



Stiller ve Veri Baęlama İşlemlerine Giriş

Öğr.Gör.Erkan HÜRNALI

Stiller (Styles) Nedir?

WPF uygulamalarında kullanılacak olan tasarım, font, renk, resim ve sayamadığımız tüm görsel özellikleri belirleyebileceğimiz yapılardır. Bu nedenle stiller web tasarımında çok tercih edilen CSS (*Cascading Style Sheets*) yapılarına çok benzemektedir. Birçok görsel özelliğin aynı yapı içerisinde kullanılması geliştiricinin zaman kaybını önleyecektir.

Stiller Nasıl Tanımlanır?

Genel olarak stil aşağıdaki şekilde tanımlanır. Burada x:Key property'si stili kullanırken çağıracağımız ismi belirtiyor. **TargetType** property'si ise stilin hangi kontrol veya elemente uygulanacağını temsil ediyor.

```
<Style x:Key="isim" TargetType="Label">  
    ...  
</Style>
```

Stiller Nasıl Tanımlanır?

Style elementinin tagları arasına Setter elementi ile stil uygulanacak nesnenin özelliğini ve o özelliğin değerini belirtiriz.

```
<Style x:Key="stil" TargetType="Label" >  
    <Setter Property="Background" Value="Green"/>  
</Style>
```

Stiller Nereye Tanımlanır?

Stiller CSS'lerde olduğu gibi istersek kontrol bazında, istersek Pencere (Window) bazında ve istersek de tabii ki uygulama bazında tanımlanabilmektedirler.

```
<Application.Resources>  
</Application.Resources>
```

```
<Window.Resources>  
</Window.Resources>
```

```
<Button.Resources>  
</Button.Resources>
```

=> Stiller de birer kaynak (resource)'tır!

Stiller Nasıl Tanımlanır - Örnekler

```
<Style TargetType="TextBlock" x:Key="onyazi">
    <Setter Property="Background" Value="Blue"/>
    <Setter Property="DockPanel.Dock" Value="Top"/>
    <Setter Property="FontSize" Value="18"/>
    <Setter Property="Foreground" Value="#4E87D4"/>
    <Setter Property="FontFamily" Value="Trebuchet MS"/>
    <Setter Property="Margin" Value="0,40,10,10"/>
</Style>
```

Stiller Nasıl Tanımlanır - Örnekler

```
<Style TargetType="Label" x:Key="icerik">
    <Setter Property="DockPanel.Dock" Value="Right"/>
    <Setter Property="FontSize" Value="8"/>
    <Setter Property="Foreground" Value="Red"/>
    <Setter Property="FontFamily" Value="Arial"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="Margin" Value="0,3,10,0"/>
</Style>
```

Stilleri Uygulamak

Kaynak olarak tanımladığımız stilleri kontrollere uygulamak için:

```
<TextBlock Style="{StaticResource onyazi}">Deneme Yazısı</TextBlock>  
<Label Style="{StaticResource icerik}">İçerik Yazısı</Label>
```


Stilleri Uygulamak - Örnek

```
<Button Style="{DynamicResource buttonStyle}" Panel.ZIndex="1" Margin ="169,84,34,0" Name="button1"
Height="23" VerticalAlignment="Top">
  <Button.Resources>
    <Style x:Key="buttonStyle">
      <Setter Property="Button.Background" Value="Gray"/>
      <Setter Property="Button.Foreground" Value="White"/>
      <Setter Property="Button.FontFamily" Value="Comic Sans MS"/>
    </Style>
  </Button.Resources>
  Button
</Button>
```

Stilleri Uygulamak



Not Bir WPF penceresini derlediğinizde, Visual Studio pencerenin içerdiği kaynakları pencere ile ilişkili bir koleksiyona ekler. Açıkça söylemek gerekirse, *Key* özelliği, stilin adı değil bu koleksiyondaki kaynağın bir tanımlayıcısını gösterir. C# kodunuzdaki kaynakları yönetmek isterseniz *Name* özelliğini belirleyebilseniz de, denetimler kaynağın *Key* değerini belirterek kaynaklara başvurur. Forma eklediğiniz denetimler ve diğer öğeler kendi *Name* özelliği ayarına sahip olmalıdır çünkü kaynaklarda olduğu gibi, kodda bu öğelere nasıl başvurulacağını gösterir.

Stilleri Uygulamak

Stiller kapsama sahiptir. Form üzerindeki ikinci düğmeden *buttonStyle* stiline başvurmaya çalışırsanız, hiçbir şey olmaz. Bunun yerine, bu stilin kopyasını oluşturarak ikinci düğmenin *Resources* öğesine ekleyebilir ve daha sonra bu kopyaya başvurabilirsiniz:

Stilleri Uygulamak

```
<Grid>
  <Button Style="{DynamicResource buttonStyle}" Panel.ZIndex="1"
    Margin="169,84,34,0" Name="button1" Height="23"
    VerticalAlignment="Top">
    <Button.Resources>
      <Style x:Key="buttonStyle">
        <Setter Property="Button.Background" Value="Gray"/>
        <Setter Property="Button.Foreground" Value="White"/>
        <Setter Property="Button.FontFamily" Value="Comic Sans MS"/>
      </Style>
    </Button.Resources>
    Button
  </Button>
  <Button Style="{DynamicResource buttonStyle}" Panel.ZIndex="1" Height="23"
    Margin="0,0,0,0" Name="button2" Width="76">
    <Button.Resources>
      <Style x:Key="buttonStyle">
        <Setter Property="Button.Background" Value="Gray"/>
        <Setter Property="Button.Foreground" Value="White"/>
        <Setter Property="Button.FontFamily" Value="Comic Sans MS"/>
      </Style>
    </Button.Resources>
    Button
  </Button>
  ...
</Grid>
```

Stilleri Uygulamak

Bununla birlikte, bu yaklaşım oldukça fazla tekrarlanabilir ve düğmelerin stilini değiştirmeniz gerekirse bakımı bir kabusla dönüşebilir. Daha iyi bir strateji, pencere için kaynak olarak bir stil tanımlamaktır. Daha sonra, o penceredeki tüm denetimlerden bu stile başvurabilirsiniz.

XAML bölmesinde, kılavuz üzerine bir `<Window.Resources>` ögesi ekleyin, `buttonStyle` stilinin tanımını bu yeni ögeye geçirin ve daha sonra her iki düğmeden `<Button.Resources>` ögesini silin. Her iki düğmeden yeni stile başvurun ve kodu daha okunulur yapmak için `button2` denetiminin tanımını birden fazla satır üzerine dağıtın. Formun tüm *XAML* tanımı için güncellenmiş kod, kaynak tanımı ve kaynağa başvuru ile birlikte aşağıdaki gibidir:

Stilleri Uygulamak

```
<Window x:Class="BellRingers.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Middleshire Bell Ringers Association - Members"
  Height="300" Width="300">
  <Window.Resources>
    <Style x:Key="buttonStyle">
      <Setter Property="Button.Background" Value="Gray"/>
      <Setter Property="Button.Foreground" Value="White"/>
      <Setter Property="Button.FontFamily" Value="Comic Sans MS"/>
    </Style>
  </Window.Resources>
  <Grid>
    <Button Style="{StaticResource buttonStyle}" Panel.ZIndex="1"
      Margin="169,84,34,0" Name="button1" Height="23" VerticalAlignment="Top">
      Button
    </Button>
```

Stilleri Uygulamak

```
<Button Style="{StaticResource buttonStyle}" Panel.ZIndex="1" Height="23"
    Margin="0,0,0,0" Name="button2" Width="76">
    Button
</Button>
<Image Panel.ZIndex="0" Margin="0,0,0,0" Name ="image1">
    <Image.Source>
        <BitmapImage UriSource="Bell.gif" />
    </Image.Source>
</Image>
</Grid>
</Window>
```

Stilleri Uygulamak



Not Yukarda girdiğiniz kod, *DynamicResource* anahtar sözcüğü yerine *StaticResource* kullanarak düğme stiline başvurur. Statik kaynakların kapsama kuralları, C#'daki ile benzerdir, başvurabilmeniz için öncelikle bir kaynağı tanımlamanızı gerektirir. Bu alıştırmamanın 1. adımında, kaynağın tanımlandığı XAML kodunun üstünde *buttonStyle* stiline başvurduğunuz. Bu kapsam dışındaki başvuru çalışır, çünkü *DynamicResource* kullanmak, çalışma zamanında kaynağa başvurunun analizinin yapılmasını kaynağın oluşturulduğu noktaya kadar erteler.

Genel olarak, statik kaynaklar, dinamik kaynaklardan daha verimlidir çünkü uygulama oluşturulduğunda başvuru analizi yapılır, bununla birlikte dinamik kaynaklar daha fazla esneklik sağlar. Örneğin, uygulama çalıştıkça kaynağın kendisi değişirse (çalışma zamanında stili değiştirmek için kod yazabilirsiniz), *StaticResource* kullanarak stile başvuran denetimler güncellenmeyecektir, fakat *DynamicResource* kullanarak stile başvuran denetimler güncellenecektir.

Statik ve dinamik kaynaklar arasında birçok başka farklılıklar ve bir kaynağa dinamik olarak başvurabilmeniz için kısıtlamalar vardır. Daha fazla bilgi için, Visual Studio 2008 tarafından sağlanan .NET Framework belgelerine başvurun.

Stilleri Uygulamak

Stil tanımında hâlâ biraz yineleme vardır; özelliklerin her biri (arka plan, ön plan ve yazı tipi) açık bir şekilde düğme özellikleri olduklarını belirtir. *Style* etiketinde *TargetType* özniteliğini belirterek bu yinelemeyi ortadan kaldırabilirsiniz.

TargetType özniteliğini belirtmek için stil tanımını aşağıdaki gibi değiştirin:

```
<Style x:Key="buttonStyle" TargetType="Button">
    <Setter Property="Background" Value="Gray"/>
    <Setter Property="Foreground" Value="White"/>
    <Setter Property="FontFamily" Value="Comic Sans MS"/>
</Style>
```

Forma istediğiniz kadar düğme ekleyebilirsiniz ve tümünü yine *buttonStyle* stilini kullanarak düzenleyebilirsiniz. Peki, etiketler ve metin kutuları gibi diğer denetimler nasıl olur?

Stilleri Uygulamak

Design View penceresinde, *Window1* formunu tıklayın ve daha sonra *Toolbox* sekmesini tıklayın. *Common* bölümünde *TextBox*'ı tıklayın ve sonra formun alt bölümünde herhangi bir yeri tıklayın.

XAML bölmesinde, metin kutusu denetiminin tanımını değiştirin ve *buttonStyle* stili uygulamaya çalışan, aşağıdaki örnekte gösterilen *Style* özniteliğini belirtin:

```
<TextBox Style="{StaticResource buttonStyle}" Height="21" Margin="114,0,44,58" Name="textBox1"
VerticalAlignment="Bottom" />
```

Stilleri Uygulamak

Düğme için tasarlanmış bir stili, bir metin kutusunun stili olarak ayarlamaya çalışmak başarısızlıkla sonuçlanır. *Design View* penceresinde "The document root element has been altered or an unexpected error has been encountered in updating the designer. Click here to reload." (Belge kök ögesi değiştirilmiş ya da tasarımcıyı güncellerken beklenmedik hata oluştu. Yeniden yüklemek için burayı tıklayın) hata iletisi görünür. Gösterildiği gibi iletii tıklarsanız, *Design View* penceresinde form kaybolur ve yerine aşağıdaki ileti gelir:



Problem Loading

The document contains errors that must be fixed before the designer can be loaded. Reload the designer after you have fixed the errors.

[Reload the designer](#)

Stilleri Uygulamak

XAML bölümünde, *Key* özelliğini düzenleyin ve stil tanımını *TargetType* yerine *Control* olacak şekilde değiştirin, daha sonra aşağıda gösterildiği gibi düğme ve metin kutusu denetimlerinde stile başvuruları düzenleyin:

```
<Window x:Class="BellRingers.Window1"
...>
<Window.Resources>
    <Style x:Key="bellRingersStyle" TargetType="Control">
        <Setter Property="Background" Value="Gray"/>
        <Setter Property="Foreground" Value="White"/>
        <Setter Property="FontFamily" Value="Comic Sans MS"/>
    </Style>
</Window.Resources>
```

Stilleri Uygulamak

```
<Grid>
  <Button Style="{StaticResource bellRingersStyle}" ...>
    Button
  </Button>
  <Button Style="{StaticResource bellRingersStyle}" ...>
    Button
  </Button>
  ...
  <TextBox ... Style="{StaticResource bellRingersStyle}" ... />
</Grid>
</Window>
```

Stilleri Uygulamak

Stil artık sadece düğmelere uygulanmayacağından, yeniden adlandırmak iyi olacaktır. Stilin *TargetType* özniteliğini *Control* olarak ayarlamak, *Control* sınıfından kalıtımla alınan denetime stilin uygulanabileceğini belirtir. WPF modelinde, metin kutuları ve düğmeleri de içeren denetimlerin birçok farklı türü, *Control* sınıfından kalıtımla alır. Bununla birlikte, sadece açık bir şekilde *Control* sınıfına ait özellikler için *Setter* öğelerini sağlayabilirsiniz. (Düğmeler, *Control* sınıfının parçası olmayan bazı ek özelliklere sahiptir; bu düğmeye ait özelliklerden birini belirlerseniz, *TargetType* özniteliğini *Control* olarak ayarlayamazsınız.)

Stil İçerisinde Trigger Kullanımı

Ne yazık ki, renklerin seçimi, metin kutusunu tıklayarak bir metin yazdığınızda metin imlecini görmenizi zorlaştırır. Bu problemi sonraki adımda çözeceksiniz.

XAML bölmesinde, *bellRingersStyle* stilini düzenleyin ve aşağıdaki kodda koyu renkle gösterilen `<Style.Triggers>` ögesini ekleyin. (*TriggerCollection*'ın mühürlendiğine (sealed olduğuna) dair bir hata iletisi alırsanız, çözümü yeniden oluşturun.)

Stil İçerisinde Trigger Kullanımı

```
<Style x:Key="bellRingersStyle" TargetType="Control">
  <Setter Property="Background" Value="Gray"/>
  <Setter Property="Foreground" Value="White"/>
  <Setter Property="FontFamily" Value="Comic Sans MS"/>
  <Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
      <Setter Property="Background" Value="Blue" />
    </Trigger>
  </Style.Triggers>
</Style>
```

Tetikleyici, bir özellik değeri değiştiğinde gerçekleştirilecek olan hareketi belirler. *bellRingersStyle* stili, fare işaretçisi üzerine gittiğinde denetimin arka plan rengini geçici olarak değiştirmek için *IsMouseOver* özelliğindeki değişikliği belirler.

Stil İçerisinde Trigger Kullanımı

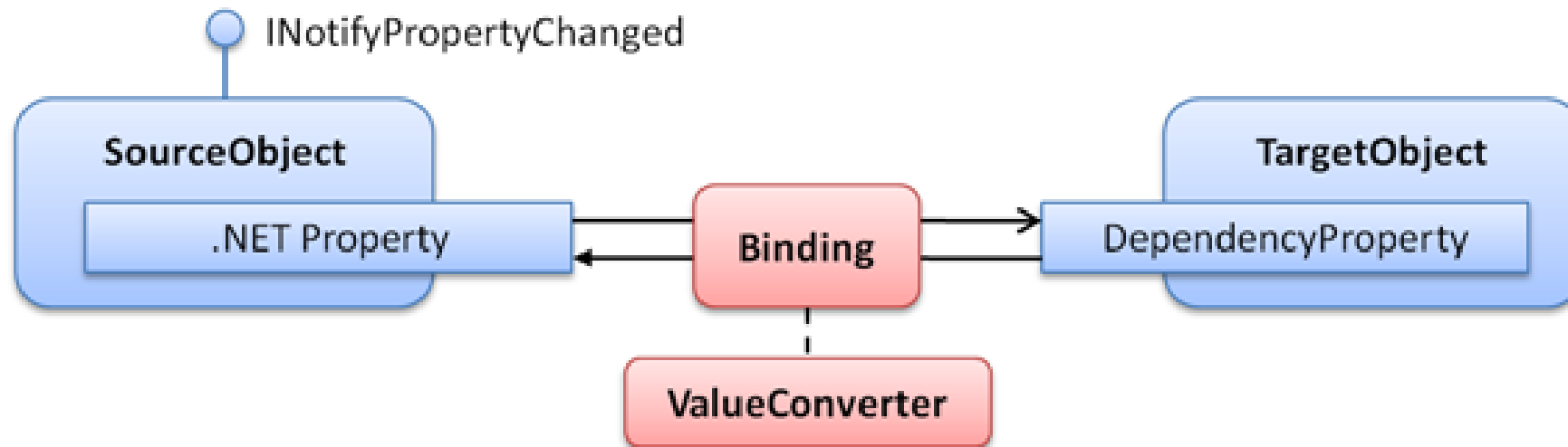


Not Tetikleyicileri olaylarla karıştırmayın. Tetikleyiciler, özellik değerlerindeki geçici değişikliklere karşılık verir. Tetikleme özelliğindeki değer tersine dönerse, tetiklenen hareket tamamlanmamış olur. Daha önce gösterilen örnekte, bir denetim için *IsMouseOver* özelliği artık *true* olmadığında, *Background* özelliği orijinal değerine geri döner. Olaylar, bir uygulamada önemli bir özel durum (kullanıcının düğmeyi tıklaması gibi) meydana geldiğinde gerçekleştirilecek bir hareketi belirtir; olay tarafından gerçekleştirilen hareketler, özel durum sona erdiğinde tamamlanmamış olur.

Veri Bağlama (Data Binding) İşlemlerine Giriş

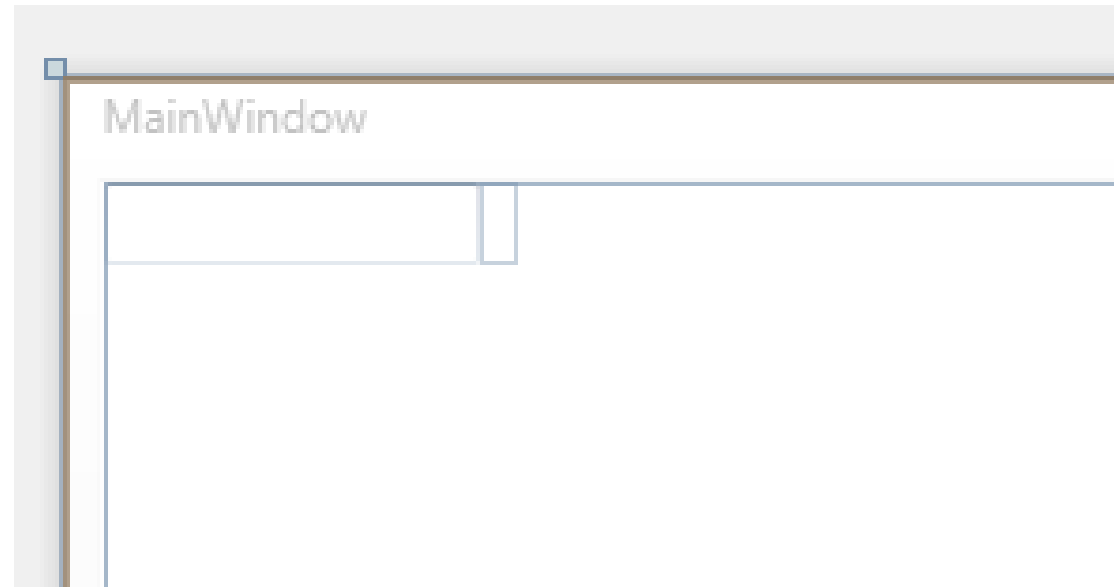
Veri Bağlama (Data Binding) Nedir?

Çalışma Modeli



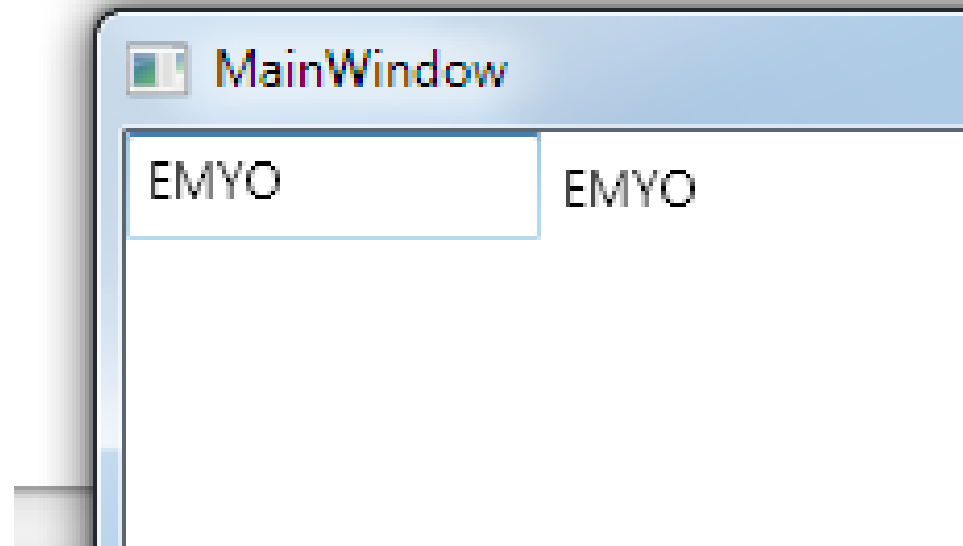
Merhaba EMY☺ - Tasarım Anı

```
<Grid>  
    <WrapPanel>  
        <TextBox Width="100"></TextBox>  
        <Label></Label>  
    </WrapPanel>  
</Grid>
```



Merhaba EMY😊 - Çalışma Zamanı

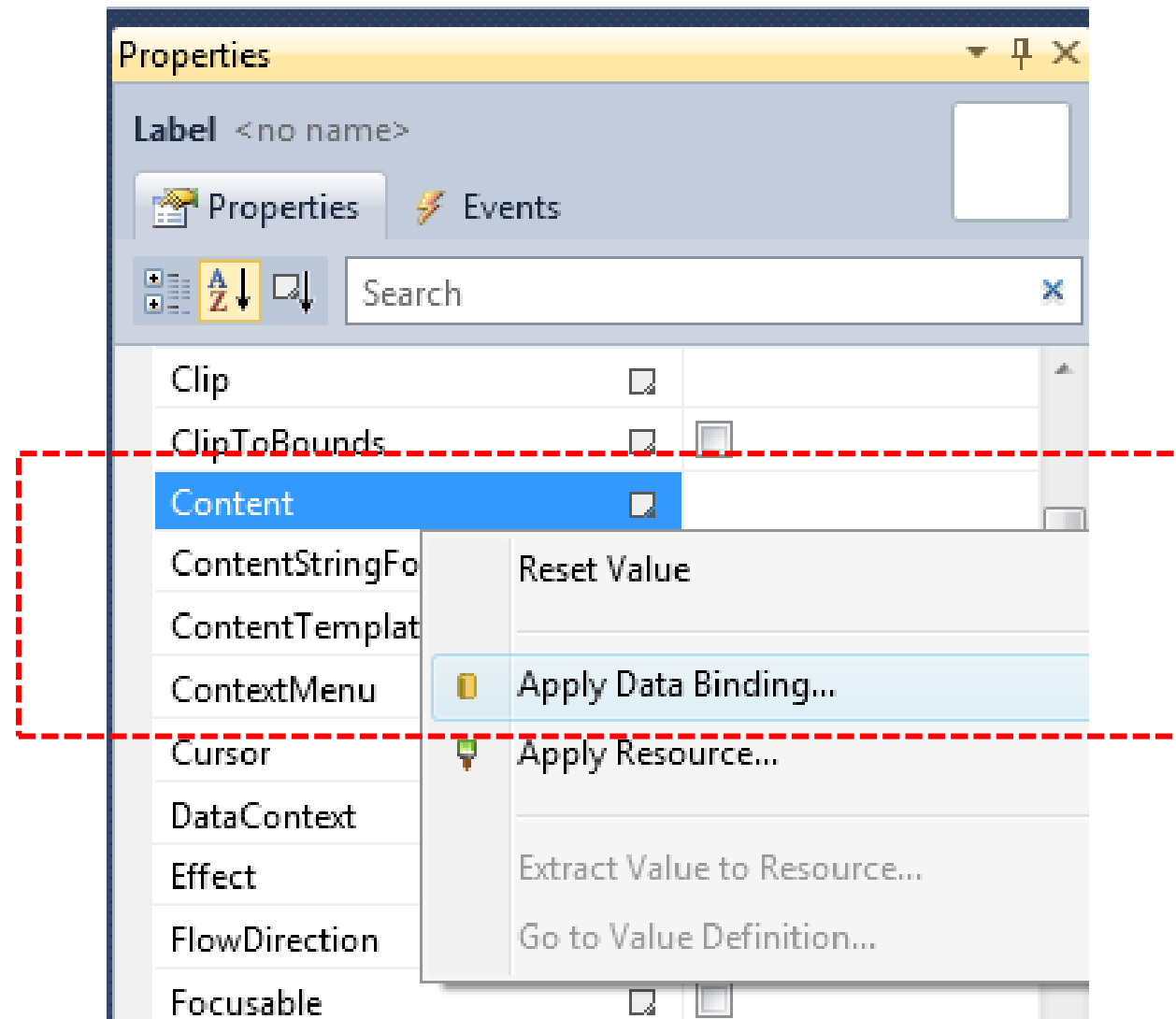
```
<Grid>
  <WrapPanel>
    <TextBox Width="100" Name="tbKaynak"></TextBox>
    <Label>
      <Label.Content>
        <Binding ElementName="tbKaynak" Path="Text"/>
      </Label.Content>
    </Label>
  </WrapPanel>
</Grid>
```



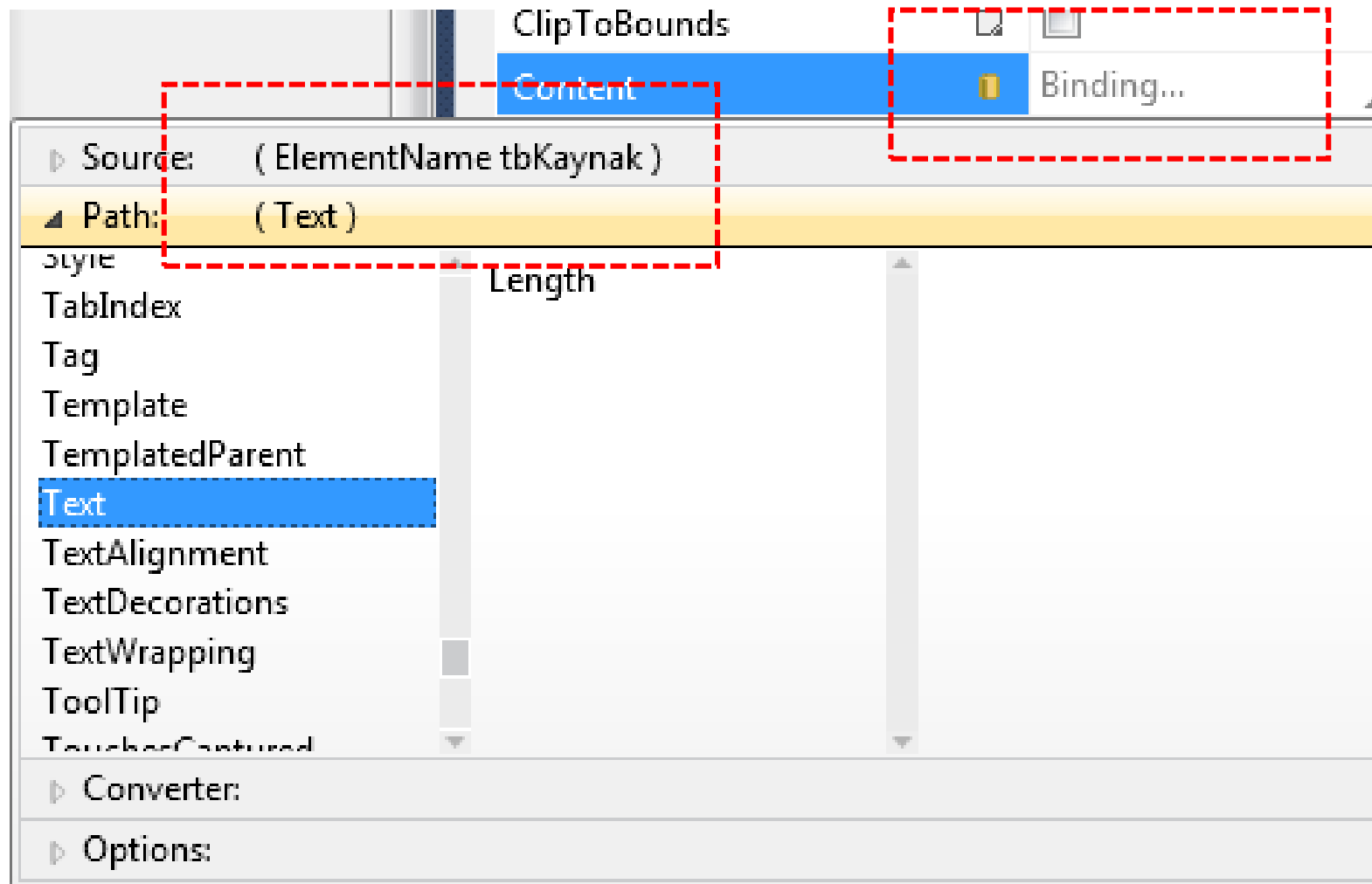
Merhaba EMY😊 - Çalışma Zamanı

```
<Grid>
    <WrapPanel>
        <TextBox Width="100" Name="tbKaynak"></TextBox>
        <Label Content="{Binding ElementName=tbKaynak, Path=Text}" />
    </WrapPanel>
</Grid>
```

```
<Label Content="{Binding ElementName=tbKaynak, Path=Text}" />
```

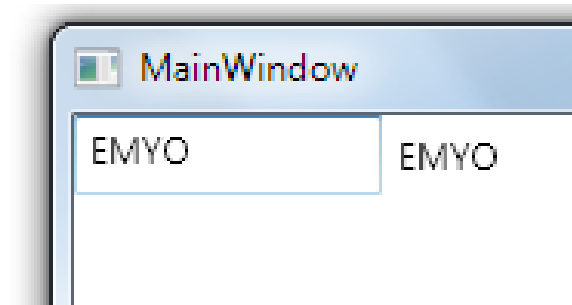


```
<Label Content="{Binding ElementName=tbKaynak, Path=Text}" />
```

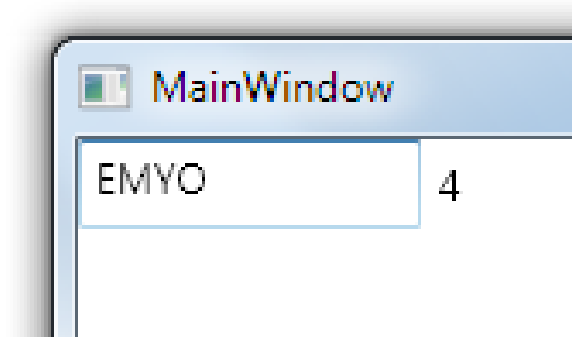


Çok Esnek

```
<Label Content="{Binding Path=Text, ElementName=tbKaynak}" />
```



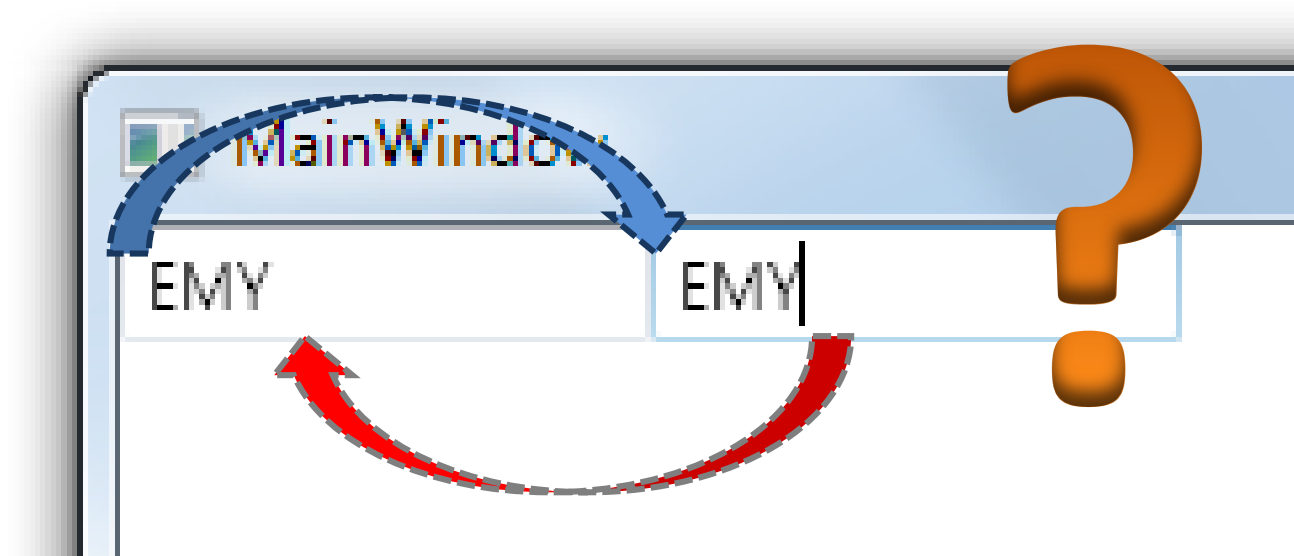
```
<Label Content="{Binding Path=Text.Length, ElementName=tbKaynak}" />
```



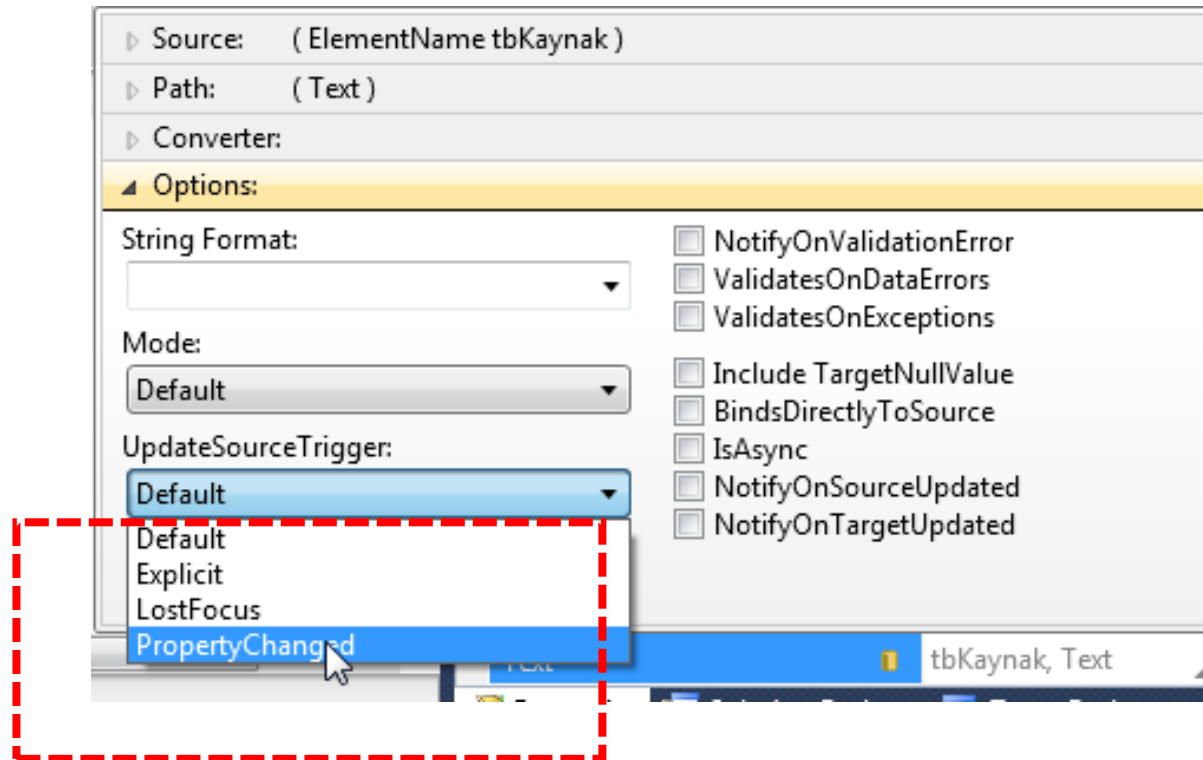
Ne Zaman Çalışsın?

```
<TextBox Width="100" Name="tbKaynak"/>
```

```
<TextBox Text="{Binding Path=Text, ElementName=tbKaynak}" Width="100" />
```



Ne Zaman Çalışsın?



```
<TextBox Text="{Binding Path=Text, ElementName=tbKaynak, UpdateSourceTrigger=PropertyChanged}" Width="100" />
```

Mode Özelliği

```
<TextBox Width="100" Name="tbKaynak"/>
```

```
<TextBox Text="{Binding Path=Text.Length,  
                        ElementName=tbKaynak,  
                        UpdateSourceTrigger=PropertyChanged}"  
Width="100" />
```

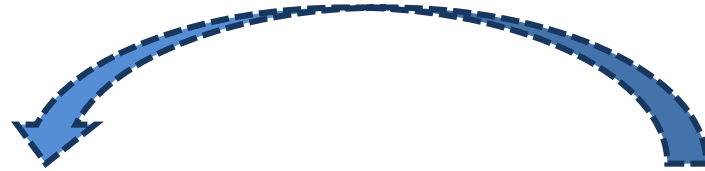


XamlParseException occurred



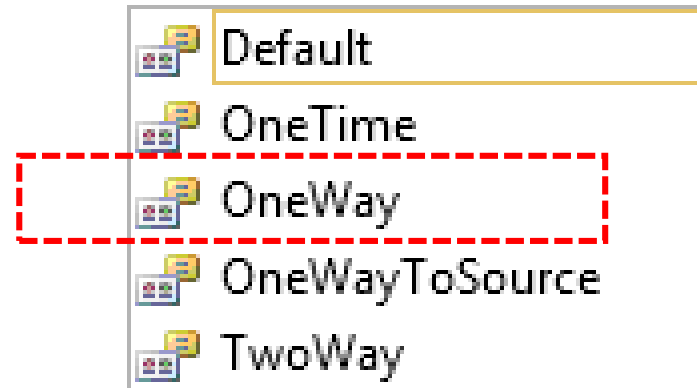
'Set property 'System.Windows.Controls.TextBox.Text' threw an exception.' Line number '10' and line position '22'.

Mode Özelliği

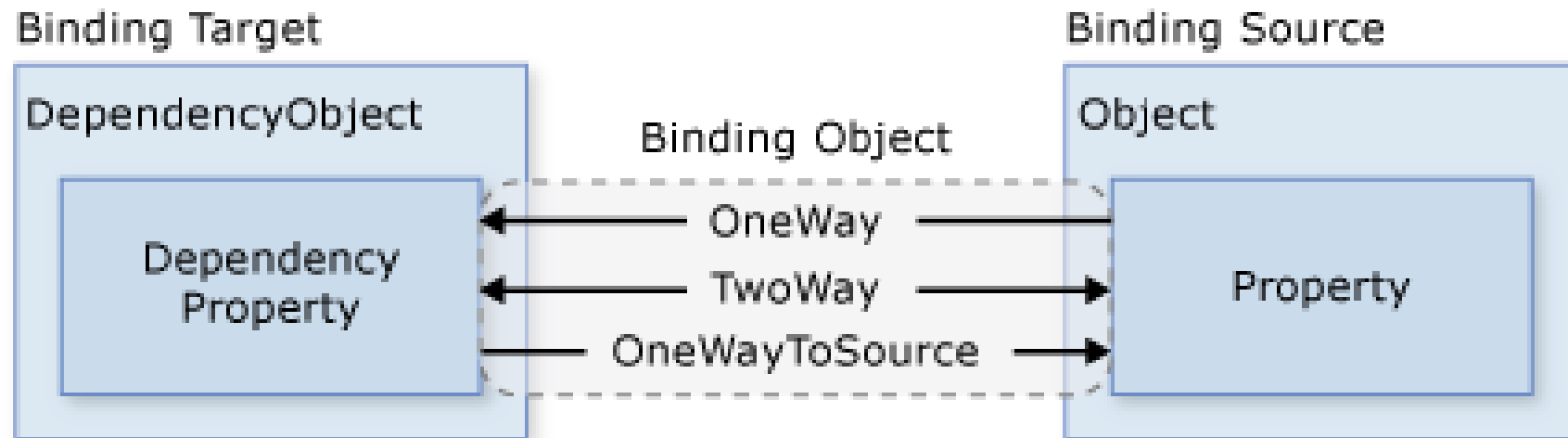


```
<TextBox Text="{Binding Path=Text.Length,  
                        ElementName=tbKaynak,  
                        UpdateSourceTrigger=PropertyChanged,  
                        Mode=}"  
Width="100" />
```

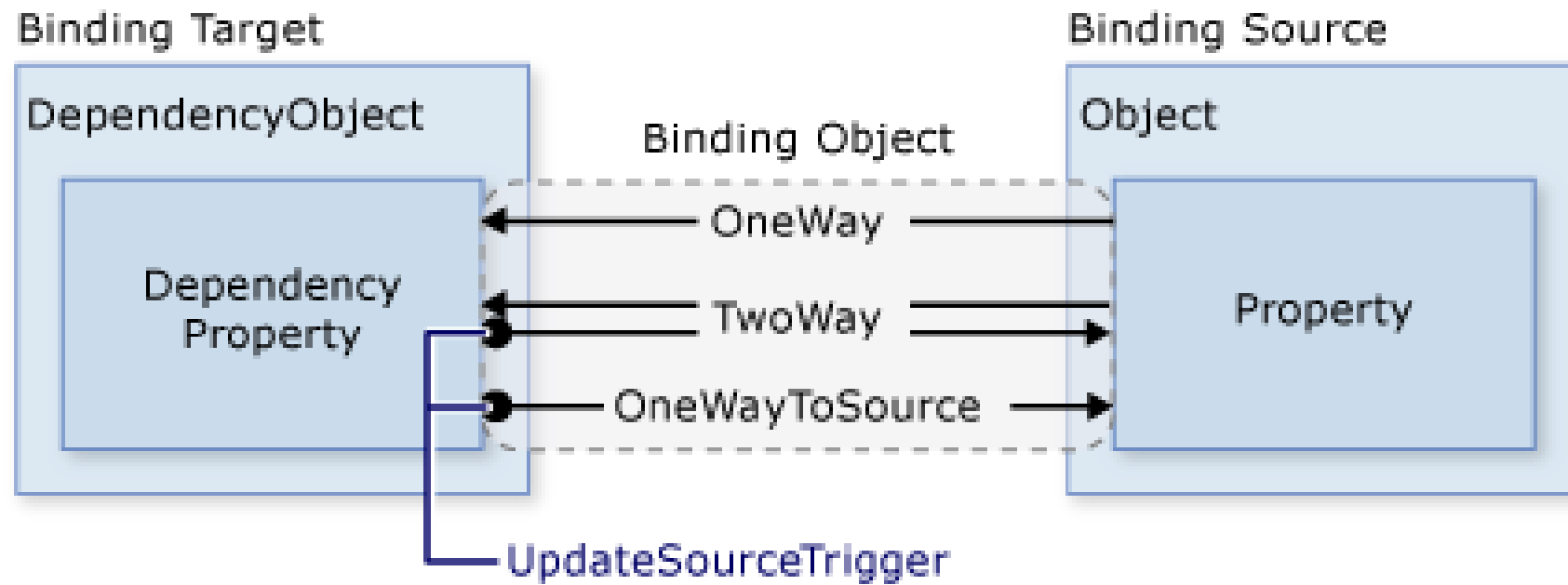
rapPanel>



Mode Özelliği



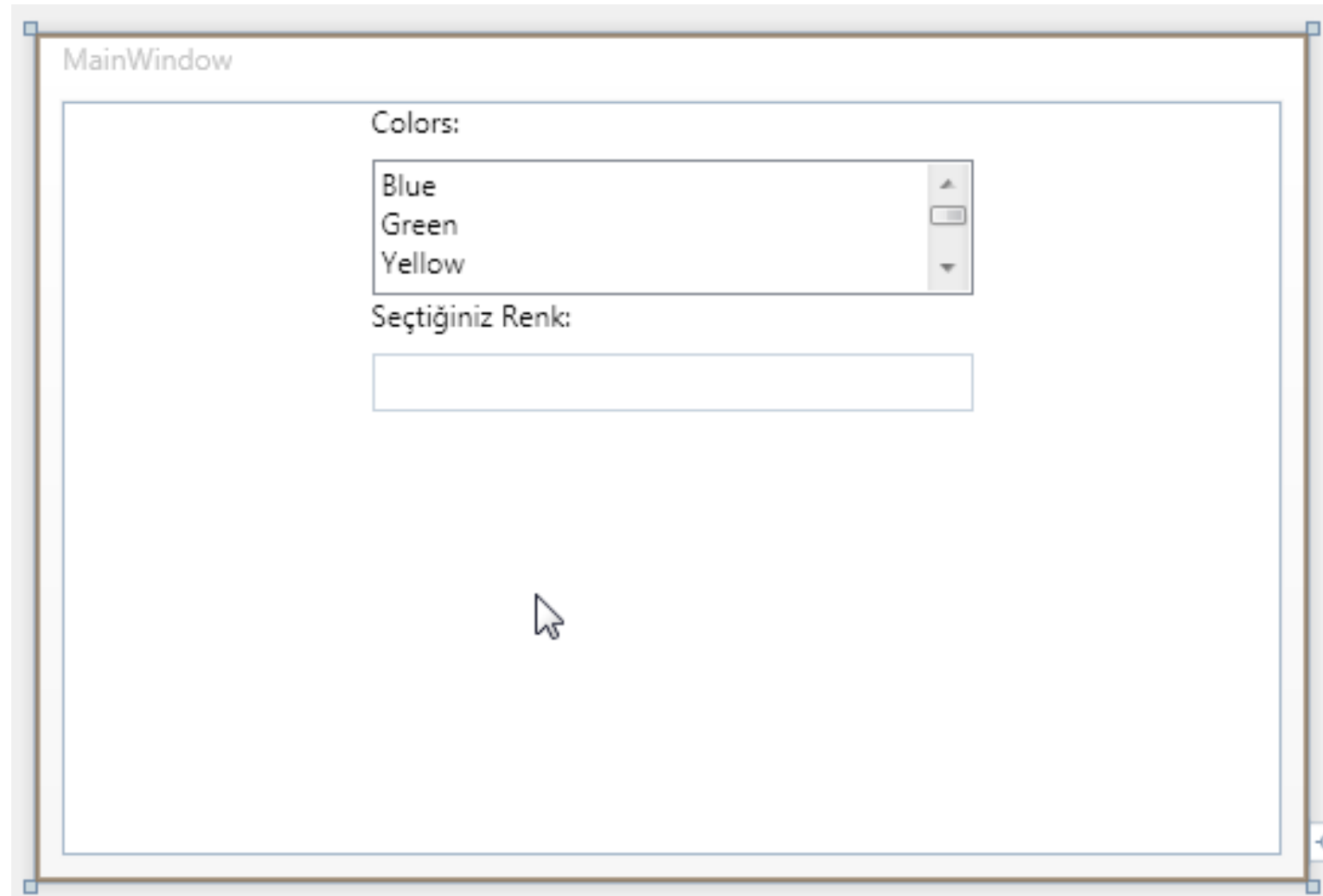
Mode – UpdateSourceTriggers



Örnek

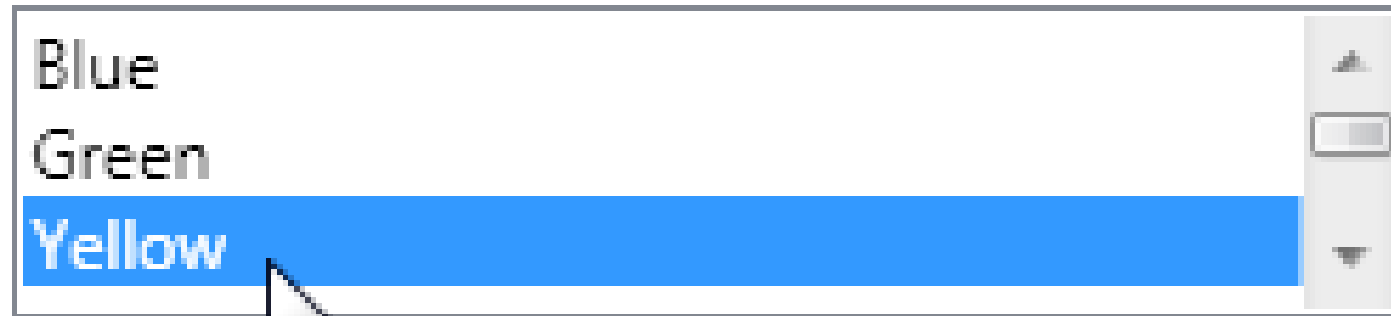
```
<StackPanel>
  <TextBlock Width="248" Height="24" Text="Colors:" TextWrapping="Wrap"/>
  <ListBox x:Name="lbColor" Width="248" Height="56">
    <ListBoxItem Content="Blue"/>
    <ListBoxItem Content="Green"/>
    <ListBoxItem Content="Yellow"/>
    <ListBoxItem Content="Red"/>
    <ListBoxItem Content="Purple"/>
    <ListBoxItem Content="Orange"/>
  </ListBox>
  <TextBlock Width="248" Height="24" Text="Seçtiğiniz Renk:" />
  <TextBlock Width="248" Height="24" Text="{Binding ElementName=lbColor,
                                                    Path=SelectedItem.Content}" />
</StackPanel>
```


Örnek



Soru

Colors:



Blue
Green
Yellow

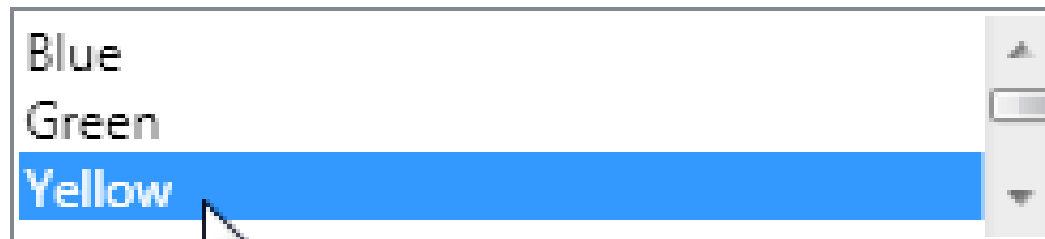
Seçtiğiniz Renk:

Yellow

Cevap

```
<TextBlock Width="248" Height="24" Text="{Binding ElementName=lbColor,  
                                                    Path=SelectedItem.Content}"  
            Background="{Binding ElementName=lbColor,  
                                Path=SelectedItem.Content}" />
```

Colors:



Seçtiğiniz Renk:

Yellow

Havada Kalmasin ☺



Teşekkürler...



*bize
katlandığınız
için!*
