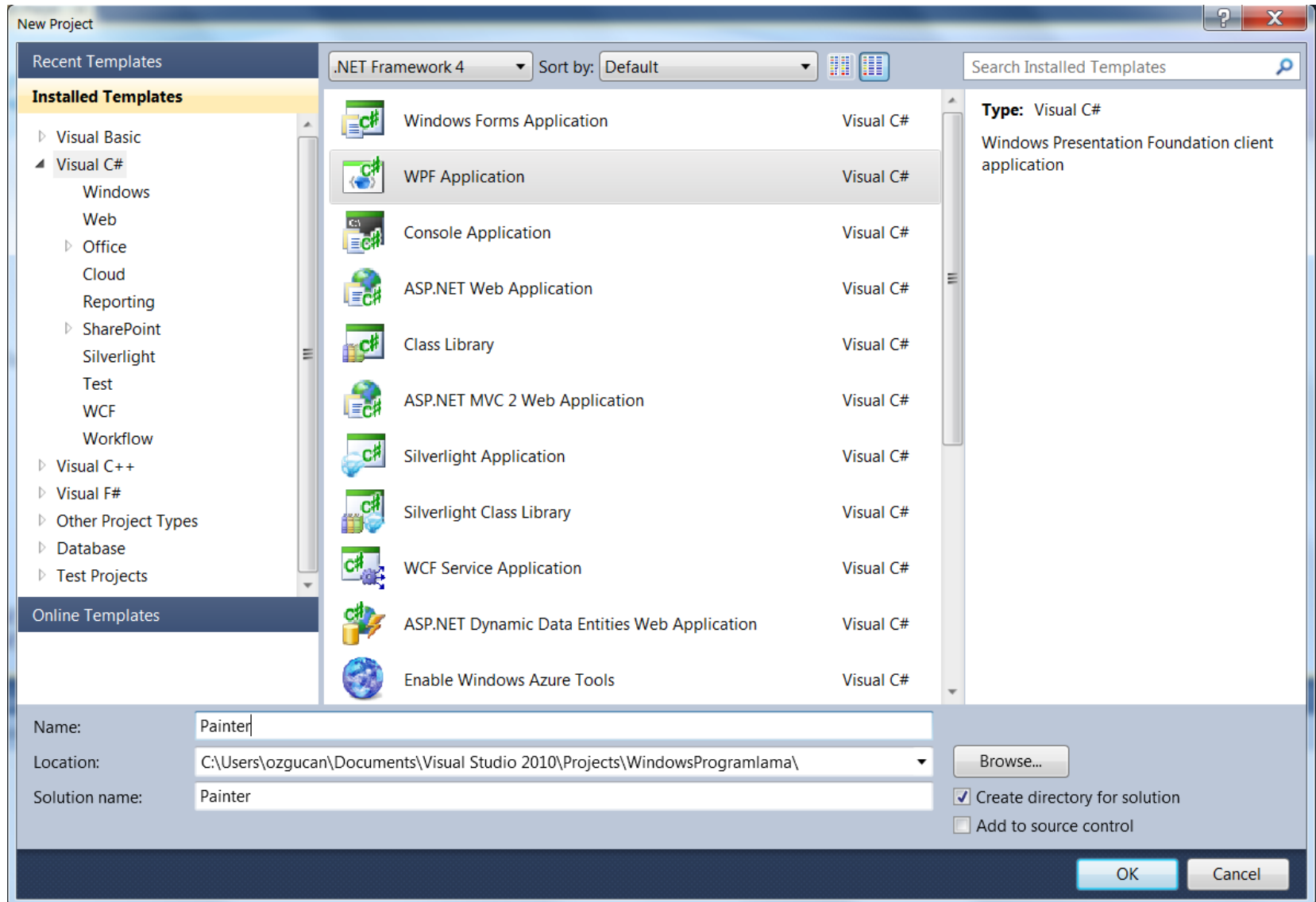
# **LAYOUT KONTROLLERI**

Control	Description
<i>Layout containers (derived from Panel)</i>	
Grid	Layout is defined by a grid of rows and columns, depending on the RowDefinitions and ColumnDefinitions properties. Elements are placed into cells.
Canvas	Layout is coordinate based. Element positions are defined explicitly by their distance from the top and left edges of the Canvas.
StackPanel	Elements are arranged in a single row or column, depending on the Orientation property.
DockPanel	Elements are positioned based on which edge they're docked to. If the LastChildFill property is True, the last element gets the remaining space in the middle.
WrapPanel	A wrapping StackPanel. Elements are arranged sequentially in rows or columns (depending on the Orientation), each row or column wrapping to start a new one when it reaches the WrapPanel's right or bottom edge, respectively.
<i>Content controls (derived from ContentControl)</i>	
Border	Adds a background or a border to the child element.
GroupBox	Surrounds the child element with a titled box.
Window	The application window. Also the root element.
Expander	Puts the child element in a titled area that collapses to display just the header and expands to display the header and the content.



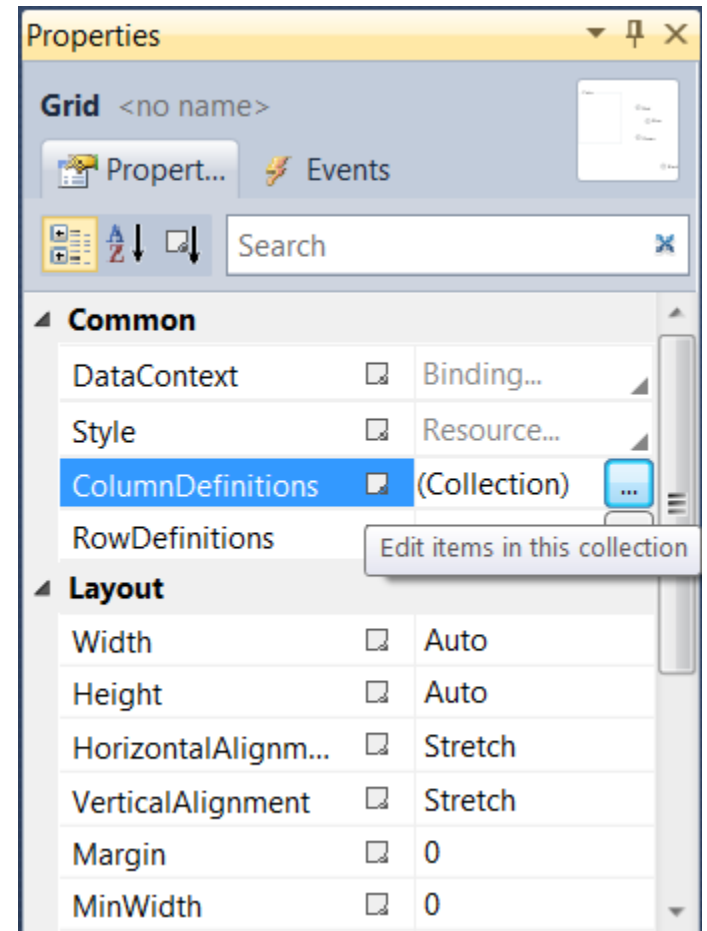
**UYGULAMA - PAINTER**

# Uygulama - Painter

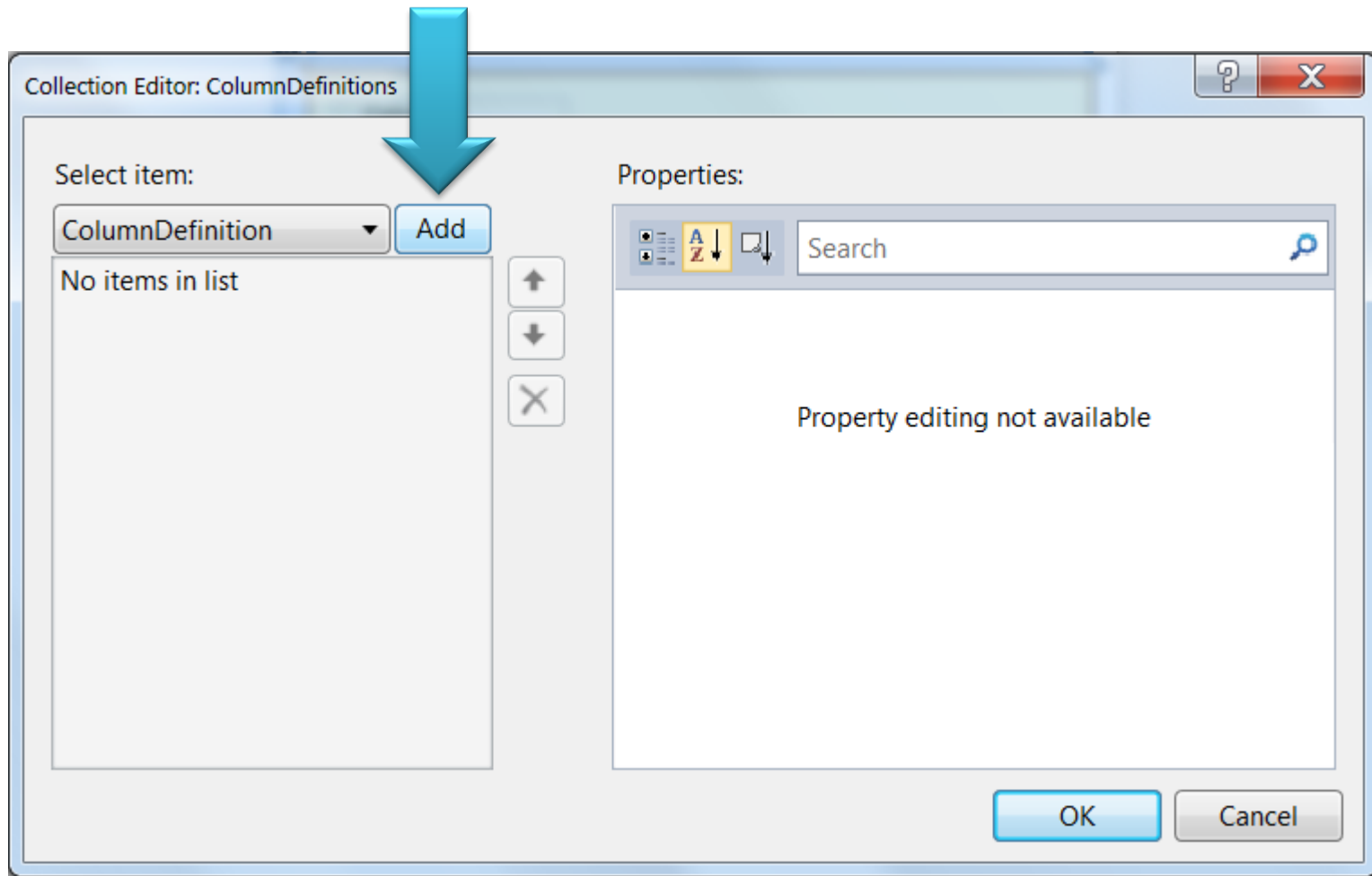


# Uygulama - Painter

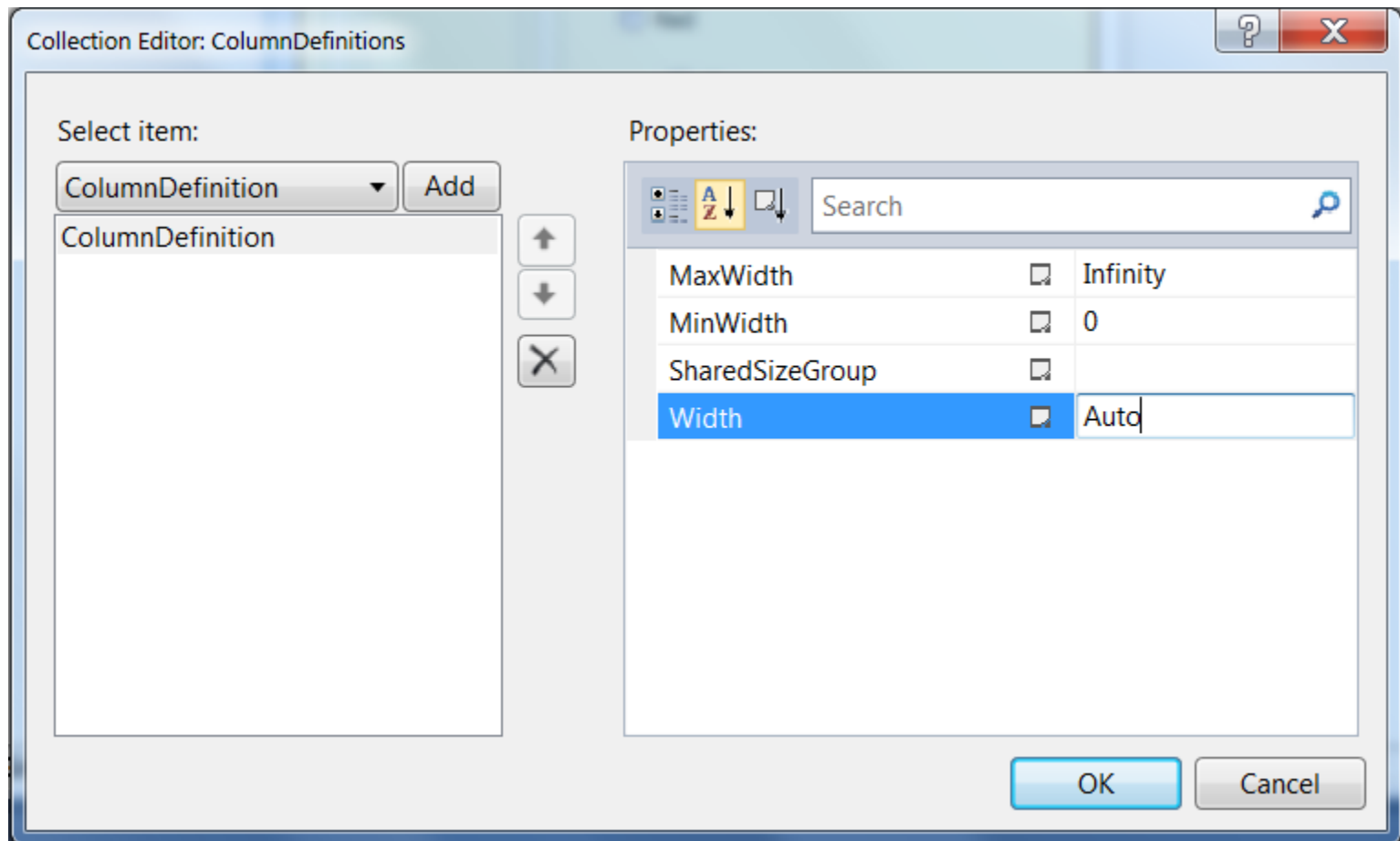
- Grid → Properties  
**ColumnDefinitions**



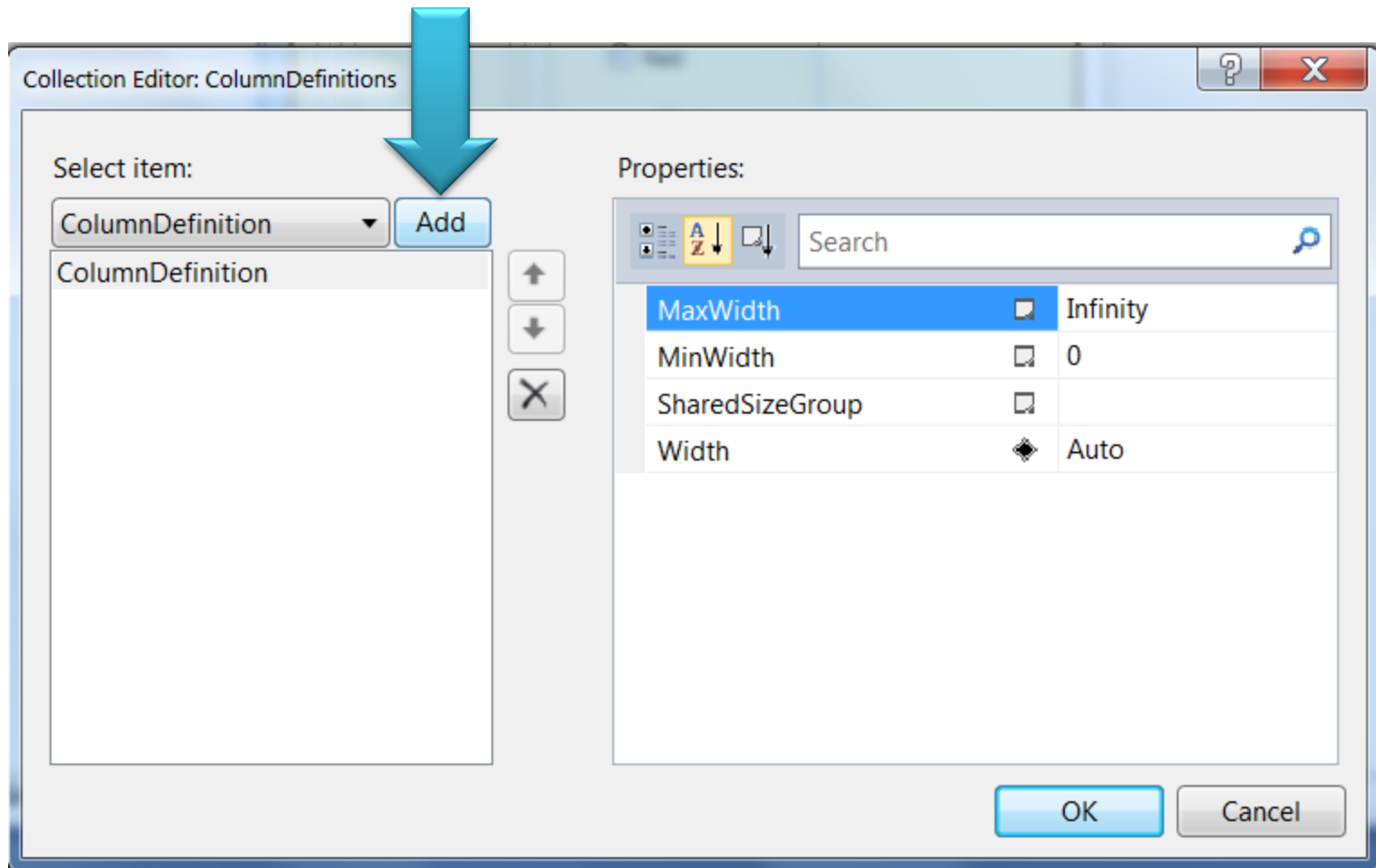
# Uygulama - Painter



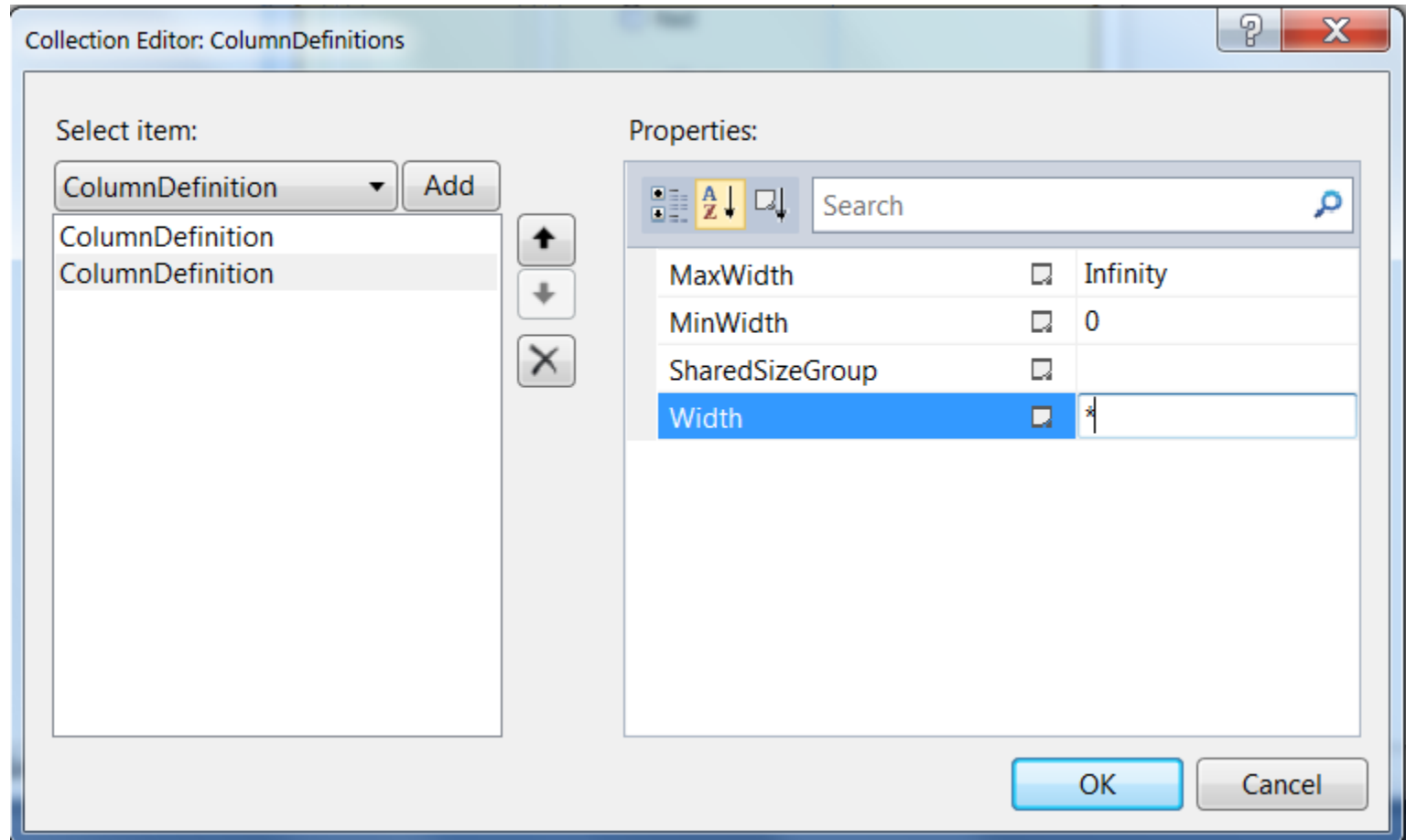
# Uygulama - Painter



# Uygulama - Painter



# Uygulama - Painter



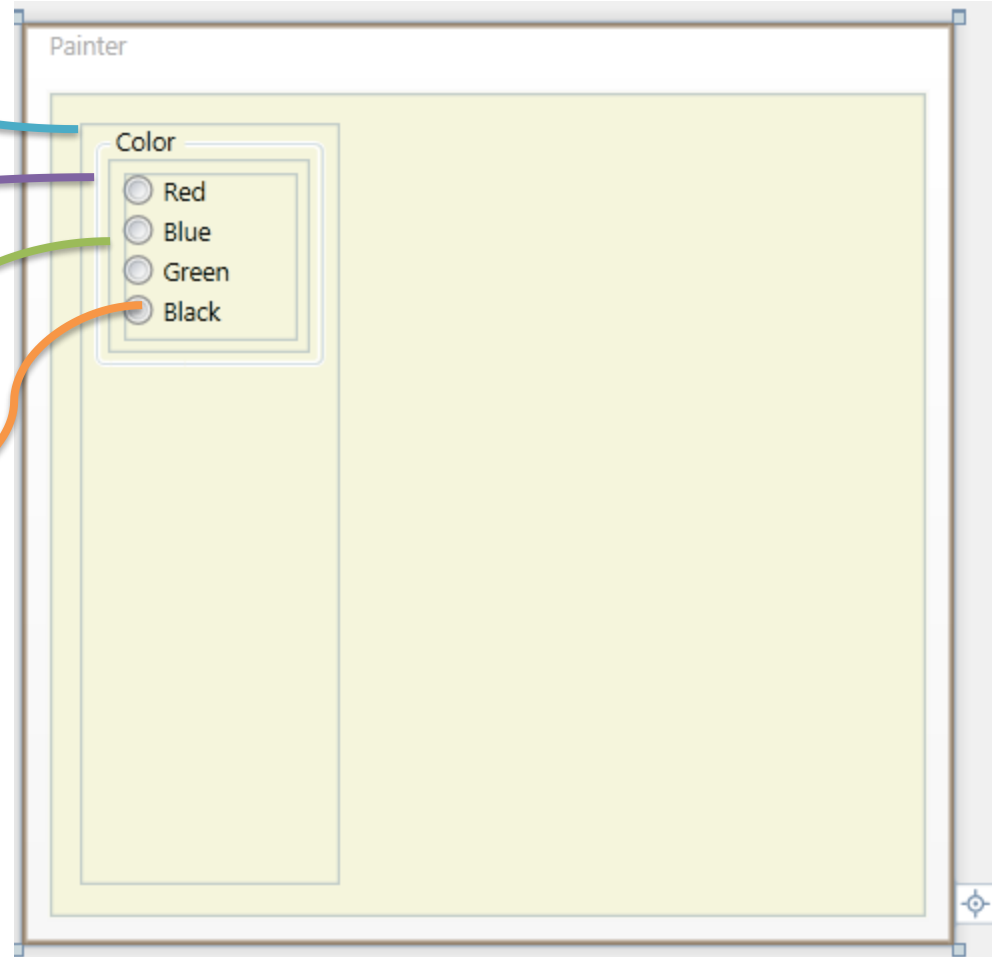
# Uygulama

```
<!-- creates a Grid -->
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" /> <!--defines a column-->
    <ColumnDefinition Width="*" />    <!--defines a column-->
  </Grid.ColumnDefinitions>
</Grid>
```

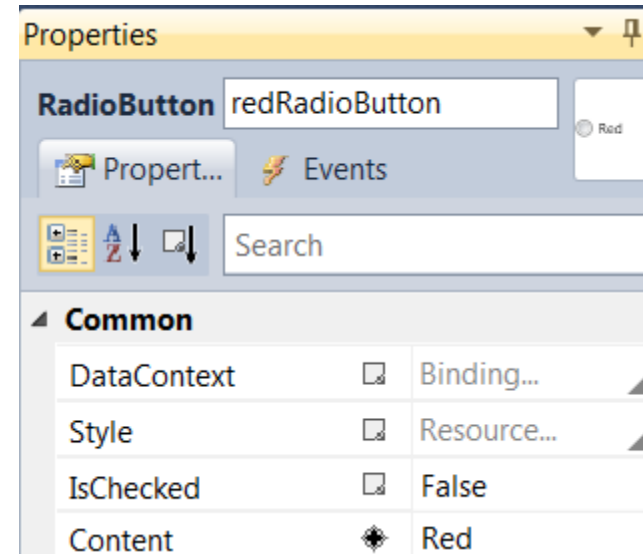
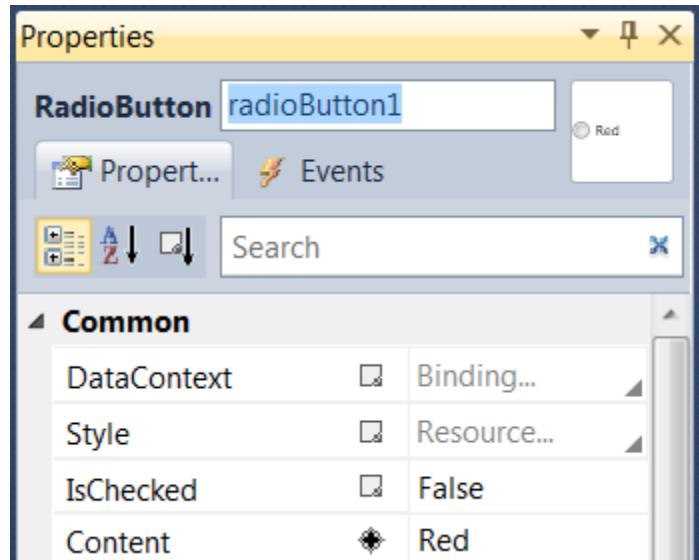


# Uygulama - Painter

- StackPanel
- GroupBox
  - Header = Color
- StackPanel
- 4 RadioButton
  - Content = Red
  - Content = Blue
  - Content = Green
  - Content = Black



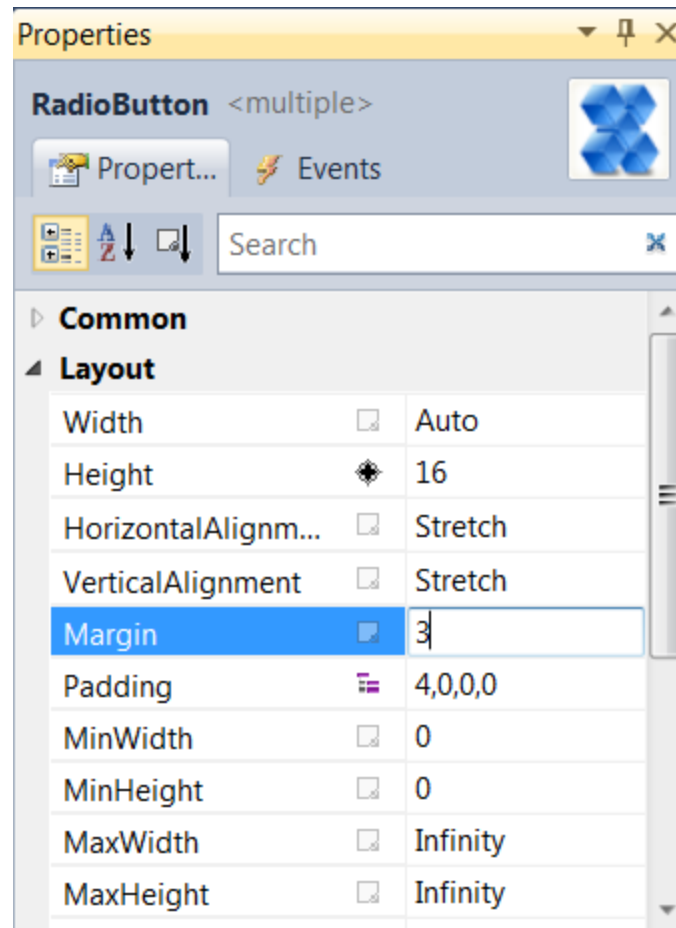
# Uygulama - Painter



**radioButton1 → redRadioButton**  
**radioButton2 → blueRadioButton**  
**radioButton3 → greenRadioButton**  
**radioButton4 → blackRadioButton**

# Uygulama - Painter

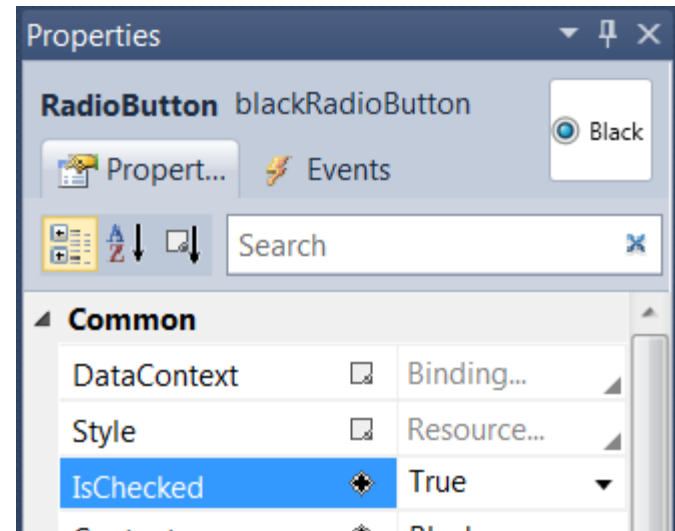
RadioButton → Properties → Layout → **Margin = 3**



# Uygulama - Painter

**blackRadioButton**

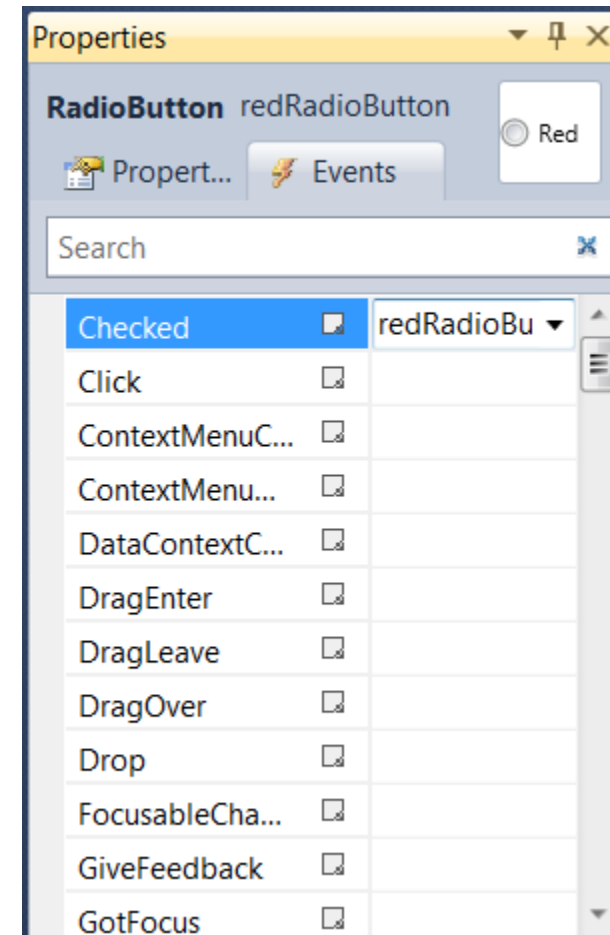
**– IsChecked = True**



# Uygulama - Painter

Button → Events → **Checked**

- **redRadioButton\_Checked**
- **blueRadioButton\_Checked**
- **greenRadioButton\_Checked**
- **blackRadioButton\_Checked**

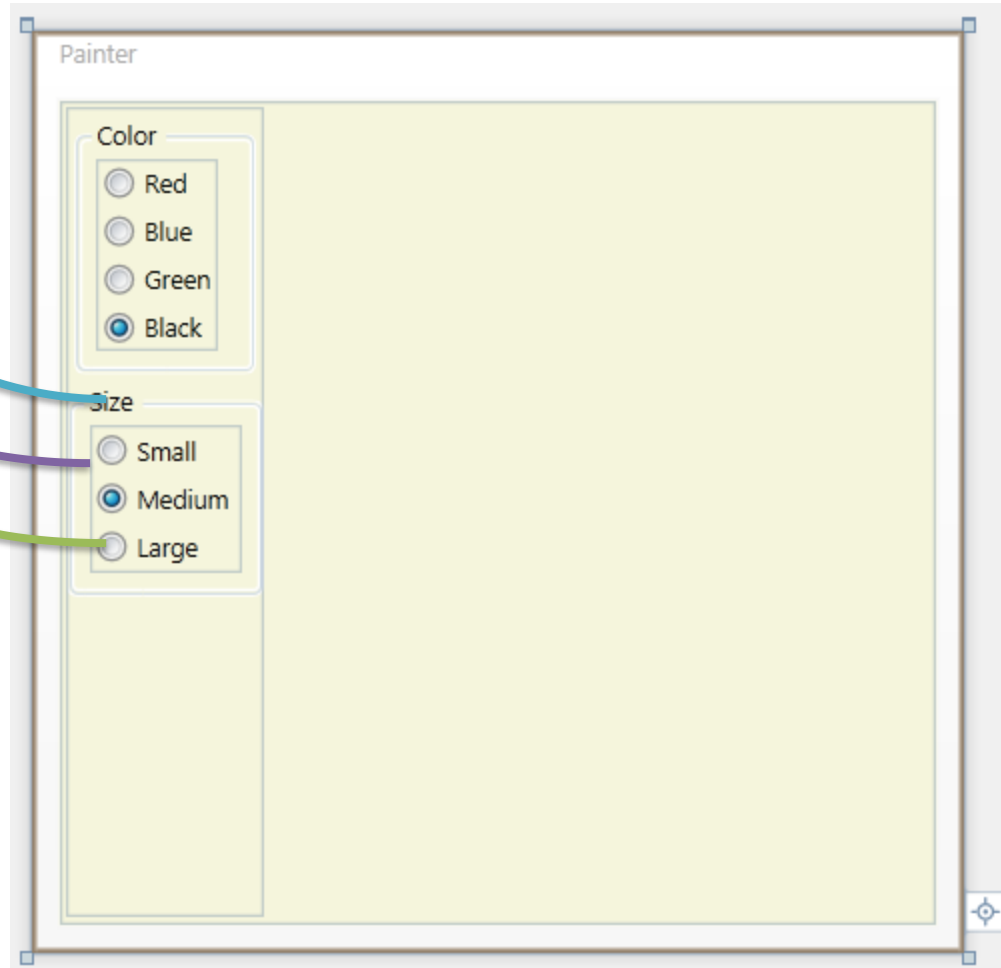


# Uygulama - Painter

```
<StackPanel Margin="3" Name="stackPanel1" >
    <GroupBox Header="Color" Name="colorGroupBox" Margin="3" HorizontalAlignment="Stretch" VerticalAlignment="Top">
        <StackPanel HorizontalAlignment="Left" Margin="3" Name="stackPanel2" VerticalAlignment="Top">
            <RadioButton Content="Red" Name="redRadioButton" Margin="3" Checked="redRadioButton_Checked"/>
            <RadioButton Content="Blue" Name="blueRadioButton" Margin="3" Checked="blueRadioButton_Checked" />
            <RadioButton Content="Green" Name="greenRadioButton" Margin="3" Checked="greenRadioButton_Checked"/>
            <RadioButton Content="Black" Name="blackRadioButton" Margin="3" Checked="blackRadioButton_Checked" IsChecked="True" />
        </StackPanel>
    </GroupBox>
</StackPanel>
```

# Uygulama - Painter

- **GroupBox**
  - Header = Size
- **StackPanel**
- **3 RadioButton**
  - Content = Small
  - Content = Medium
  - Content = Large



# Uygulama - Painter

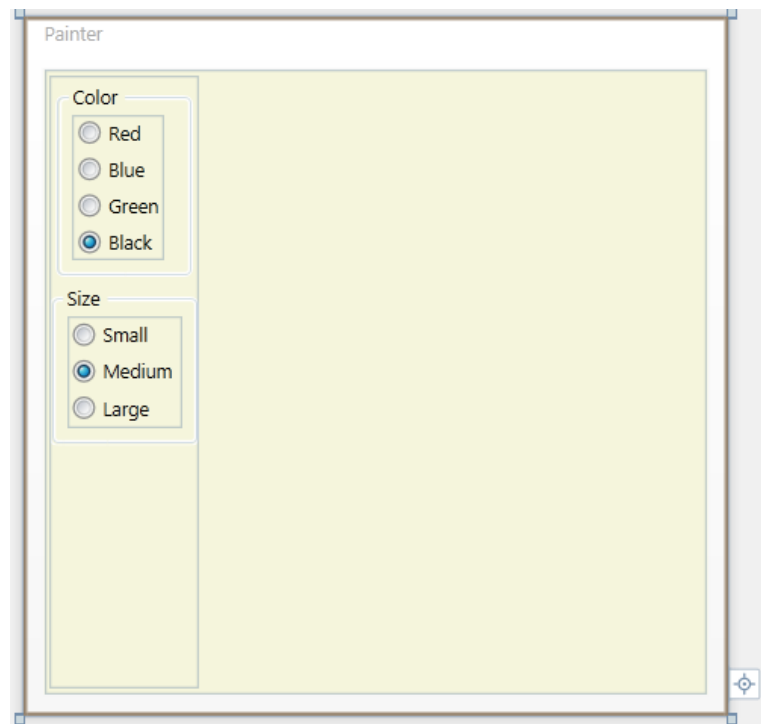
```
<GroupBox Header="Size" Height="86">
  <StackPanel Margin="3" Height="59" Width="61">
    <RadioButton Content="Small" Name="smallRadioButton" Margin="3" Checked="smallRadioButton_Checked"/>
    <RadioButton Content="Medium" Name="mediumRadioButton" IsChecked="True" Checked="mediumRadioButton_Checked" Margin="3"/>
    <RadioButton Content="Large" Name="largeRadioButton" Margin="3" Checked="largeRadioButton_Checked"/>
  </StackPanel>
</GroupBox>
```



```

<!-- creates a Grid -->
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" /> <!--defines a column-->
    <ColumnDefinition Width="*" /> <!--defines a column-->
  </Grid.ColumnDefinitions>
  <StackPanel Margin="3" Name="stackPanel1" >
    <GroupBox Header="Color" Name="colorGroupBox" Margin="3" HorizontalAlignment="Stretch" VerticalAlignment="Top">
      <StackPanel HorizontalAlignment="Left" Margin="3" Name="stackPanel2" VerticalAlignment="Top">
        <RadioButton Content="Red" Name="redRadioButton" Margin="3" Checked="redRadioButton_Checked"/>
        <RadioButton Content="Blue" Name="blueRadioButton" Margin="3" Checked="blueRadioButton_Checked" />
        <RadioButton Content="Green" Name="greenRadioButton" Margin="3" Checked="greenRadioButton_Checked"/>
        <RadioButton Content="Black" Name="blackRadioButton" Margin="3" Checked="blackRadioButton_Checked" IsChecked="True" />
      </StackPanel>
    </GroupBox>
    <GroupBox Header="Size" Height="86">
      <StackPanel Margin="3" Height="59" Width="61">
        <RadioButton Content="Small" Name="smallRadioButton" Margin="3" Checked="smallRadioButton_Checked"/>
        <RadioButton Content="Medium" Name="mediumRadioButton" IsChecked="True" Checked="mediumRadioButton_Checked" Margin="3"/>
        <RadioButton Content="Large" Name="largeRadioButton" Margin="3" Checked="largeRadioButton_Checked"/>
      </StackPanel>
    </GroupBox>
  </StackPanel>
</Grid>

```



# Uygulama - Painter

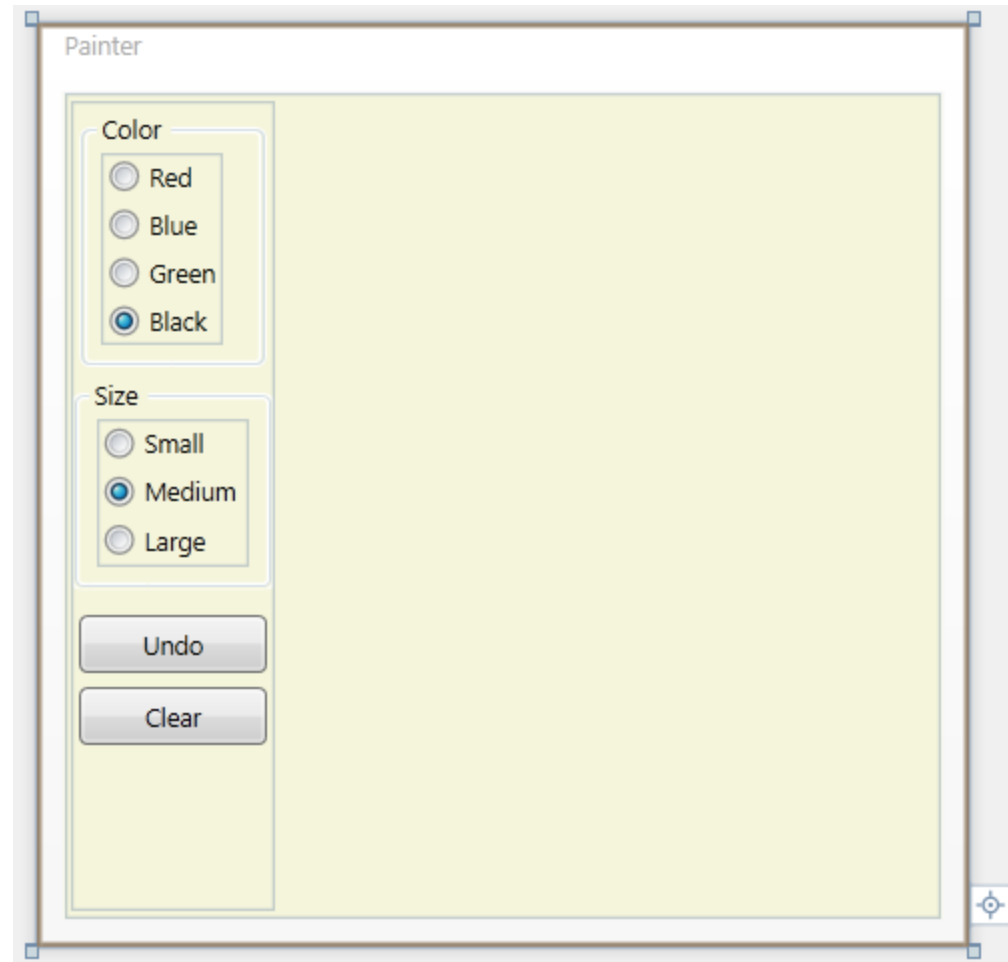
## 2 Button

1. Content = Undo  
Event

undoButton\_Click

2. Content = Clear  
Event

clearButton\_Click



```

<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" /> <!--defines a column-->
    <ColumnDefinition Width="*" /> <!--defines a column-->
  </Grid.ColumnDefinitions>
  <StackPanel Margin="3" Name="stackPanel1" >
    <GroupBox Header="Color" Name="colorGroupBox" Margin="3" HorizontalAlignment="Stretch" VerticalAlignment="Top">
      <StackPanel HorizontalAlignment="Left" Margin="3" Name="stackPanel2" VerticalAlignment="Top">
        <RadioButton Content="Red" Name="redRadioButton" Margin="3" Checked="redRadioButton_Checked"/>
        <RadioButton Content="Blue" Name="blueRadioButton" Margin="3" Checked="blueRadioButton_Checked" />
        <RadioButton Content="Green" Name="greenRadioButton" Margin="3" Checked="greenRadioButton_Checked"/>
        <RadioButton Content="Black" Name="blackRadioButton" Margin="3" Checked="blackRadioButton_Checked" IsChecked="True" />
      </StackPanel>
    </GroupBox>
    <GroupBox Header="Size" Height="86">
      <StackPanel Margin="3" Height="59" Width="61">
        <RadioButton Content="Small" Name="smallRadioButton" Margin="3" Checked="smallRadioButton_Checked"/>
        <RadioButton Content="Medium" Name="mediumRadioButton" IsChecked="True" Checked="mediumRadioButton_Checked" Margin="3"/>
        <RadioButton Content="Large" Name="largeRadioButton" Margin="3" Checked="largeRadioButton_Checked"/>
      </StackPanel>
    </GroupBox>

    <Button Height="23" Content="Undo" Name="undoButton" Width="75" Margin="3,10,3,3" Click="undoButton_Click"/>

    <Button Height="23" Content="Clear" Name="clearButton" Width="75" Margin="3" Click="clearButton_Click"/>

  </StackPanel>
</Grid>

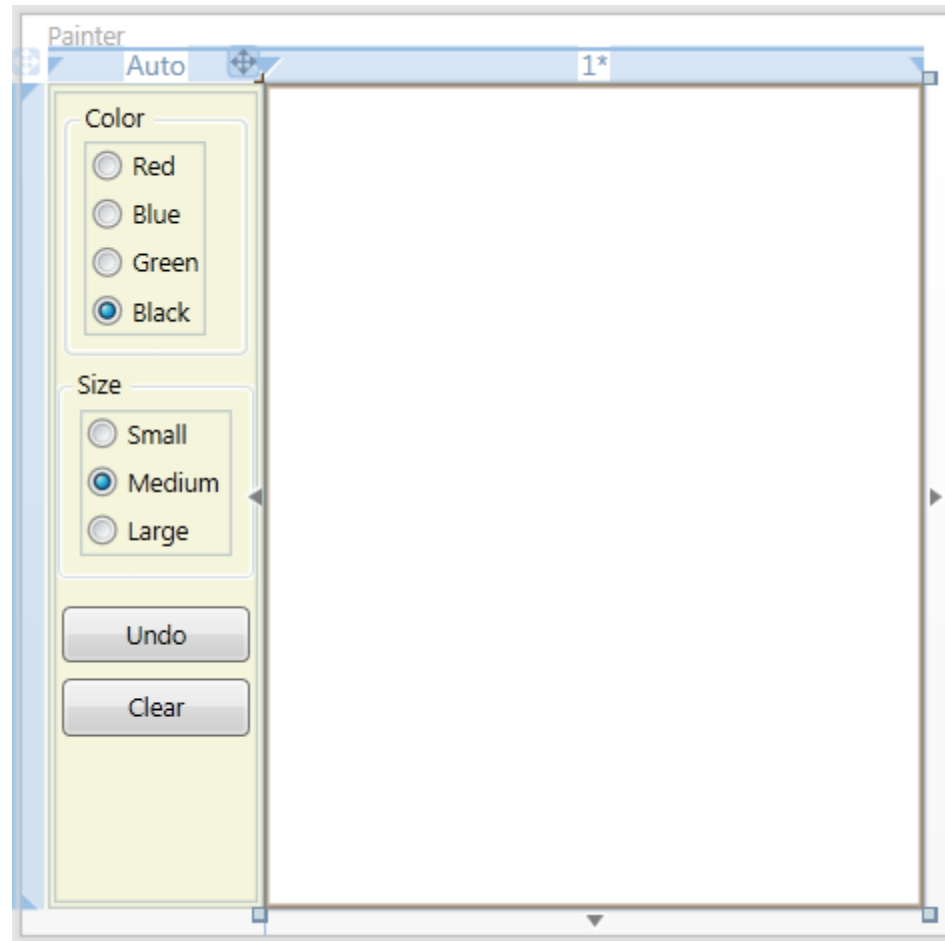
```

# Uygulama - Painter

**Canvas**

**Name = paintCanvas**

**Background = White**



# Uygulama - Painter

paintCanvas → Events

paintCanvas\_MouseLeftButtonDown

paintCanvas\_MouseRightButtonDown

paintCanvas\_MouseLeftButtonUp

paintCanvas\_MouseRightButtonUp

paintCanvas\_MouseMove

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" /> <!--defines a column-->
    <ColumnDefinition Width="*" />    <!--defines a column-->
  </Grid.ColumnDefinitions>
  <StackPanel Margin="3" Name="stackPanel1" >
    <GroupBox Header="Color" Name="colorGroupBox" Margin="3" HorizontalAlignment="Stretch" VerticalAlignment="Top">
      <StackPanel HorizontalAlignment="Left" Margin="3" Name="stackPanel2" VerticalAlignment="Top">
        <RadioButton Content="Red" Name="redRadioButton" Margin="3" Checked="redRadioButton_Checked"/>
        <RadioButton Content="Blue" Name="blueRadioButton" Margin="3" Checked="blueRadioButton_Checked" />
        <RadioButton Content="Green" Name="greenRadioButton" Margin="3" Checked="greenRadioButton_Checked"/>
        <RadioButton Content="Black" Name="blackRadioButton" Margin="3" Checked="blackRadioButton_Checked" IsChecked="True" />
      </StackPanel>
    </GroupBox>
    <GroupBox Header="Size" Height="86">
      <StackPanel Margin="3" Height="59" Width="61">
        <RadioButton Content="Small" Name="smallRadioButton" Margin="3" Checked="smallRadioButton_Checked"/>
        <RadioButton Content="Medium" Name="mediumRadioButton" IsChecked="True" Checked="mediumRadioButton_Checked" Margin="3"/>
        <RadioButton Content="Large" Name="largeRadioButton" Margin="3" Checked="largeRadioButton_Checked"/>
      </StackPanel>
    </GroupBox>

    <Button Height="23" Content="Undo" Name="undoButton" Width="75" Margin="3,10,3,3" Click="undoButton_Click"/>

    <Button Height="23" Content="Clear" Name="clearButton" Width="75" Margin="3" Click="clearButton_Click"/>

  </StackPanel>
  <Canvas Grid.Column="1" Margin="0" Name="paintCanvas" Background="White" MouseMove="paintCanvas_MouseMove"
    MouseLeftButtonDown="paintCanvas_MouseLeftButtonDown" MouseLeftButtonUp="paintCanvas_MouseLeftButtonUp"
    MouseRightButtonDown="paintCanvas_MouseRightButtonDown" MouseRightButtonUp="paintCanvas_MouseRightButtonUp"/>
</Grid>
```

# **UYGULAMA – PAINTER**

## **OLAY İŞLEME (EVENT HANDLING)**

# Uygulama - Painter

```
public partial class MainWindow : Window
{
    private int diameter = 8; // set diameter of circle
    private Brush brushColor = Brushes.Black; // set the drawing color
    private bool shouldErase = false; // specify whether to erase
    private bool shouldPaint = false; // specify whether to paint

    private enum Sizes // size constants for diameter of the circle
    {
        SMALL = 4,
        MEDIUM = 8,
        LARGE = 10
    }

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



# Uygulama - Painter

```
// paints a circle on the Canvas
private void PaintCircle( Brush circleColor, Point position )
{
    Ellipse newEllipse = new Ellipse(); // create an Ellipse

    newEllipse.Fill = circleColor; // set Ellipse's color
    newEllipse.Width = diameter; // set its horizontal diameter
    newEllipse.Height = diameter; // set its vertical diameter

    // set the Ellipse's position
    Canvas.SetTop( newEllipse, position.Y );
    Canvas.SetLeft( newEllipse, position.X );

    paintCanvas.Children.Add( newEllipse );
}
```

# Uygulama - Painter

```
// handles paintCanvas's MouseLeftButtonDown event
private void paintCanvas_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    shouldPaint = true; // OK to draw on the Canvas
}

// handles paintCanvas's MouseLeftButtonUp event
private void paintCanvas_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    shouldPaint = false; // do not draw on the Canvas
}
```

# Uygulama - Painter

```
// handles paintCanvas's MouseRightButtonUp event
private void paintCanvas_MouseRightButtonUp(object sender, MouseButtonEventArgs e)
{
    shouldErase = false; // do not erase the Canvas
}

// handles paintCanvas's MouseRightButtonDown event
private void paintCanvas_MouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    shouldErase = true; // OK to erase the Canvas
}
```

# Uygulama - Painter

```
private void paintCanvas_MouseMove(object sender, MouseEventArgs e)
{
    if (shouldPaint)
    {
        // draw a circle of selected color at current mouse position
        Point mousePosition = e.GetPosition(paintCanvas);
        PaintCircle(brushColor, mousePosition);
    }
    else if (shouldErase)
    {
        // erase by drawing circles of the Canvas's background color
        Point mousePosition = e.GetPosition(paintCanvas);
        PaintCircle(paintCanvas.Background, mousePosition);
    }
}
```

# Uygulama - Painter

```
private void redRadioButton_Checked(object sender, RoutedEventArgs e)
{
    brushColor = Brushes.Red;
}
```

```
private void greenRadioButton_Checked(object sender, RoutedEventArgs e)
{
    brushColor = Brushes.Green;
}
```

```
private void blueRadioButton_Checked(object sender, RoutedEventArgs e)
{
    brushColor = Brushes.Blue;
}
```

```
private void blackRadioButton_Checked(object sender, RoutedEventArgs e)
{
    brushColor = Brushes.Black;
}
```

# Uygulama - Painter

```
private void smallRadioButton_Checked(object sender, RoutedEventArgs e)
{
    diameter = (int)Sizes.SMALL;
}
```

```
private void mediumRadioButton_Checked(object sender, RoutedEventArgs e)
{
    diameter = (int)Sizes.MEDIUM;
}
```

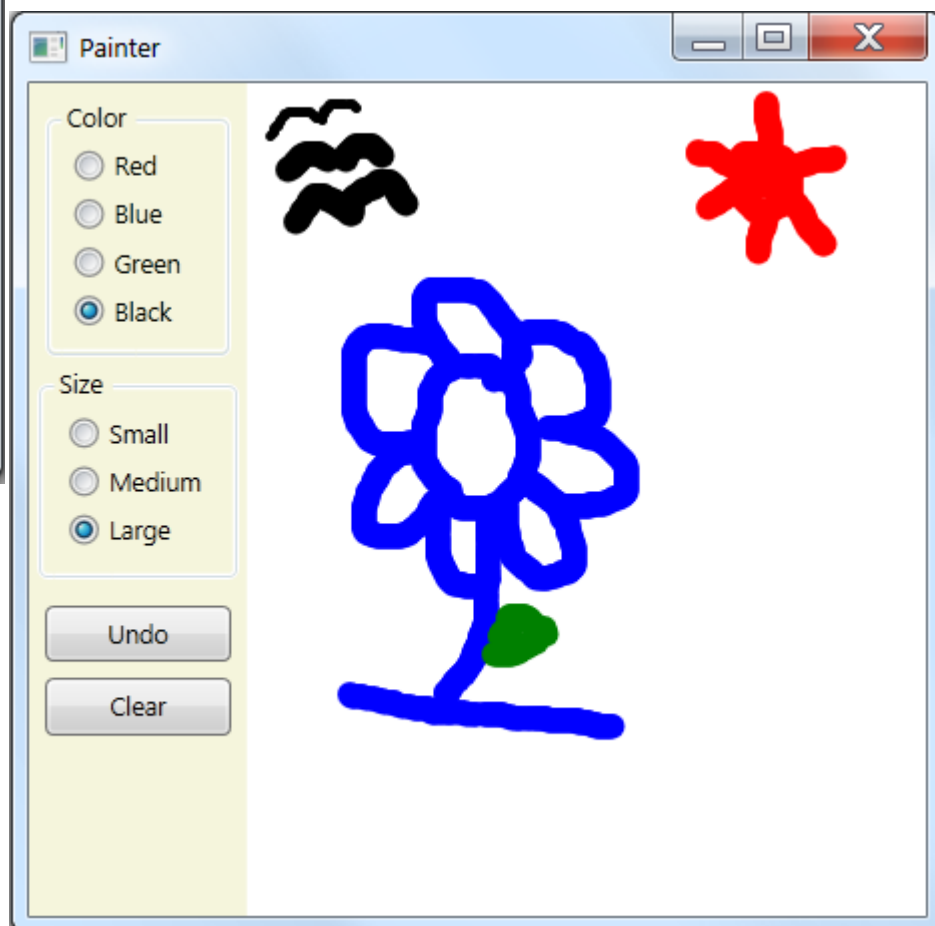
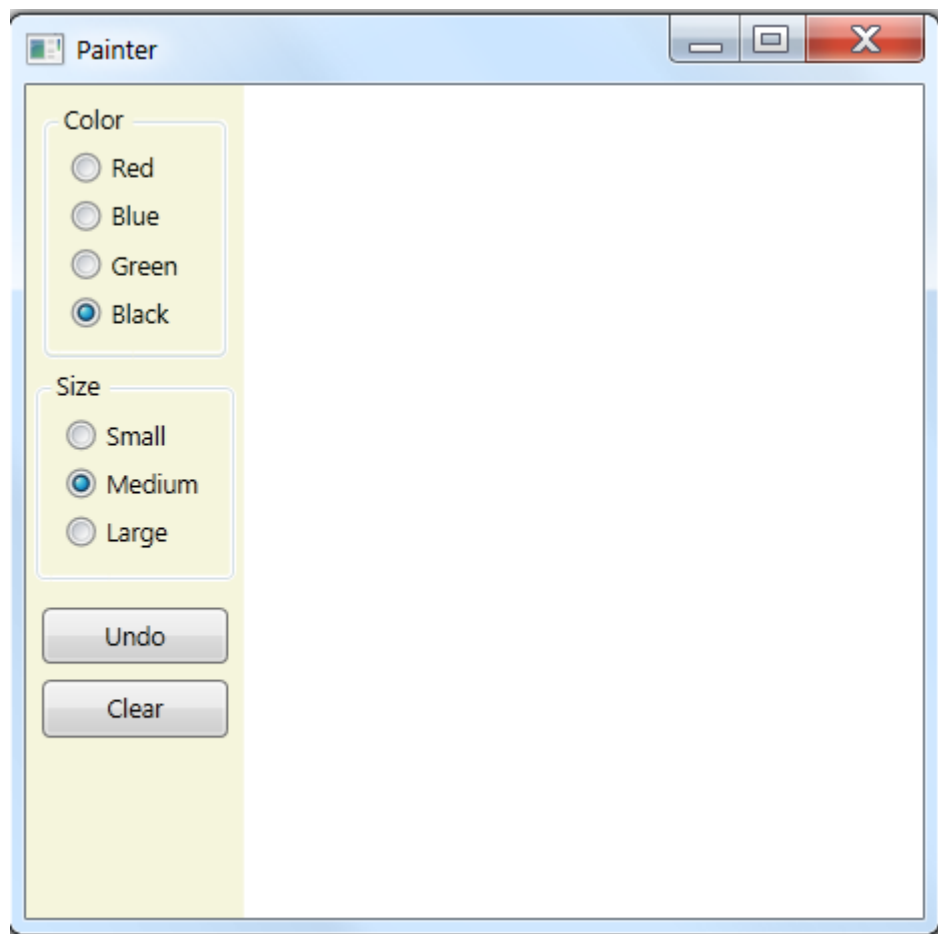
```
private void largeRadioButton_Checked(object sender, RoutedEventArgs e)
{
    diameter = (int)Sizes.LARGE;
}
```

# Uygulama - Painter

```
private void undoButton_Click(object sender, RoutedEventArgs e)
{
    int count = paintCanvas.Children.Count;

    // if there are any shapes on Canvas remove the last one added
    if (count > 0)
        paintCanvas.Children.RemoveAt(count - 1);
}

private void clearButton_Click(object sender, RoutedEventArgs e)
{
    paintCanvas.Children.Clear(); // clear the canvas
}
```





**WPF KOMUT KÜTÜPHANESİ**

# WPF Komut Kütüphanesi

- WPF, komut (command) tekniğini desteklemektedir.
  - Bir eylem ya da görev (task), birçok kullanıcı etkileşimi tarafından tetiklenebilir (trigger).
- Komutlar, **Icommand** arayüzünün (interface) gerçekleştirimidir.

# WPF Komut Kütüphanesi

- Komutlar, bir görevin kullanılabilirlik (availability) durumunu **senkronize** etmenize olanak sağlar.
  - ÖR: Herhangi bir metin seçilmemiş ise, tanımlanan “Kopyala” komutu otomatik olarak etkisiz (disabled) olacaktır.

# WPF Komut Kütüphanesi

Common built-in commands from the WPF command library

## *ApplicationCommands properties*

New	Open	Save	Close
Cut	Copy	Paste	

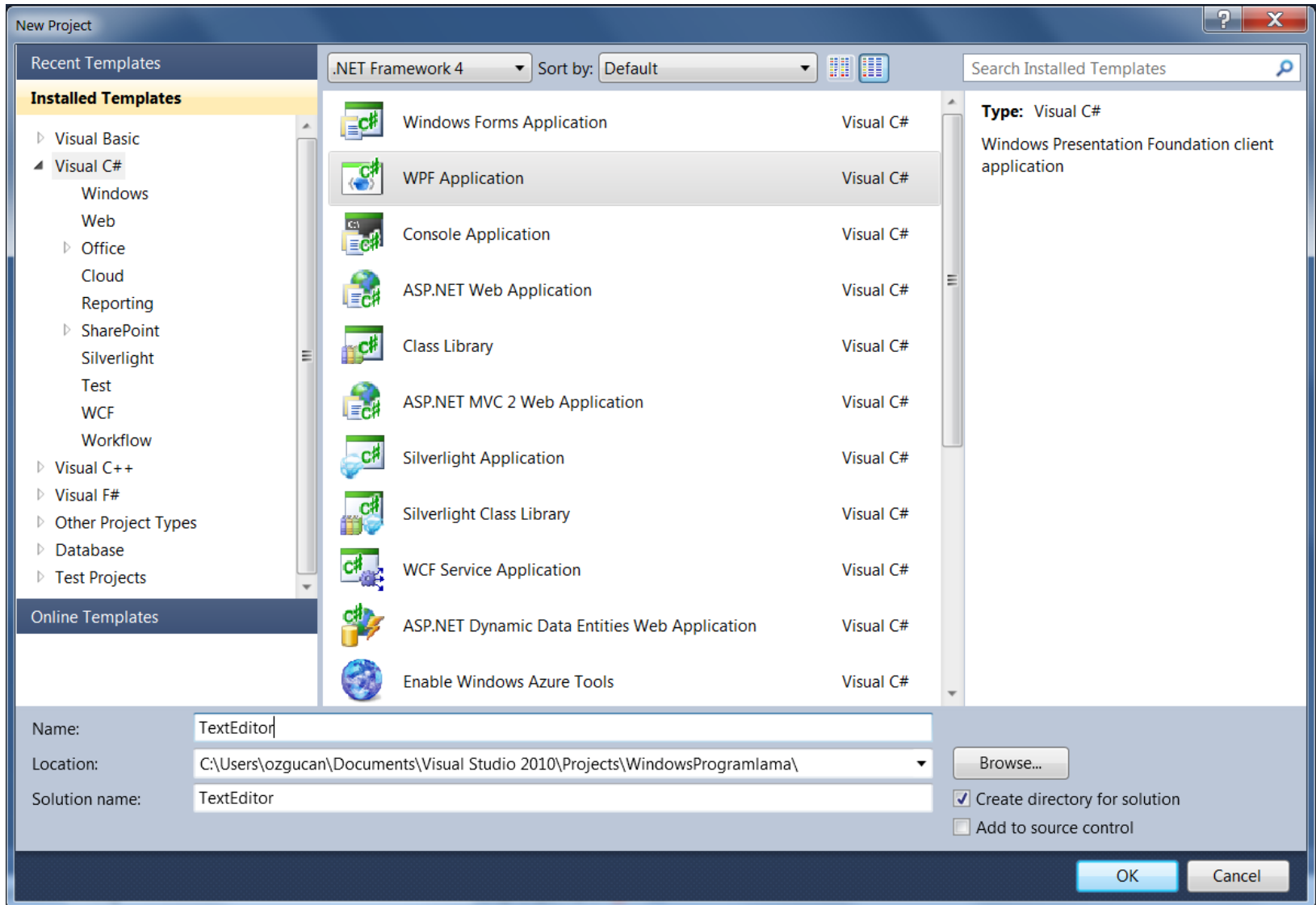
## *EditingCommands properties*

ToggleBold	ToggleItalic	ToggleUnderline
------------	--------------	-----------------

## *MediaCommands properties*

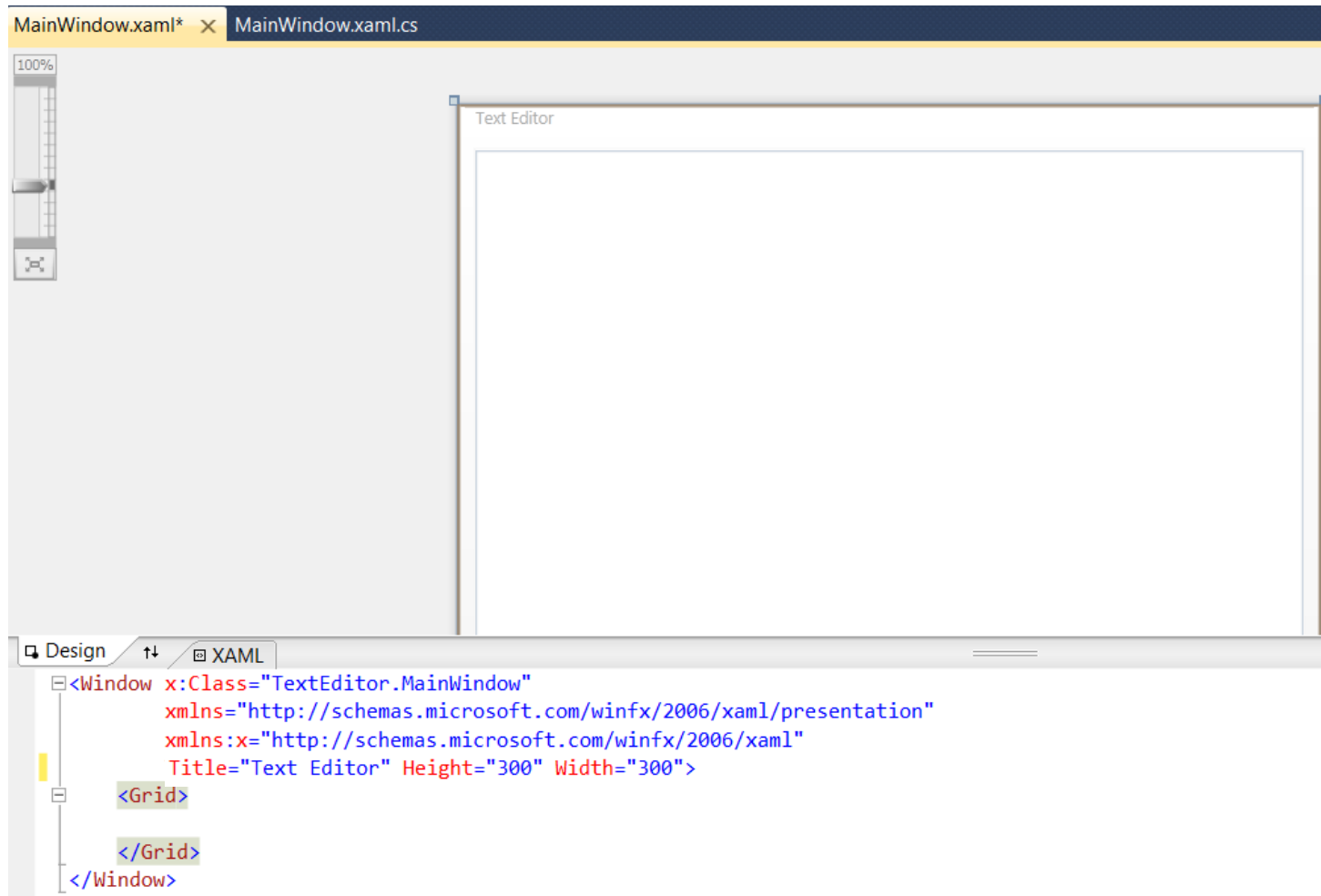
Play	Stop	Rewind	FastForward
IncreaseVolume	DecreaseVolume	NextTrack	PreviousTrack

# Uygulama – Text Editor

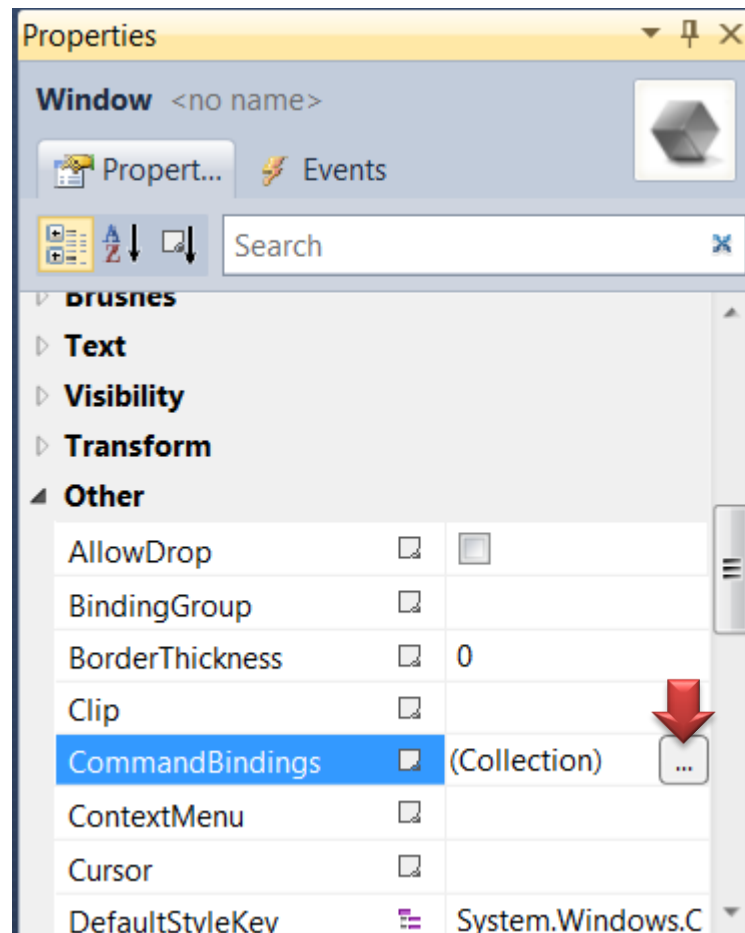


# Uygulama – Text Editor

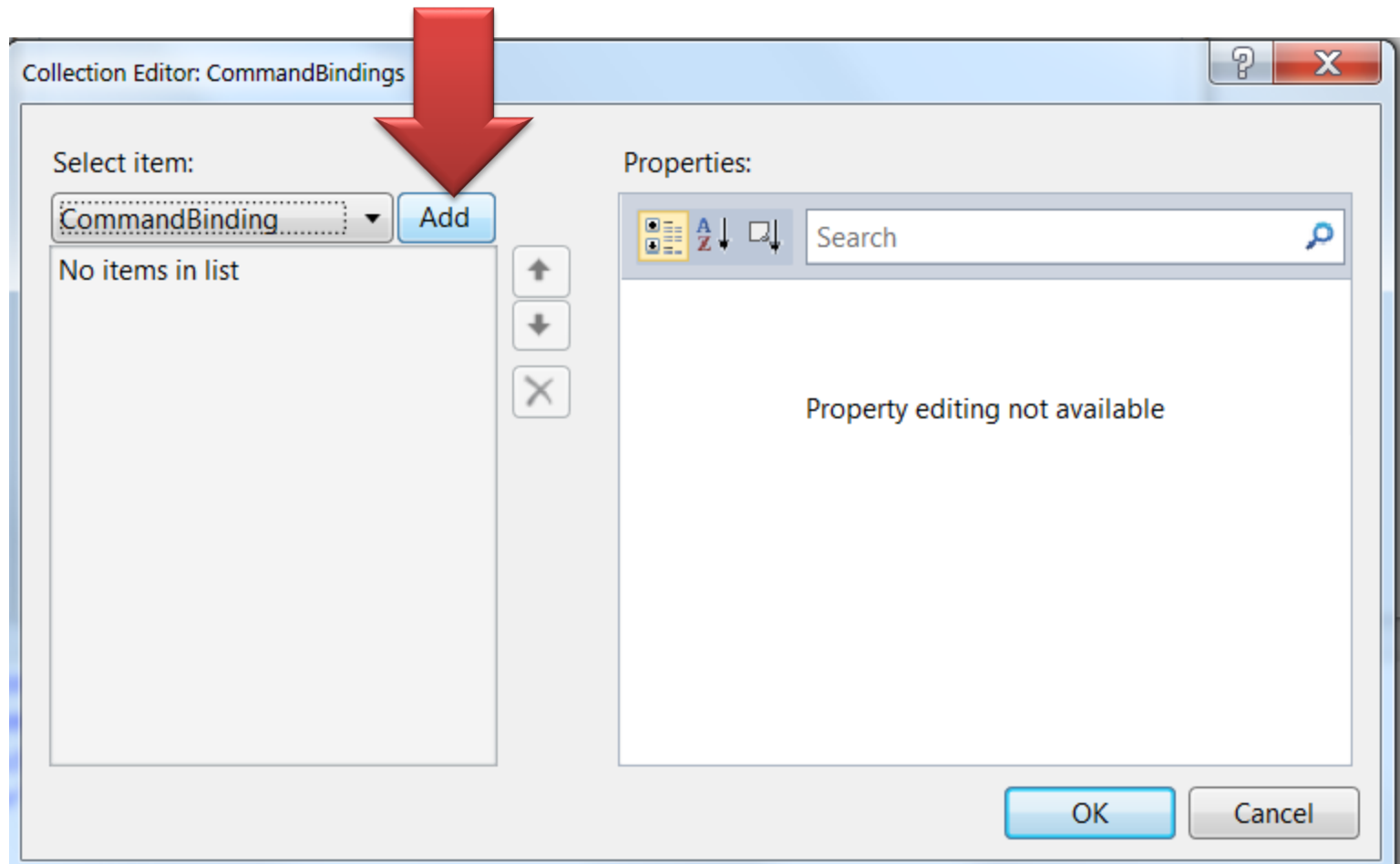
Title = Text Editor, Height & Width = 300



# Uygulama – Text Editor

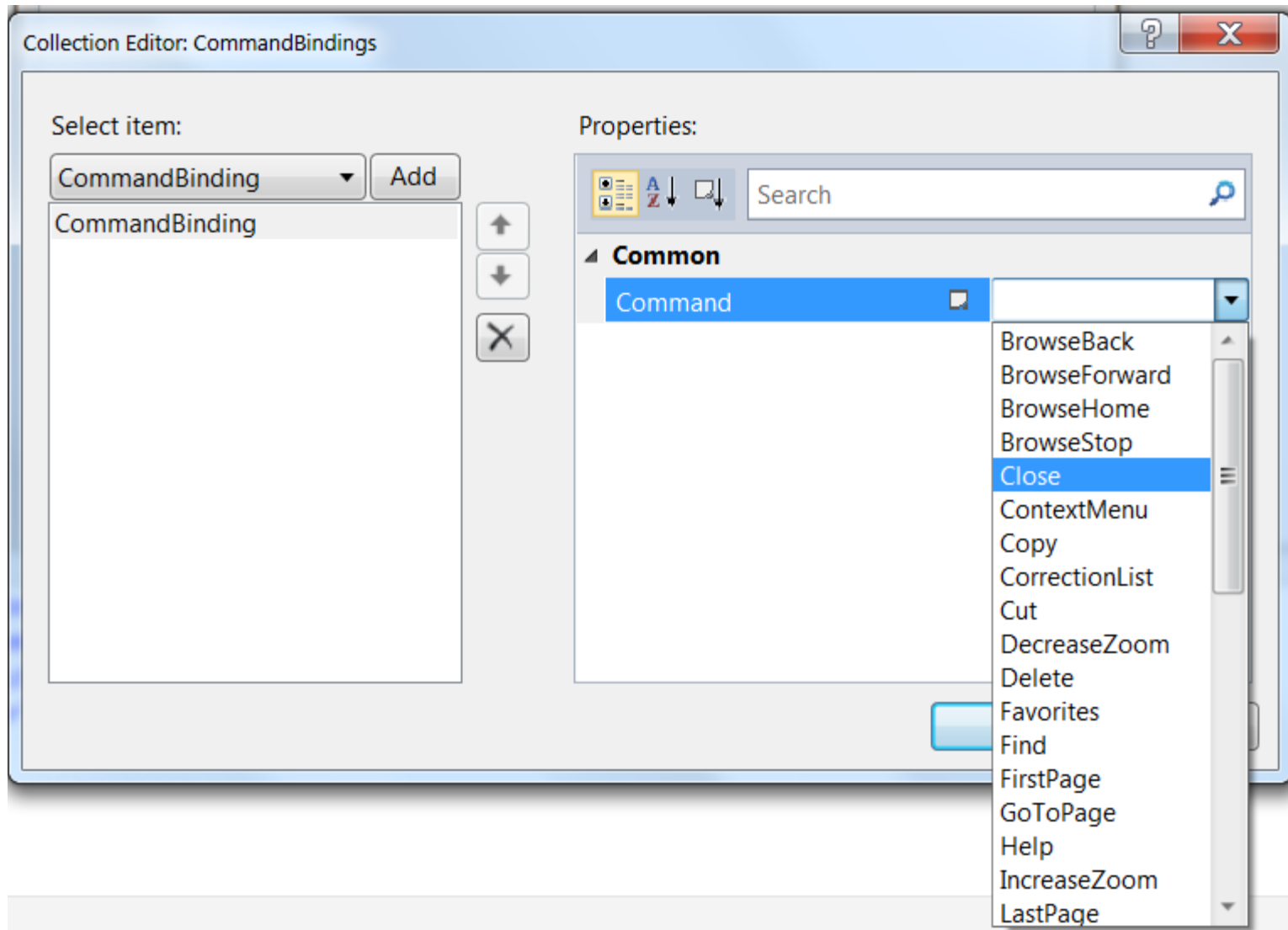


# Uygulama – Text Editor

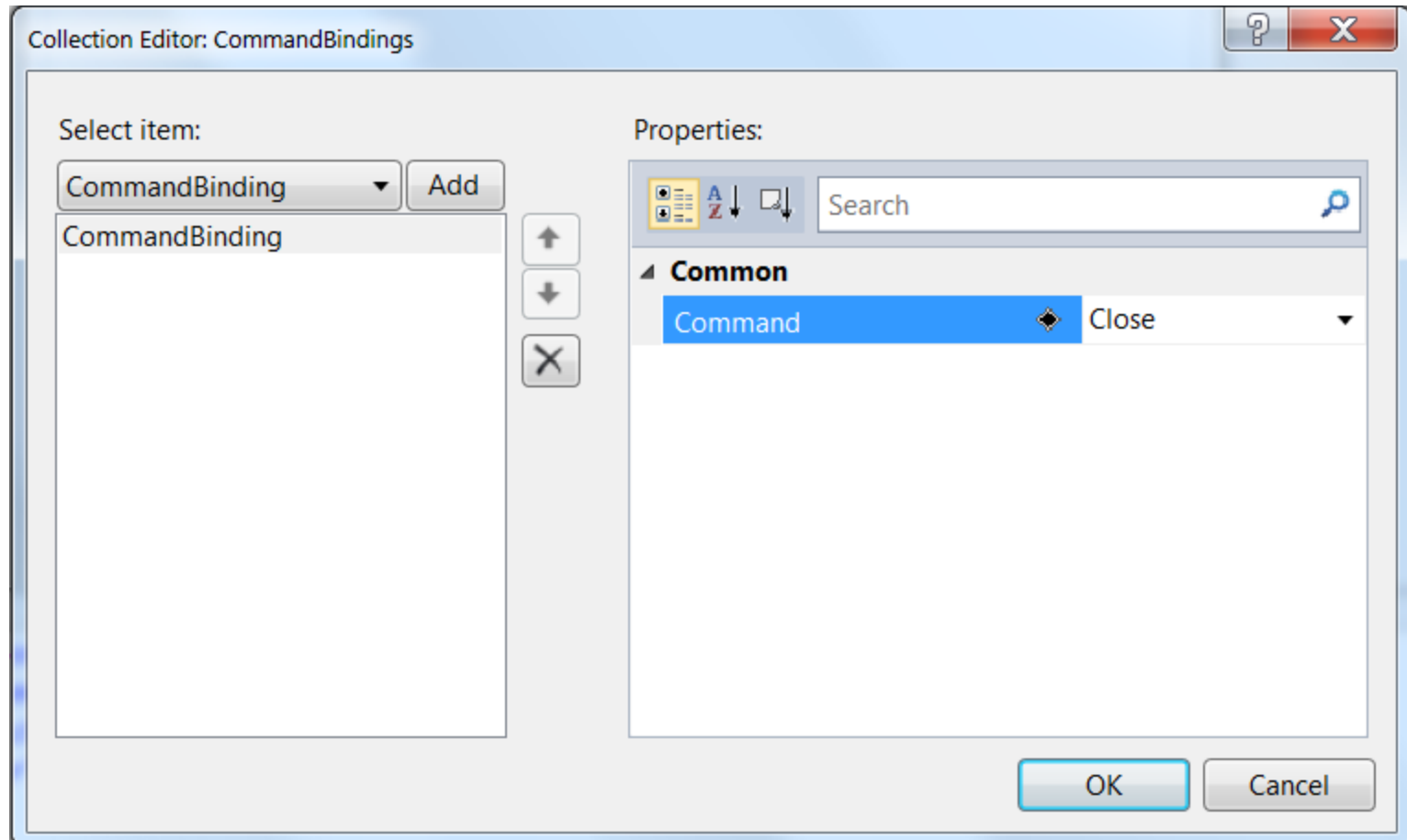




# Uygulama – Text Editor



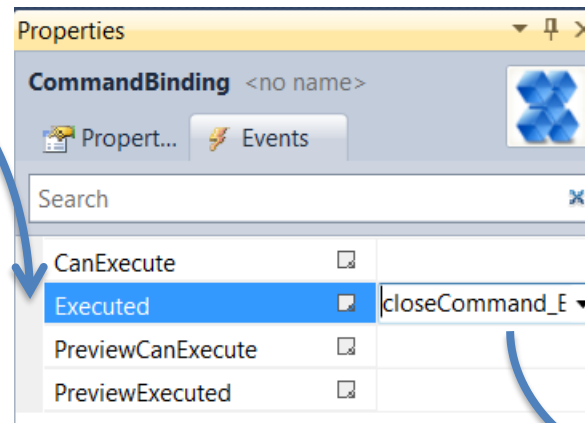
# Uygulama – Text Editor



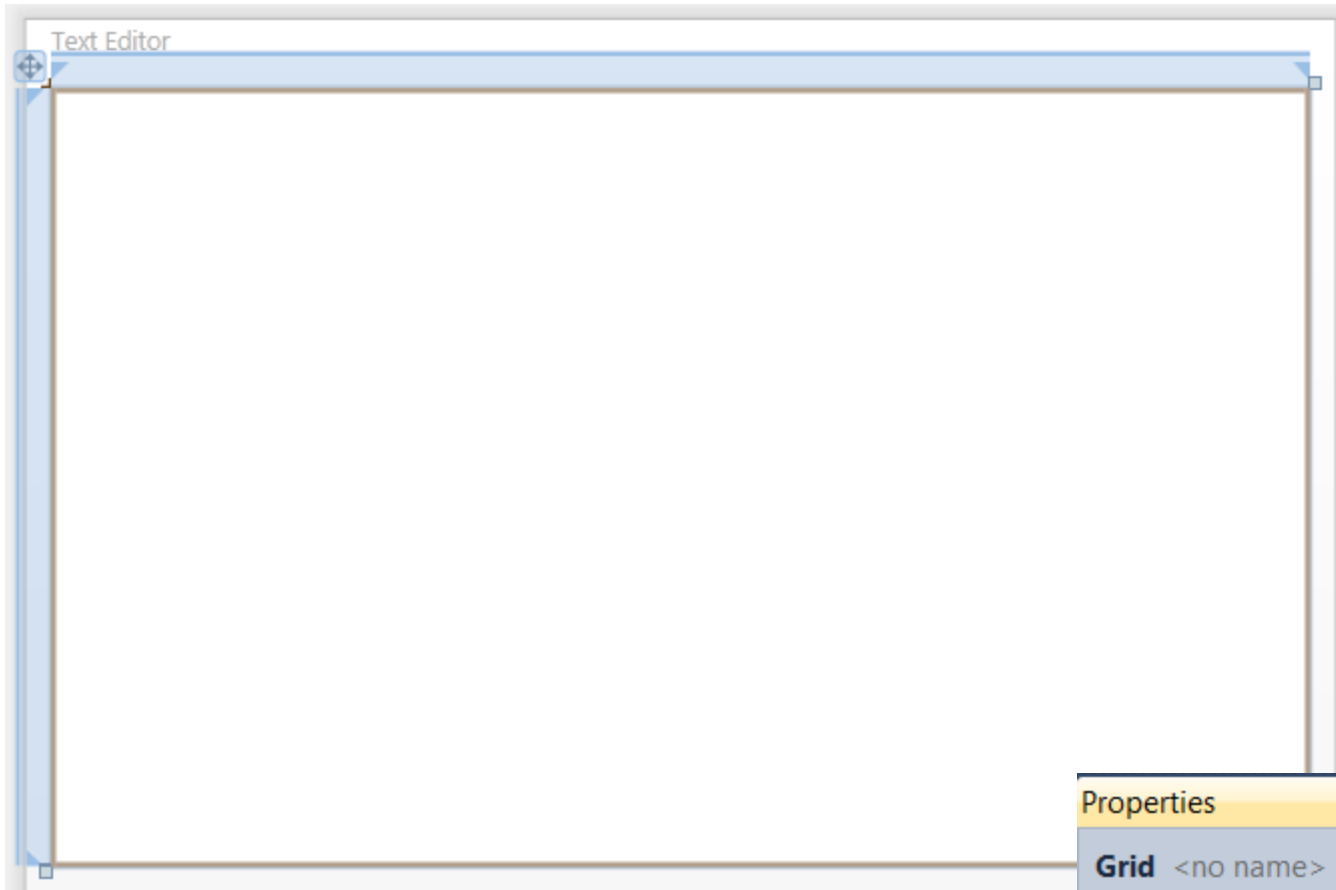
# Uygulama – Text Editor



```
<Window.CommandBindings>  
  <CommandBinding Command="ApplicationCommands.Close" />  
</Window.CommandBindings>
```

Events








```
<Window.CommandBindings>  
  <CommandBinding Command="ApplicationCommands.Close" Executed="closeCommand_Executed" />  
</Window.CommandBindings>
```











Properties  

**Grid** <no name>

 Property...  Events

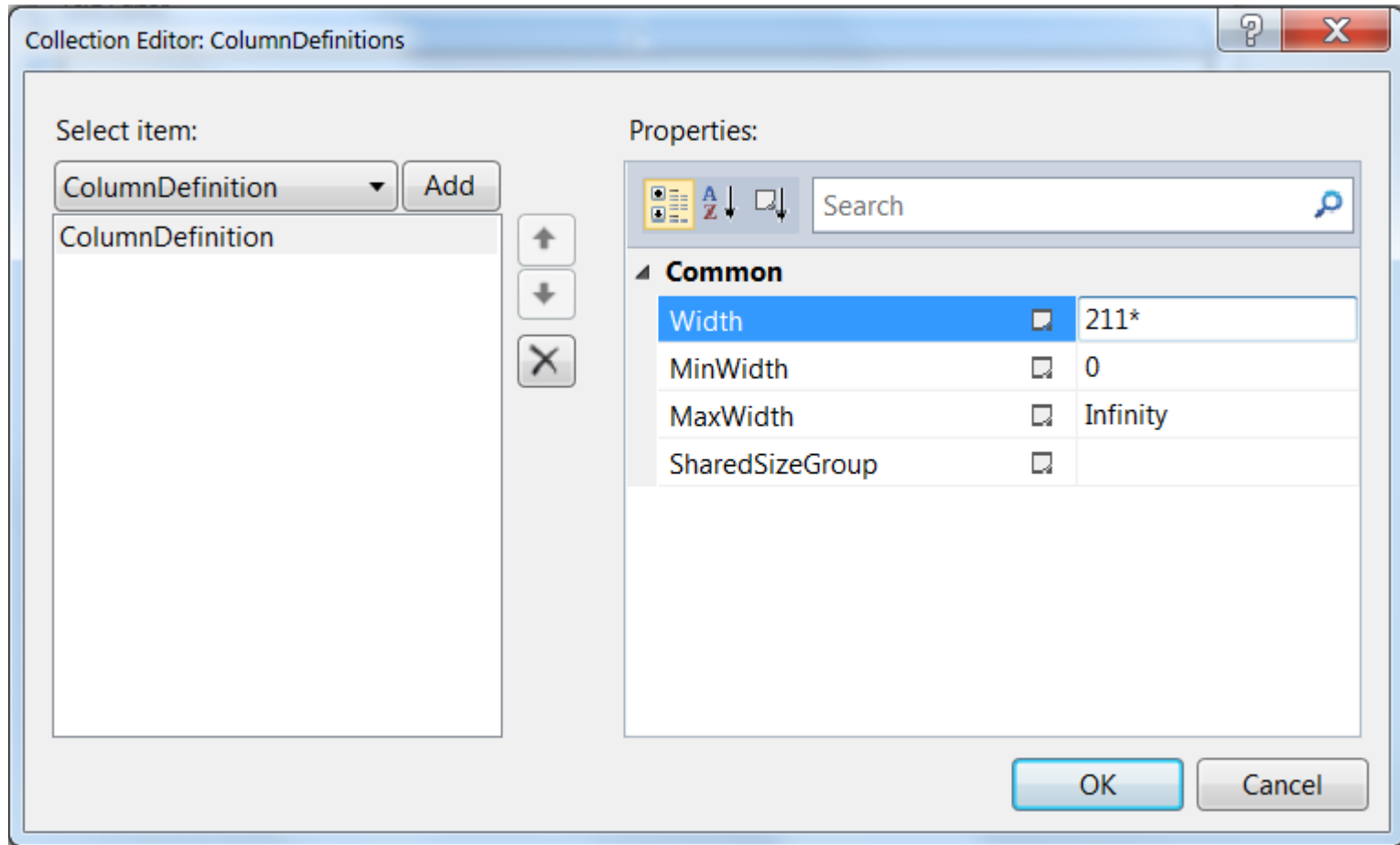
   Search

▲ **Common**

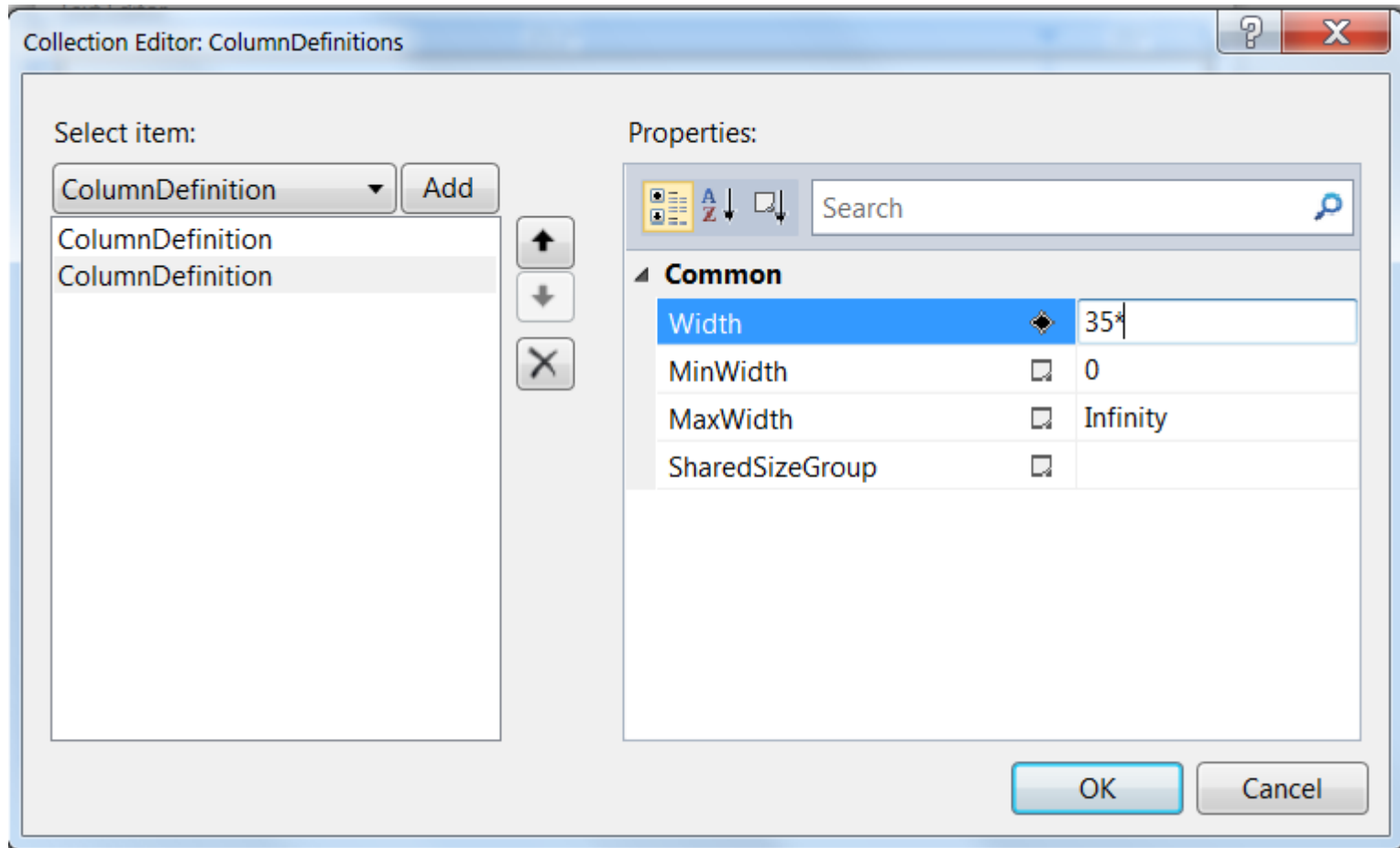
DataContext		Binding...	
Style		Resource...	
ColumnDefinitions		(Collection)	
RowDefinitions		(Collection)	

▶ **Layout**

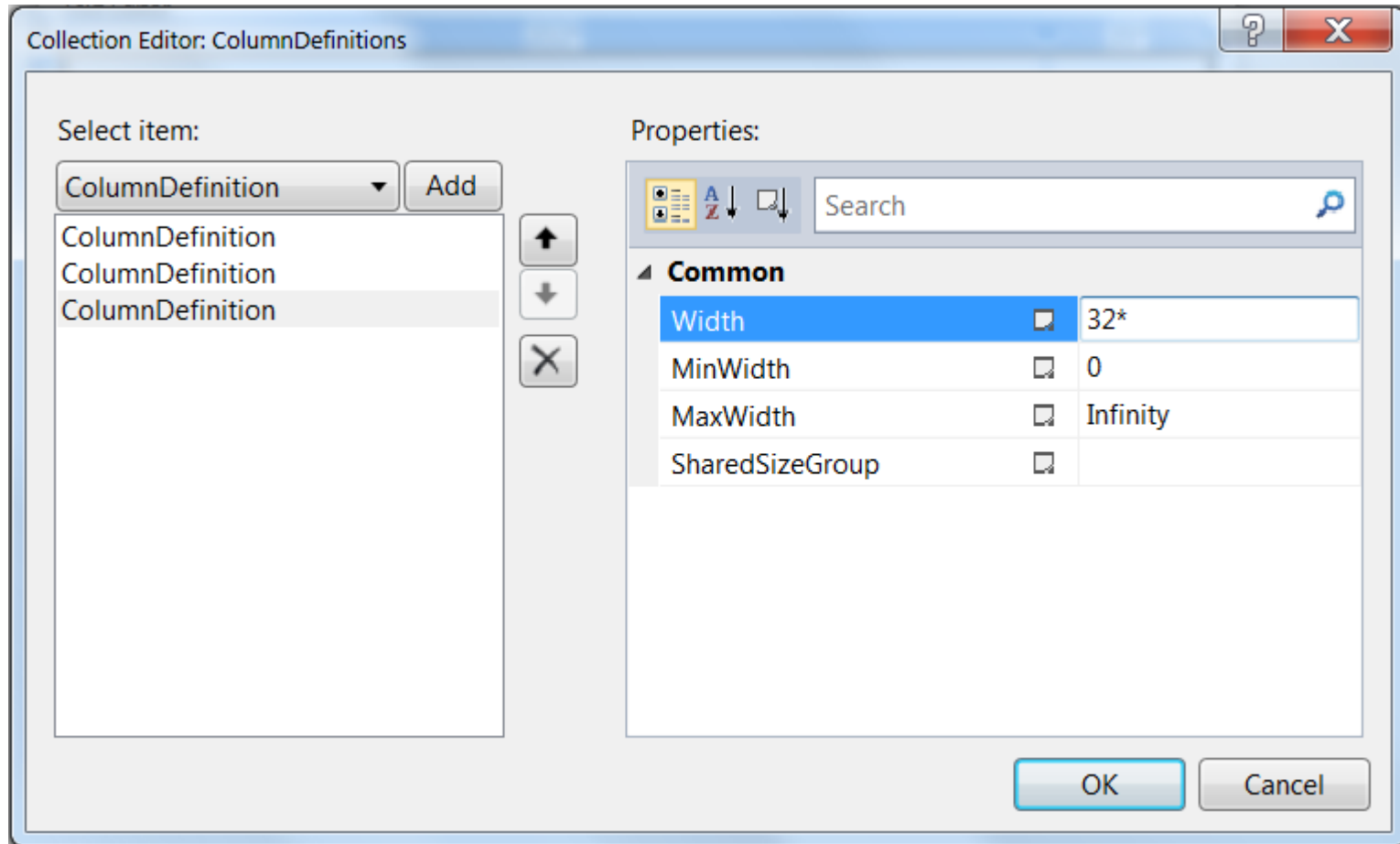
# Uygulama – Text Editor

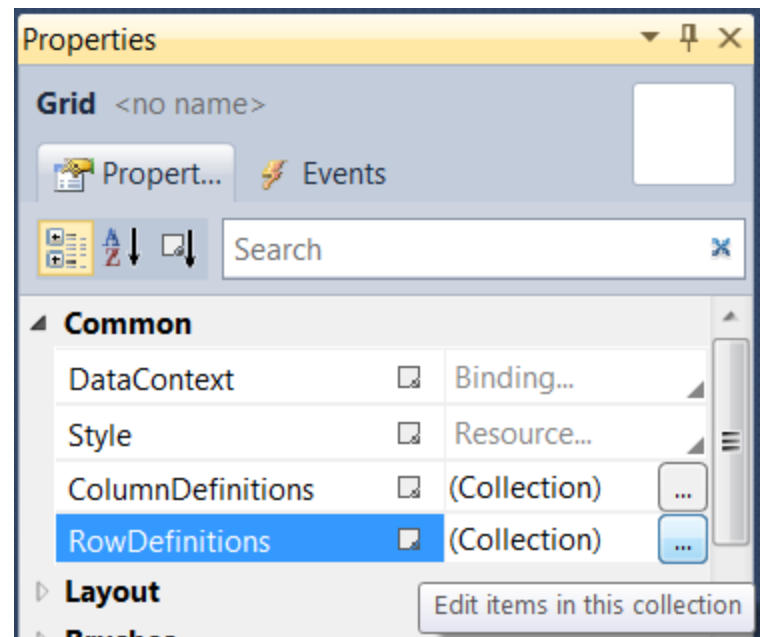
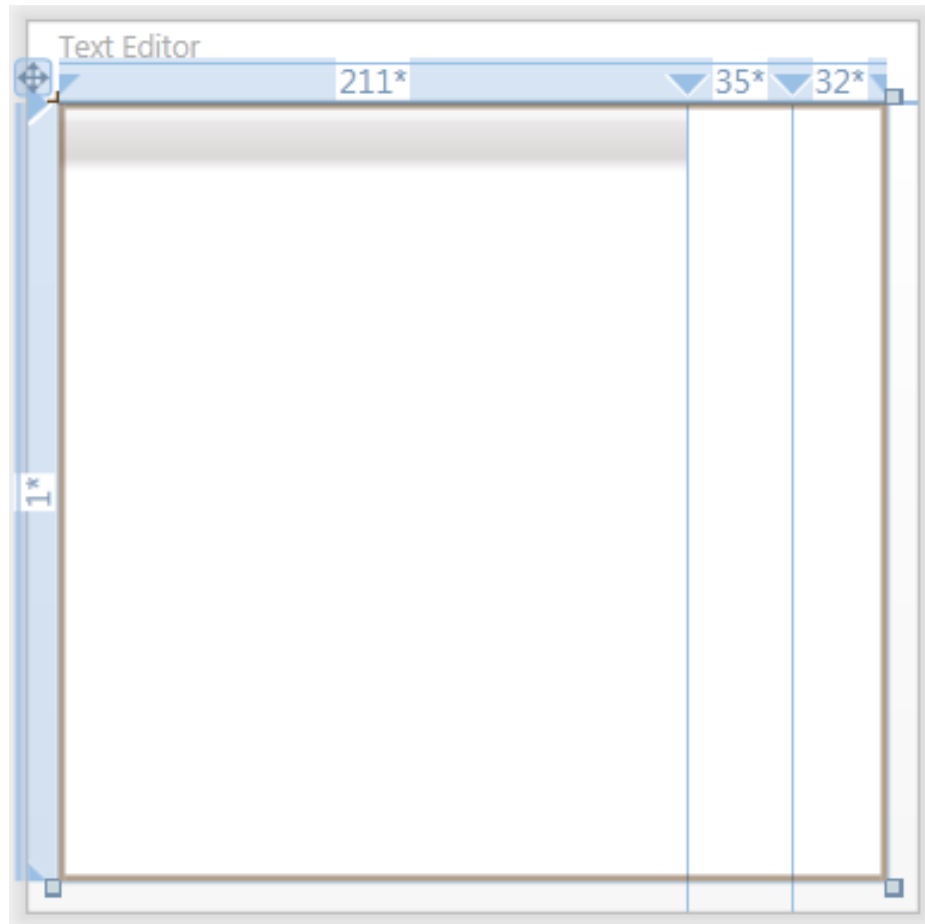


# Uygulama – Text Editor



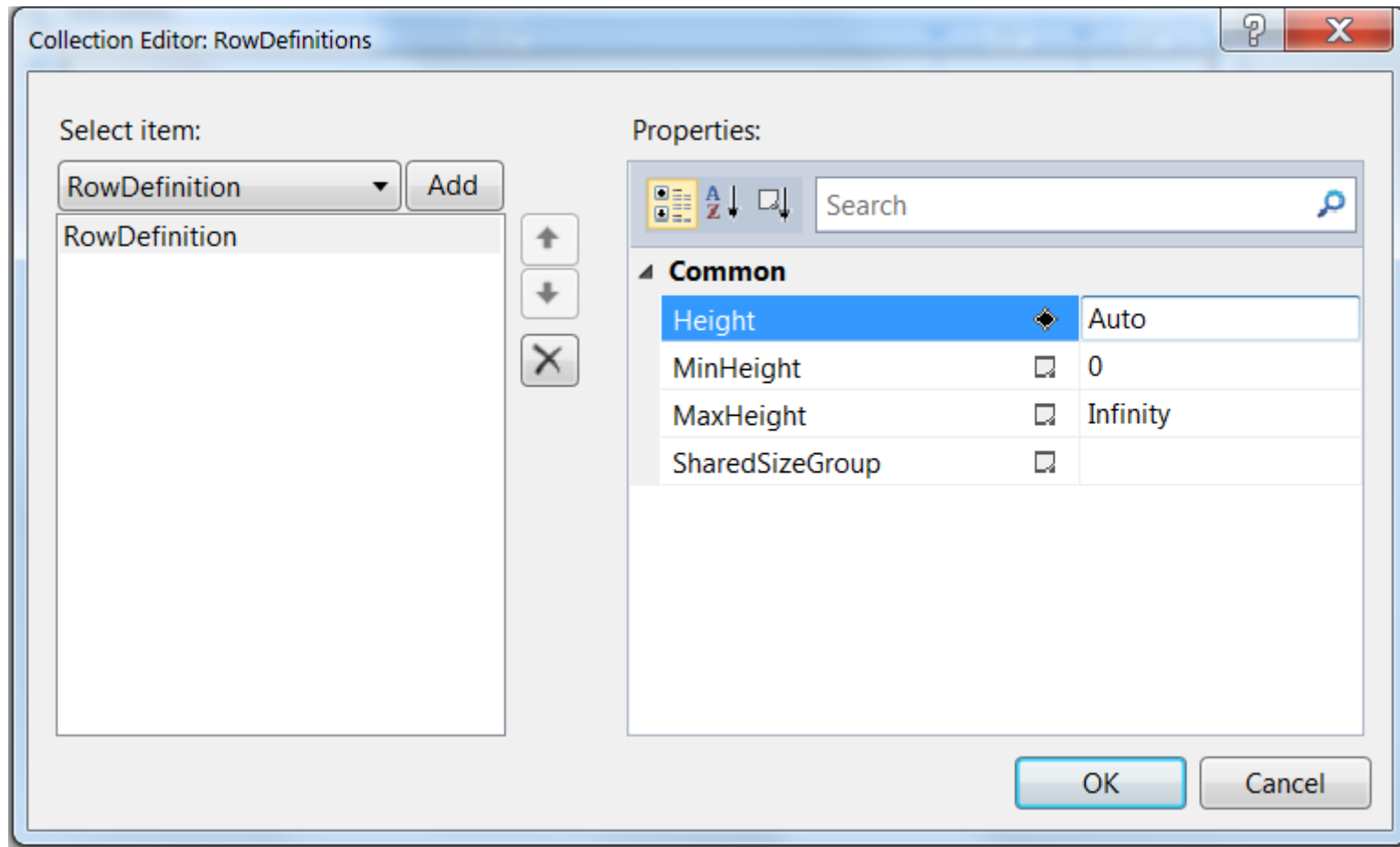
# Uygulama – Text Editor



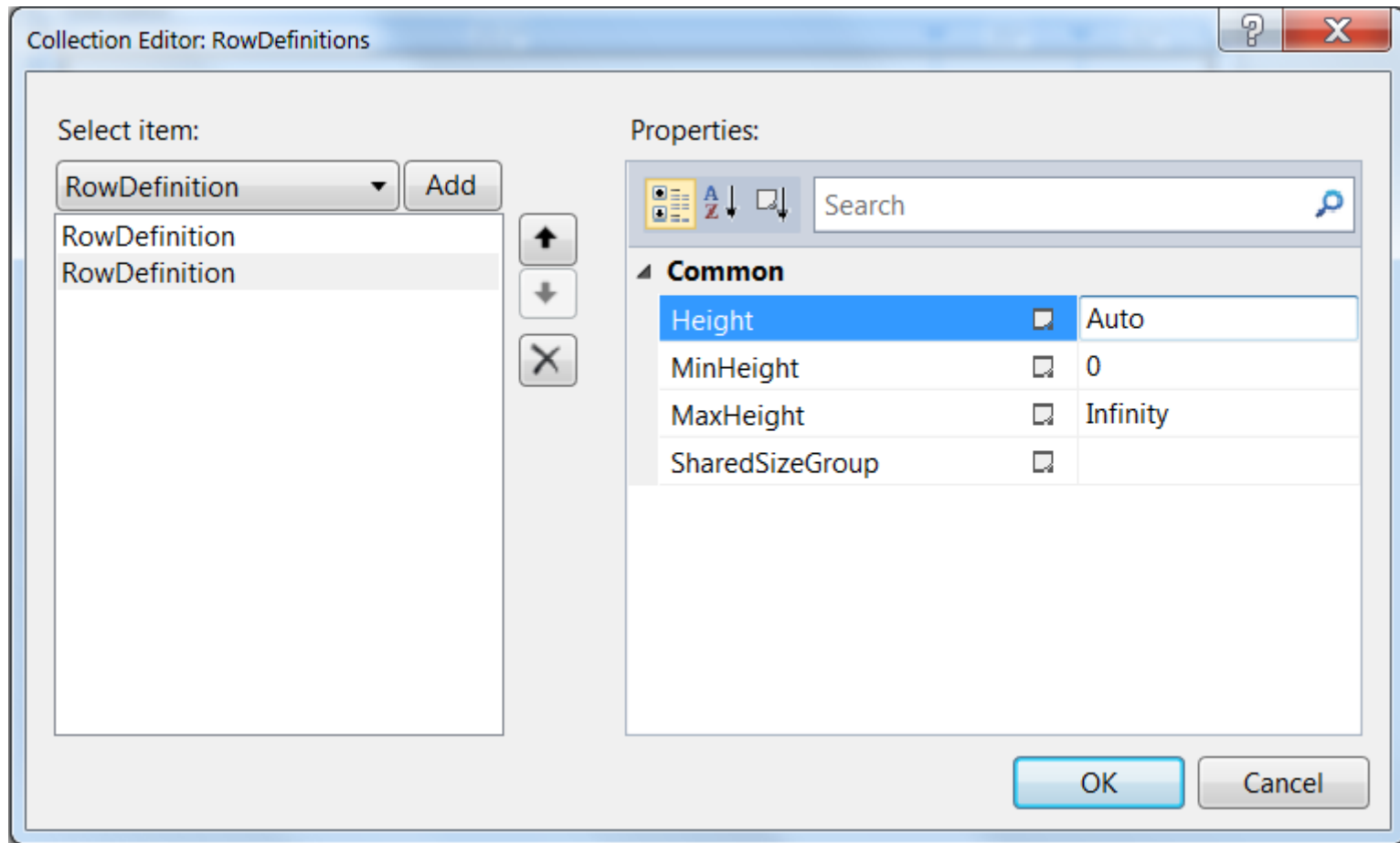




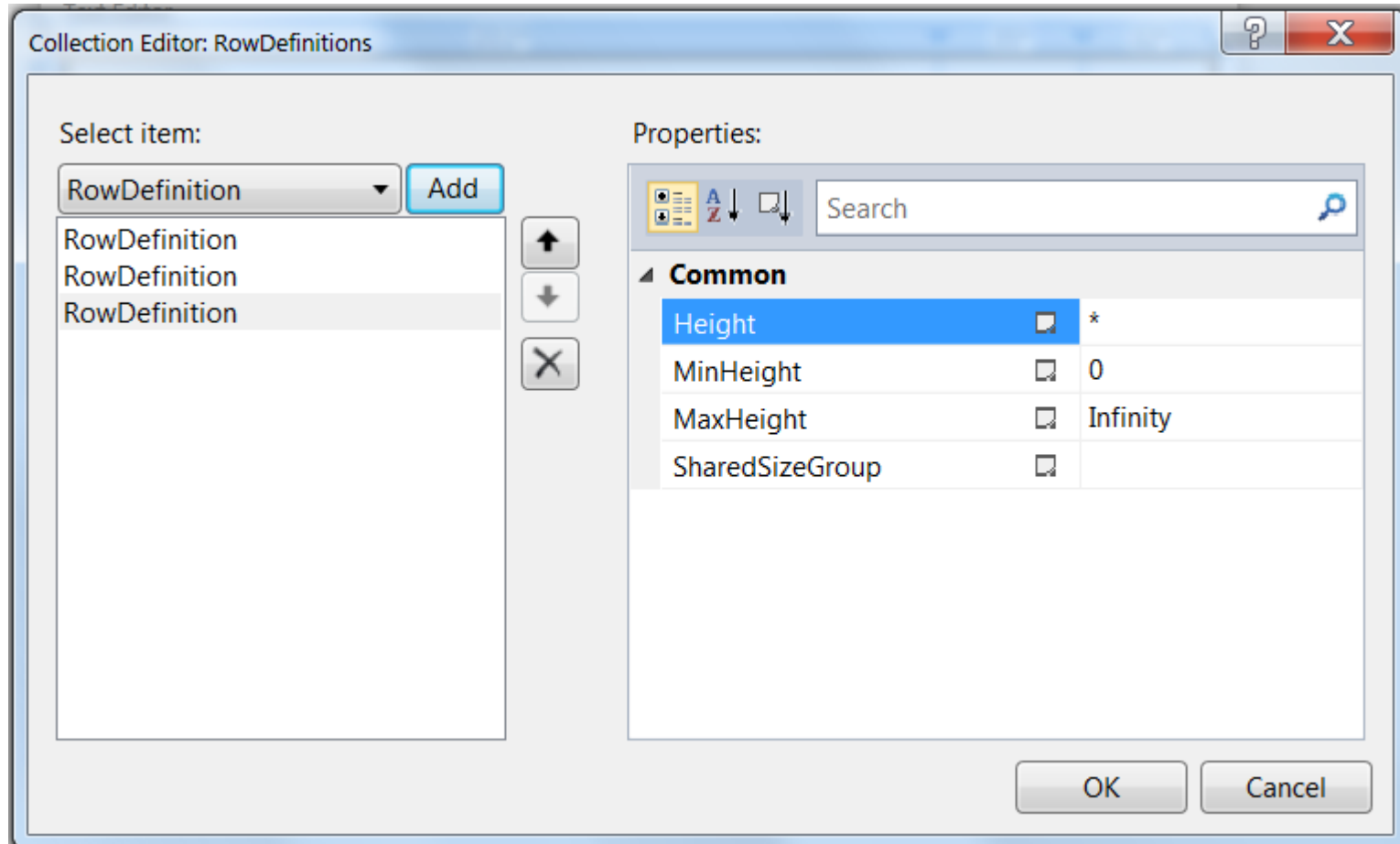
# Uygulama – Text Editor

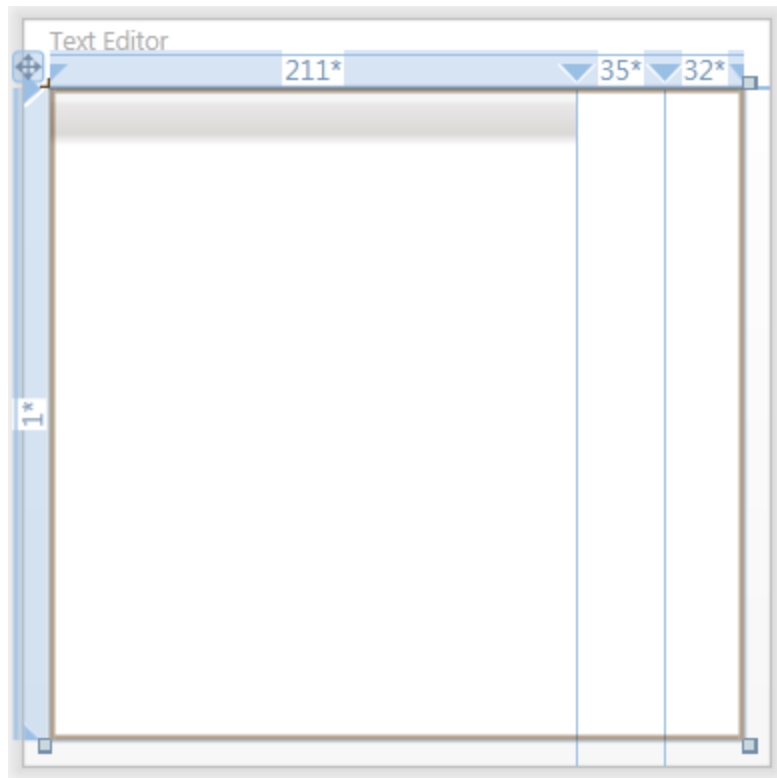


# Uygulama – Text Editor



# Uygulama – Text Editor

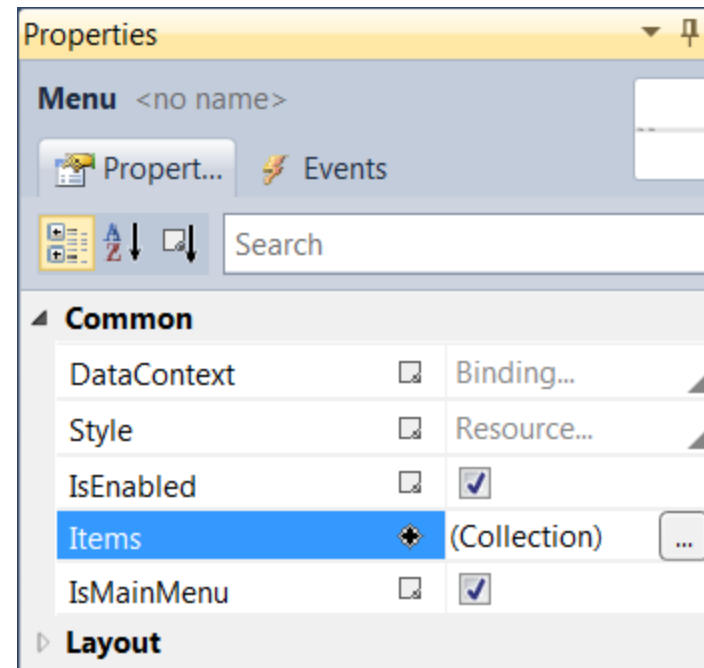
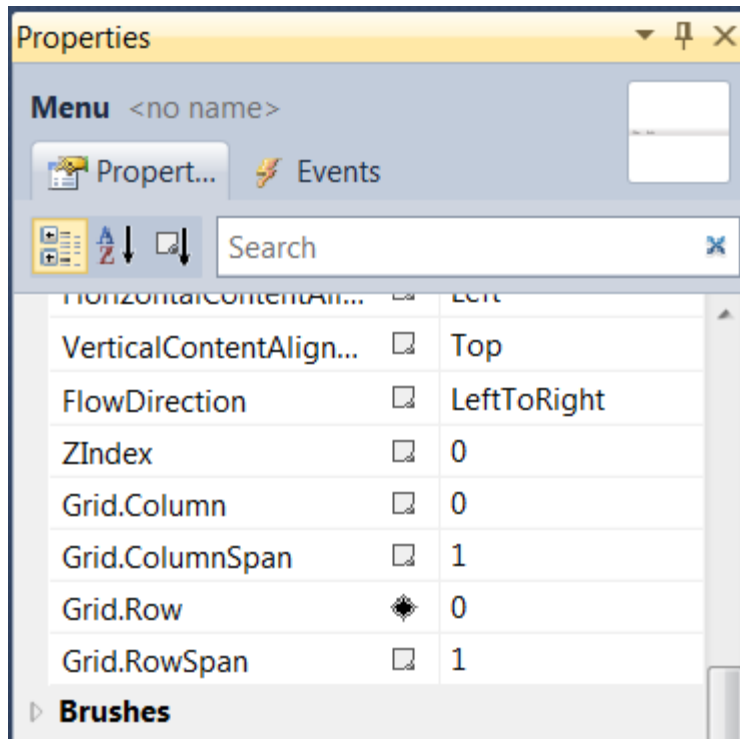




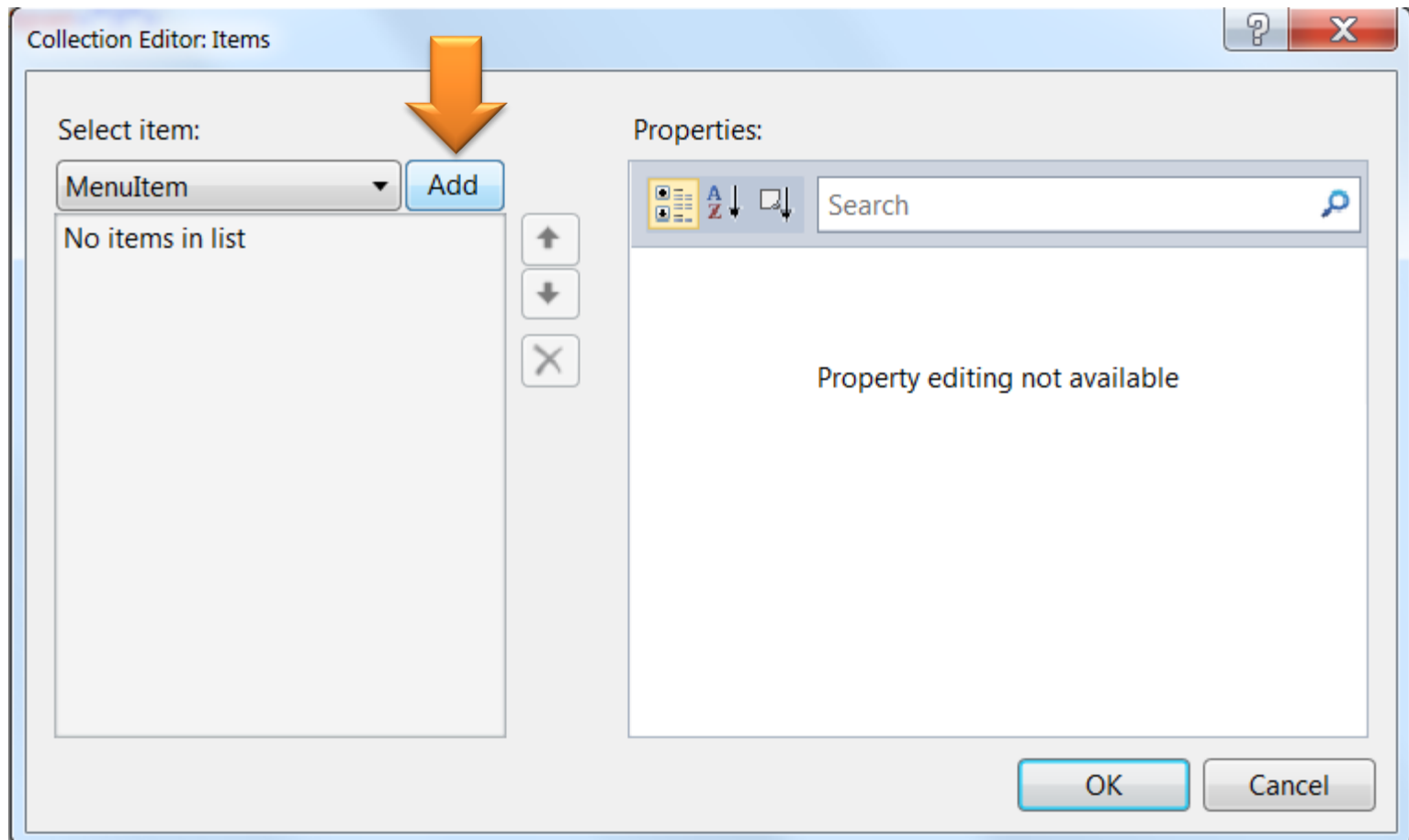
```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="211*" />
    <ColumnDefinition Width="35*" />
    <ColumnDefinition Width="32*" />
  </Grid.ColumnDefinitions>
</Grid>
```

# Uygulama – Text Editor

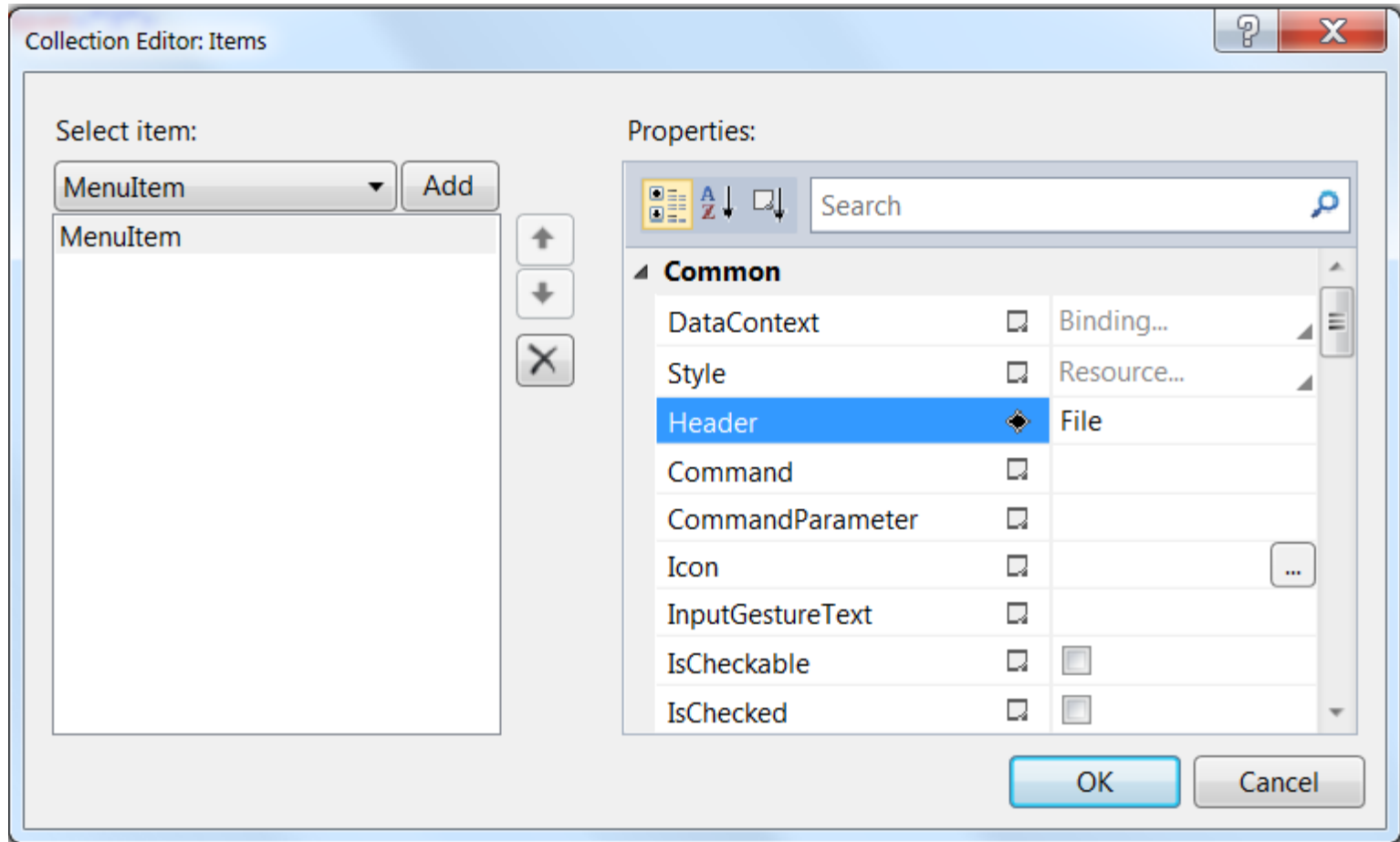
- **Menu**



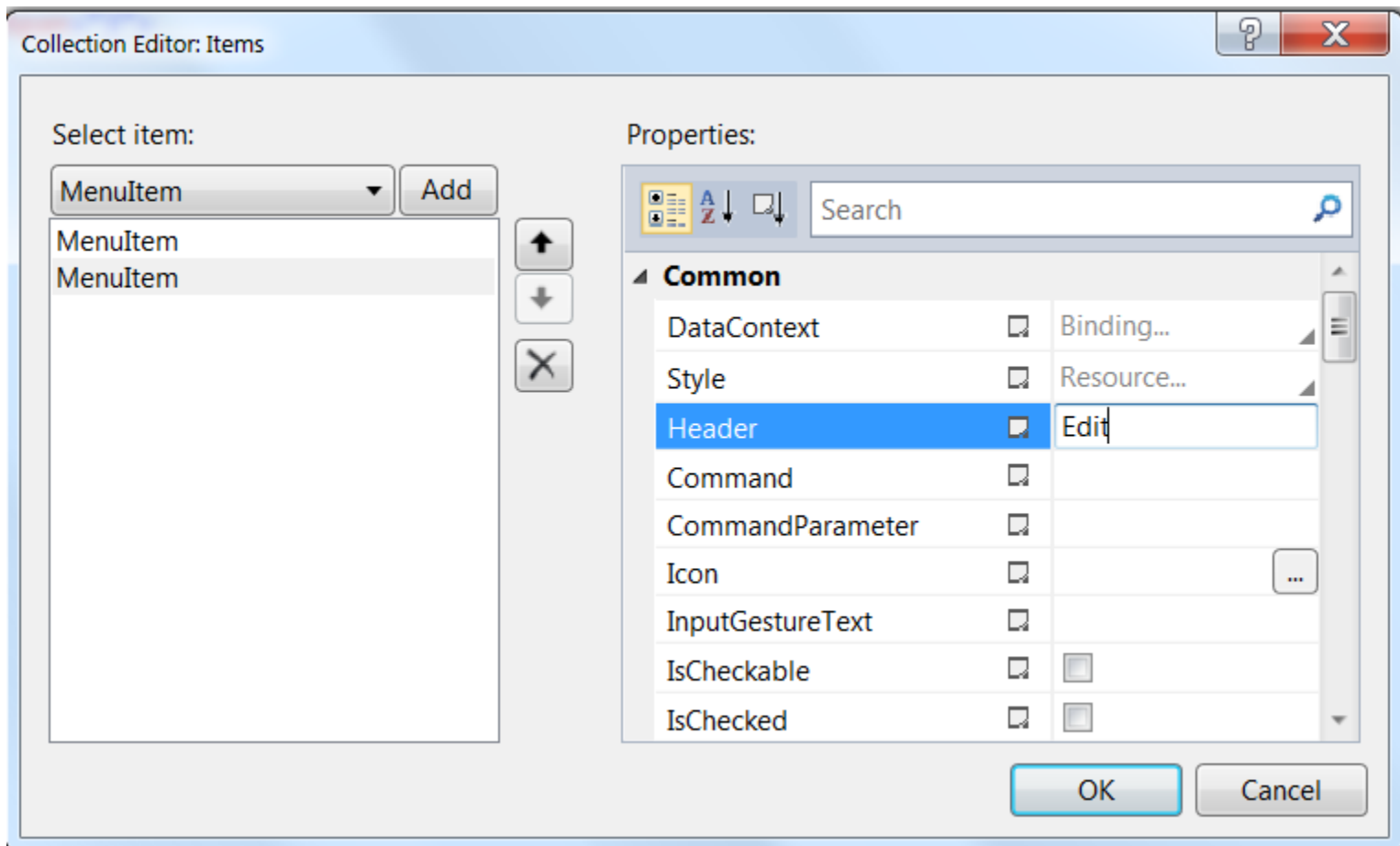
# Uygulama – Text Editor



# Uygulama – Text Editor



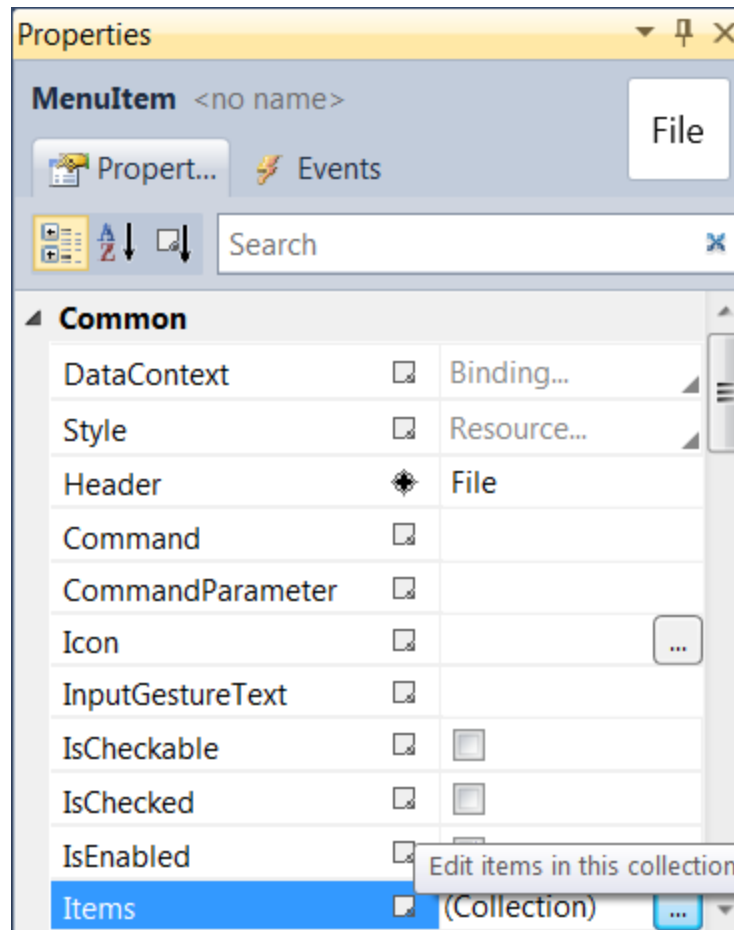
# Uygulama – Text Editor



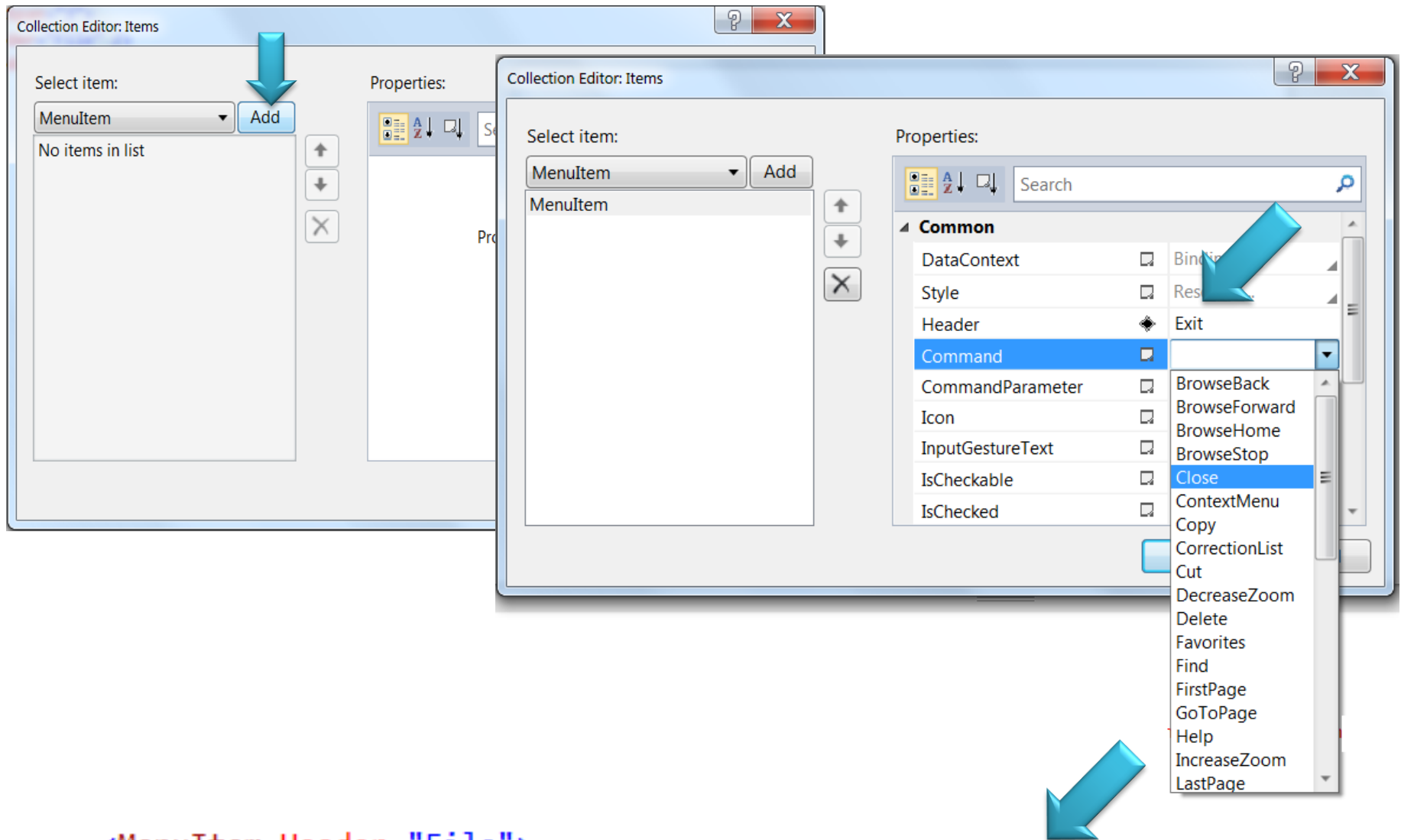


# Uygulama – Text Editor

- **File** → Properties → **Items**



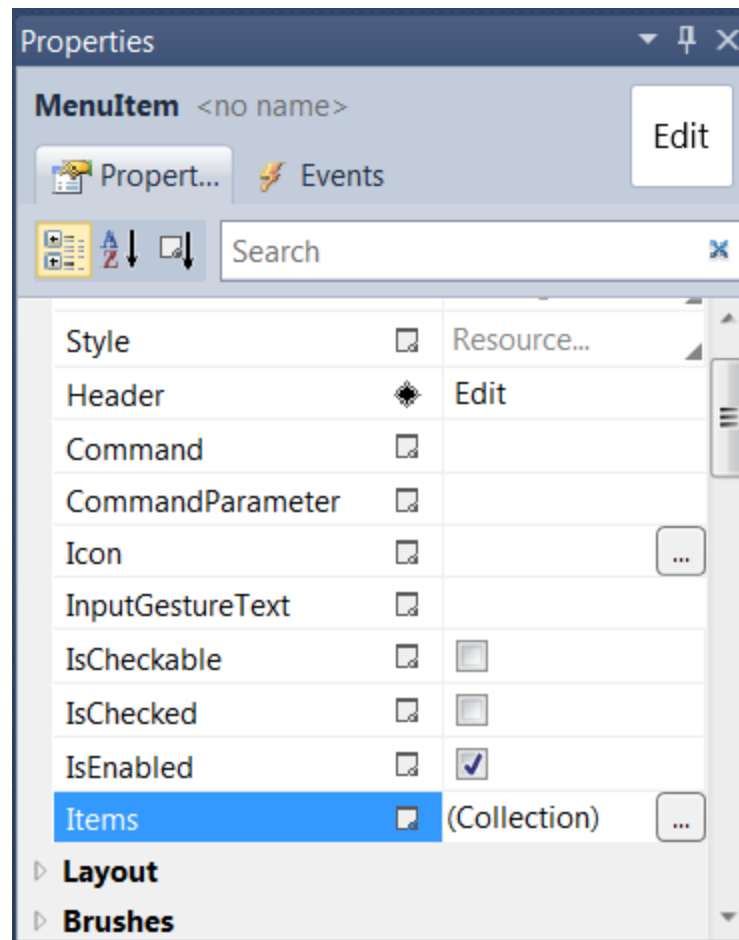
# Uygulama – Text Editor



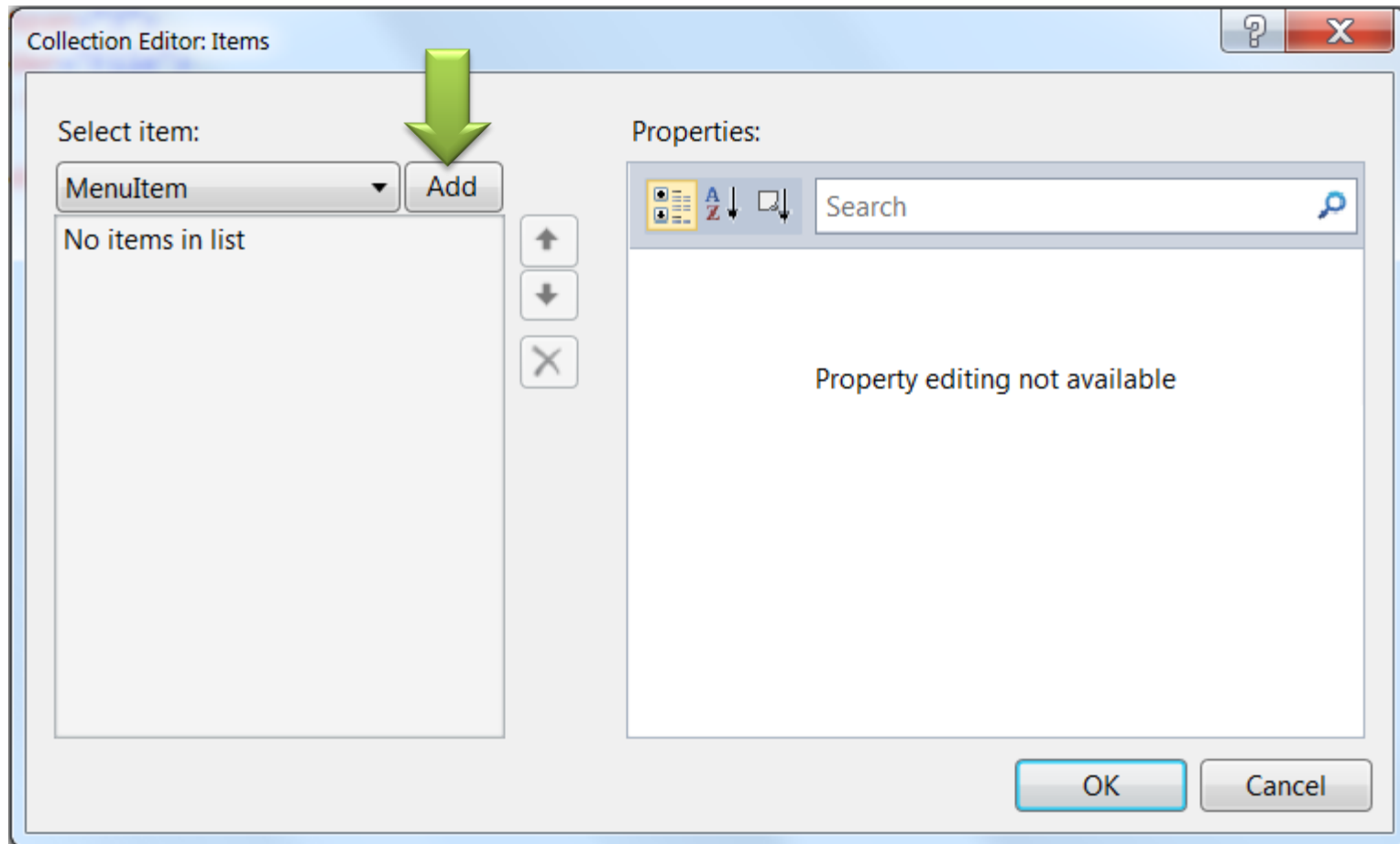
```
<MenuItem Header="File">  
    <MenuItem Command="ApplicationCommands.Close" Header="Exit" />  
</MenuItem>
```

# Uygulama – Text Editor

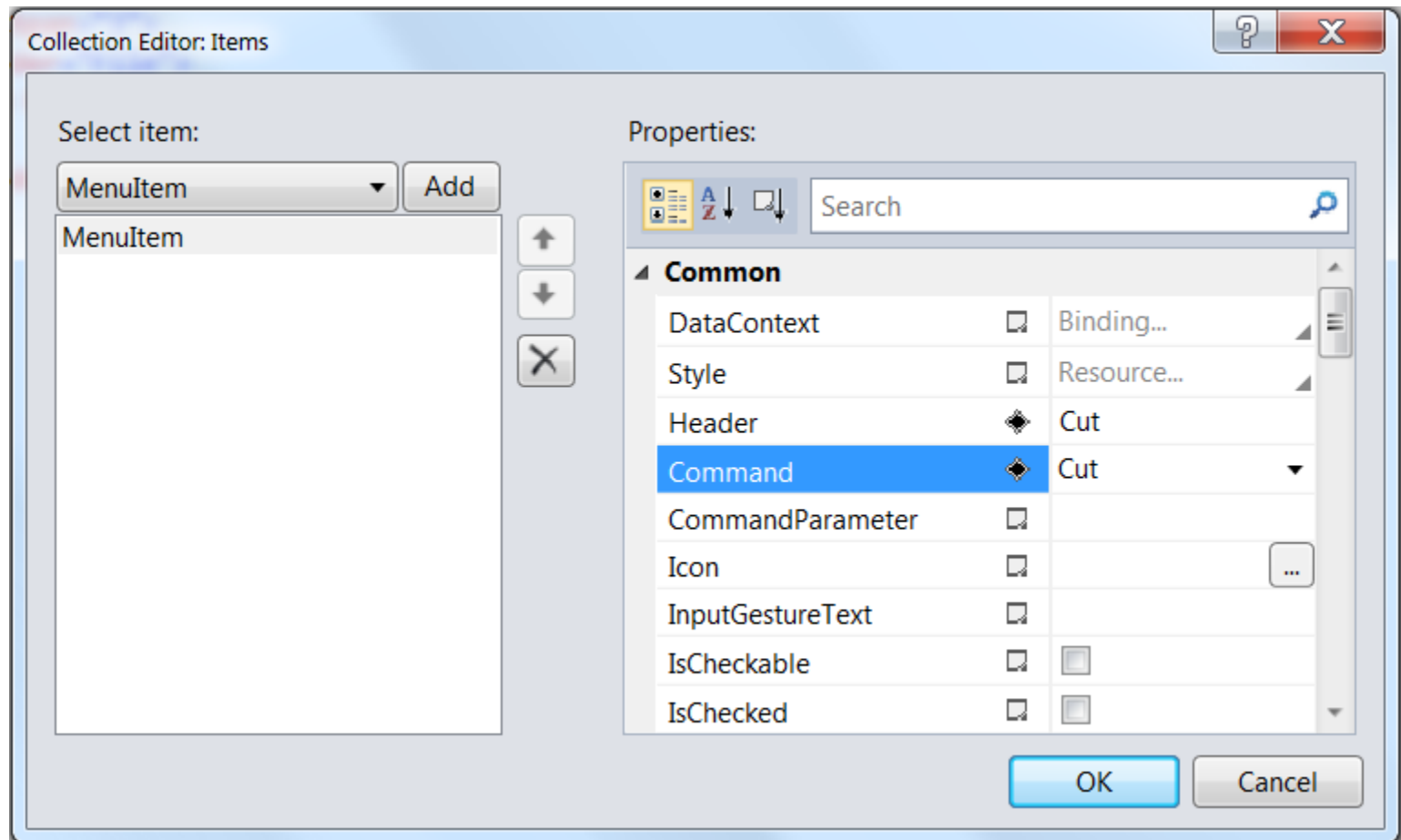
- **Edit** → **Properties** → **Items**



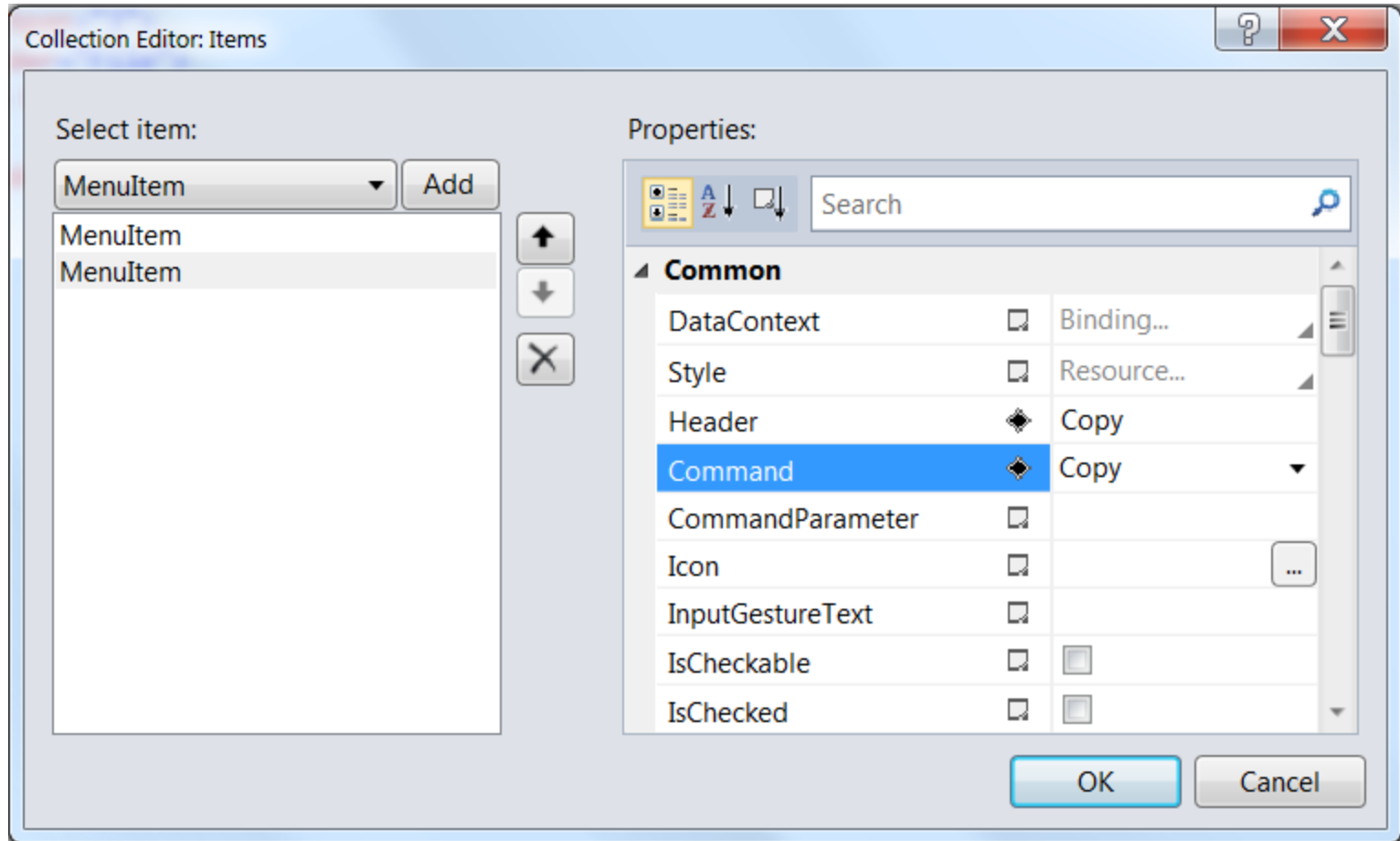
# Uygulama – Text Editor



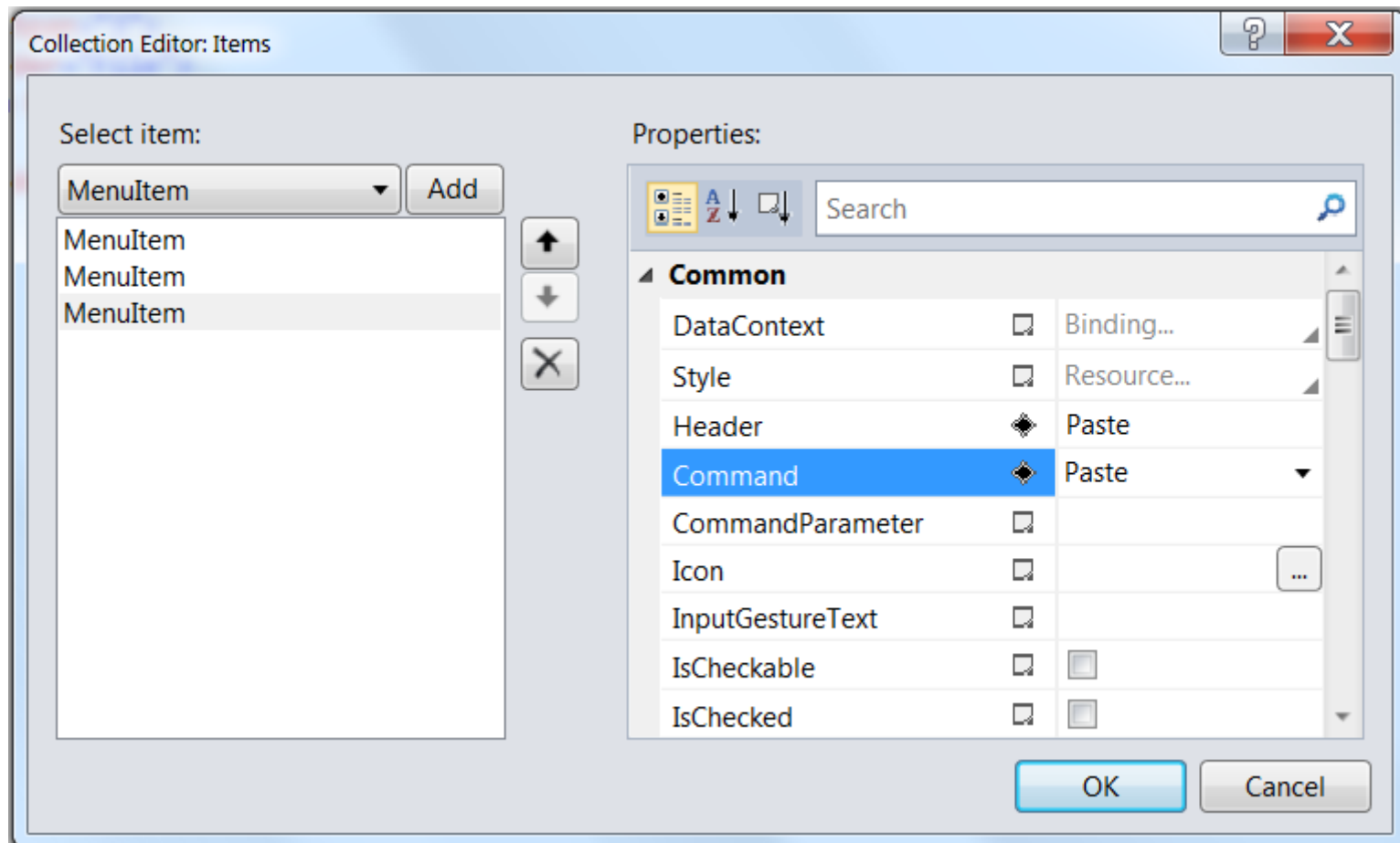
# Uygulama – Text Editor

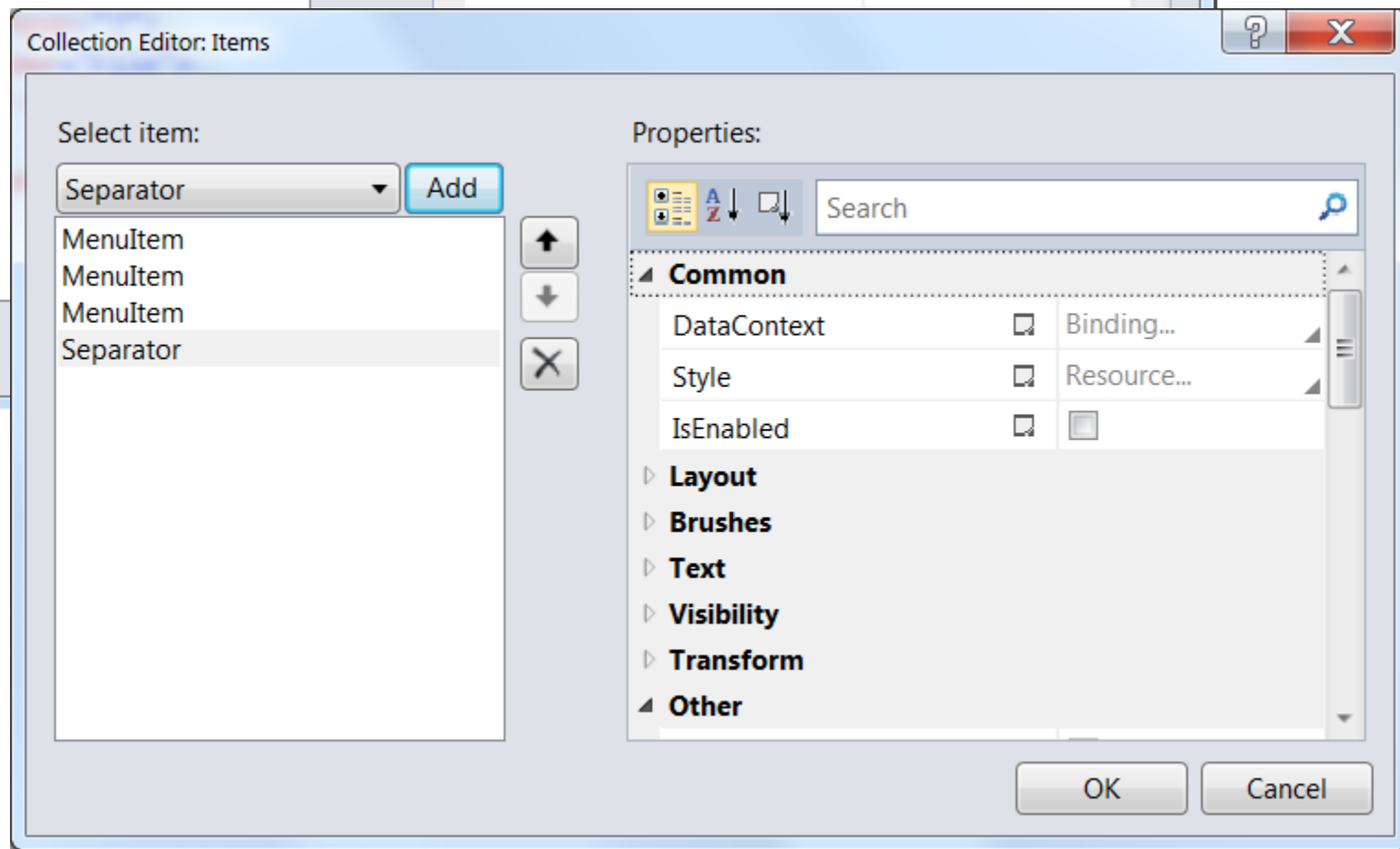
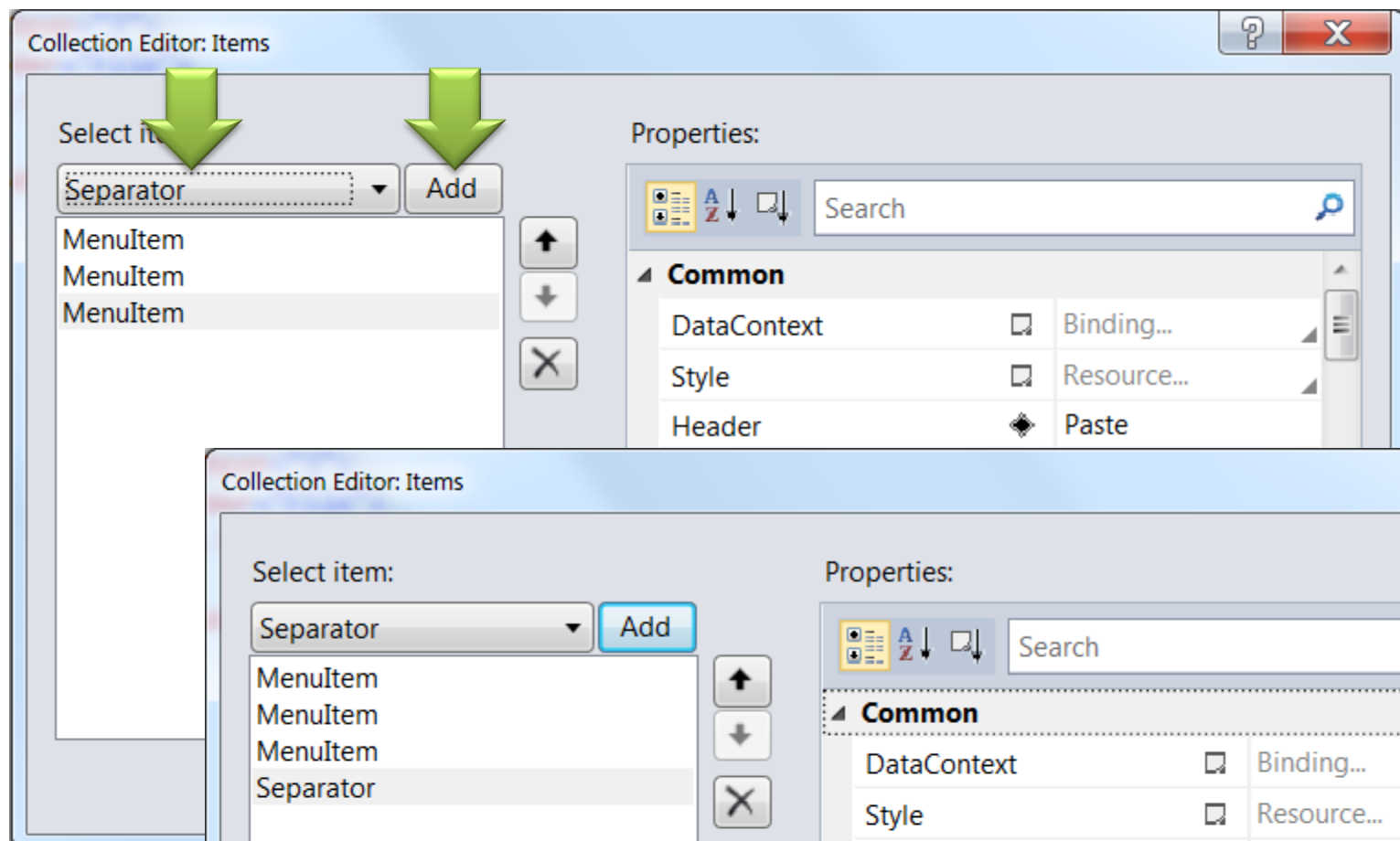


# Uygulama – Text Editor

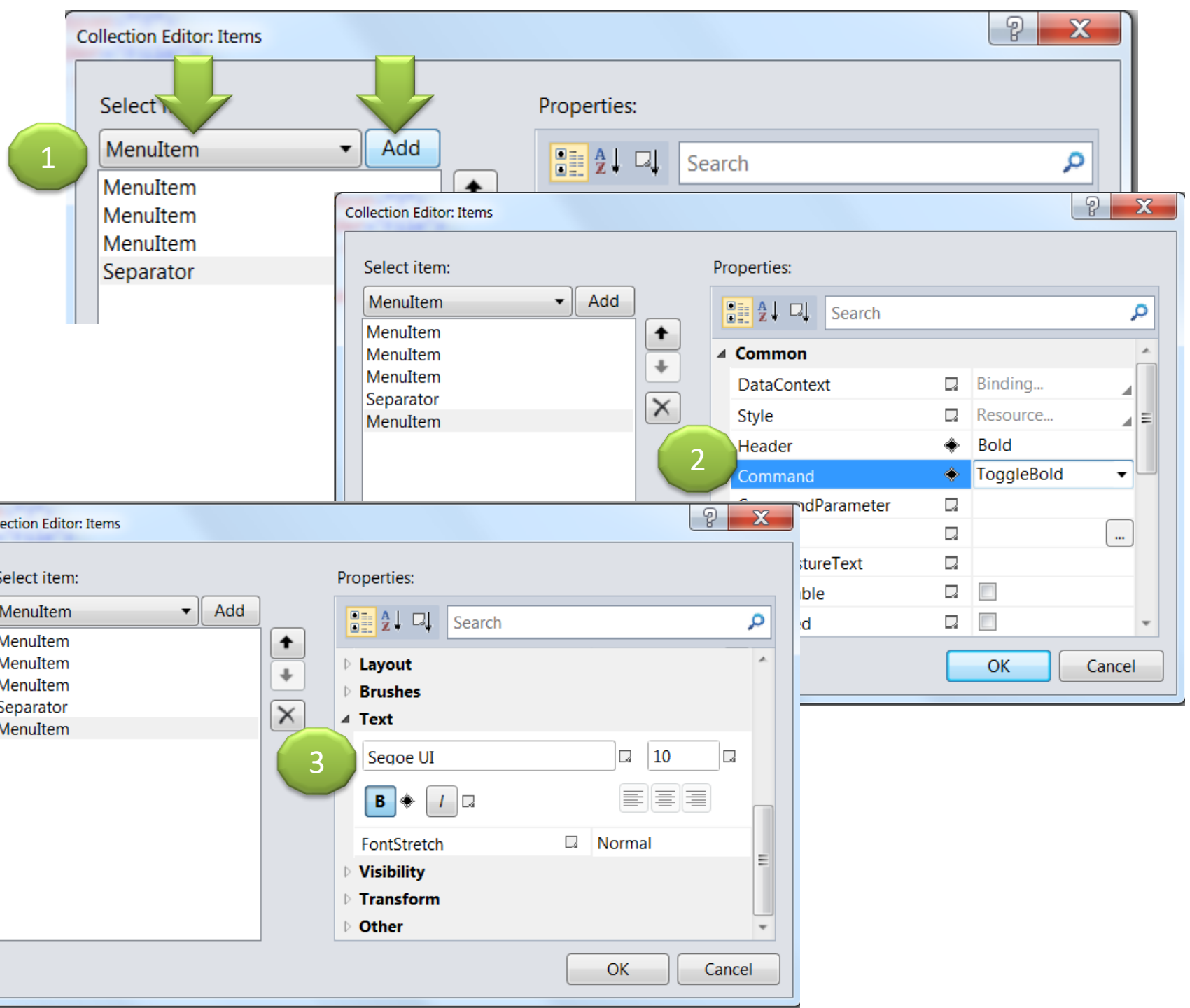


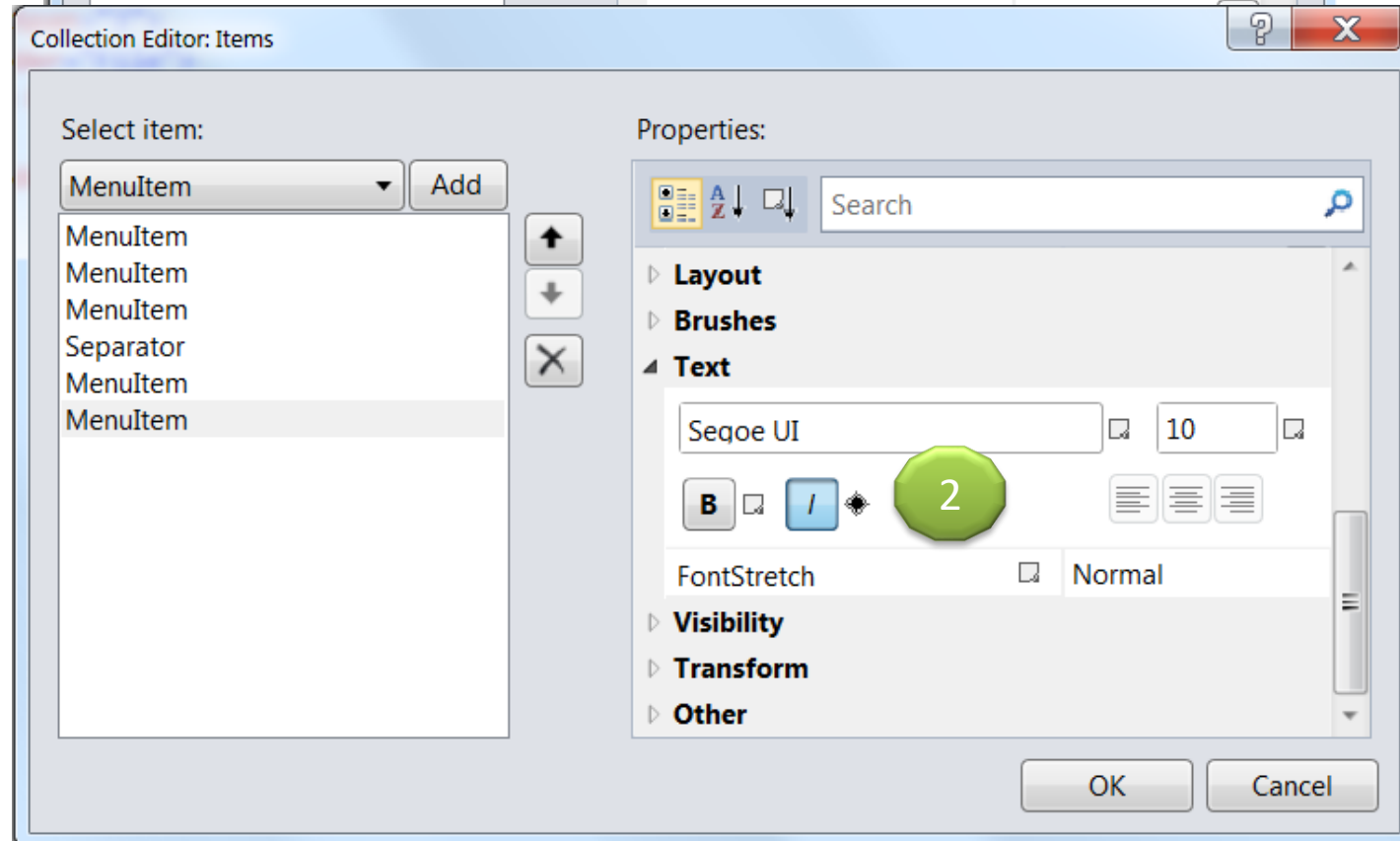
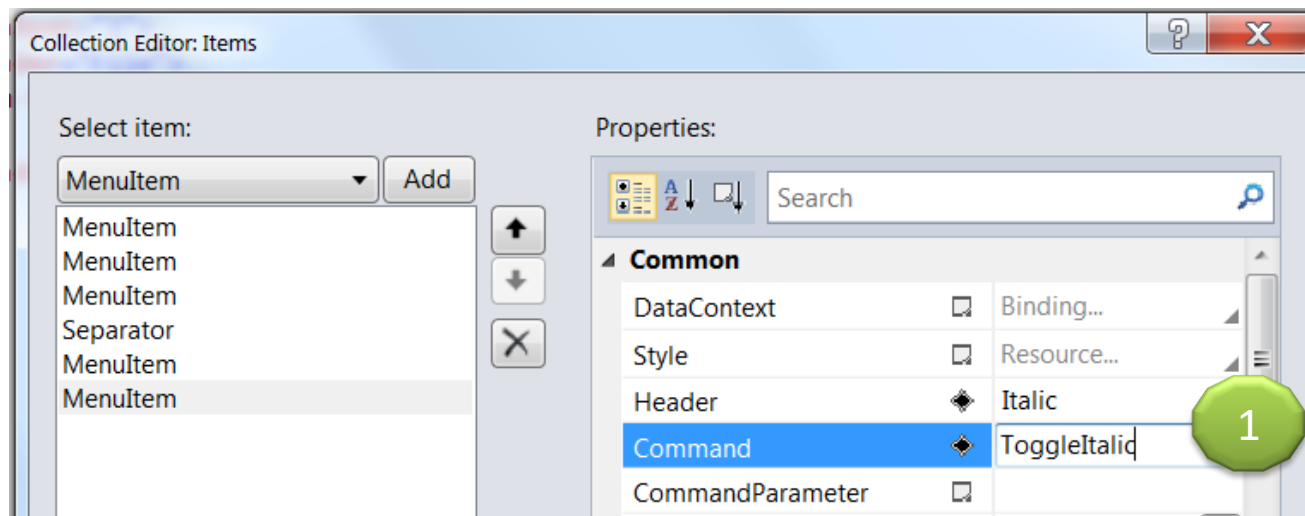
# Uygulama – Text Editor











```

<Menu Grid.Row="0">
  <MenuItem Header="File">
    <MenuItem Command="Close" Header="Exit" />
  </MenuItem>
  <MenuItem Header="Edit">
    <MenuItem Command="Cut" Header="Cut" />
    <MenuItem Command="Copy" Header="Copy" />
    <MenuItem Command="Paste" Header="Paste" />
    <Separator />
    <MenuItem Command="ToggleBold" FontWeight="Bold" Header="Bold" />
    <MenuItem Command="ToggleItalic" FontStyle="Italic" Header="Italic" />
  </MenuItem>
</Menu>

```

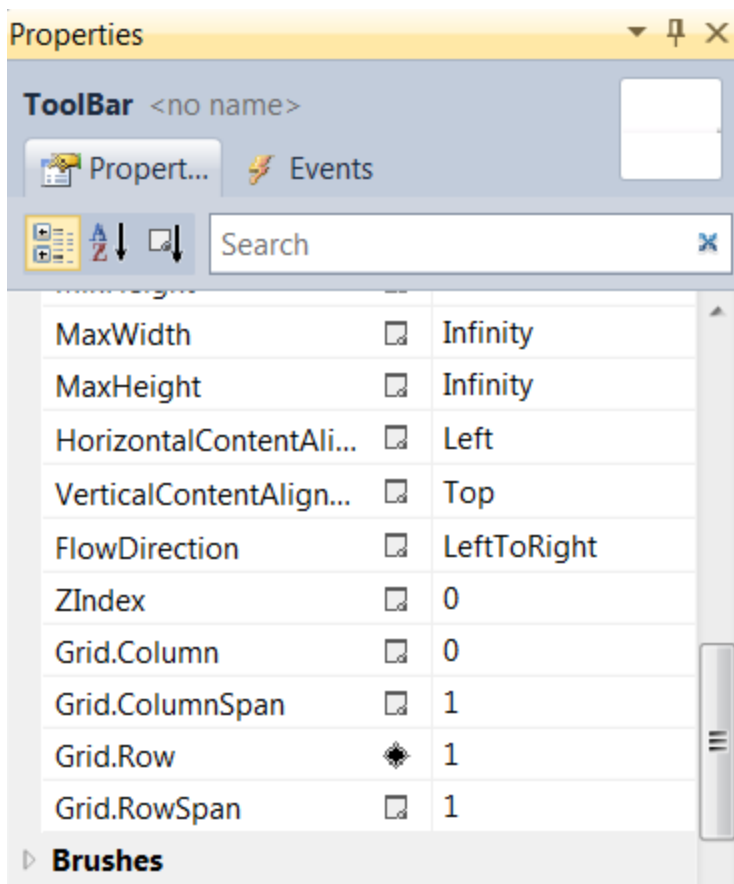


# Uygulama – Text Editor

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="211*" />
    <ColumnDefinition Width="35*" />
    <ColumnDefinition Width="32*" />
  </Grid.ColumnDefinitions>
  <Menu Grid.Row="0">
    <MenuItem Header="File">
      <MenuItem Command="Close" Header="Exit" />
    </MenuItem>
    <MenuItem Header="Edit">
      <MenuItem Command="Cut" Header="Cut" />
      <MenuItem Command="Copy" Header="Copy" />
      <MenuItem Command="Paste" Header="Paste" />
      <Separator />
      <MenuItem Command="ToggleBold" FontWeight="Bold" Header="Bold" />
      <MenuItem Command="ToggleItalic" FontStyle="Italic" Header="Italic" />
    </MenuItem>
  </Menu>
</Grid>
```

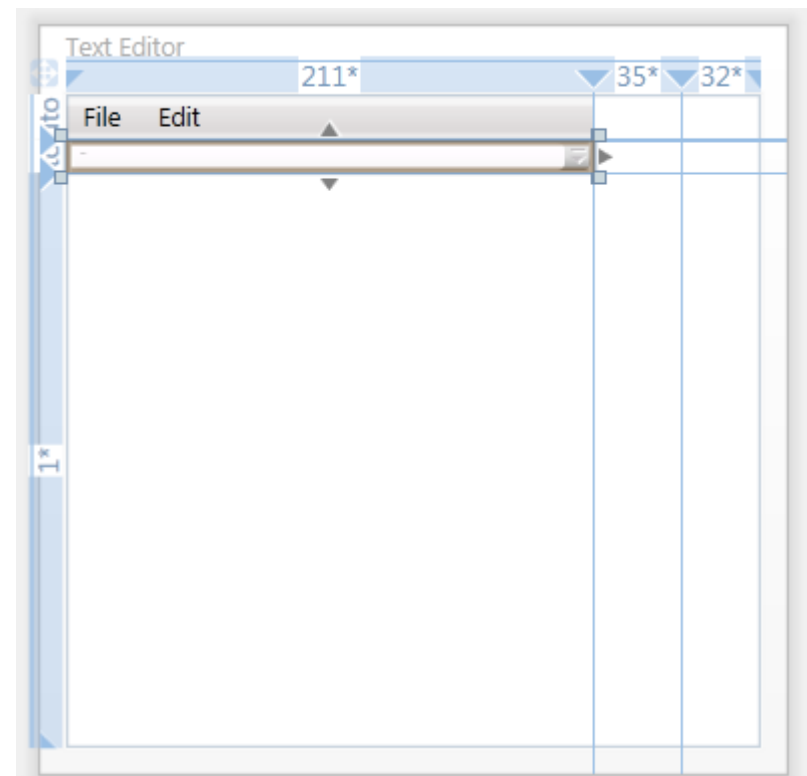
# Uygulama – Text Editor

- **ToolBar**



```
<ToolBar Grid.Row="1">
```

```
</ToolBar>
```

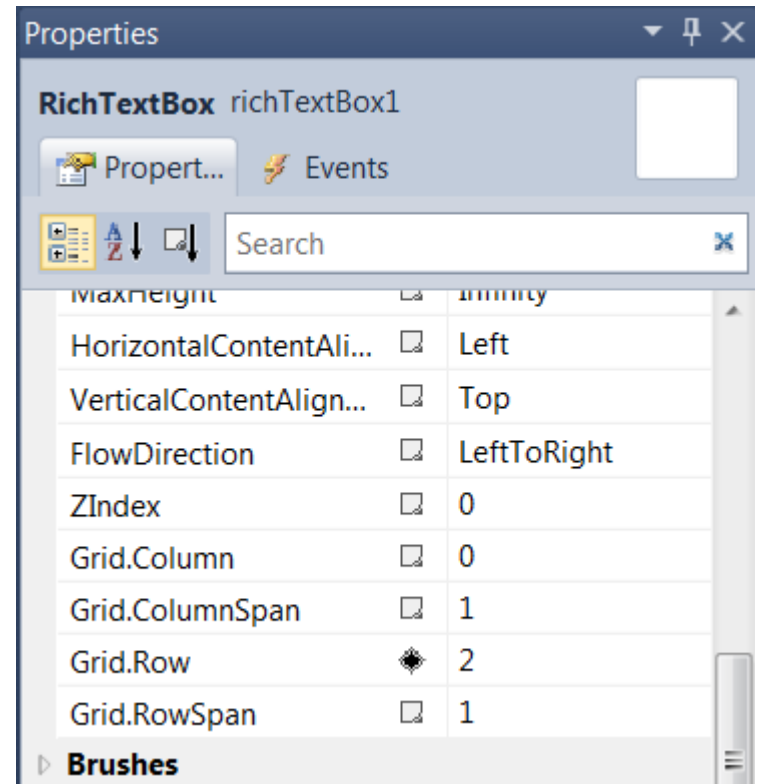
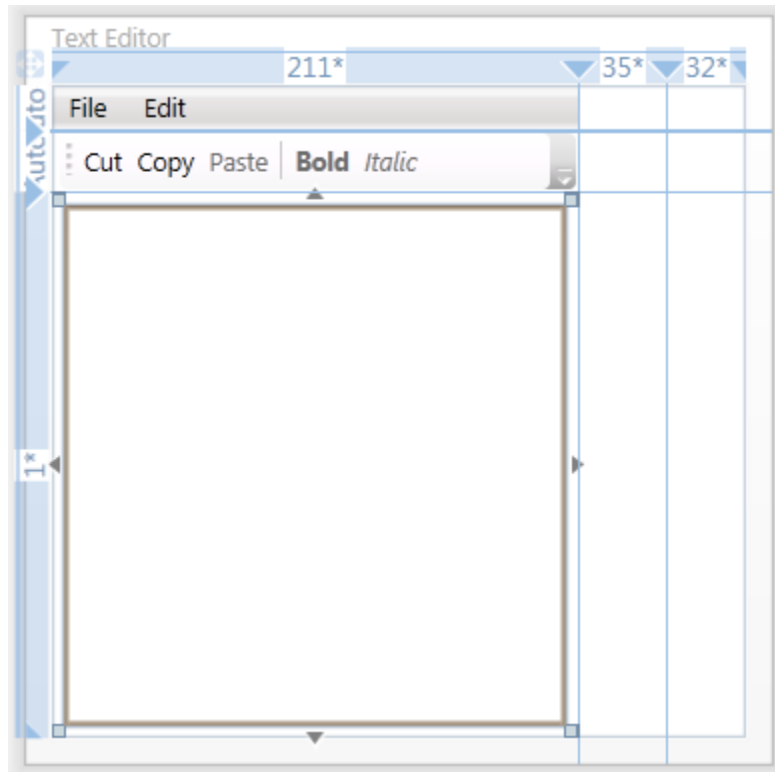


# Uygulama – Text Editor

```
<ToolBar Grid.Row="1">
  <Button Command="Cut">Cut</Button>
  <Button Command="Copy">Copy</Button>
  <Button Command="Paste">Paste</Button>
  <Separator /> <!-- separates groups of toolbar items -->
  <Button FontWeight="Bold" Command="ToggleBold">Bold</Button>
  <Button FontStyle="Italic" Command="ToggleItalic">Italic</Button>
</ToolBar>
```



- RichTextBox
  - Grid.Row = "2"
  - Margin = "5"



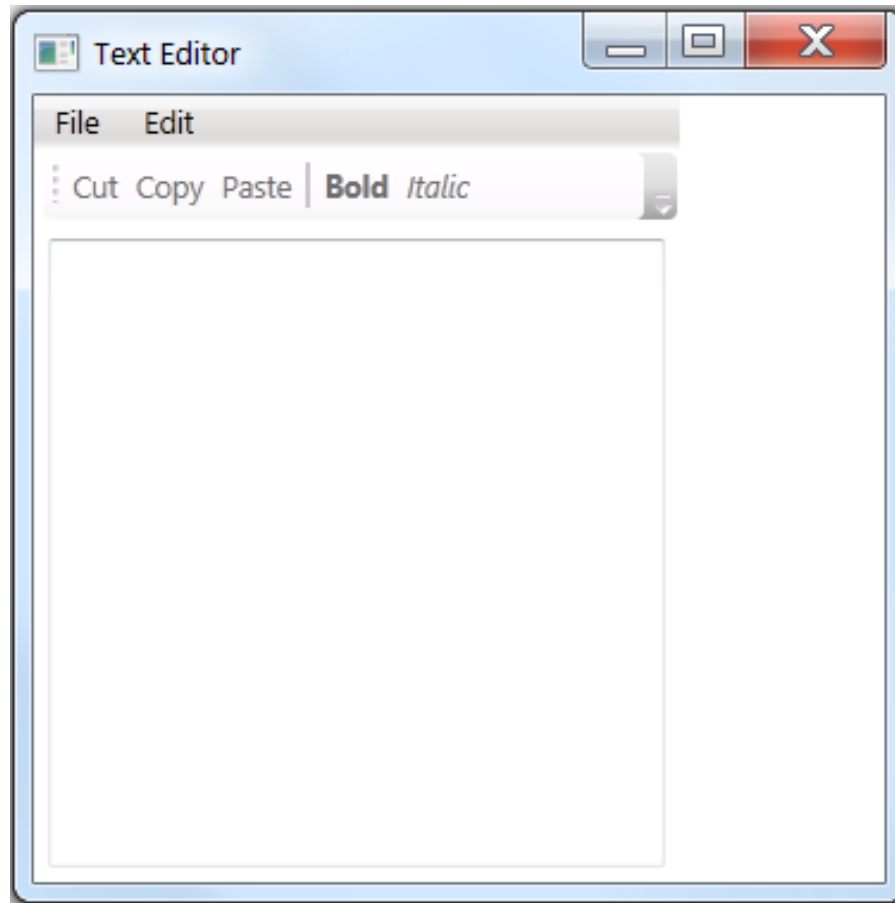
```
<RichTextBox Grid.Row="2" Margin="5" />
```

# Uygulama – Text Editor

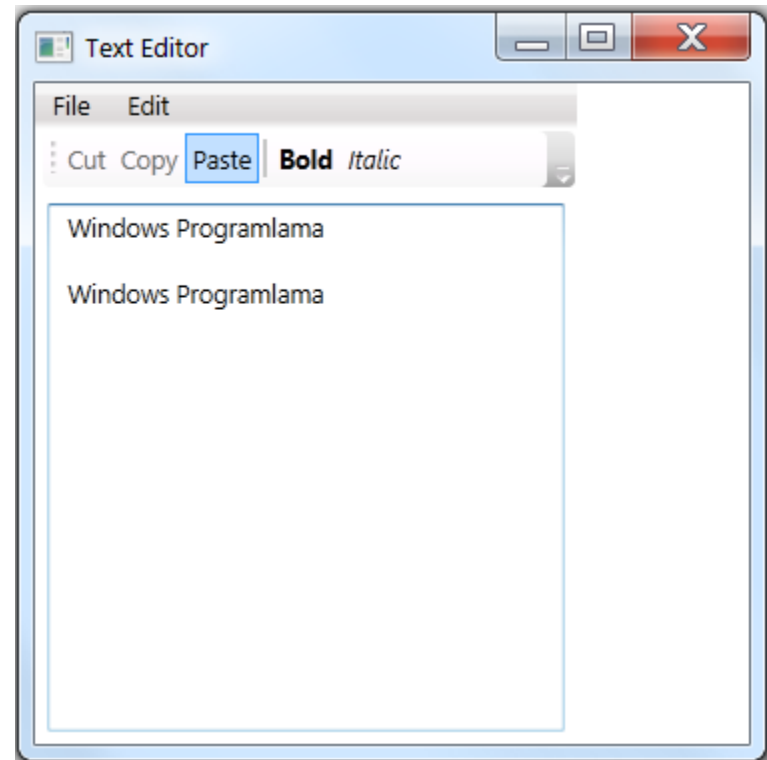
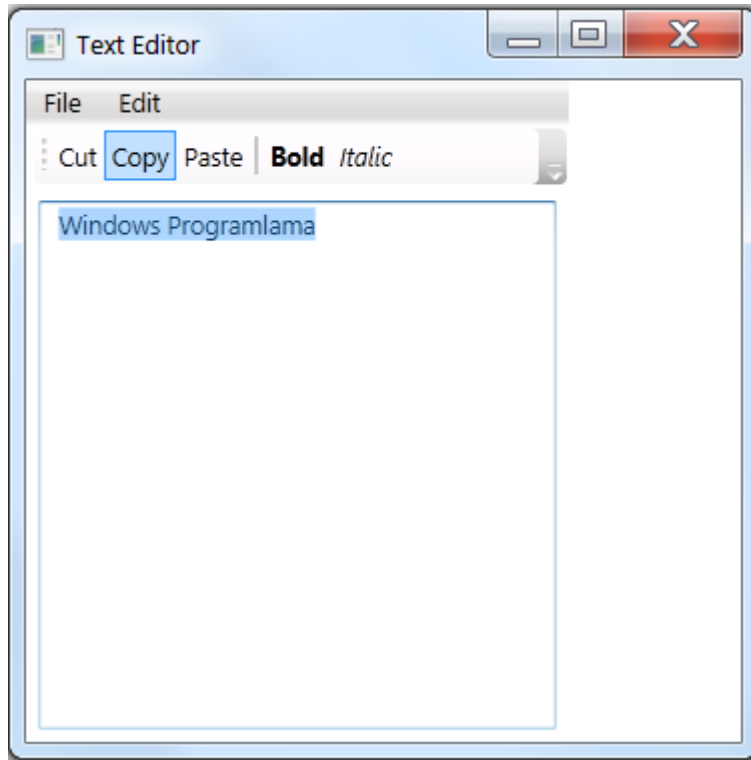
```
private void closeCommand_Executed(object sender, ExecutedRoutedEventArgs e)
{
    Application.Current.Shutdown(); // exit the application
}
```



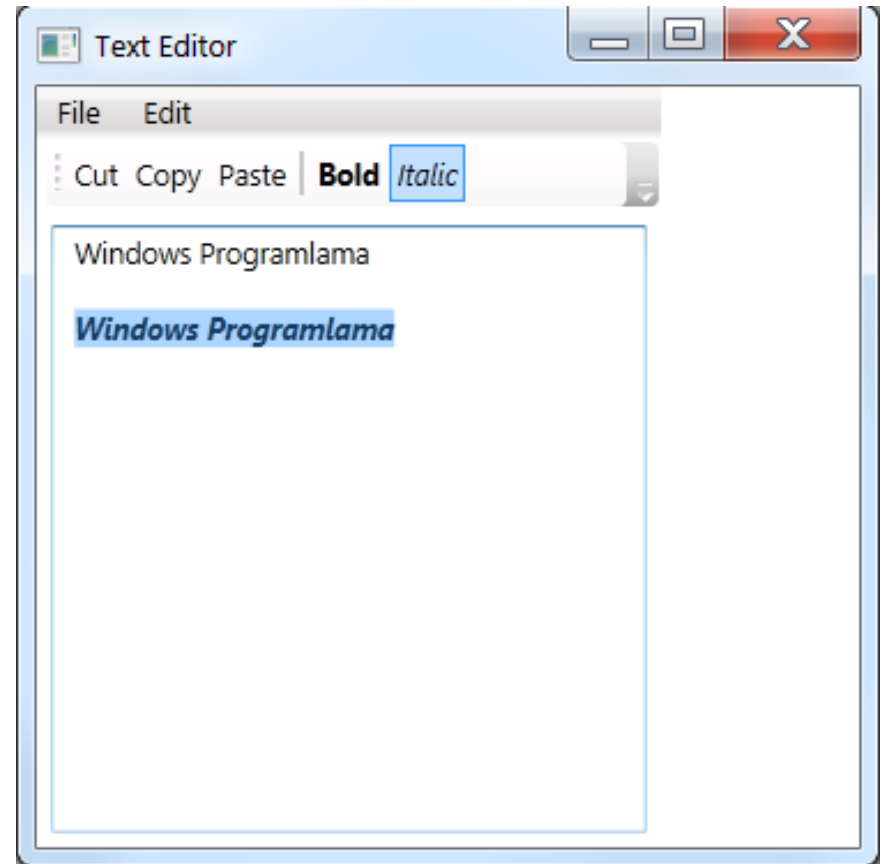
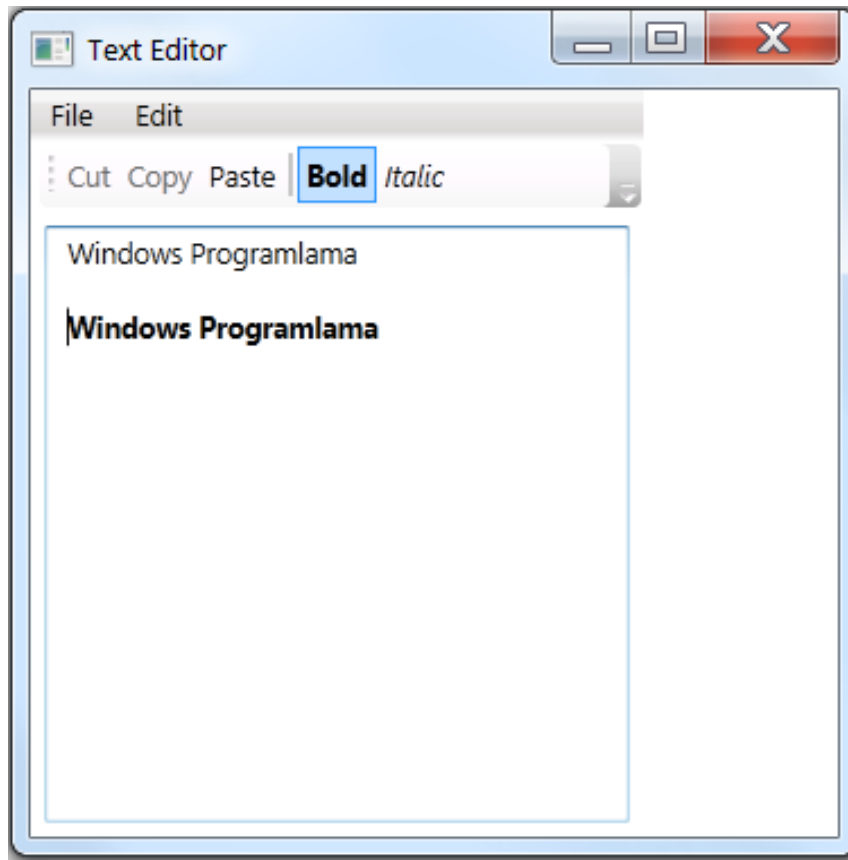
# Uygulama – Text Editor



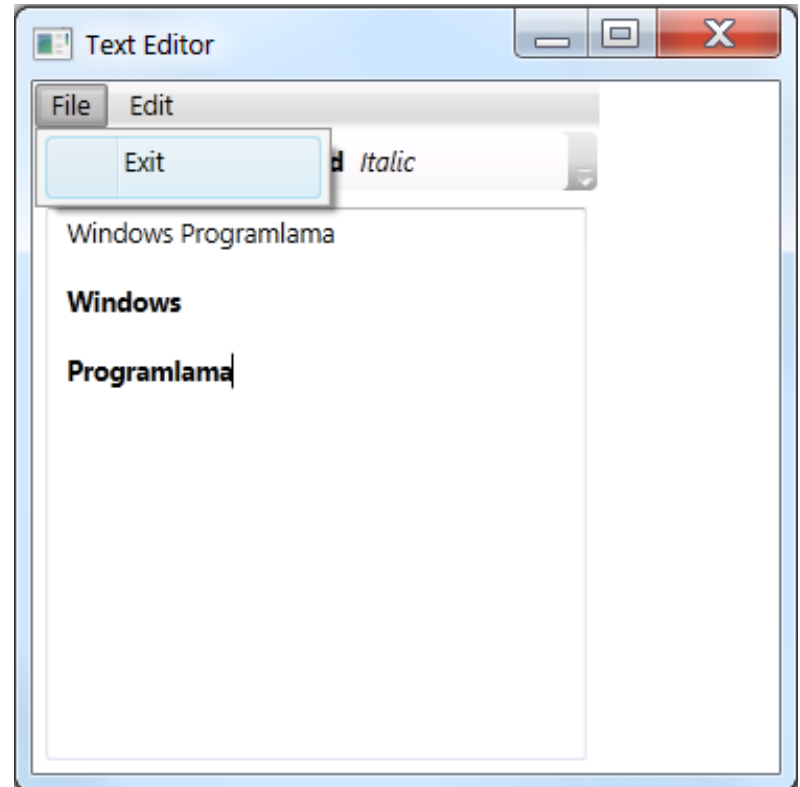
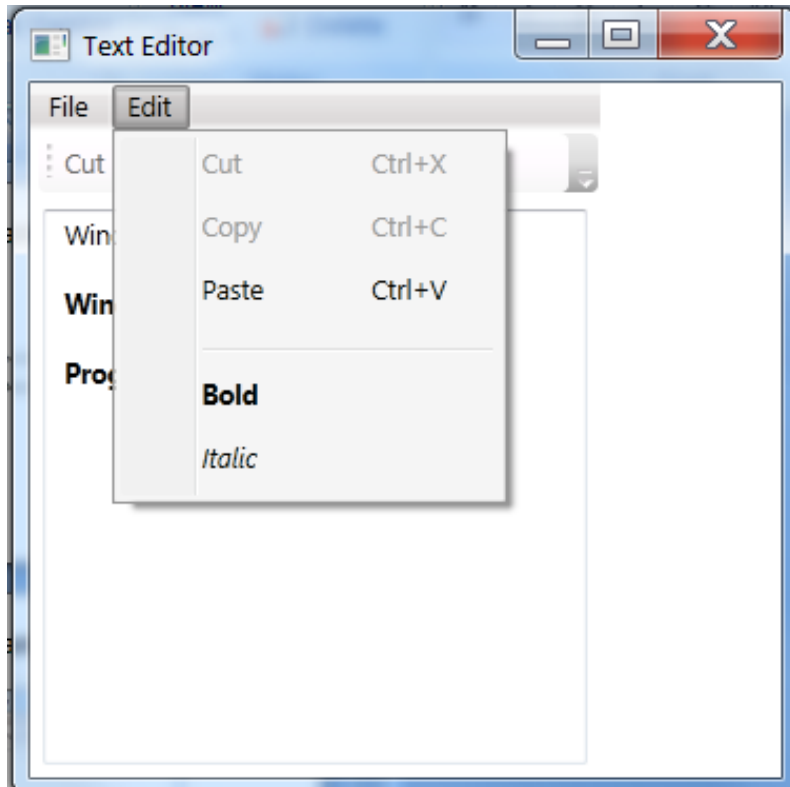
# Uygulama – Text Editor



# Uygulama – Text Editor



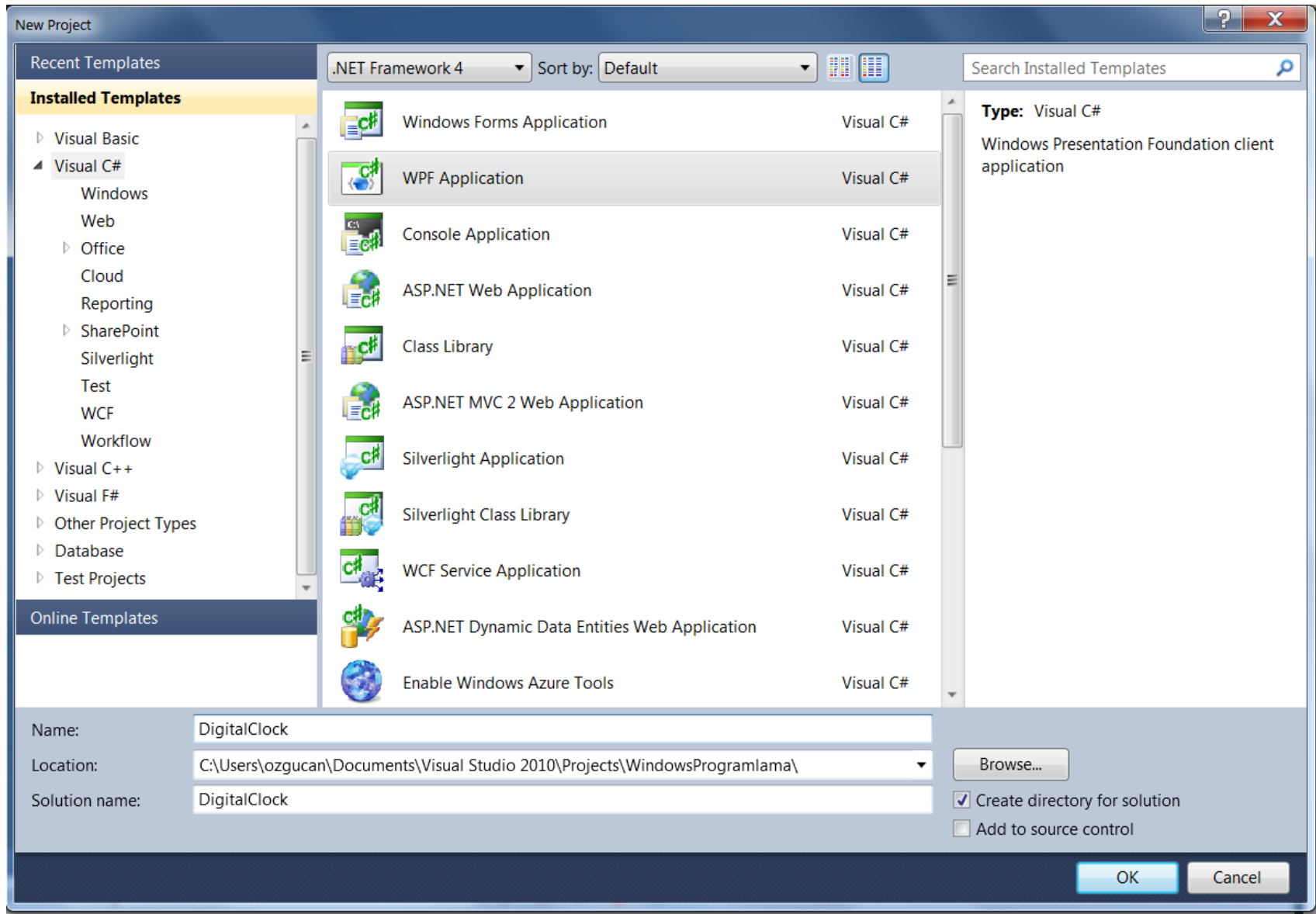
# Uygulama – Text Editor



Digital Saat Uygulaması

# **UYGULAMA PENCERELERİN UYARLANMASI**

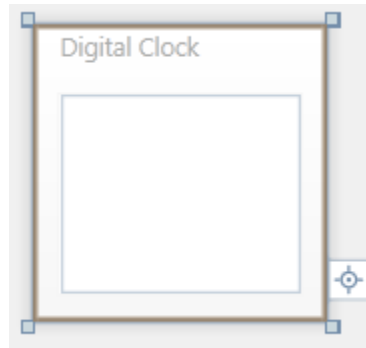
# Uygulama – Dijital Saat



# Uygulama – Dijital Saat

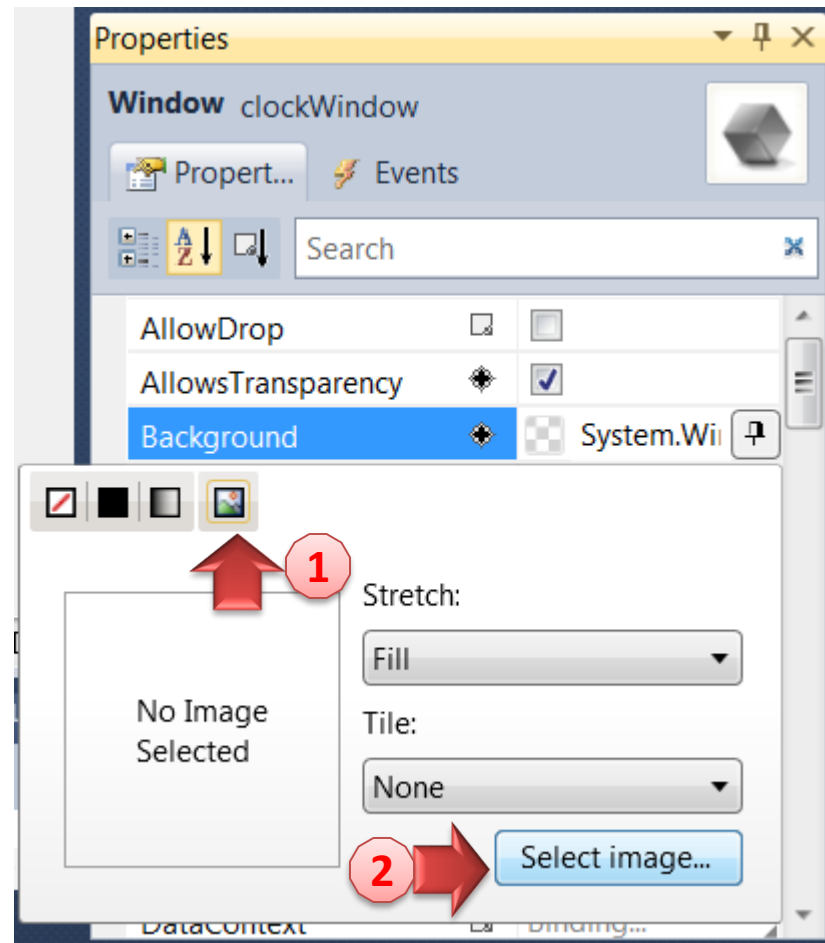
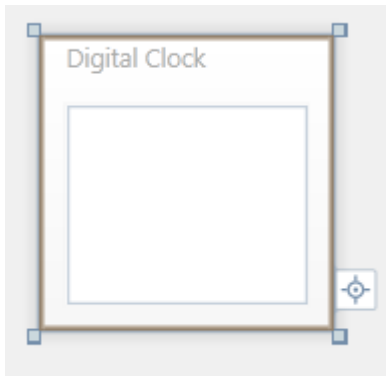
```
Title="Digital Clock" Name="clockWindow" Height="118" Width="118" WindowStyle="None" AllowsTransparency="True"  
MouseLeftButtonDown="clockWindow_MouseLeftButtonDown">
```

Event



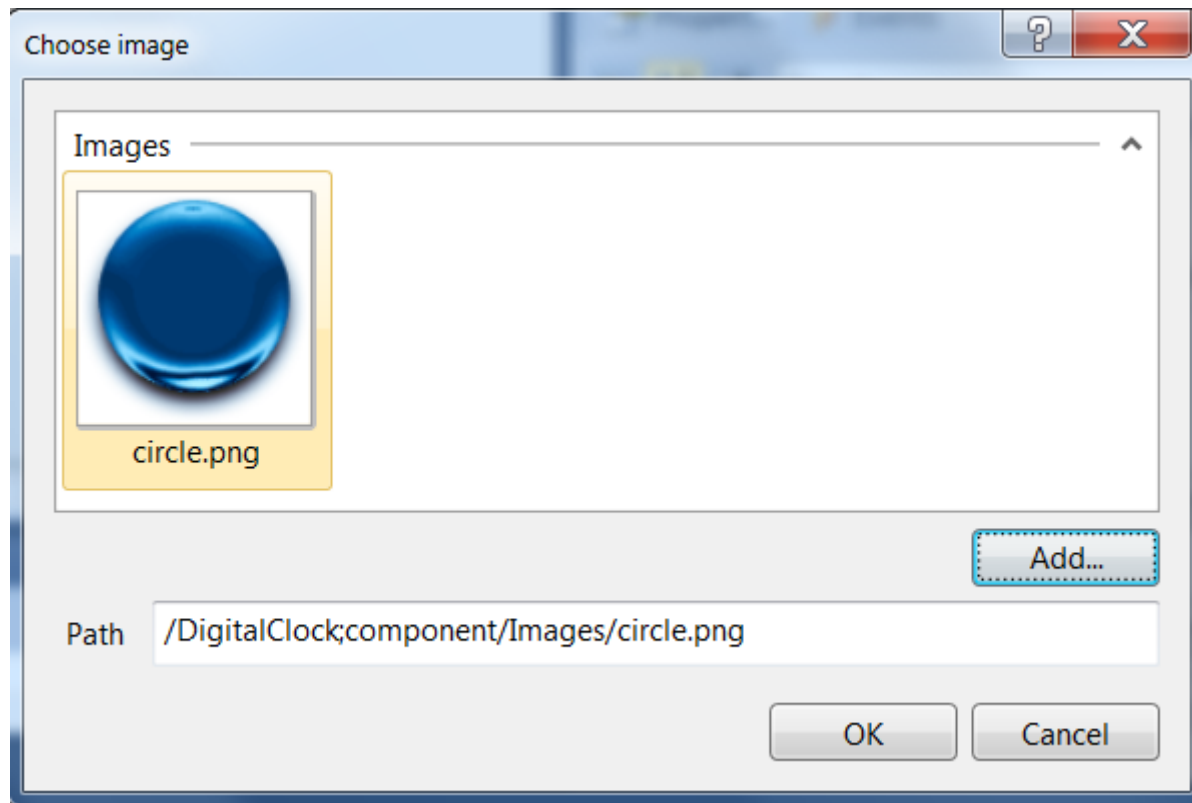
# Uygulama – Dijital Saat

- Window → Properties → **Background**

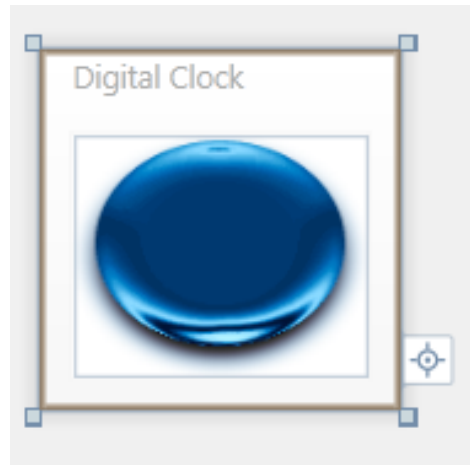




# Uygulama – Dijital Saat



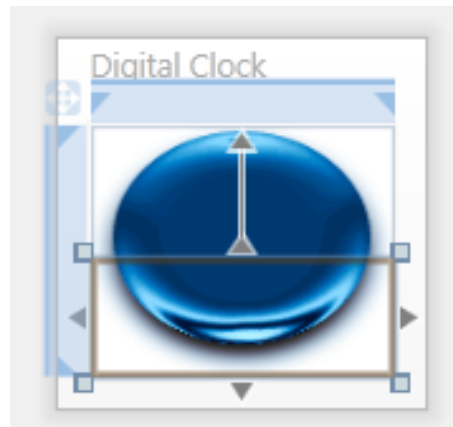
# Uygulama – Dijital Saat



```
<Window.Background>  
    <ImageBrush ImageSource="/DigitalClock;component/Images/circle.png" />  
</Window.Background>
```

# Uygulama – Dijital Saat

- **TextBox**



```
<Grid>  
    <TextBox Name="timeTextBox" Margin="0,42,0,0"  
        Background="Transparent" TextAlignment="Center"  
        FontWeight="Bold" Foreground="White" FontSize="16"  
        BorderThickness="0" Cursor="Arrow" Focusable="False" />  
</Grid>
```

# Uygulama – Dijital Saat

```
<Window x:Class="DigitalClock.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Digital Clock" Name="clockWindow" Height="118" Width="118" WindowStyle="None" AllowsTransparency="True"
        MouseLeftButtonDown="clockWindow_MouseLeftButtonDown">
    <Grid>
        <TextBox Name="timeTextBox" Margin="0,42,0,0"
            Background="Transparent" TextAlignment="Center"
            FontWeight="Bold" Foreground="White" FontSize="16"
            BorderThickness="0" Cursor="Arrow" Focusable="False" />
    </Grid>
    <Window.Background>
        <ImageBrush ImageSource="/DigitalClock;component/Images/circle.png" />
    </Window.Background>
</Window>
```

# Uygulama – Dijital Saat

```
public partial class MainWindow : Window
{
    // create a timer to control clock
    private System.Windows.Threading.DispatcherTimer timer = new System.Windows.Threading.DispatcherTimer();

    public MainWindow()
    {
        InitializeComponent();

        timer.Interval = TimeSpan.FromSeconds(1); // tick every second
        timer.IsEnabled = true; // enable timer
        timer.Tick += timer_Tick;
    }
}
```

# Uygulama – Dijital Saat

```
private void timer_Tick(object sender, EventArgs e)
{
    DateTime currentTime = DateTime.Now; // get the current time

    // display the time as hh:mm:ss
    timeTextBox.Text = currentTime.ToLongTimeString();
}

// drag Window when the left mouse button is held down
private void clockWindow_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    this.DragMove(); // moves the window
}
```

# Uygulama – Dijital Saat

