Düzen Panelleri Ve Nesne Kompozisyonları*

Öğr.Gör.Erkan HÜRNALI

*Ahmet Ali SÜZEN'in WPF kitabından derlenmiştir.

T.C. ANKARA ÜNİVERSİTESİ
MYO
elmadağ meslek yüksekokulu

Düzen Panelleri (Layout Controls)

- Windows form uygulamalarından alışık olduğumuz sabit piksel koordinatları, WPF ile son buldu.
- WPF uygulamaları farklı çözünürlüklerde çalışsalar bile görüntü kaybı olmamaktadır. Bu yüzden tasarım yapılarının daha gelişmiş olması gerekmektedir. Bu noktada Layout (*yerleşim*) kontrolleri gelmektedir.
- Bunları form uygulamalarında bulunan paneller olarak bilmekteyiz.
- Tabi WPF'de bu kontroller daha zengin bir şekilde kullanılacaktır

Düzen Panelleri (Layout Controls)

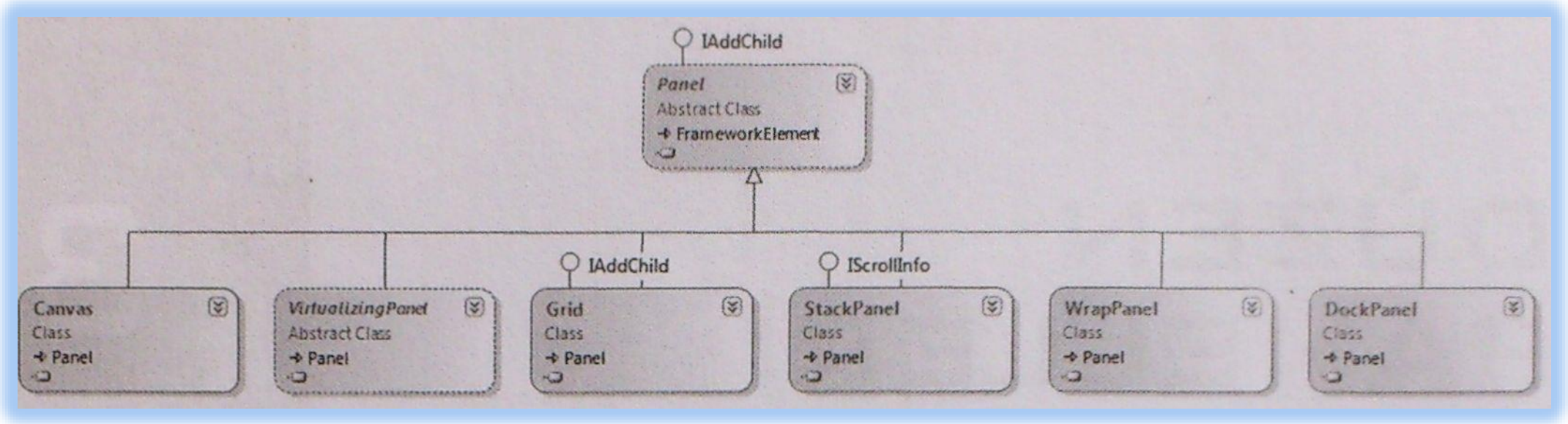
- Layout kontrollerinin ortak özelliği Container yapısında olmalarıdır.
- Yani içerisine Children türünde UIElemet'ler alabiliyor olmalarıdır.
- Layout kontrollerinin kendine has bir şekilde kontrol gösterme ve yerleştirme özelliği vardır.
- WPF'de her element veya kontrol mutlaka bir Layout içerisinde bulunmak zorundadır.

Düzen Panelleri (Layout Controls)

- Layout kontrolleri içerisinde kullanılacak kontrollere sabit koordinat veya sabit büyüklük vermekten kaçınmamız gerekiyor.
- Ayrıca bu kontrolleri işlevine göre kullanmak uygulama kalitesini artıracaktır.
- Örneğin; Canvas isimli layout'u kullanacaksanız bu kontrol içerisinde vektörel grafikler veya hareketli yapılar oluşturmanız daha doğru olacaktır.

Düzen Panelleri (Layout Controls)

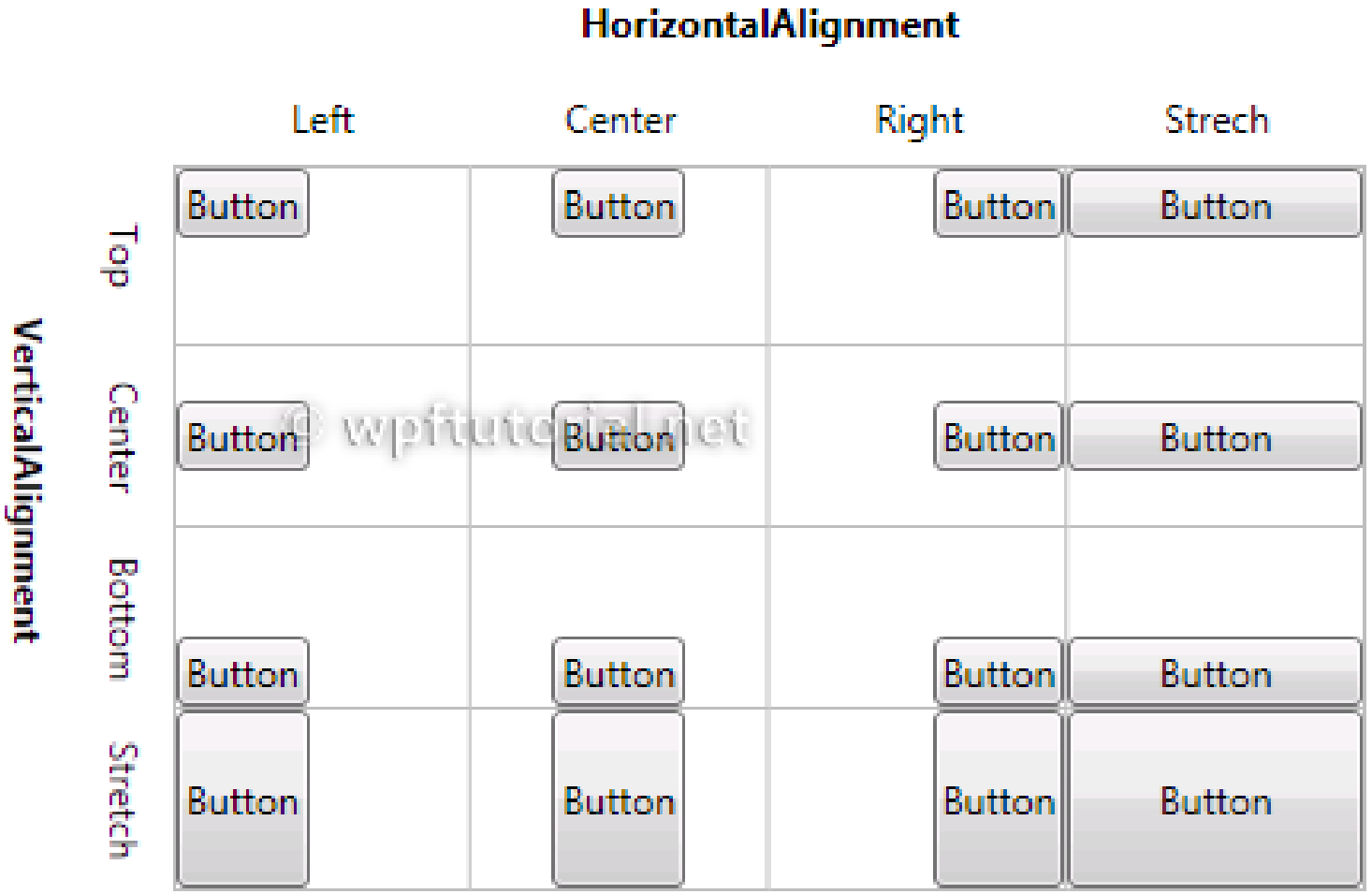
- Layout kontrollerini aslında birer panel olarak düşünebiliriz.
- Zaten Layout kontrolleri Panel isimli soyut sınıftan türemişlerdir.
- Layout kontrollerini **System.**win**dows.**Controls uzayının altında bulabilirsiniz.



Yatay ve Düşey Hizalama

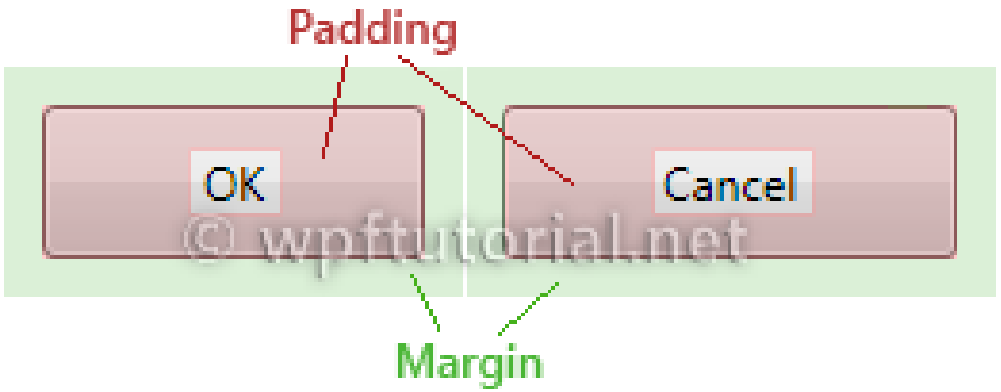
- WPF uygulamalarında her kontrolün bir Container içerisinde olması gerektiğini söylemiştik.
- Kontroller, içerisinde bulundukları panelin konumuna göre kendini tanımlar.
- Her kontrolde **HorizontalAlignment** ve **VerticalAlignment** property'leri mevcuttur.
- Bu property'ler sayesinde içinde bulunduğu panele hizalama işlemi gerçekleşir.
- **HorizontalAlignment** yatayda **Left, Right, Center** ve **Stretch** enum'ları ile hizalama gerçekleştirir.
- **VerticalAlignment** dikeyde **Top, Bottom, Çenter** ve **Stretch** enum'ları ile hizalama gerçekleştirir.

Yatay ve Düşey Hizalama



Margin ve Padding

- **Margin** property'si, kontrolün içinde bulunduğu panel ile arasındaki boşluğu ayarlamamız için kullanılır.
- **Padding** property'si ise kontrolün, kendisine ait content ile arasındaki mesafeyi düzenler.



UniformGrid

- Layout kontrollerine bakıldığı zaman belki de en basit kullanıma sahip olan container'dır.
- Normal bir Grid kontrolünde hücre, satır ve sütunlar mevcuttur.
- UniformGrid kontrolünde de yine hücre, satır ve sütun vardır. Ama bu yapılar **Container'in** içerisine aldığı kontrollere göre tanımlanmaktadır.
- Kısaca her eklenen kontrol için hücre, satır ve sütun oluşmaktadır.

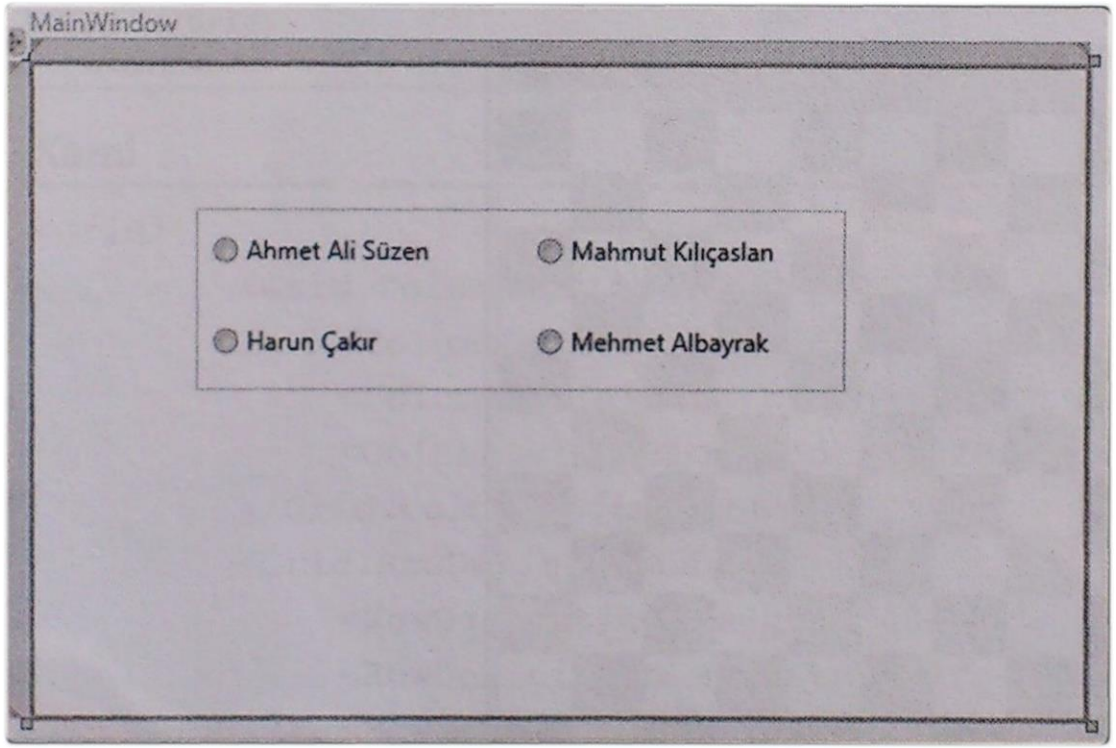
UniformGrid

Xaml:

```
<UniformGrid Height="86" HorizontalAlignment="Left" Margin="79,69,0,0"
Name="uniformGrid1" VerticalAlignment="Top" Width="309">
            <RadioButton Content="Ahmet Ali Süzen" Height="19"
Name="radioButton1" Width="137" />
            <RadioButton Content="Mahmut Kılıçaslan" Height="19"
Name="radioButton6" Width="137" />
            <RadioButton Content="Harun Çakır" Height="19"
Name="radioButton8" Width="137" />
            <RadioButton Content="Mehmet Albayrak" Height="19"
Name="radioButton9" Width="137" />
</UniformGrid>
```

UniformGrid

Ekran Çıktısı:



UniformGrid – Örnek Uygulama

Basit bir şekilde codebehid tarafında UniformGrid kullanarak satranç tahtası görünümünü elde edelim.

CodeBehind:

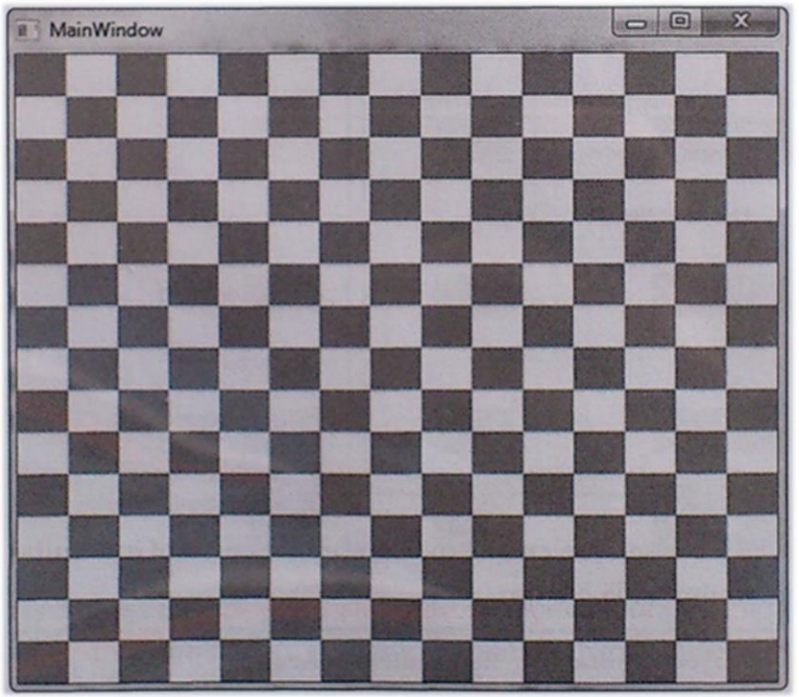
```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
        {
            System.Windows.Controls.Primitives.UniformGrid cerceve =
                new System.Windows.Controls.Primitives.UniformGrid();
            cerceve.Rows = 15;
            cerceve.Columns = 15;
            for (int i = 0; i < 225; i++)
```

UniformGrid – Örnek Uygulama

CodeBehind:

```
    for (int i = 0; i < 225; i++)
    {
        Rectangle rt = new Rectangle();
        rt.Height = 50;
        rt.Width = 50;
        if (i%2==0)
        {
            rt.Fill = new SolidColorBrush(Colors.Red);
        }
        else
        {
            rt.Fill = new SolidColorBrush(Colors.Yellow);
        }
        cerceve.Children.Add(rt);
    }
    this.alan.Children.Add(cerceve);
}
```

Ekran Çıktısı:



Grid

- UniformGrid kontrolünde içerisine kontrol eklendikten sonra oluşturulan hücre, satır ve sütun; Grid kontrolünde geliştirici aracılığıyla oluşturulmaktadır.
- **Grid** kontrolü, **UniformGrid** kontrolüne göre daha gelişmiş bir yapıdadır.
- RowDefinitions property'si satır oluşturmamızı sağlar.
- ColumnDefinitions property'si sütun oluşturmamızı sağlar.
- ShowGridLines property'si ile tanımlanan satır ve sütunları ekranda görebiliriz.

Grid

- Yeni açılan **WPF** uygulamalarının default olarak gelen layout kontrolü Grid'tir.

Design XAML

```
<Window x:Class="WpfApplication3.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>
```

Grid

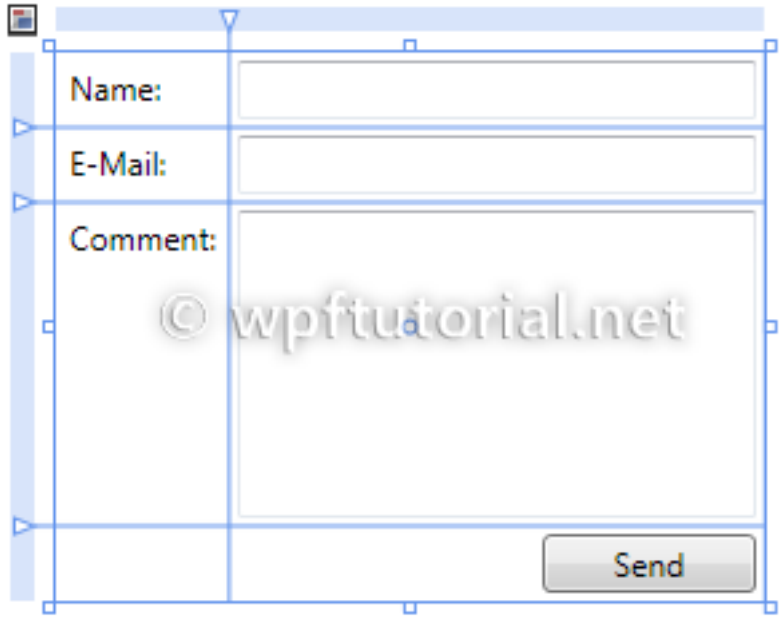
Xaml:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="28" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="200" />
    </Grid.ColumnDefinitions>
</Grid>
```

- Yukarıdaki XAML kodunda görüldüğü üzere Grid içerisine satır ve sütunları oluşturduk.

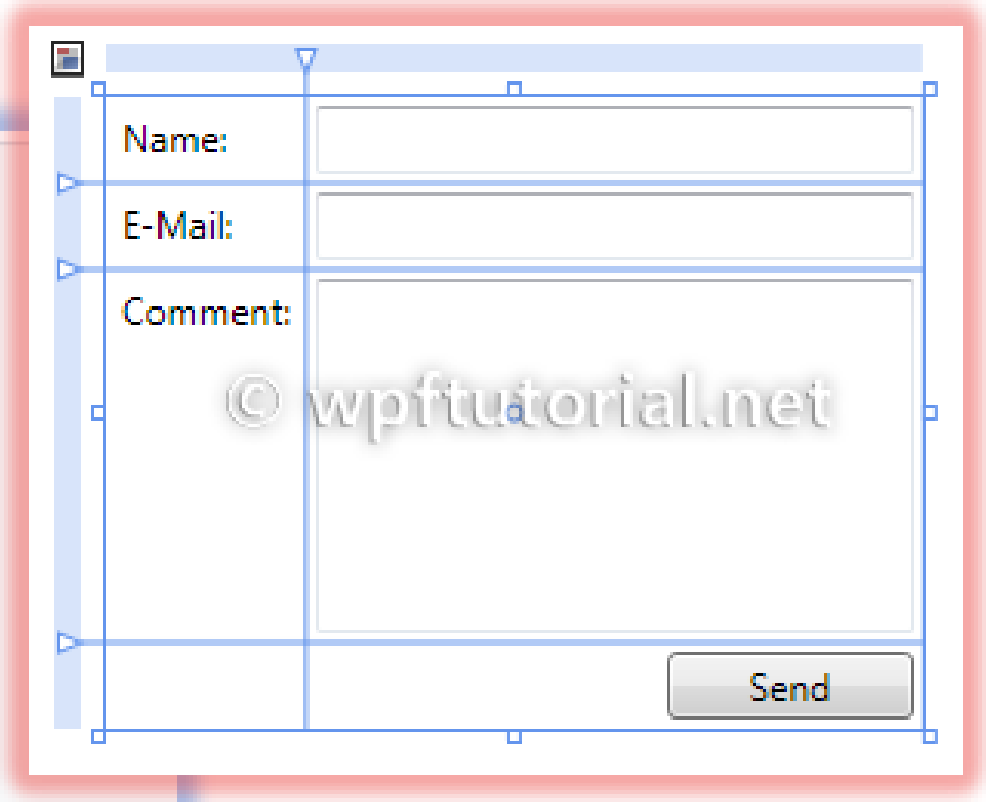
Grid

- Burada unutulmaması gereken bilgi, her oluşan satır ve sütunun indeksinin 0'dan başlamasıdır.
- Grid'i incelediğimizde **Auto** ve * kavramlarını görmekteyiz.
- **Auto;** Grid içerisindeki hücrelerin büyüklüğü, içerisindeki kontrolün genişliği ve yüksekliği ile doğru orantılı olarak ayarlanması modudur.
- * işaretinin işlevi ise büyüklüğü belirtilmiş hücreler dışında kalan tüm alanları kullanmasıdır



Grid

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="28" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="200" />
    </Grid.ColumnDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Content="Name:"/>
    <Label Grid.Row="1" Grid.Column="0" Content="E-Mail:"/>
    <Label Grid.Row="2" Grid.Column="0" Content="Comment:"/>
    <TextBox Grid.Column="1" Grid.Row="0" Margin="3" />
    <TextBox Grid.Column="1" Grid.Row="1" Margin="3" />
    <TextBox Grid.Column="1" Grid.Row="2" Margin="3" />
    <Button Grid.Column="1" Grid.Row="3" HorizontalAlignment="Right"
            MinWidth="80" Margin="3" Content="Send"  />
</Grid>
```



Grid

Codebehid tarafından programatik olarak da Grid içerisine satır ve sütun oluşturabiliriz

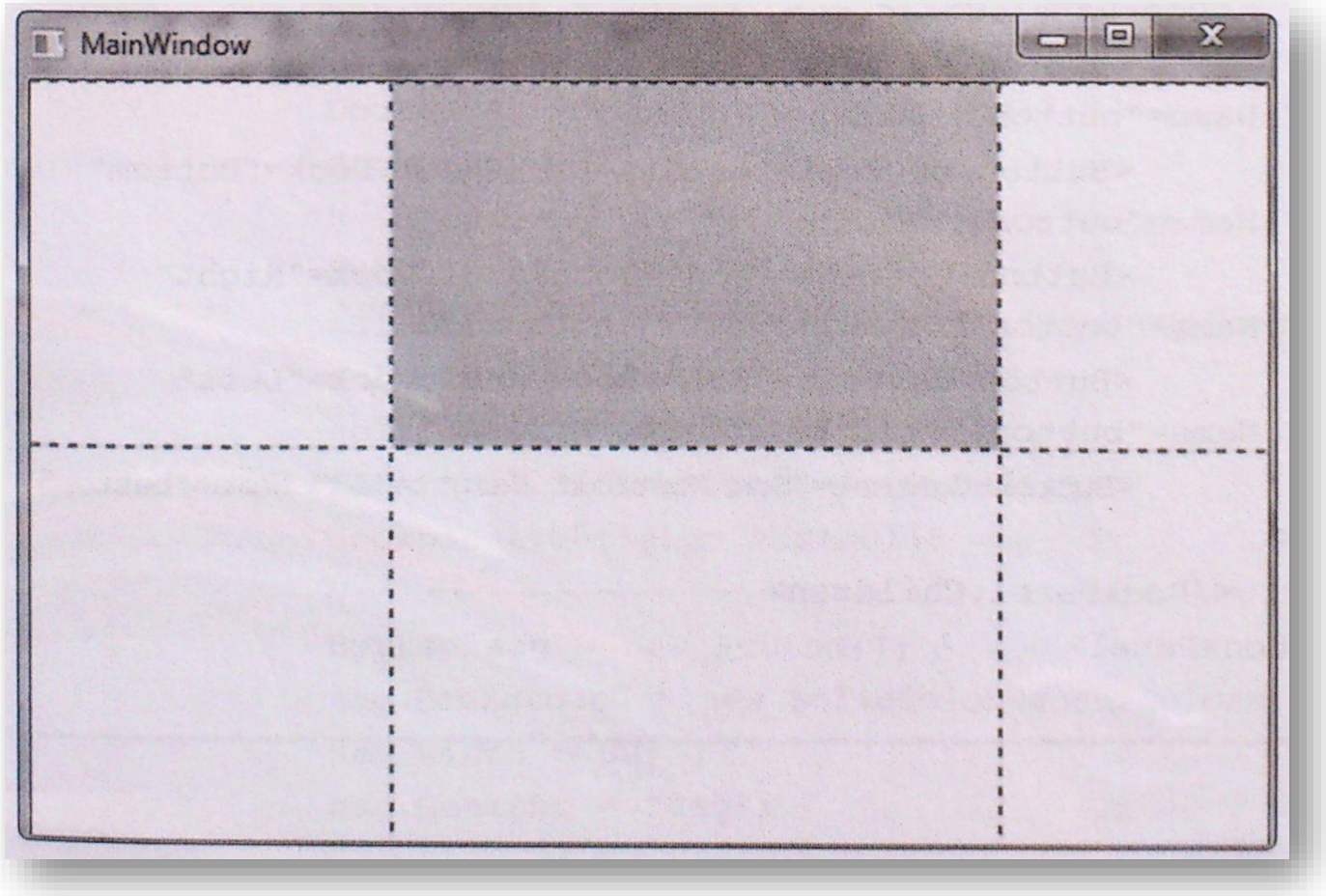
```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
        {
            Grid grid = new Grid();

            //Grid satırlarını oluşturuyoruz.

            RowDefinition r = new RowDefinition();
            r.Height = new GridLength(100, GridUnitType.Auto);
            grid.RowDefinitions.Add(r);

            r = new RowDefinition();
            r.Height = new GridLength(150, GridUnitType.Pixel);
            grid.RowDefinitions.Add(r);

            r = new RowDefinition();
            r.Height = new GridLength(200, GridUnitType.Star);
            grid.RowDefinitions.Add(r);

            //Grid sütunlarını oluşturuyoruz.
```

Grid

```
        ColumnDefinition c = new ColumnDefinition();
        c.Width = new GridLength(150, GridUnitType.Pixel);
        grid.ColumnDefinitions.Add(c);

        c = new ColumnDefinition();
        c.Width = new GridLength(250, GridUnitType.Pixel);
        grid.ColumnDefinitions.Add(c);

        c = new ColumnDefinition();
        c.Width = new GridLength(50, GridUnitType.Pixel);
        grid.ColumnDefinitions.Add(c);

        grid.ShowGridLines = true;
        this.Content = grid;

        // Grid içerisine yerleştireceğimiz kontrolü tanımlıyoruz.

        Rectangle rec = new Rectangle();
        rec.Fill = new SolidColorBrush(Colors.Pink);
        rec.HorizontalAlignment = HorizontalAlignment.Stretch;
        grid.Children.Add(rec);

        //Kontrolün grid'in hangi satır ve sütunlarına yerleştirileceğini belirliyoruz.

        Grid.SetRow(rec, 1);
        Grid.SetColumn(rec, 1);
    }
```

Grid

Ekran Çıktısı:



Grid Uzunluklarının Kod Yardımıyla Ayarlanması

CodeBehind:

Auto sized	`GridLength.Auto`
Star sized	`new GridLength(1,GridUnitType.Star)`
Fixed size	`new GridLength(100,GridUnitType.Pixel)`

```
Grid grid = new Grid();

ColumnDefinition col1 = new ColumnDefinition();
col1.Width = GridLength.Auto;
ColumnDefinition col2 = new ColumnDefinition();
col2.Width = new GridLength(1,GridUnitType.Star);

grid.ColumnDefinitions.Add(col1);
grid.ColumnDefinitions.Add(col2);
```

DockPanel

- Geliştiricilerin form uygulamalarından bildiği Dock property'si WPF'de karşımıza bir panel yapısında çıkmaktadır.
- Bildiğiniz üzere form uygulamalarında görsel her kontrolün Dock property'si vardır.
- Fakat WPF uygulamalarında kontrollerin Dock property'si yoktur.
- DockPanel elementleri veya kontrolleri yapısı içerisinde **Left, Right, Top, Bottom** gibi yaslayabilme özelliklerine sahiptir.
- DockPanel içerisindeki element veya kontrollerin **DockPanel.Dock** attached property'si vardır.
- Bu property sayesinde DockPanel içerisindeki konumlar tanımlanır.

DockPanel

- DockPanel içerisindeki son element veya kontrolün **DockPanel.Dock** property'si yoktur.
- Bu kontrol veya element için DockPanel'in **LastchildFill** property'si vardır.
- Bu property **true** olduğu zaman son kontrol panelin ortasını kaplayacaktır.
- **False** olması durumunda **DockPanel** default özellikleri tanımlanacaktır.
- Yani son kontrol sola dayalı şekilde konumlandınlacaktır

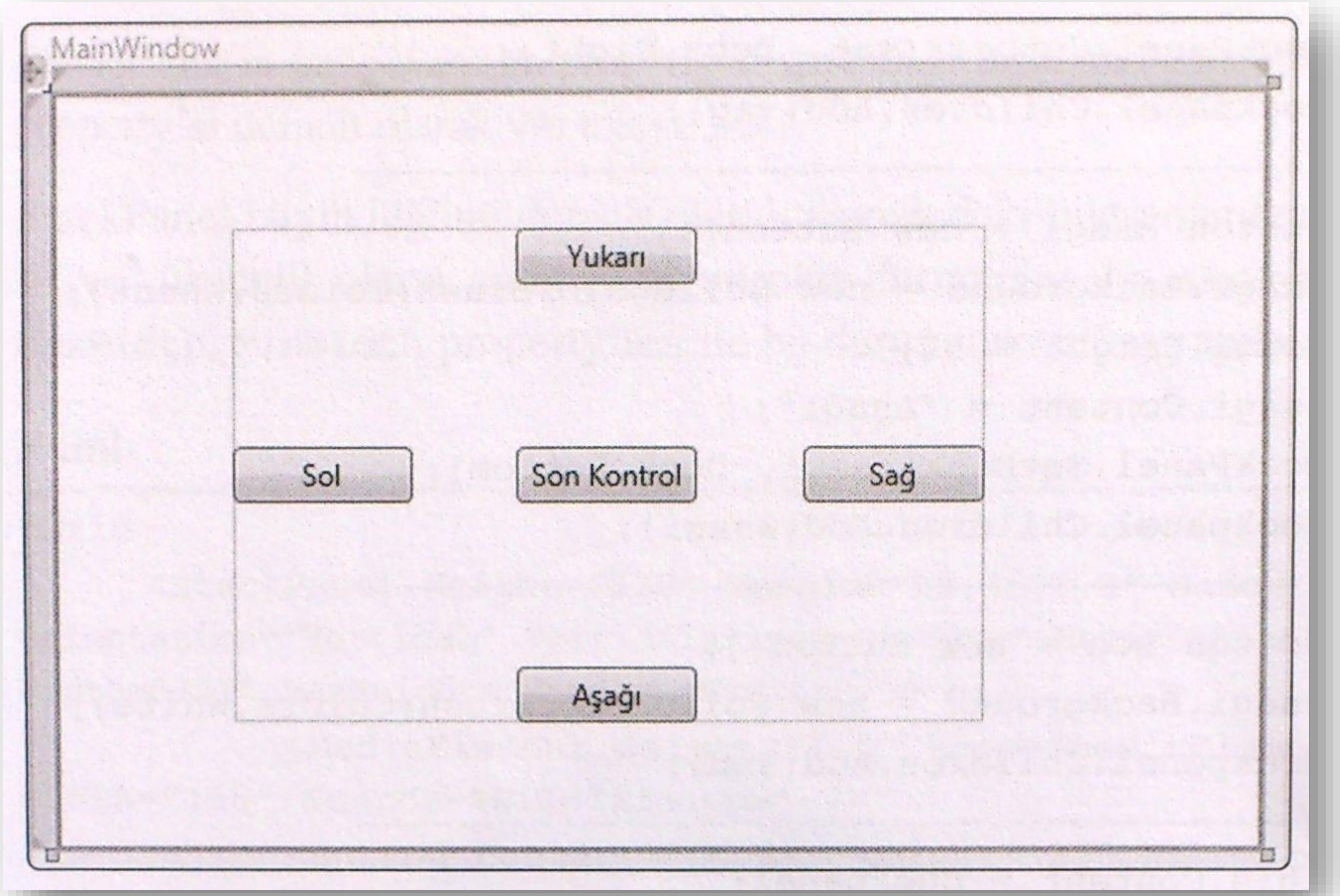
DockPanel

Xaml:

```
<Grid>
        <DockPanel Height="202" LastChildFill="True"
HorizontalAlignment="Left" Margin="75,56,0,0" Name="dockPanel1"
VerticalAlignment="Top" Width="312">
            <DockPanel.Children>
                <Button Content="Yukarı" DockPanel.Dock="Top"
Height="23" Name="button1" Width="75" />
                <Button Content="Aşağı" DockPanel.Dock="Bottom"
Height="23" Name="button2" Width="75" />
                <Button Content="Sağ" DockPanel.Dock="Right"
Height="23" Name="button3" Width="75" />
                <Button Content="Sol" DockPanel.Dock="Left"
Height="23" Name="button4" Width="75" />
                <Button Content="Son Kontrol" Height="23" Name="button5"
Width="75" />
            </DockPanel.Children>
        </DockPanel>
</Grid>
```

DockPanel

Ekran Çıktısı:



DockPanel

Ayrıca programatik olarak DockPanel kullanıldığı zaman kontrolleri veya elementleri SetDock metodu ile istenilen köşede konumlandırabiliriz

```
private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            DockPanel dockpanel = new DockPanel();
            //-----------------------------------------------------------------
            Button ust = new Button();
            ust.Background = new SolidColorBrush(Colors.Red);
            ust.Height = 50;
            ust.Content = "Üst";
            DockPanel.SetDock(ust, Dock.Top);
            dockpanel.Children.Add(ust);
            //-----------------------------------------------------------------
            Button sol = new Button();
            sol.Background = new SolidColorBrush(Colors.Yellow);
            sol.Width = 50;
            sol.Content = "Sol";
            DockPanel.SetDock(sol, Dock.Left);
            dockpanel.Children.Add(sol);
            //-----------------------------------------------------------------
            Button sag = new Button();
            sag.Background = new SolidColorBrush(Colors.AliceBlue);
            sag.Width = 50;
            sag.Content = "Sağ";
```

DockPanel

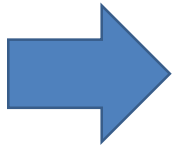
CodeBehind:

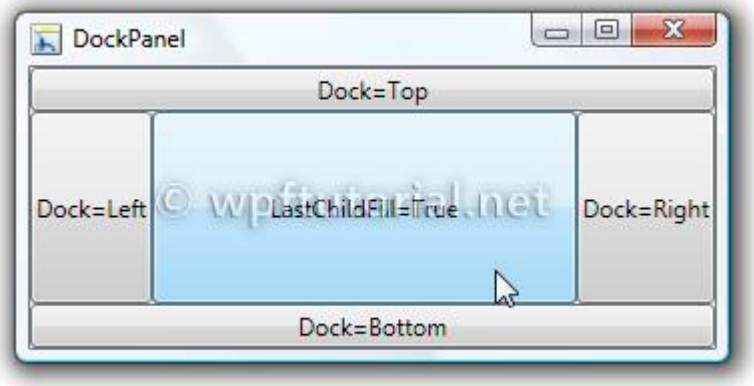
```
        DockPanel.SetDock(sag, Dock.Right);
        dockpanel.Children.Add(sag);
        //-------------------------------------------------------------
        Button asagi = new Button();
        asagi.Background = new SolidColorBrush(Colors.Wheat);
        asagi.Height = 50;
        asagi.Content = "Aşağı";
        DockPanel.SetDock(asagi, Dock.Bottom);
        dockpanel.Children.Add(asagi);
        //-------------------------------------------------------------
        Button son = new Button();
        asagi.Background = new SolidColorBrush(Colors.White);
        dockpanel.Children.Add(son);
        //-------------------------------------------------------------
        this.Content = dockpanel;
}
```

DockPanel - Örnekler

Xaml:

```
<DockPanel LastChildFill="True">
    <Button Content="Dock=Top" DockPanel.Dock="Top"/>
    <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>
    <Button Content="Dock=Left"/>
    <Button Content="Dock=Right" DockPanel.Dock="Right"/>
    <Button Content="LastChildFill=True"/>
</DockPanel>
```





DockPanel - Örnekler

Xaml:

```
<DockPanel LastChildFill="True">
    <Button Content="Dock=Left"/>
    <Button Content="Dock=Left"/>
    <Button Content="Dock=Top" DockPanel.Dock="Top"/>
    <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>
    <Button Content="Dock=Right" DockPanel.Dock="Right"/>
    <Button Content="LastChildFill=True"/>
</DockPanel>
```

StackPanel

- StackPanel, form uygulamalanndan bildiğimiz **FlowLayoutPanel** kontrolüne çok benzemektedir.
- İçerisine aldığı child elementlerini veya kontrolleri yatay veya dikey olarak hizalandırabilir.
- WPF'de her kontrolün yatay ve dikey hizalama özellikleri mevcuttur.
- Buna StaekPanel'inde dahil olduğunu düşünürsek, StackPanel'i Window'a göre de hizalayabiliriz.

StackPanel

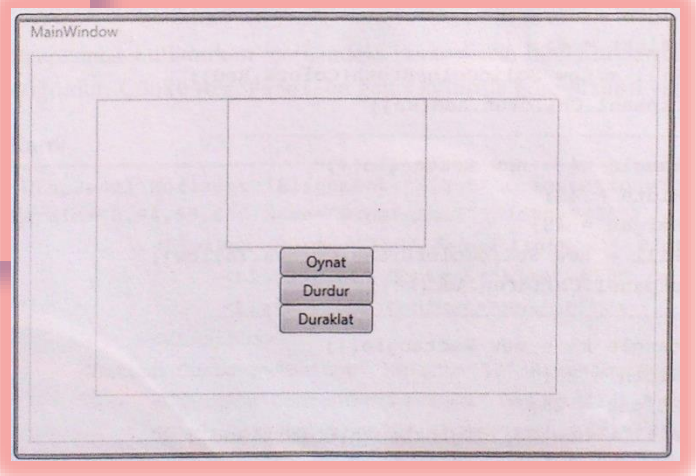
- StackPanel'in **Orientation** property'si sayesinde hizalama yapabiliriz.
- **Orientation** property'si default olarak **Vertical** gelir.
- StackPanel büyüklüğünü default olarak içerisindeki bileşenlerden alır.
- Yani genişlik ve yükseklik olarak en büyük kontrolün durumunu alır.
- **MaxHeight, MinHeight, MaxWidth, MinWidth** property'leri ile bu durum tekrardan yapılandırılabilir

StackPanel

Xaml:

```
<Grid>
      <StackPanel Height="238" Margin="53,36,0,0" Name="stackPanel1"
Orientation="Vertical" VerticalAlignment="Top" HorizontalAlignment="Left"
Width="371" MaxHeight="750" MaxWidth="750">
            <MediaElement Height="120" Name="mediaElement1"
Width="160" Source="Wildlife.wmv" />
            <Button Content="Oynat" Height="23" Name="button1"
Width="75" HorizontalAlignment="Center" />
            <Button Content="Durdur" Height="23" Name="button2"
Width="75" HorizontalAlignment="Center"  />
            <Button Content="Duraklat" Height="23" Name="button3"
Width="75" HorizontalAlignment="Center"  />
      </StackPanel>
</Grid>
```

Ekran Çıktısı:



StackPanel

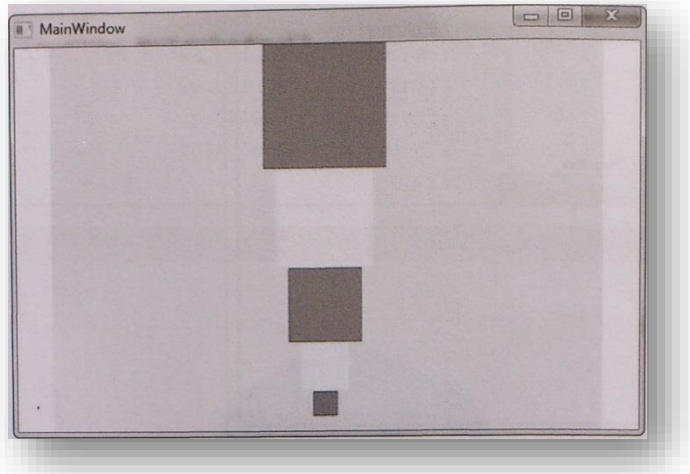
Codebehind tarafından baktığımızda StackPanel kontrolünün programatik oluşu¬mu oldukça kolaydır. Children property sayesinde istenilen kontrolleri içerisine ekleyebiliriz.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            StackPanel stackpanel = new StackPanel();
            stackpanel.Width = 450;
            stackpanel.Height = 600;
            stackpanel.Orientation = Orientation.Vertical;
            stackpanel.Background = new
SolidColorBrush(Colors.AntiqueWhite);
            //-----------------------------------------------------------------
            Rectangle k1 = new Rectangle();
            k1.Width = 105;
            k1.Height = 105;
            k1.Fill = new SolidColorBrush(Colors.Red);
            stackpanel.Children.Add(k1);
            //-----------------------------------------------------------------
            Rectangle k2 = new Rectangle();
            k2.Width = 85;
            k2.Height = 85;
            k2.Fill = new SolidColorBrush(Colors.Yellow);
            stackpanel.Children.Add(k2);
```

StackPanel

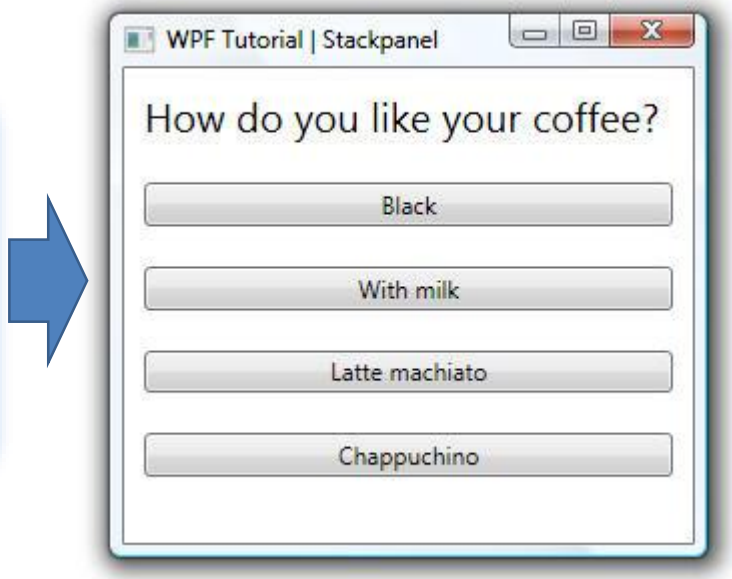
CodeBehind:

```
            Rectangle k3 = new Rectangle();
            k3.Width = 65;
            k3.Height = 65;
            k3.Fill = new SolidColorBrush(Colors.Red);
            stackpanel.Children.Add(k3);
            //-----------------------------------------------------------------
            Rectangle k4 = new Rectangle();
            k4.Width = 45;
            k4.Height = 45;
            k4.Fill = new SolidColorBrush(Colors.Yellow);
            stackpanel.Children.Add(k4);
            //-----------------------------------------------------------------
            Rectangle k5 = new Rectangle();
            k5.Width = 25;
            k5.Height = 25;
            k5.Fill = new SolidColorBrush(Colors.Red);
            stackpanel.Children.Add(k5);
            //-----------------------------------------------------------------
            this.Content = stackpanel;
        }
```



StackPanel - Örnekler

```
<StackPanel>
  <TextBlock Margin="10" FontSize="20">How do you like your coffee?</TextBlock>
  <Button Margin="10">Black</Button>
  <Button Margin="10">With milk</Button>
  <Button Margin="10">Latte machiato</Button>
  <Button Margin="10">Chappuchino</Button>
</StackPanel>
```



StackPanel - Örnekler

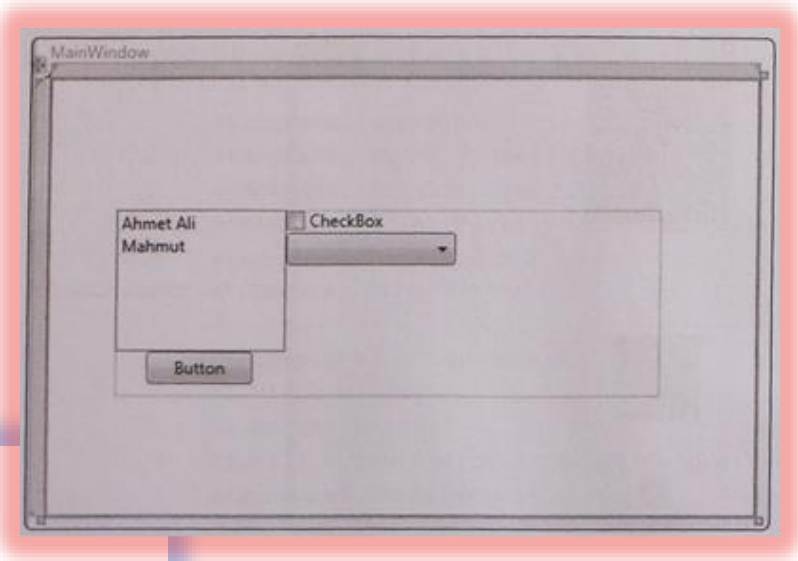
```
<StackPanel Margin="8" Orientation="Horizontal">
    <Button MinWidth="93">OK</Button>
    <Button MinWidth="93" Margin="10,0,0,0">Cancel</Button>
</StackPanel>
```

WrapPanel

- **WrapPanel** içerisindeki bileşenleri yatay veya dikey olarak hizalamaktadır.
- **StackPanel**'de olduğu gibi **Orientation** property'si sayesinde hangi yönde hizalama yapacağımızı ayarlayabiliriz.
- StackPanel kontrolüne oldukça benzemektedir. Aralarındaki fark; WrapPanel'in hizalama yaparken sığmayan kontrolleri yeni bir satıra kaydırabiliyor olmasıdır. Bu denkleme işlemine wrap denir.
- **WrapPanel** kullanırken içerisindeki kontrollerin büyüklüklerini eşit yapmak daha doğrudur. Çünkü **wrapPanel,** en geniş kontrole göre dizilim yapar

WrapPanel

Ekran Çıktısı:



Xaml:

```
<WrapPanel HorizontalAlignment="Right" Orientation="Vertical"
Margin="0,94,69,85" Name="wrapPanel1" Width="385">
            <ListBox Height="100" Name="listBox1" Width="120">
                  <ListBoxItem Content="Ahmet Ali" />
                  <ListBoxItem Content="Mahmut" />
            </ListBox>
      <Button Content="Button" Height="23" Name="button1" Width="75" />
            <CheckBox Content="CheckBox" Height="16" Name="checkBox1" />
            <ComboBox Height="23" Name="comboBox1" Width="120" />
</WrapPanel>
```

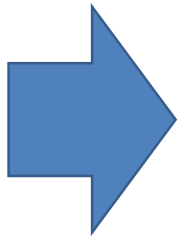
WrapPanel

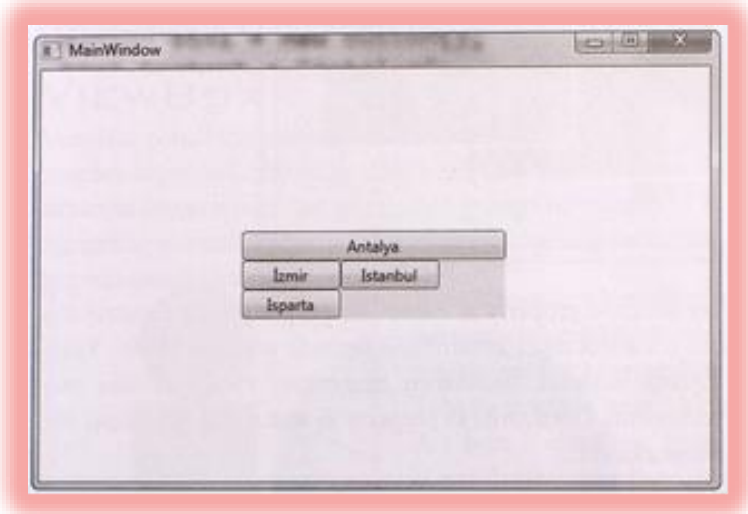
WrapPanel'i programatik olarak oluşturabiliriz. Dört adet button tanımlıyoruz. Daha sonra her kontrolün genişliklerini farklı bir şekilde veriyoruz.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            WrapPanel wp = new WrapPanel();
            wp.Background = new SolidColorBrush(Colors.AntiqueWhite);
            wp.Orientation = Orientation.Horizontal;
            wp.Width = 200;
            wp.HorizontalAlignment = HorizontalAlignment.Center;
            wp.VerticalAlignment = VerticalAlignment.Center;

            Button btn1 = new Button();
            btn1.Content = "Antalya";
            btn1.Width = 200;
            Button btn2 = new Button();
            btn2.Content = "İzmir";
            btn2.Width = 75;
            Button btn3 = new Button();
            btn3.Content = "Istanbul";
            btn3.Width = 75;
            Button btn4 = new Button();
            btn4.Content = "Isparta";
            btn4.Width = 75;
```

WrapPanel

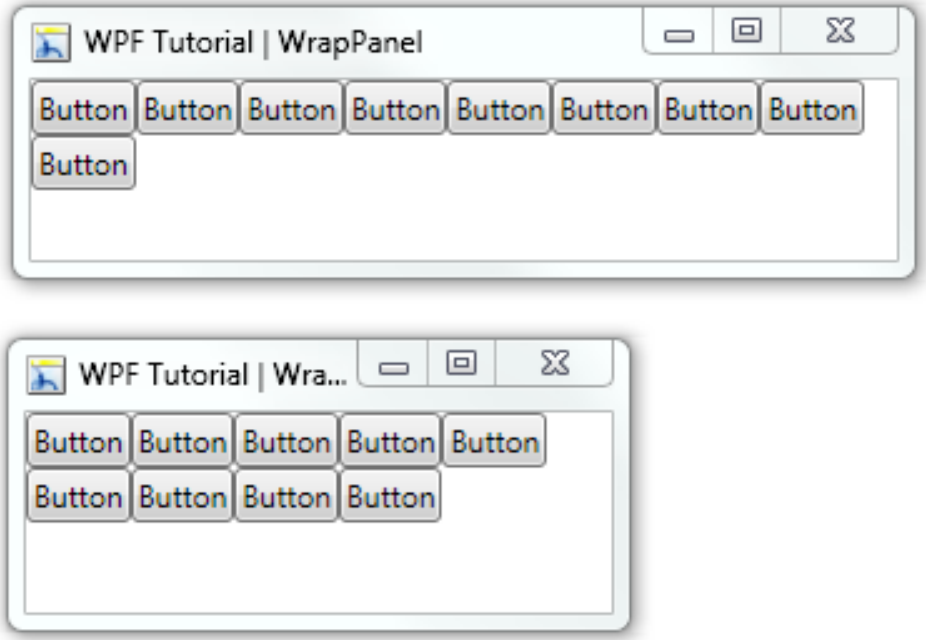
```
        wp.Children.Add(btn1);
        wp.Children.Add(btn2);
        wp.Children.Add(btn3);
        wp.Children.Add(btn4);
        this.Content = wp;
}
```





WrapPanel - Örnekler

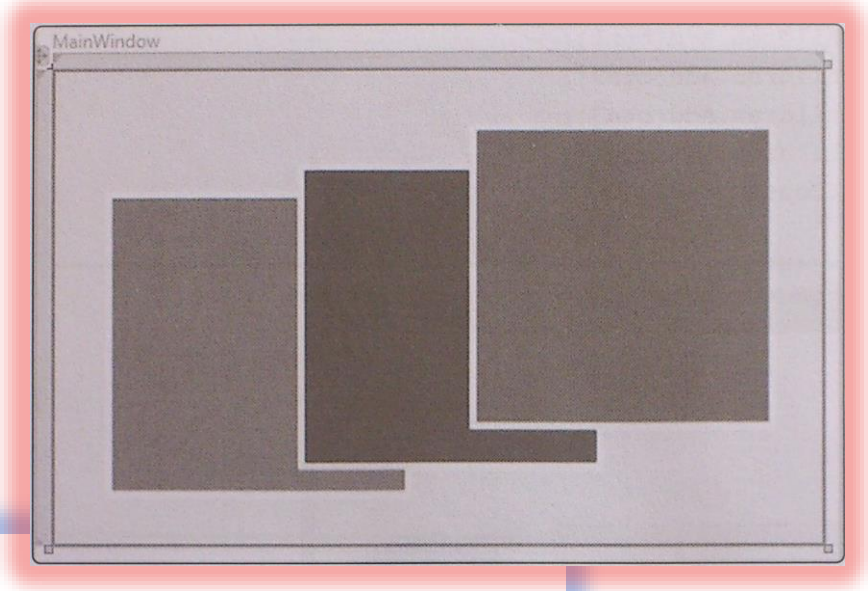
```
<WrapPanel Orientation="Horizontal">
    <Button Content="Button" />
    <Button Content="Button" />
    <Button Content="Button" />
    <Button Content="Button" />
    <Button Content="Button" />
</WrapPanel>
```



Canvas

- Canvas paneli daha çok grafiksel işlemlerde veya animasyon yapımında tercih ediliyor.
- Aslında bunun sebebi, Canvas panelinin içerisine alacağı kontrolü noktasal olarak belirtmesidir.
- Canvas paneli içerisine alacağı kontrolleri herhangi bir köşeden belirtilen uzaklıktaki noktaya yerleştirir.
- Canvas paneli içerisine yerleştirilecek her kontrolün **Left, Right, Top, Bottom** isimli attached property'leri vardır.
- Bu property'ler sayesinde bileşenlerin konumun noktasal olarak belirtebiliriz.

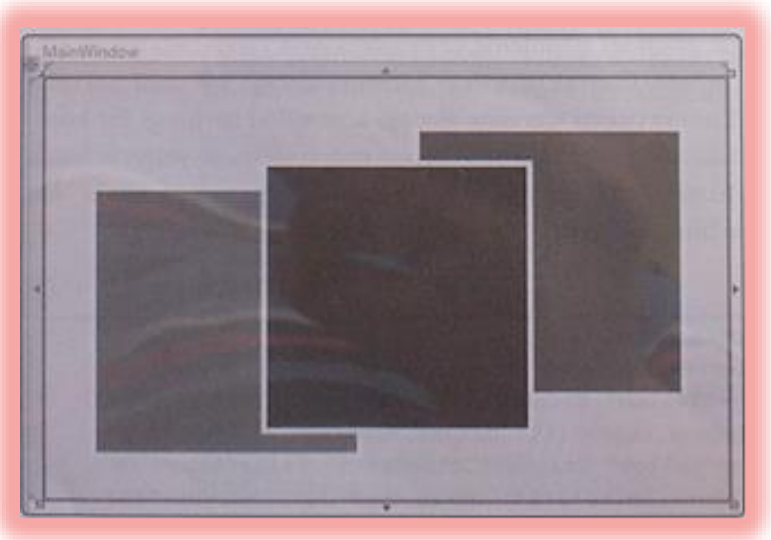
Canvas



Xaml:

```
<Canvas Background="AntiqueWhite" >
      <Rectangle Canvas.Left="35" Canvas.Top="80"  Height="200"
Width="200" Stroke="Yellow" StrokeThickness="5" Fill="Red" />
      <Rectangle Canvas.Left="160" Canvas.Top="62" Height="200"
Width="200" Stroke="Yellow" StrokeThickness="5" Fill="Blue" />
      <Rectangle Canvas.Left="272" Canvas.Top="36"  Height="200"
Width="200" Stroke="Yellow" StrokeThickness="5" Fill="Green" />
</Canvas>
```

Canvas



- Canvas panelin **zindex** attached property'si vardır.
- Bu property hangi kontrol içerisinde 1 değerini alırsa o kontrol diğer kontrollerin üstünde görünmektedir.
- Yukarıdaki örneğimizde ortada bulunan dikdörtgen nesnesinin **Panel.zindex** property'sini 1 olarak ayarlayalım.
- Görüyoruz ki property'si atanan element diğer elementlerin üstünde görünmektedir.

```
<Rectangle Canvas.Left="160" Canvas.Top="62" Panel.ZIndex="1" Height="200"
Width="200" Stroke="Yellow" StrokeThickness="5" Fill="Blue" />
```

Canvas

- Canvas panelin programatik olarak nesnesi alınır.
- Daha sonra içerisine eklenecek kontrolleri tanımlanırız.
- Son olarak **Canvas.SetLeft, Canvas.SetRight,** Canvas. SetTop, Canvas. SetBottam metodları ile panel içerisine noktasal konumlandırma yapmalıyız.

```
Canvas c = new Canvas();
Label l = new Label();
l.Content = "Ahmet Ali SÜZEN";
Canvas.SetLeft(l, 50);
this.Content = c;
```

Canvas - Örnekler

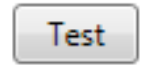

```
<Canvas>
    <Ellipse Fill="Green" Width="60" Height="60" Canvas.Left="30" Canvas.Top="20"
             Canvas.ZIndex="1"/>
    <Ellipse Fill="Blue"  Width="60" Height="60" Canvas.Left="60" Canvas.Top="40"/>
</Canvas>
```

ViewBox

- ViewBox paneli içerisindeki element veya kontrollerin görselliklerini, vektörel olarak yeniden boyutlandırabilme özelliği vardır.
- Kısaca ViewBox, zoom veya unzoom işlemlerini gerçekleştirir.
- Grid içerisinde yapılan bir çalışmada, Grid büyüklüğü değiştiği zaman otomatik olarak seçilen köşe sabit kalarak bir büyüme gerçekleşmektedir.
- Aynı bileşenleri ViewBox içerisinde büyüttüğümüz zaman panelin büyüklüğü ile doğru orantılı şekilde vektörel bir büyüme gerçekleşmektedir.
- Tabi bu noktada şuna dikkat etmek gerekmektedir; bazı kontrollerin panel içerisinde dinamik olarak boyutlandırılmasnı istiyorsak **Grid** kullanmak daha doğru olacaktır. Örneğin; bir **Grid Label'in** genişliği ve yüksekliğini üzerinde yaparken, **viewBox** genişliği ve yüksekliğini görselliği üzerinden sağlamaktadır.

ViewBox - Örnekler

```
<Button Content="Test" />

<Viewbox Stretch="Uniform">
    <Button Content="Test" />
</Viewbox>
```



Button

Button in a ViewBox

Nesne Kompozisyonları - Örnekler

```
<Button Margin="3">
  <StackPanel>
    <TextBlock Margin="3">Image and text button</TextBlock>
    <Image Source="happyface.jpg" Stretch="None" />
    <TextBlock Margin="3">Courtesy of the StackPanel</TextBlock>
  </StackPanel>
</Button>
```

Nesne Kompozisyonları - Örnekler

```
<Button>Absolutely No Padding</Button>
<Button Padding="3">Well Padded</Button>
```

Nesne Kompozisyonları - Örnekler

```
<Button>
  <Button.ToolTip>
    <StackPanel>
      <TextBlock Margin="3" >Image and text</TextBlock>
      <Image Source="happyface.jpg" Stretch="None" />
      <TextBlock Margin="3" >Image and text</TextBlock>
    </StackPanel>
  </Button.ToolTip>
  <Button.Content>I have a fancy tooltip</Button.Content>
</Button>
```

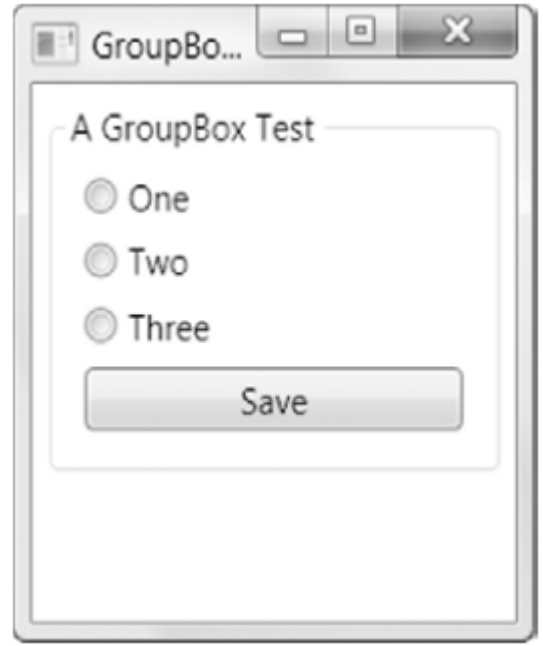
Nesne Kompozisyonları - Örnekler

```
<Popup Name="popLink" StaysOpen="False" Placement="Mouse" MaxWidth="200"
 PopupAnimation="Slide" AllowsTransparency="True">
  <Border BorderBrush="Beige" BorderThickness="2" Background="White">
    <TextBlock Margin="10" TextWrapping="Wrap">
      For more information, see
      <Hyperlink NavigateUri="http://en.wikipedia.org/wiki/Term"
       Click="lnk_Click">Wikipedia</Hyperlink>
    </TextBlock>
  </Border>
</Popup>
```

```
private void run_MouseEnter(object sender, MouseEventArgs e)
{
    popLink.IsOpen = true;
}

private void lnk_Click(object sender, RoutedEventArgs e)
{
    Process.Start(((Hyperlink)sender).NavigateUri.ToString());
}
```

Nesne Kompozisyonları - Örnekler

```
<GroupBox Header="A GroupBox Test" Padding="5"
  Margin="5" VerticalAlignment="Top">
  <StackPanel>
    <RadioButton Margin="3">One</RadioButton>
    <RadioButton Margin="3">Two</RadioButton>
    <RadioButton Margin="3">Three</RadioButton>
    <Button Margin="3">Save</Button>
  </StackPanel>
</GroupBox>
```



Nesne Kompozisyonları - Örnekler

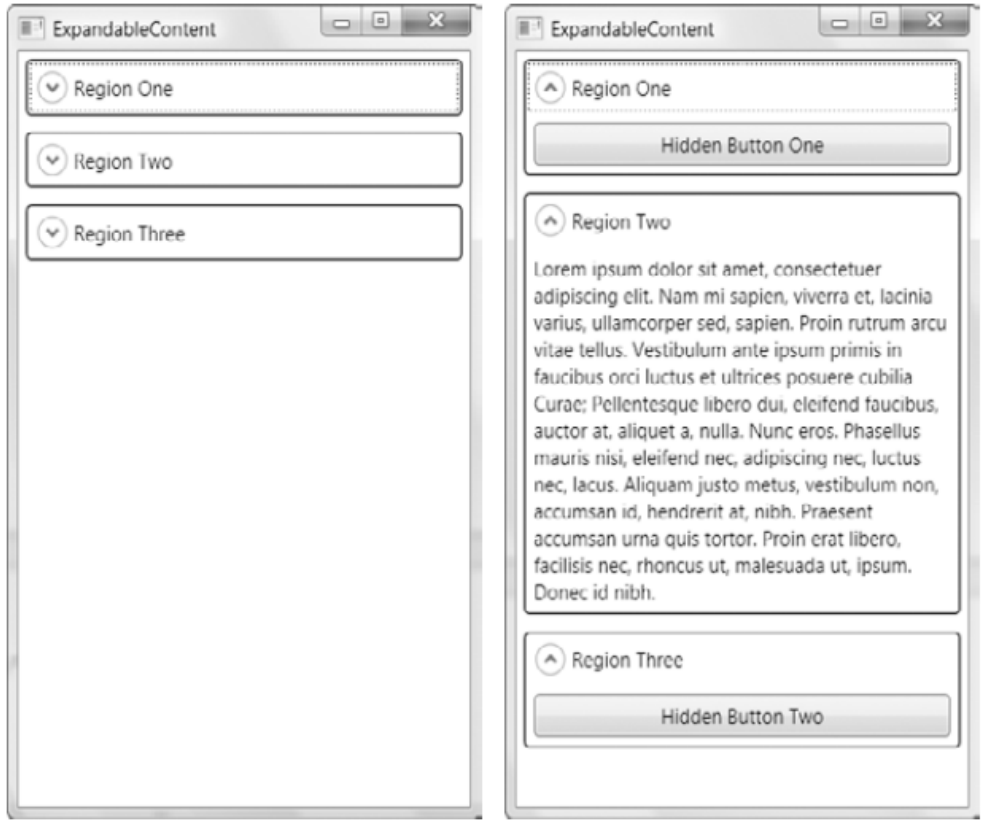
```
<TabControl Margin="5">
  <TabItem Header="Tab One">
    <StackPanel Margin="3">
      <CheckBox Margin="3">Setting One</CheckBox>
      <CheckBox Margin="3">Setting Two</CheckBox>
      <CheckBox Margin="3">Setting Three</CheckBox>
    </StackPanel>
  </TabItem>
  <TabItem Header="Tab Two">
    ...
  </TabItem>
</TabControl>
```

Nesne Kompozisyonları - Örnekler

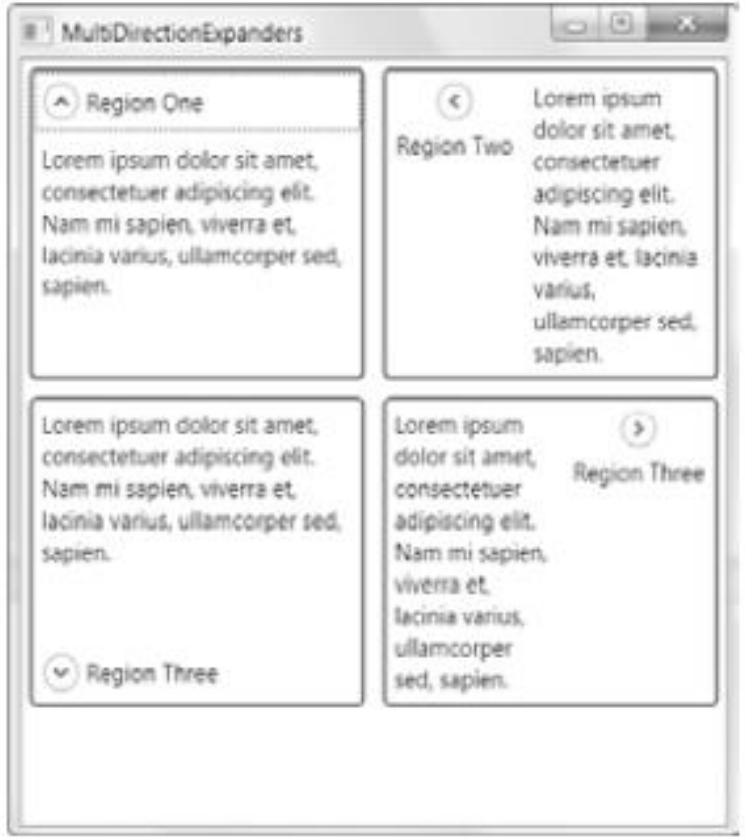
```
<TabControl Margin="5">
  <TabItem>
    <TabItem.Header>
      <StackPanel>
        <TextBlock Margin="3" >Image and Text Tab Title</TextBlock>
        <Image Source="happyface.jpg" Stretch="None" />
      </StackPanel>
    </TabItem.Header>

    <StackPanel Margin="3">
      <CheckBox Margin="3">Setting One</CheckBox>
      <CheckBox Margin="3">Setting Two</CheckBox>
      <CheckBox Margin="3">Setting Three</CheckBox>
    </StackPanel>
  </TabItem>

  <TabItem Header="Tab Two"></TabItem>
</TabControl>
```

Nesne Kompozisyonları - Örnekler

```
<StackPanel>
  <Expander Margin="5" Padding="5" Header="Region One">
    <Button Padding="3">Hidden Button One</Button>
  </Expander>
  <Expander Margin="5" Padding="5" Header="Region Two" >
    <TextBlock TextWrapping="Wrap">
      Lorem ipsum dolor sit amet, consectetuer adipiscing elit ...
    </TextBlock>
  </Expander>
  <Expander Margin="5" Padding="5" Header="Region Three">
    <Button Padding="3">Hidden Button Two</Button>
  </Expander>
</StackPanel>
```

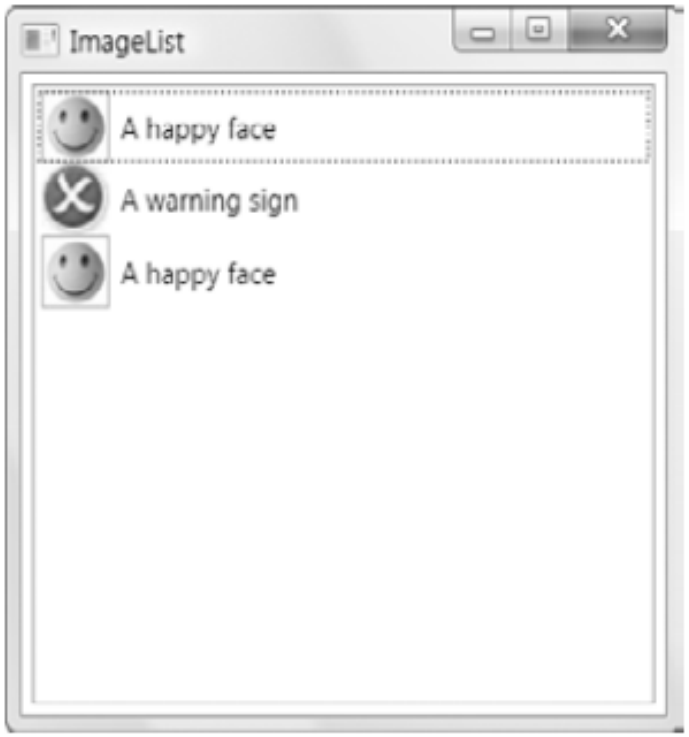


Nesne Kompozisyonları - Örnekler



Nesne Kompozisyonları - Örnekler

```
<ListBox>
  <StackPanel Orientation="Horizontal">
    <Image Source="happyface.jpg"  Width="30" Height="30"></Image>
    <Label VerticalContentAlignment="Center">A happy face</Label>
  </StackPanel>
  <StackPanel Orientation="Horizontal">
    <Image Source="redx.jpg" Width="30" Height="30"></Image>
    <Label VerticalContentAlignment="Center">A warning sign</Label>
  </StackPanel>
  <StackPanel Orientation="Horizontal">
    <Image Source="happyface.jpg"  Width="30" Height="30"></Image>
    <Label VerticalContentAlignment="Center">A happy face</Label>
  </StackPanel>
</ListBox>
```



Havada Kalmasın ☺

Teşekkürler…

bize katlandığınız için!