WPF Control Sınıf Kavramı

System.Windows.Control sınıfından türüyen sınıflar **Control** sınıflarıdır. WPF uygulamalarında görsel arayüz elementleri kullanılır. Bu elementlerden bazıları ise control olarak adlandırılır. Bir control genelde user-interactive element olarak tanımlanır. Yani bir element mouse veya klavye aracılığı ile etkileşimde bulunuyorsa bu element control elementidir. Örnek olarak text box'lar ve buttonlar birer controldür.

Controller aşağıdaki şekilde gruplandırılır:

Content controls: İçerisinde sayısız miktarda element barındırabilen controllerdir. Bunlara örnek olarak olarak **Label**, **Button**, **ToolTip** ve **ScrollView** sınıfları verilebilir.

Headered content controls: Ayrı title kısmına ve ana bölüme sahip olan controllerdir. Bu controller genelde daha büyük blokları barındırmak için kullanılır. Örnek olarak TabItem, GroupBox ve **Expander** sınıfları verilebilir.

Text controls: Kullanıcının input girebildiği controllerdir. Bu controllere örnek olarak TextBox, PasswordBox ve **RichTextBox** sınıfları verilebilir.

List controls: **Collection**'ları liste halinde gösteren controllerdir. Örnek olarak ListBox ve **ComboBox** sınıfları verilebilir.

Range-based controls: Bu controller'in ortak noktası Value property'sidir. Örnek olarak Slider ve **ProgressBar** sınıfları verilebilir.

Date controls: Tarih seçmek için kullanılan controllerdir. Örnek olarak Calendar ve **DatePicker** sınıfları verilebilir.

Control sınıflarının ortak özellikleri şunlardır:

1. Alignment
2. Tab order
3. **Background**, foreground ve border
4. Text için size ve font

Background and Foreground Brushes

Tüm controller background ve foreground özelliklerine sahiptirler. **Background** yüzeyi temsil ederken, foreground yazıyı temsil eder. Genelde bu özelliklerin color nesnesine sahip olmasını bekleriz. Fakat color nesnesine sahip olmaktan ziyade daha gelişmiş olan **Brush** nesnesini kullanılırlar.

Örnek:

```
1    cmd.Background= new SolidColorBrush(Colors.AliceBlue);
```

Burada cmd değişkeni **Button** nesnesini ifade etmektedir.

Not: Bir control sınıfını tamamen değiştirmek için, örneğin **Button** control'üne tıklanıldığında rengini değiştirmek için, templates kullanılması gereklidir.

Bir rengi R, G, B değerlerine göre oluşturmak için aşağıdaki kod kullanılır:

```
1    int red = 0;
2    int green = 255;
3    int blue = 0;
4    cmd.Foreground= new SolidColorBrush(Color.FromRgb(red, green, blue));
```

Rengi transparent yapmak istiyorsak, alpha değerini kullanabiliriz. Alpha değerini set edebilmek için **Color.FromArgb()** metodunu kullanmalıyız. Alpha değeri 255'ten 0'a yaklaştıkça transparent değeri **artar.**

XAML dosyasında renk belirtmek için aşağıdaki gibi kullanabiliriz:

```
1    <Button Background="Red">A Button</Button>
```

Renk kodu kullanmak istersek aşağıdaki gibi yazabiliriz:

```
<Button Background="#FFFF0000">A Button</Button>
```

LinearGradientBrush gibi farklı türden brush kullanabilmek için aşağıdaki gibi yazabiliriz:

```
<Button>
    A Button
    <Button.Background>
        <SolidColorBrush Color="Red" />
    </Button.Background>
</Button>
```

Font Embedding

Kullandığımız font işletim sisteminde bulunmazsa, işletim sisteminin kendi fontu kullanılır. Bu tarz problemleri önlemek için dosyadan font kullanabilme özelliği eklenmiştir. Bu sayede font her bilgisayarda korunmuş olur. Custom font kullanabilmek için .ttf dosyasına XAML control'ü içinde referans verebiliriz:

```
<Label FontFamily="./#BenimFontDosyam" FontSize="20">This is an embedded font</Label>
```

Başlangıçta yazılan **"./"** karakterleri o anki klasörü temsil eder. Font ismi kullanabilmek için ise bu karakterlere ek olarak **"#"** karakteri kullanılması gereklidir.

Text Formatting Mode

TextOptions.TextFormattingMode property ile Font size'ı küçük olan yazılar ekranda bulanık gösterilebilir. Bu tarz bir problemle karşılaşmamak için TextOptions.TextFormattingMode property kullanılır.

Örnek:

Problem Oluşturabilecek TextBlock:

```
<TextBlock FontSize="12" Margin="5">
    This is a Test. Ideal text is blurry at small sizes.
</TextBlock>
```

Çözüm:

```
<TextBlock FontSize="12" Margin="5" TextOptions.TextFormattingMode="Display">
    This is a Test. Display text is crisp at small sizes.
</TextBlock>
```

Not: Bu property sadece **small** yazılarda işe yarar. 15 point üstü yazılarda karakterler arasındaki boşluk, gösteriş biçimi vs düzgün olmayabilir.