Kontroller*

Öğr.Gör.Erkan HÜRNALI

*Ahmet Ali SÜZEN'in WPF kitabından derlenmiştir.

T.C. ANKARA ÜNİVERSİTESİ
MYO
elmadağ meslek yüksekokulu

Kontroller

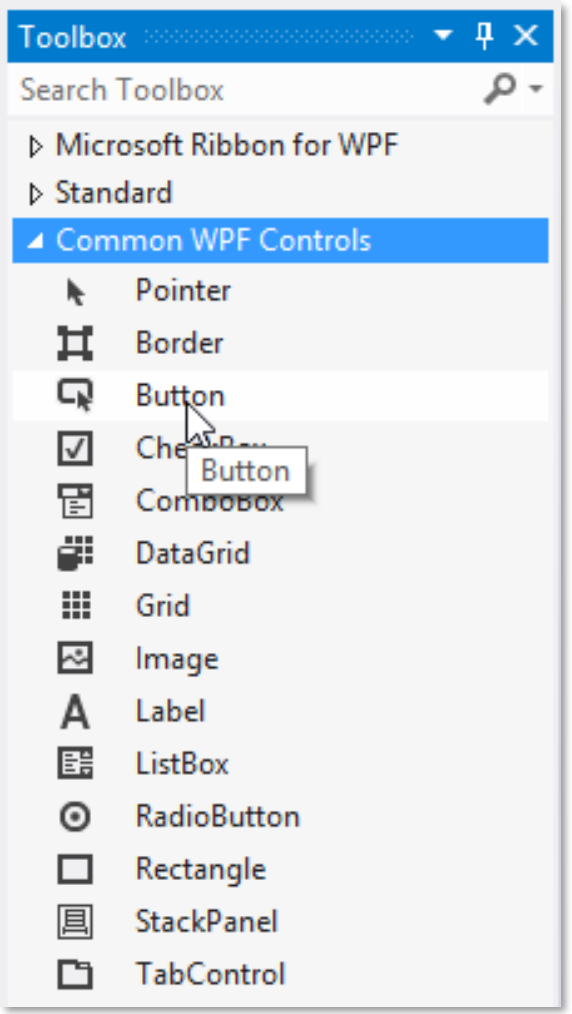
2003 yılında P.D.C.'de WPF'in duyurulması ile beraber yazılım alanında farklılaşmanın başladığını gördük. İlk değişim, tasarım kodlama alanının codebehind'dan ayrılması oldu. Tasarım alanının kendi içerisinde güçlenmesi ile beraber WPF kontrollerinin kullanıcı etkileşimi, form uygulamalarına göre olumlu bir değişim gösterdi. WPF'de kontrollerin hem codebehind tarafından hem de XAML ile eklenmesine imkan sağlandı. Form uygulamalarında bulunan kontrollerin kullanıcı etkileşimi oldukça düşüktür. Fakat WPF'e ait kontroller kullanıcı etkileşimli yapıya sahiptir

Kontroller

Kontroller, bir uygulamanın en önemli parçasıdır. Bu kozu uygulamalarda doğru ve etkin bir biçimde kullanmak kaliteyi artırmaktadır. Bu bölümde VVPF'e ait kontrollerin her birisini farklı örneklerle inceleyeceğiz.

Kontroller, yeni uygulamaya başlayanların zevkle incelediği bir kısımdır. Bir kontrolü ihtiyaca göre farklı yerlerde oluşturacağız. Kontrolleri standart olarak Toolbox penceresinde görmekteyiz. Buradan rahatlıkla tasarım alanının istenilen noktasına yerleştirebiliriz. Unutmayalım ki Toolbox'dan kullanılan her kontrol XAML alanında tanımlanır.

WPF'e ait kontroller genel olarak System.windows.Controls uzayından gelmektedir. Her bir kontrol aslında birer sınıf yapısındadır. Kontroller amaçlarına göre farklı sınıflardan türeyebilir. Kontrolleri anlatırken her kontrolün yapısı hakkında bilgiler vereceğiz.



Kontroller

Property	Açıklama
Name	Kontroller için benzersiz erişim isimleri.
Width	Kontrole ait genişliği belirler.
Height	Kontrole ait yüksekliği belirler.
HorizontalAlignment	Kontrolün Window'daki yatay referansını belirler.
VerticalAlignment	Kontrolün Window'daki dikey referansını belirler.
Margin	Kontrolün Window ekranındaki konumun belirler.
Font	Kontrole ait yazı biçimini belirler.
Foreground	Kontroller ait yazı rengini veya yazının zemin resmini belirler.
Backgroud	Kontrolün arkaplan rengini veya resmini belirler.
Opacity	Kontrolün Window alanındaki görünürlüğünü belirler. 0 ile 1 arasında double değerler alır.
ContextMenu	Kontrollere ait menüler oluşturmak için kullanılır.

Button

- Programlamanın en alt seviyesinden en üst seviyesine kadar her noktada tercih edilen bir kontroldür.
- ButtonBase sınıfında türeyen bir sınıftır.
- Genellikle tek amaçlı yönetimlerde kullanırız.
- En sık kullanılan property'si Content, en çok ihtiyaç duyulan event'i ise Click'tir. Yine codebehind ve XAML kısımlarında oluşturulabilir.

Button

Xaml

```
<Button Content="Tıkla" Height="78" HorizontalAlignment="Left"
Margin="115,102,0,0" Name="button1" VerticalAlignment="Top" Width="196"
FontSize="45" Foreground="#FF694A4A" Click="button1_Click"></Button>
```

Codebehind

```
private void button1_Click(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("Tıklandı!");
        }
```

Button

Örnekte görüldüğü gibi button kontrolünü XAML ile oluşturduk. Daha sonra button için gerekli property'lere ilk değer tanımlaması gerçekleştirdik.

```
Button b = new Button();
b.Name = "button1";
b.HorizontalAlignment = HorizontalAlignment.Left;
b.VerticalAlignment = VerticalAlignment.Top;
b.Margin = new Thickness(150, 150, 0, 0);
this.window.Children.Add(b);
```

Button sınıfın nesnesini alıp gerekli property değerlerini verdikten sonra Window isimli panele kontrolümüzü ekledik.

CheckBox

- ToogleButton sınıfından türeyen CheckBox kullanıcı tercihlerini belirlemek için kullanılan bir kontroldür.
- ChechBox, boolen mantığına göre çalışmaktadır.
- Yani true ya da false değerlerini taşımaktadır.
- isChecked property'si ile başlangıç durumunu belirleyebiliriz.
- Ayrıca Content property'si ile de görünün metnini değiştirebiliriz.
- En çok tercih edilen event'i ise Checked'dır.

CheckBox

Xaml

```
<CheckBox Content="CheckBox" Height="19" HorizontalAlignment="Left"
Margin="70,76,0,0" Name="checkBox1" VerticalAlignment="Top" Width="135"
Checked="checkBox1_Checked" />
<Label Content="Label" Height="28" HorizontalAlignment="Left"
Margin="70,101,0,0" Name="label1" VerticalAlignment="Top" />
```

Codebehind

```
private void checkBox1_Checked(object sender, RoutedEventArgs e)
      {
          if (checkBox1.IsChecked==true)
          {
               label1.Content = "İşaretli";
          }
          else
          {
               label1.Content = "İşaretli değil.";

          }
      }
```

ComboBox

- Genellikle tanımlı veriler içerisinde kullanıcı seçimi yaptırmak için tercih edilen bir kontroldür.
- Selector sınıfında türemiştir.
- Genel olarak form uygulamalarındaki özelliklere benzemektedir.
- Ama dikkat çeken bir özelliği vardır ki bu özellik, içerisine sadece veri değil uiElement alabiliyor olmasıdır.
- SelectedItem, SelectedIndex, SelectedValue ve Items property'lerini daha çok kullanmaktayız.

ComboBox

Xaml

```
<ComboBox Height="23" HorizontalAlignment="Left" Margin="77,95,0,0"
Name="comboBox1" VerticalAlignment="Top" Width="120"
SelectionChanged="comboBox1_SelectionChanged">
      <ComboBoxItem Content="Donanım" />
      <ComboBoxItem Content="Yazılım" />
</ComboBox>
```

ComboBox

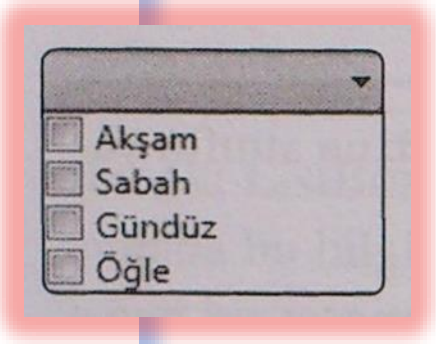
Codebehind

```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
//Açılışta seçili item'ı belirleyebiliriz.
      comboBox1.SelectedIndex = 1;
      comboBox1.Text = "İller";

}

private void comboBox1_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
      ComboBoxItem c = comboBox1.SelectedItem as ComboBoxItem;
      MessageBox.Show(c.Content.ToString());
}
```

ComboBox - Örnek

ComboBox kontrolünü UIElemet'leri bünyesine alabileceğiniz söylemiştik. Şöyle bir örnek ile daha net açıklayalım.

```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
      string[] gazeteler = {"Akşam","Sabah","Gündüz","Öğle"};

      for (int i = 0; i < gazeteler.GetLength(0); i++)
      {
            CheckBox secim = new CheckBox();
            secim.Content = gazeteler[i];
            comboBox1.Items.Add(secim);
      }
}
```



Image

- Image kontrolü, FrameworkElement sınıfından türemiştir.
- Bu kontrolün Source property'sine imageSource sınıfı türünden atama yapabiliriz.
- Ayrıca BitmapImage sınıfı ile oluşturulan resim türlerini Image kontrolüne aktarabiliriz.

imageSourceConverter sınıfında yararlanarak URI olarak da resimleri kontrolümüze aktarabiliriz

```
image1.Source = (ImageSource)new
ImageSourceConverter().ConvertFromString("www.a2syazilim.com/kodlab.png");
```

Image

Diğer yöntem ise Bitmaplmage sınıfını ile resim türlerini rahatlıkla kontrolümüze atayabiliriz

```
BitmapImage resim = new BitmapImage(new Uri("F:\\deneme.jpg"));
image1.Source = resim;
```

Xaml

```
<Image Height="150" HorizontalAlignment="Left" Margin="81,71,0,0"
Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="200"
Source="/WpfApplication2;component/Images/1.jpg" />
```

Border

- **Border** sınıfı, UIElement'leri içerisine alarak, bu kontrollere dış çerçeve oluşturmamızı sağlar.
- **CornerRadius** property'si ile çerçevenin köşe kenarlarının biçimin ayarlamak için kullanılır.
- **BorderThickness** ile kenar çerçevelerinin kalınlıklarını belirleriz.
- **Padding** ile **Border** içerisinde bulunan kontrollerin **Border** 'ın sol, sağ. yukarı, aşağı durumlarındaki yerlerini belirler

Border

Xaml

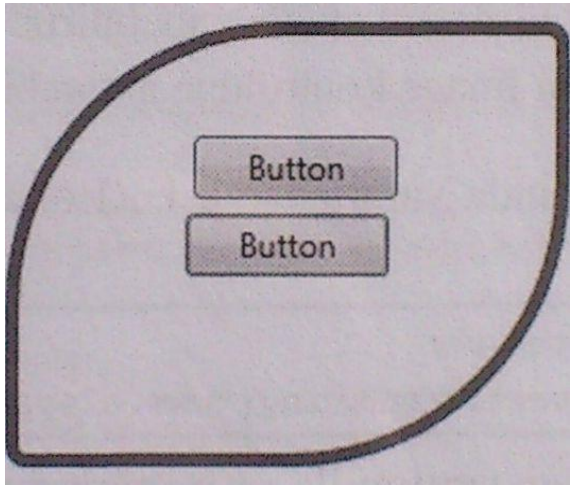
```
<Border BorderBrush="#FF9D1212" BorderThickness="5" Height="161"
HorizontalAlignment="Left" Margin="46,92,0,0" Name="border1"
VerticalAlignment="Top" Width="207" CornerRadius="25"
Background="{x:Null}">
      <Image Height="141" Name="image1" Stretch="Fill" Width="179"
Source="/WpfApplication2;component/Images/1.jpg" />
 </Border>
```

Border

Ayrıca Border kontrolünün CornerRadius property'sine ComerRadius sınıfın yapıcı metodunda her köşe için farklı değerler belirleyebiliriz.

Codebehind kısmından herhangi bir kontrolü child property'si ile değiştirebiliriz

```
CornerRadius kose = new CornerRadius(100, 5, 100, 5);
border1.CornerRadius = kose;
```



ListBox

- ListBox kontrolü işlev olarak ComboBox'a çok benzemektedir.
- Yine bu kontrol içerisine sadece veri değil, UIElement de alabilir.

Metod	Açıklama
SelectedItem	Seçili veriyi almak için kullanılır. ListBoxItem olarak kullanabiliriz.
SelectedIndex	Seçili olan verinin indeks sayısını alabiliriz.
SelectedValue	Seçili olan verinin değerini alabiliriz.
Items	ListBox kontrolüne veri eklemek için kullanabiliriz.
SelectionMode	ListBox içerisindeki verilerin seçim tipini belirleyebiliriz. Tek veya çoklu seçim imkanı sağlayabiliriz.

ListBox

Xaml

```
<ListBox Height="272" HorizontalAlignment="Left" Margin="334,12,0,0"
Name="listBox1" VerticalAlignment="Top" Width="157"
SelectionChanged="listBox1_SelectionChanged">
        <ListBoxItem Content="Antalya" />
        <ListBoxItem Content="izmir" />
        <ListBoxItem Content="Isparta" />
 </ListBox>
```

Örnek: ListBox içerisine UIElement yani kontrol alabilen bir yapıdadır. Bu uygulamada bu bilgiyi kullanarak bir mini resim albümü yapalım. Öncelikle albüm için genel bir tasarım yapıyoruz. Gerekli kontrolleri Window üzerine yerleştiriyoruz.

ListBox - Örnek

Xaml

```
<Grid x:Name="window">
      <ListBox Height="349" HorizontalAlignment="Left"
Margin="38,12,0,0" Name="listBox1" VerticalAlignment="Top"
Width="156" SelectionChanged="listBox1_SelectionChanged"
FlowDirection="LeftToRight" VerticalContentAlignment="Top">
            <Image Height="100" HorizontalAlignment="Left"
Margin="10,10,0,0" Name="image1" Stretch="Fill" VerticalAlignment="Top"
Width="125" Source="/WpfApplication2;component/Images/1.jpg" />
            <Image Height="100" HorizontalAlignment="Left"
Margin="10,10,0,0" Name="image2" Stretch="Fill" VerticalAlignment="Top"
Width="125" Source="/WpfApplication2;component/Images/03-25-Cool-Windows-
8-Wallpapers.png" />
            <Image Height="100" HorizontalAlignment="Left"
Margin="10,10,0,0" Name="image3" Stretch="Fill" VerticalAlignment="Top"
Width="125" Source="/WpfApplication2;component/Images/arkaplan2.png" />
      </ListBox>
      <Image Height="276" HorizontalAlignment="Left" Margin="213,48,0,0"
Name="image4" Stretch="Fill" VerticalAlignment="Top" Width="303" />
      <Button Content="Değiştir" Height="23" HorizontalAlignment="Left"
Margin="262,358,0,0" Name="button1" VerticalAlignment="Top" Width="75" />
   </Grid>
```

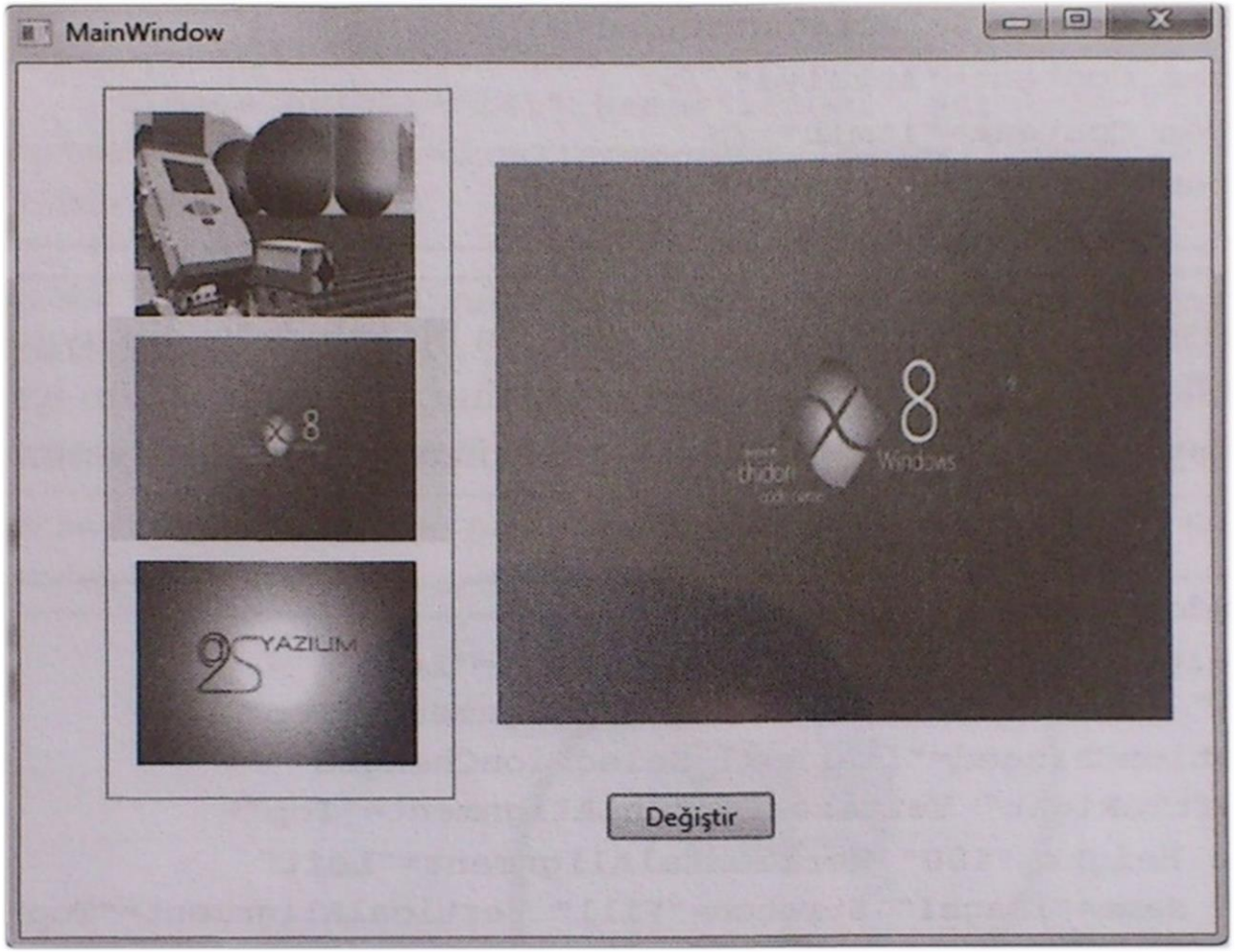
ListBox - Örnek

Codebehind

```
private void button1_Click(object sender, RoutedEventArgs e)
    {
        listBox1.SelectedIndex = new Random().Next(2);
        Image resim = (Image)listBox1.SelectedItem;
        image4.Source = resim.Source;
    }
```

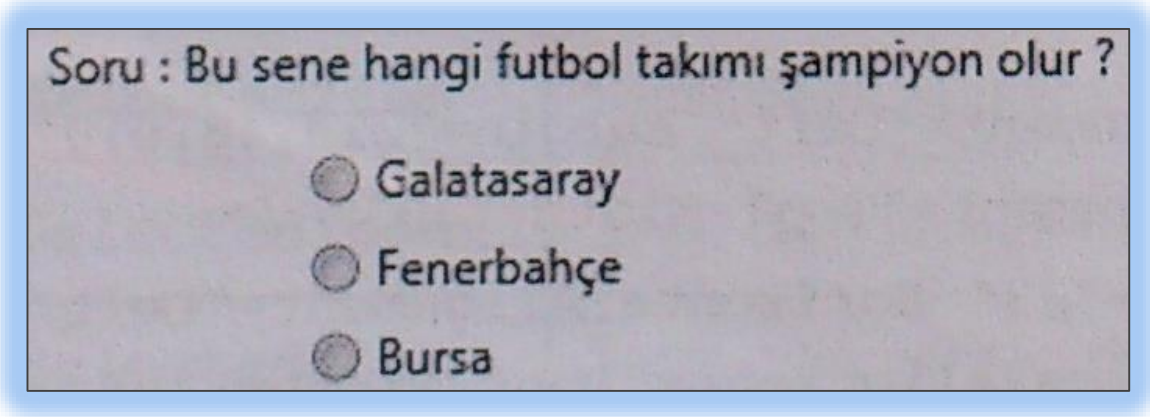
ListBox - Örnek

Sonuç



RadioButton

- Çoklu seçim alanlarından sadece tek seçim yapılmasını sağlayan kontroldür.
- RadioButton içeren Container içerisinde sadece bir tane seçim yapabiliriz.
- ToogleButton sınıfından türemiştir.
- Özellik olarak CheckBox'a benzemektedir.
- Her bir RadioButton'un Checked event'ını tek bir metod ile birleştirdik



RadioButton

Xaml

```
<Grid x:Name="window">
        <RadioButton Content="Galatasaray" Height="16"
HorizontalAlignment="Left" Margin="148,132,0,0" Name="radioButton1"
VerticalAlignment="Top" Checked="secim_Checked" />
        <RadioButton Content="Fenerbahçe" Height="16"
 HorizontalAlignment="Left" Margin="148,154,0,0" Name="radioButton2"
 VerticalAlignment="Top" Checked="secim_Checked" />
        <RadioButton Content="Bursa" Height="16" HorizontalAlignment="Left"
 Margin="148,176,0,0" Name="radioButton3" VerticalAlignment="Top"
 Checked="secim_Checked" />
        <Label Content="Soru : Bu sene hangi futbol takımı şampiyon olur ?"
 Height="28" HorizontalAlignment="Left" Margin="78,98,0,0" Name="label1"
 VerticalAlignment="Top" />
    </Grid>
```

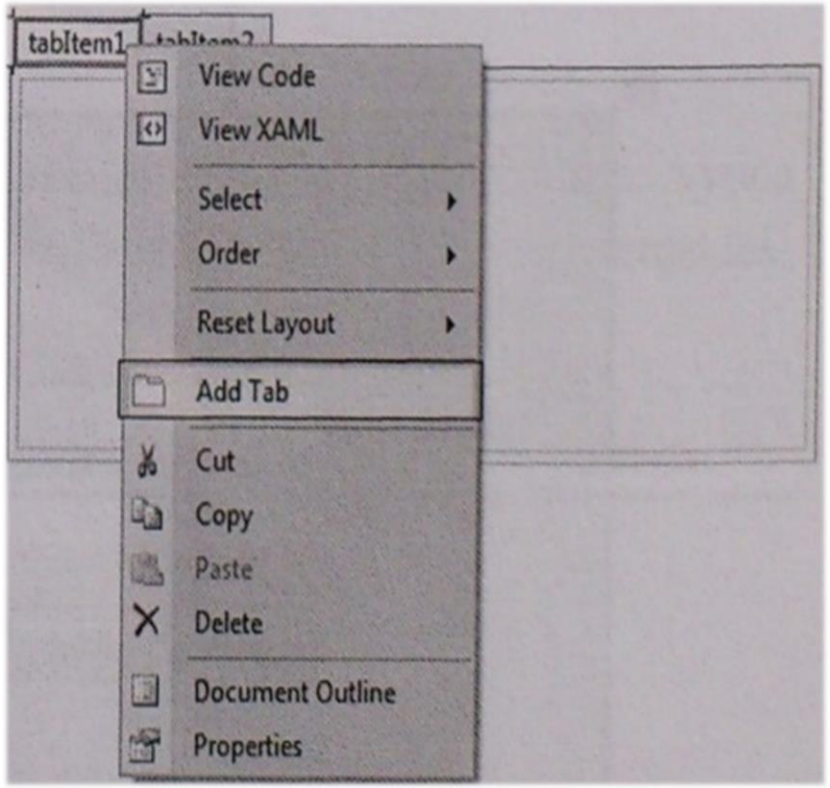
RadioButton

Codebehind

```
private void secim_Checked(object sender, RoutedEventArgs e)
    {
        RadioButton secim = (RadioButton)sender;
        if (secim.IsChecked==true)
        {
            MessageBox.Show(secim.Content.ToString());
        }
    }
```

TabControl

- Çalışma alanınızı daha verimli bir şekilde kullanmak için bu kontrolü tercih ediyoruz.
- TabControl, alan üzerinde istenilen sayı kadar sekme oluşturmamıza izin verir.
- Oluşturulan her sekme için yeni bir tasarım alanı oluşturulur.
- Selector sınıfında türemiştir.
- Item property'si kullanaraktan TabControl'e yeni TabItem'lar oluşturabiliriz.
- Her oluşturulan Tabitem Grid türünde Container içerir.
- Ayrıca TabControl'e sağ tıklayıp, Add Tab dediğimizde de yeni bir TabItem ekleyebiliriz.



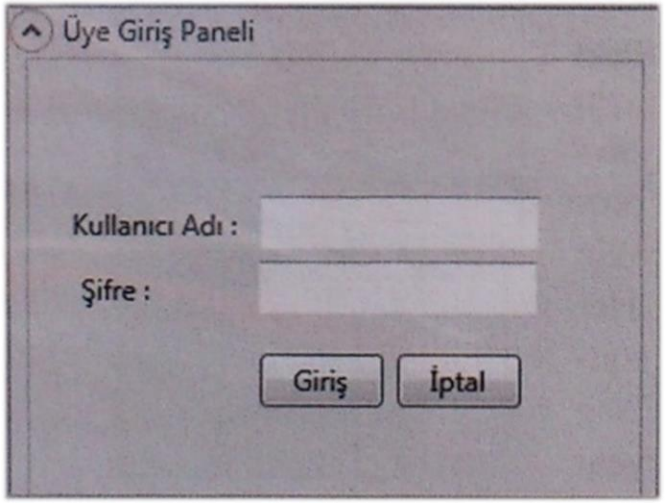
TabControl

Xaml

```
<Grid x:Name="window">
        <TabControl Height="182" HorizontalAlignment="Left"
Margin="21,110,0,0" Name="tabControl1" VerticalAlignment="Top" Width="393">
        <TabItem Header="tabItem1" Name="tabItem1">
            <Grid />
        </TabItem>
        <TabItem Header="tabItem2" Name="tabItem2">
            <Grid />
        </TabItem>
    </TabControl>
</Grid>
```

Expander

- Kullanıcı etkileşimli uygulamalar geliştirmek, kullanıcının uygulamayı etkin bir şekilde kullanmasından geçer. Bazen yaptığımız uygulamalarda bazı bilgilerin kullanıcı isteği doğrultusunda gösterilmesini isteyebiliriz. Bu gibi rollerde Expander kontrolüne ihtiyaç duyanz.
- Expander'e en güzel örnek class diyagram'de sınıfı temsil eden yapılardır.
- HeaderedContentControl sınıfından türemiştir.
- Expander içerisinde Grid bulundurur.
- Header property'si ile Expander için başlık tanımlaması yapabiliriz.
- isExpended property'si ile kontrolün başlangıçtaki içeriğini gösterir durumunda veya kapalı durumda olacağını belirleyebiliriz.
- ExpandDirection property'si ile açılır panelin yönünü belirtebiliriz



Expander

Xaml

```
<Expander Header="Üye Giriş Paneli" Height="213" HorizontalAlignment="Left"
Margin="35,39,0,0" Name="expander1" VerticalAlignment="Top" Width="281"
IsExpanded="False" ExpandDirection="Down" Background="#FFDCE5AA"
FontStyle="Normal" Foreground="#FFC10F0F">
            <Grid Height="189" Width="264">
                <TextBox Height="23" HorizontalAlignment="Left"
Margin="100,59,0,0" Name="textBox1" VerticalAlignment="Top" Width="120" />
                <TextBox Height="23" HorizontalAlignment="Left"
Margin="99,87,0,0" Name="textBox2" VerticalAlignment="Top" Width="120" />
                <Button Content="Giriş" Height="23"
HorizontalAlignment="Left" Margin="100,127,0,0" Name="button1"
VerticalAlignment="Top" Width="53" />
                 <Button Content="İptal" Height="23"
HorizontalAlignment="Left" Margin="159,127,0,0" Name="button2"
VerticalAlignment="Top" Width="53" />
                 <Label Content="Kullanıcı Adı :" Height="28"
HorizontalAlignment="Left" Margin="15,59,0,0" Name="label1"
VerticalAlignment="Top" />
                 <Label Content="Şifre :" Height="28"
HorizontalAlignment="Left" Margin="19,87,0,0" Name="label2"
VerticalAlignment="Top" />
             </Grid>
</Expander>
```

Frame

- HTML’den bildiğimiz bu kontrolü sayfa içerisinde sayfa açma gibi düşünebiliriz.
- Yani WPF tasarım alanı içerisinde bir web sitesini veya WPF’e ait Page’leri görüntüleyebiliriz.
- **ContentControl** sınıfında türemiştir.
- **Source** property’sine bir web sitesi veya Page türünden linkler verebiliriz.
- Ayrıca codebehind kısmında **Navigate** metodunu kullanaraktan aynı işlemleri gerçekleştirebiliriz.

Frame

Xaml

```
<Grid>
        <Frame Height="178" HorizontalAlignment="Left"
Margin="60,173,0,0" Name="frame1" VerticalAlignment="Top"
Width="361" Source="Page1.xaml" /
</Grid>
```

Frame

İstersek codebehind tarafından da Frame kontrollüne bir sayfayı bağlayabiliriz.

Codebehind

```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
frame1.Navigate(new Uri("/page1.xaml",UriKind.RelativeOrAbsolute));
}
```

ScroolBar

- Daha çok kaydırma çubuğu olarak bilinen **Scroll Bar'ın** en büyük özelliği **value** değerinin değişmesi ve bu değişime göre kullanıcının işlemler yapmasıdır.
- Bu **value** değerlerinin sınırlarını **Minimum** ve **Maximum** property'leri ile belirleyebiliriz.
- **Orientation** property'si ise yatay veya dikey konumlarını belirler.
- Ayrıca value değişikliğini yakalayan **ValueChanged** eventine sahiptir.

ScroolBar - Örnek

- Tasarım alanına üç tane ScrollBar'ı dikey olarak ekledikten sonra bir tane **Regtangle** ekliyoruz.
- Amaç; ScrollBar'lardan bir renk oluşumu için gerekli olan R, G, B değerlerini almak olacaktır.
- Red, Green, Blue değerlerinin 0-255 arasında değer aldığını unutmamak gerekir.
- Daha sonra her **ScrollBar'm ValueChanged** event'i ile istenilen değeri Color sınıfının metoduna aktararak renk oluşturma işlemini tamamlıyoruz

ScroolBar

Xaml

```
<Grid>
        <ScrollBar Height="238" HorizontalAlignment="Left"
Margin="42,74,0,0" Name="R" VerticalAlignment="Top" Width="18"
Maximum="255" ValueChanged="R_ValueChanged" SmallChange="10"
LargeChange="10" />
        <ScrollBar Height="238" HorizontalAlignment="Left"
Margin="66,74,0,0" Name="G" VerticalAlignment="Top" Width="18"
Maximum="255" ValueChanged="G_ValueChanged" SmallChange="10"
LargeChange="10" />
        <ScrollBar Height="238" HorizontalAlignment="Left"
Margin="90,74,0,0" Name="B" VerticalAlignment="Top" Width="18"
Maximum="255" ValueChanged="B_ValueChanged" SmallChange="10"
LargeChange="10" />
        <Rectangle Height="238" HorizontalAlignment="Left"
Margin="132,74,0,0" Name="rectangle1" Stroke="Black"
VerticalAlignment="Top" Width="160" Fill="#FFBE5858" />
        <Label Content="0" Height="28" HorizontalAlignment="Left"
Margin="42,318,0,0" Name="label1" VerticalAlignment="Top" />
```

ScroolBar

Xaml devamı

```
        <Label Content="0" Height="28" HorizontalAlignment="Left"
Margin="68,318,0,0" Name="label2" VerticalAlignment="Top" />
        <Label Content="0" Height="28" HorizontalAlignment="Left"
Margin="92,318,0,0" Name="label3" VerticalAlignment="Top" />
</Grid>
```

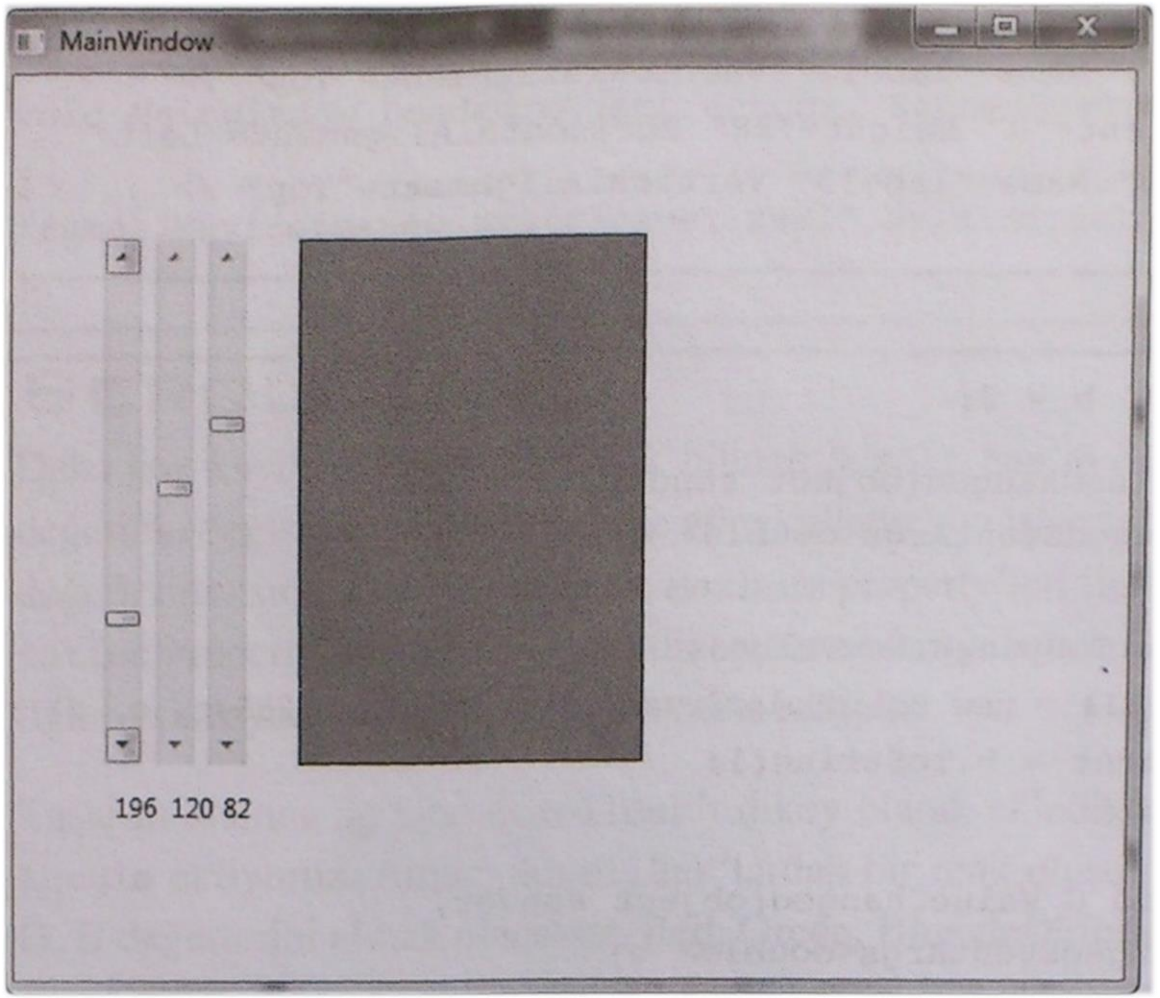
ScroolBar

Codebehind

```
byte r = 0, g = 0, b = 0;

private void B_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
        b = Convert.ToByte(e.NewValue);
        rectangle1.Fill = new SolidColorBrush(Color.FromArgb(255,r, g, b));
        label3.Content = b.ToString();
}

        private void G_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
        g = Convert.ToByte(e.NewValue);
        rectangle1.Fill = new SolidColorBrush(Color.FromArgb(255, r, g, b));
        label2.Content = g.ToString();
}

private void R_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
        r = Convert.ToByte(e.NewValue);
        rectangle1.Fill = new SolidColorBrush(Color.FromArgb(255,r, g, b));
        label1.Content = r.ToString();
}
```

ScroolBar

Sonuç



PasswordBox

- **Control** sınıfından türeyen sealed bir sınıftır.
- **TextBox** kontrolüne çok benzemektedir.
- Kullanıcı tarafından girilen veriyi herhangi bir **char** ile şifreleyebiliriz.
- Genellikle şifreli işlemlerde tercih edilmektedir

```
<PasswordBox Height="23" HorizontalAlignment="Left" Margin="136,133,0,0"
Name="passwordBox1" VerticalAlignment="Top" Width="120" PasswordChar="*" />
```

Menü

- Eski form uygulamalarında tamdık gelen bir kontroldür.
- Her uygulamada ihtiyaç **du**yulabilir.
- Uygulama ekranının üst kısmında tercih edilir.
- **Menü** içerisine **Menultem** türünde yapılar almaktadır.

Menü

Xaml

```
<Grid>
        <Menu Height="23" HorizontalAlignment="Left" Name="menu1"
VerticalAlignment="Top" Width="497">
            <MenuItem Header="Müşteri" >
                <MenuItem Header="Müşteri Bul" Click="MenuItem_Click" />
                <MenuItem Header="Müşteri İşlem" Click="MenuItem_Click_1" />
            </MenuItem>
            <MenuItem Header="Banka" >
                <MenuItem Header="Müşteri Bul" Click="MenuItem_Click_2"  />
                <MenuItem Header="Müşteri İşlem"  Click="MenuItem_Click_3"/>
            </MenuItem>
</Menu>
</Grid>
```

Popup

- İçerisine aldığı kontrolleri mini pencere ile Window üstünde gösterebilen kontroldür.
- Popup kontrolü layout'lardan bir kontrol alarak daha kolay düzenleme yapmamızı sağlar.
- isOpen property'si sayesinde popup açabilir veya kapatabiliriz

Popup

Xaml

```
<Grid Background="White">
        <Button Content="Üye Kayıt" Height="23" HorizontalAlignment="Left"
Margin="0,12,0,0" Name="button1" VerticalAlignment="Top" Width="75"
Click="button1_Click" />
        <Button Content="Üye Giriş" Height="23" HorizontalAlignment="Left"
Margin="96,12,0,0" Name="button2" VerticalAlignment="Top" Width="75" />

        <Popup HorizontalAlignment="Left" Margin="73,143,0,0"
Name="popup1" VerticalAlignment="Top" Height="194" Width="334" />

            <StackPanel Background="#FFDEAEAE">
                <TextBox Height="23" HorizontalAlignment="Left"
Margin="148,320,0,0" Name="textBox2" VerticalAlignment="Top" Width="120" />
                <TextBox Height="23" HorizontalAlignment="Left"
Margin="148,349,0,0" Name="textBox3" VerticalAlignment="Top" Width="120" />
                <TextBox Height="23" HorizontalAlignment="Left"
Margin="148,291,0,0" Name="textBox1" VerticalAlignment="Top" Width="120" />

                <Button Content="Kaydet" Click="button3_Click"
Height="23" HorizontalAlignment="Left" Margin="168,386,0,0" Name="button3"
VerticalAlignment="Top" Width="75" />

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

Popup

Codebehind

```
private void button1_Click(object sender, RoutedEventArgs e)
{
        popup1.IsOpen = true;
}

private void button3_Click(object sender, RoutedEventArgs e)
{
   popup1.IsOpen = false;
   }
```

Media Element

- FrameworkElement sınıfında türeyen Media Element kontrolü uygulamalarda video ve ses dosyalarınızı kullanmak için kullanılır.

```
<MediaElement Height="120" Margin="137,111,218,0"
Name="mediaElement1" VerticalAlignment="Top" Source="Wildlife.wmv" />
```

Play, Pause, Stop metodlan ile media kontrolünü rahatlıkla sağlayabiliriz.

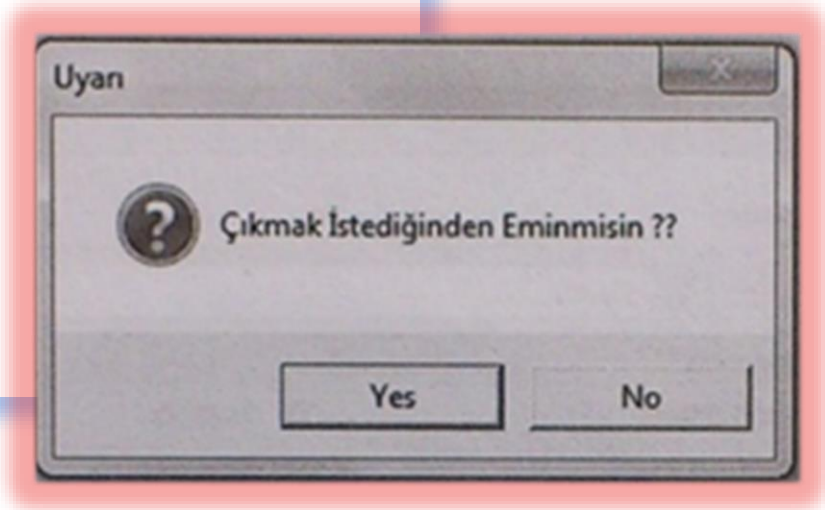
```
mediaElement1.Play();
mediaElement1.Stop();
mediaElement1.Pause();
```

Diyalog Penceresi

- Uygulamalardan geri dönüşü olmayan işlemler öncesi kullanıcı onayı almak için kullanılan yapıdır.
- Bildiğiniz üzere **MessageBox** sınıfı ile uyarı kutucukları oluşturabiliyorduk. Fakat uyarı sonucunda kullanıcının onayını alma işlemi gerçekleştirilmiyordu.
- Bu işlemi **MessageBox** sınıfın **Show** metoduna parametre yollayarak sağlıyacağız.
- Kullanıcının seçimini ise **MessageBoxResult** sınıfı ile karşılayabiliriz.

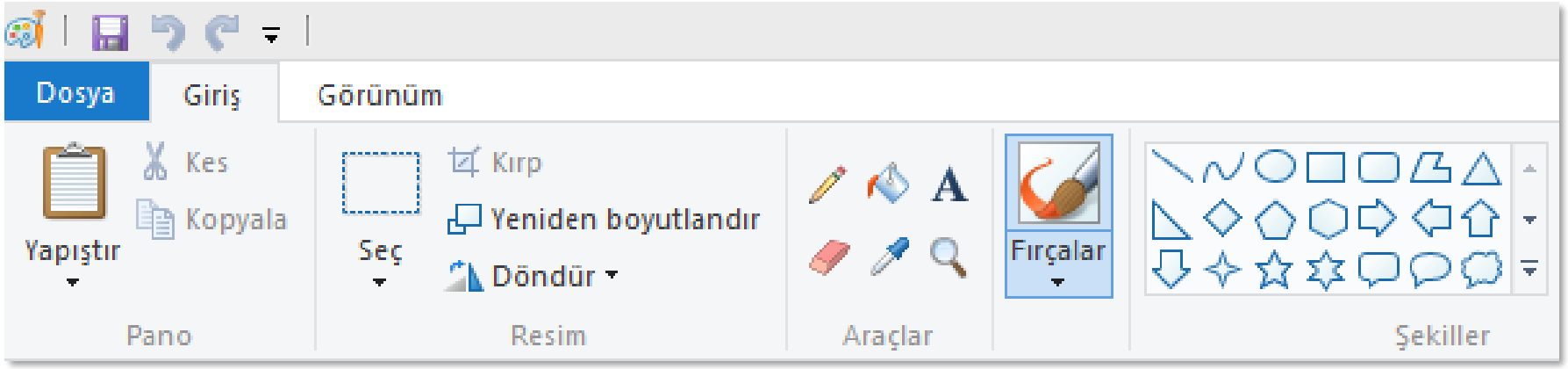
Diyalog Penceresi

```
void MainWindow_Closing(object sender,CancelEventArgs e)
{
MessageBoxResult sonuc
          = MessageBox.Show("Çıkmak İstediğinden Emin misin?",
                            "Uyarı",
                            MessageBoxButton.YesNo,
                            MessageBoxImage.Question);

              if (sonuc == MessageBoxResult.No)
              {
                   e.Cancel = true;
              }
}
```



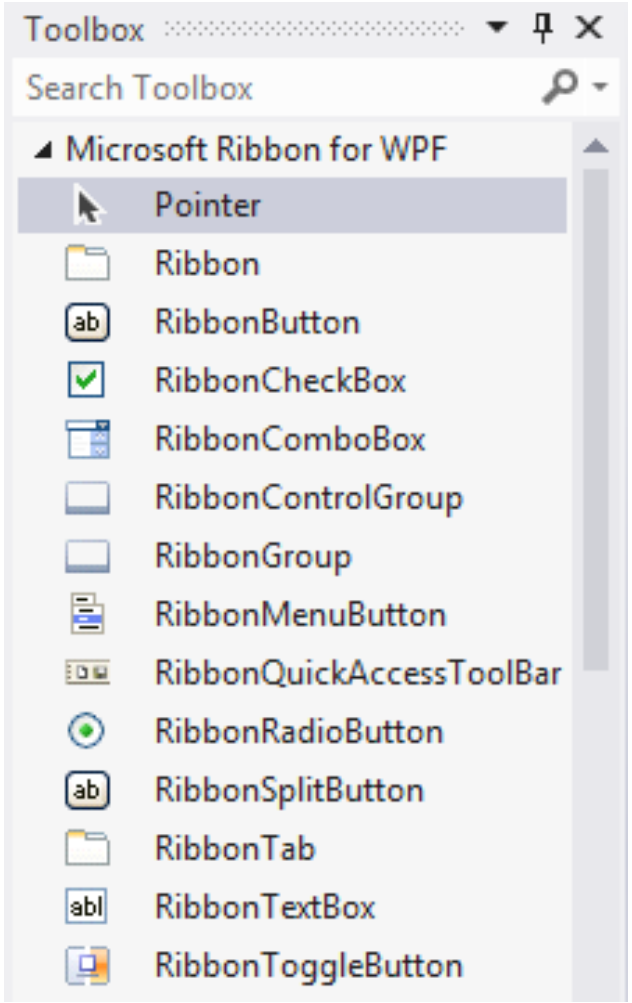
Ribbon Kontrolleri

- Uygulamaların sürekli kullanıcı etkileşiminin yüksek olması gerektiğini söylemiştik. Bu noktada Ribbon kontrolleri gelmektedir.
- Ribbon kontrolleri ilk olarak Office 2007 ile beraber gelmiştir.
- Office 2007, 2010 ve Paint vb. yazılımlarındaki ana menüler ribbon kontrollerinden oluşmuştur



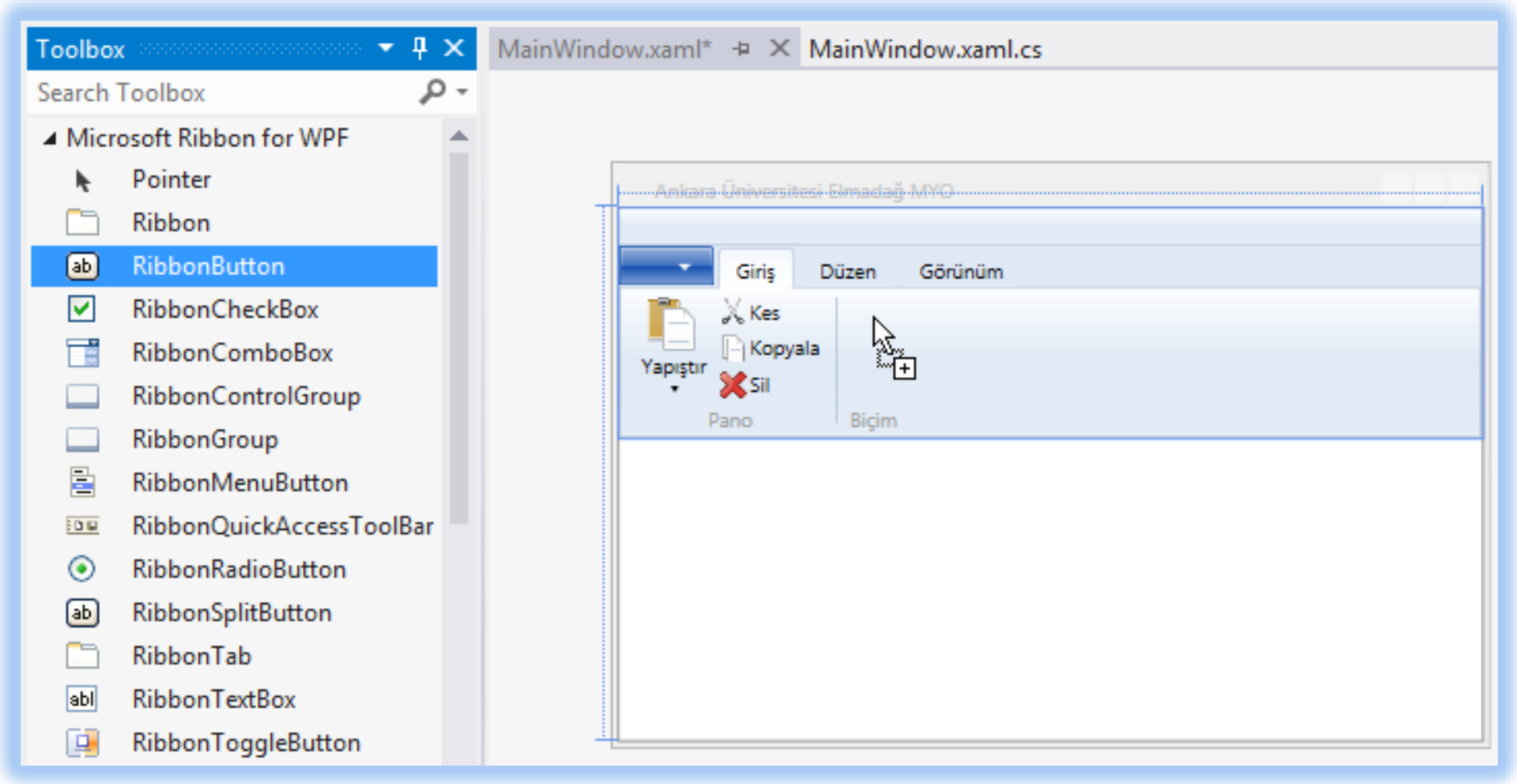
Ribbon Kontrolleri

- Ribbon Kontrolleri WPF'te kullanmak istiyorsak öncelikle RibbonControlsLibrary.dll isimli kütüphaneyi indirmek gerekmektedir.
- Microsoft.Windows.Controls.Ribbon namespace'ini uygulamalara referans verdikten sonra Toolbox penceresine Microsoft Ribbon for WPF sekmesi eklenecektir.
- Ribbon kontrolleri WPF kontrolleri ile aynı işlevselliğe sahiptir. Tek farklılık Ribbon kontrollerinin Ribbon isimli kontrol içerisinde bulunmalarıdır.



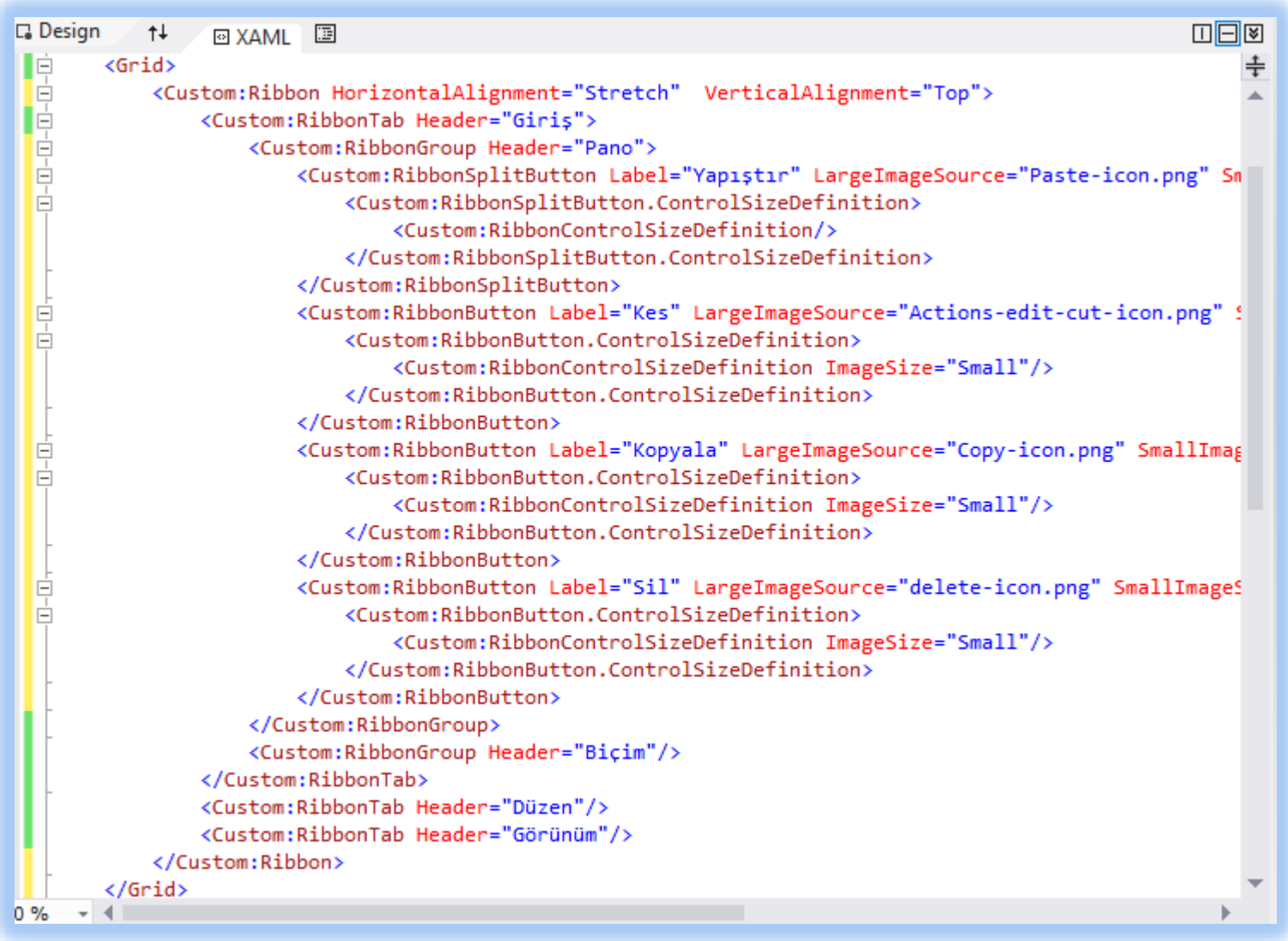
Ribbon Kontrolleri

Tasarım:



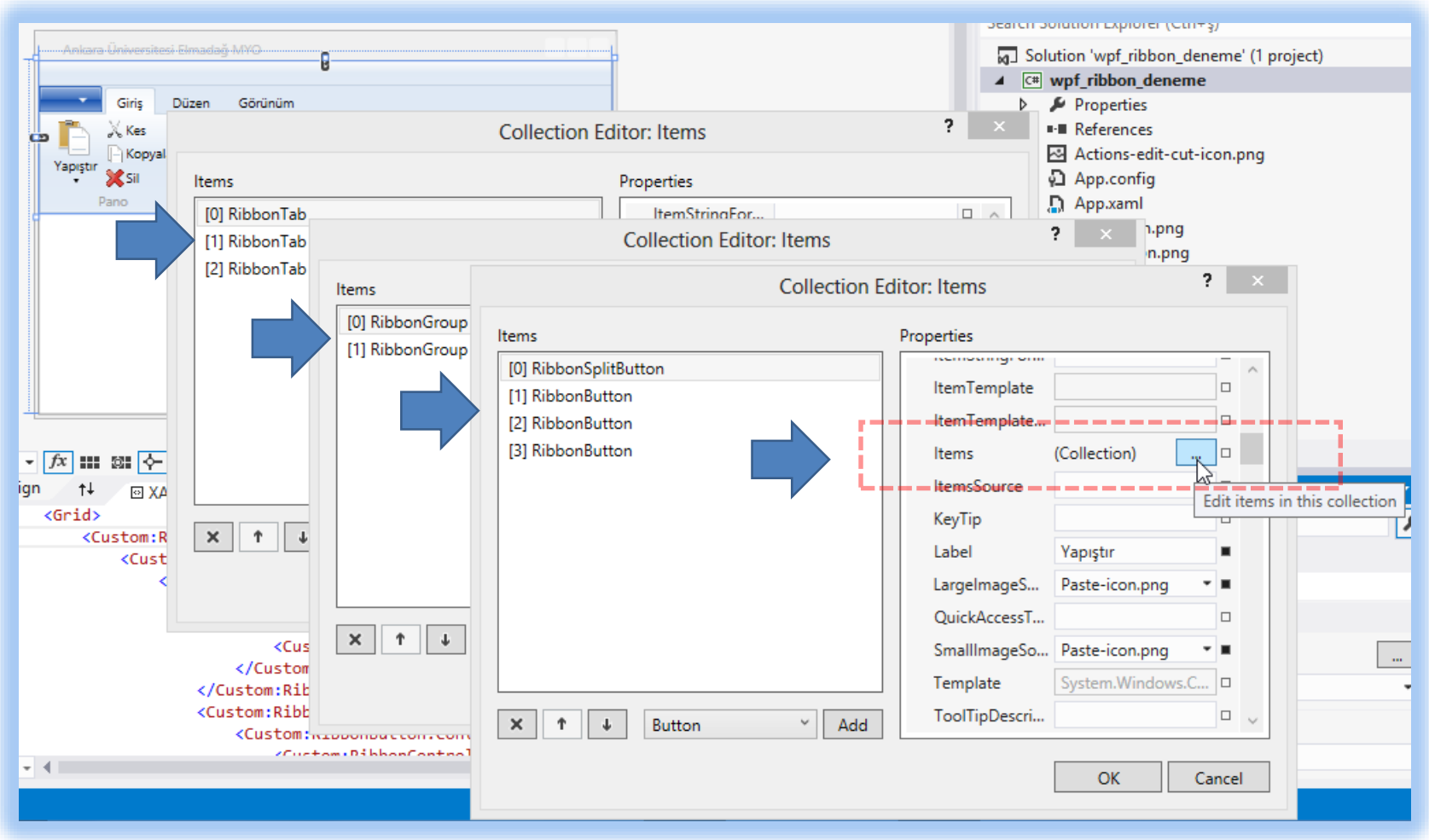
Ribbon Kontrolleri

Xaml:

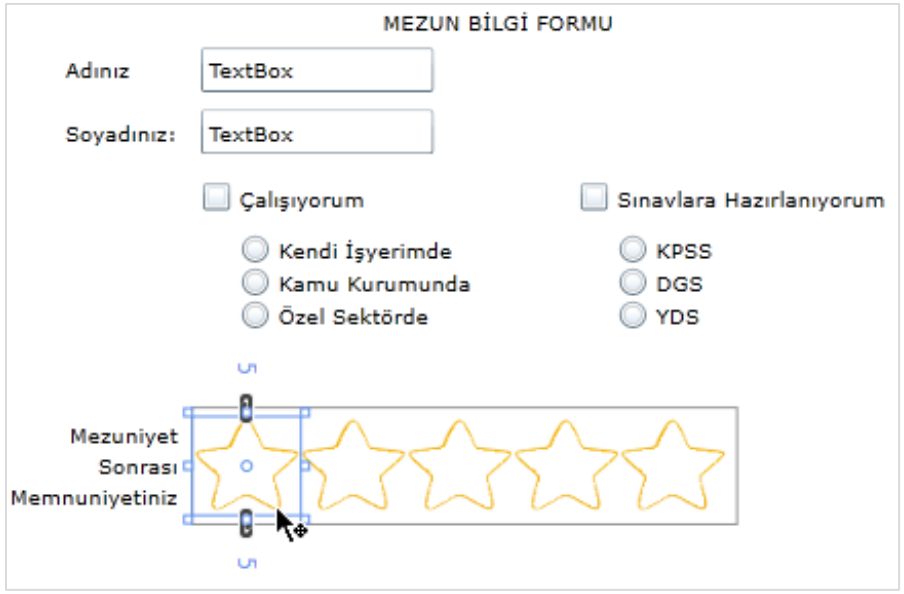


Ribbon Kontrolleri

- Ribbon kontrolünün items property'si ile Ribbon sekmeleri oluşturabiliriz.

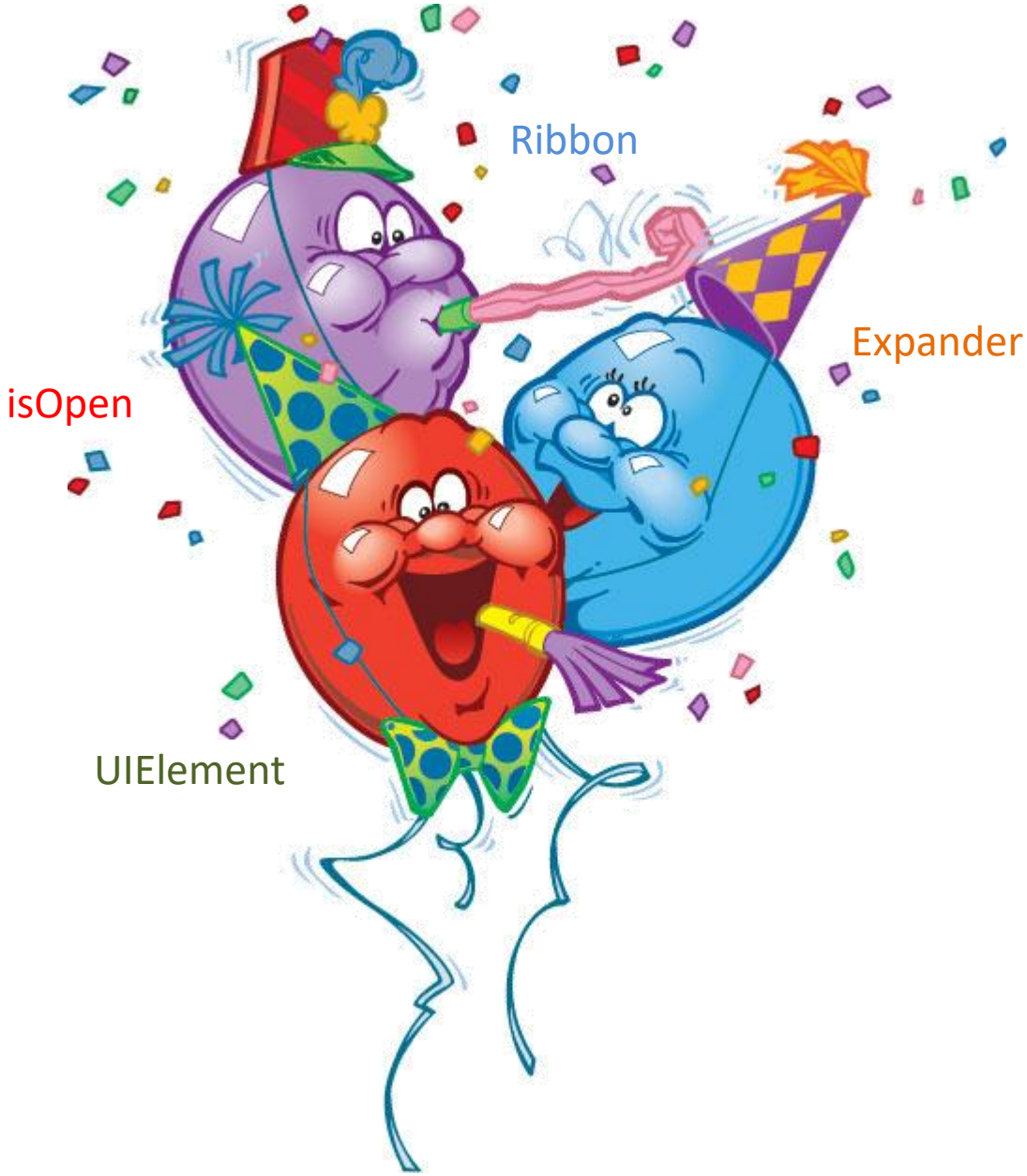


Uygulama Ödevi



WPF, XBAP, SilverLight ?

Havada Kalmasın ☺



Teşekkürler…

bize katlandığınız için!