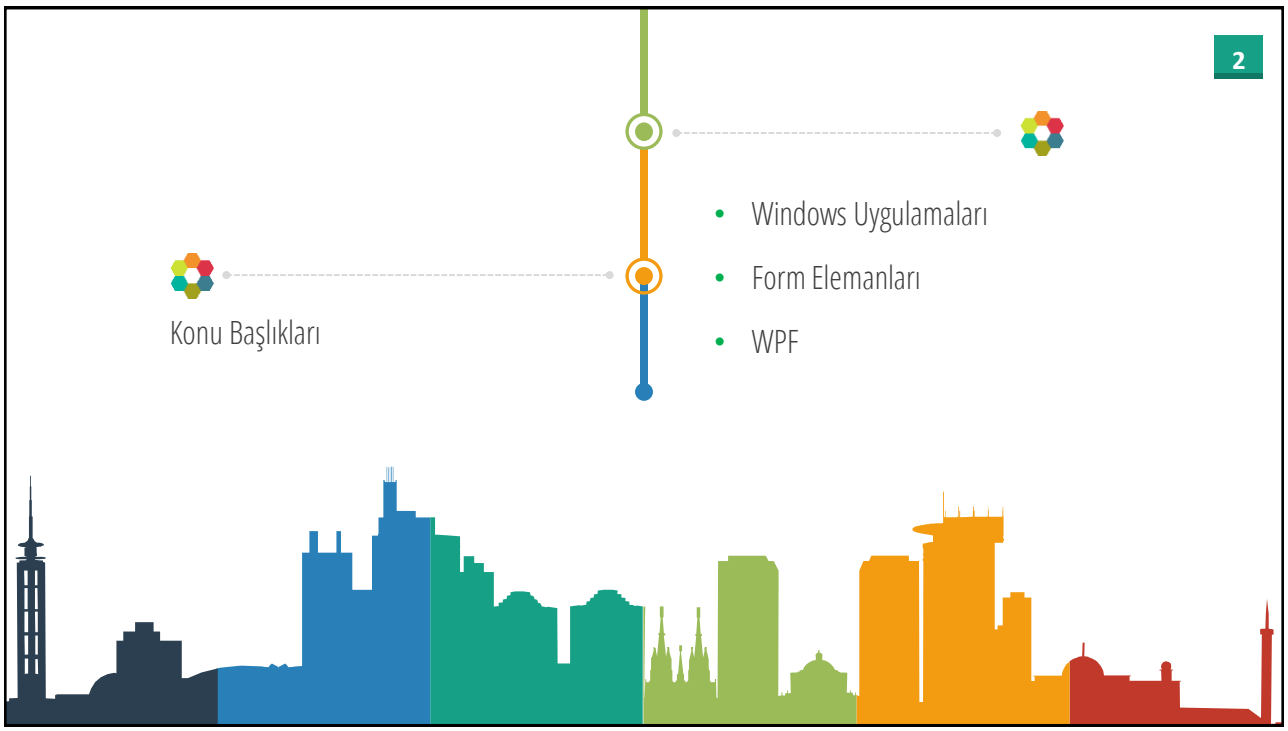

Fırsatlar Sunar
WPF Formlar


2
Konu Başlıkları
Windows Uygulamaları
Form Elemanları
WPF

3

WPF

Yazılım projeleri, yoğun olarak katmansal yapıda inŞa edilir. Bütün katmanlarda (veri katmanı, iŞ katmanı vb.) yapılagelen iŞlerinizde aldığınız tüm tasarım kararlarının toplamını, sunum katmanında kullanıcı ile iletiŞime geçmek için kullanırsınız.

WPF (Windows Presentation Foundation)'in temeli, modern grafik donanımının avantajlarından yararlanmak için oluŞturulan, çözünürlükten bağımsız ve vektör tabanlı iŞleme altyapısıdır.

4

Önceleri kullanılan “WindowsForms” kütüphanelerinden farklı “GDI+” yerine “DirectX” grafik kütüphanesini kullanır. Daha çok bilgisayar oyunlarından hatırladığınız bu kütüphane sayesinde ekran kartları sürücüleri daha etkin kullanılmakta ve daha hızlı görüntü iŞlemek mümkün olmaktadır.

5

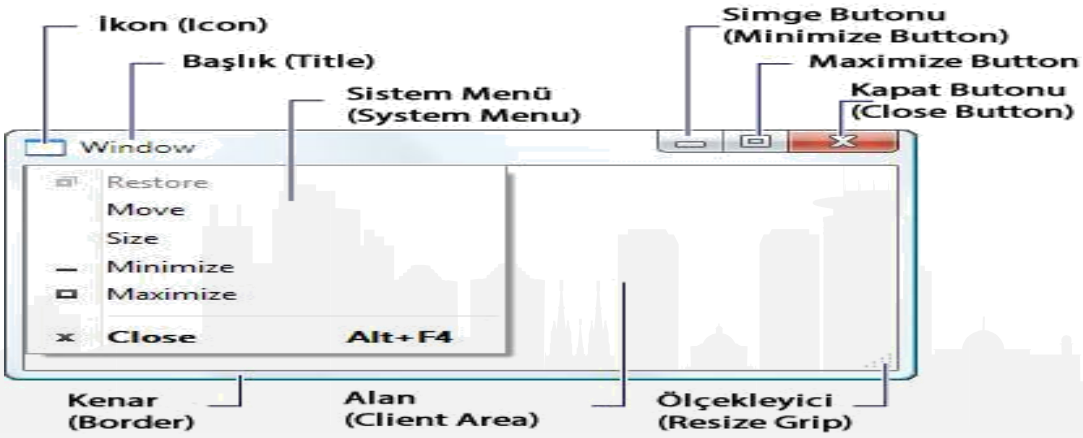
WPF Formlar

WPF, GeniŞletilebilir Uygulama Biçimlendirme Dili (XAML) isimli XML tabanlı, bildirime dayalı iŞaretleme dili ile denetimler, veri bağlama, düzen, 2-B ve 3-B grafikler, video, animasyonlar, stiller, Şablonlar, belgeler, ortam, metin ve tipografi içeren kapsamlı uygulama geliŞtirme özellikleri kümesi ile söz konusu temeli iŞler ve geniŞletir. WPF, ".NET Framework" içinde bulunur. Böylece ".NET Framework" sınıf kitaplığının diğer öğelerini bir araya getiren uygulamalar oluŞturabilirsiniz.

6

Temel anlamda bir form ve onu oluŞturan bileŞenler resim 1.1"de verildiği gibidir.
Formlar WPF kontrolleri arasında bulunur ve kullanıcı ile iletiŞimde bulunacak denetimleri kapsamakla görevlidir.

7



8

Tasarımda nihai amaç; ürün – kullanıcı entegrasyonunu baŞarı ile sağlayacak arayüzün iŞlevsel, kullanılabilir, görsel, açıdan kusursuz olmasını sağlamak olmalıdır. Bu sebepten yazılım üretim aŞamasında teknik olarak, uygulama arayüzü programcılar olduğu kadar grafik tasarımcılar da karar sahibidir. Farklı çalıŞma disiplinlerine sahip bu iki geliŞtirici (programcılar ve tasarımcılar) çoğu zaman birlikte çalıŞmak zorunda kalabilirler. WPF ile yapılan arayüz çalıŞmalarında bu iki takım üyesinin birbirinden bağımsız çalıŞabilmesine olanak tanımak ve çalıŞma ortamı verimliliğini arttırmak için tasarım kodu "XAML" ve arayüze işlevselliği sağlayacak program kodu (C# veya VB) birbirinden ayrılmıŞtır.

9

Kullanıcı ile iletiŞime geçme ortamınız çoğu kez farklılık gösterebilir. Örneğin uygulamanız masaüstü bir yazılım olabileceği gibi mobil iletiŞim cihazları üzerinde çalıŞan (tablet bilgisayarlar, akıllı telefonlar) veya bir *internet* uygulaması Şeklinde sunulabilir. Aralarında çok az farklılık bulunan bu platformlar için "XAML" tasarım odaklı ortak bir dil olma özelliğini geleceğe taŞımaktadır. Bu sayede WPF ile geliŞtirdiğiniz program arayüzleri üzerinde çok fazla değiŞikliğe gitmeden tasarım kodlarınızı diğer ortamlara rahatlıkla taŞıyabilirsiniz.

10

XAML

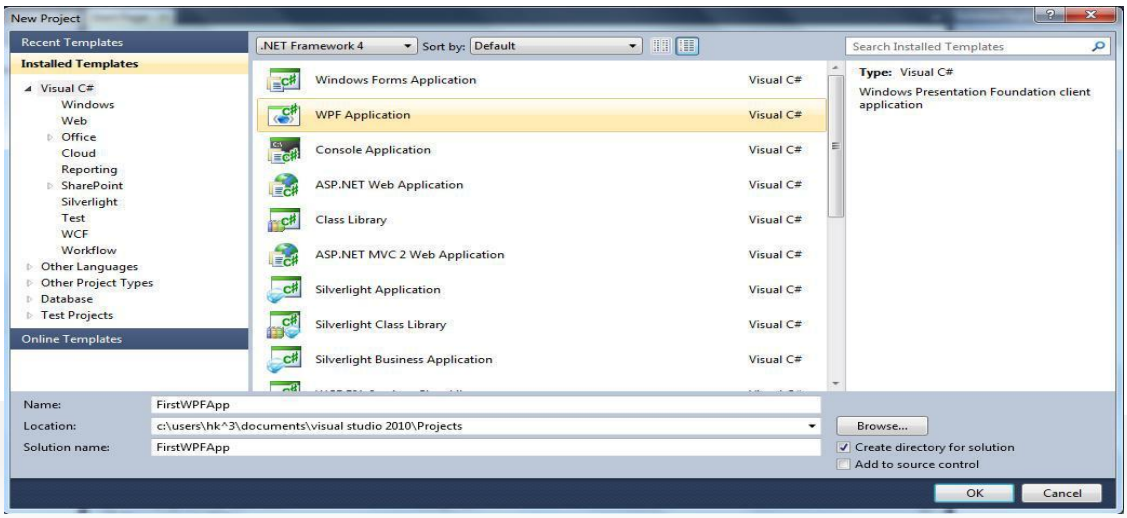
Bütün tasarımınızı "XAML" kodu yazarak oluŞturmak mümkündür ancak bu oldukça zahmetli bir iŞtir. "Visual Studio" belirli bir ölçekte sürükle bırak tekniği ile tasarım ekranında çalıŞmanıza olanak verir. Ancak bu yeterli değildir. Bunun sebebi "Visual Studio" tasarım aracı değil uygulama geliŞtirme aracı olarak hizmet etmesidir. Dolayısıyla arayüz tasarımını daha etkin ve hızlı olarak üretebilmeniz adına "Expression Studio" adı verilen ve bir dizi tasarım aracı üretmiŞtir. Siz uygulama arayüzünüz için görsel tasarımınızı yaparken sizin adınıza gereken "XAML" kodunu uygulamanız için oluŞturacaktır.

11

WPF Uygulaması Oluşturma

Bir WPF uygulaması baŞlatmanız oldukça basittir. Öncelikle uygulama geliŞtirme aracınız olan “Visual Studio” programı içinde dosya (File) menüsünden yeni bir proje (New Project) emri veriniz. Resim 1.2„de görüldüğü gibi ekrana gelen iletiŞim penceresinden ”WPF Application” Şablon seçimini yapın, projeniz için bir isim verin ve onaylayın.

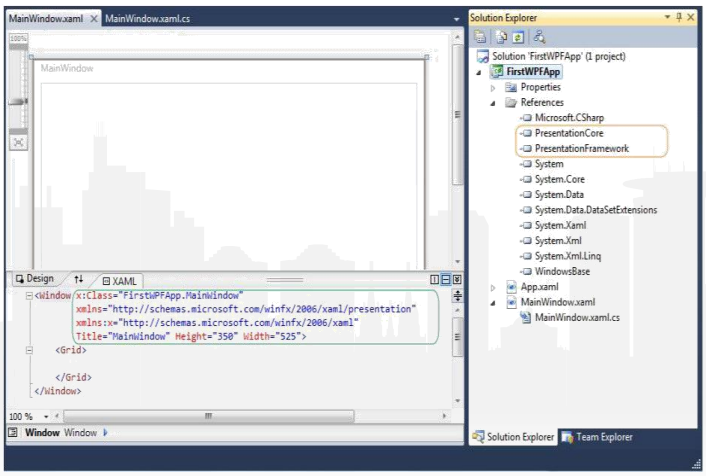
12



13

Yeni bir WPF uygulaması oluŞturduğunuzda Visual Studio sizin adınıza gereken referansları projeye ekleyecek ve tasarım yapmanıza olanak sağlayan arayüzü Şablonunu resimde görüldüğü gibi oluŞturacaktır.

14



15

Öncelikle “Solution Explorer” penceresini incelediğinizde “App.xaml” ve “MainWindow.xaml” dosyalarının oluşturulduğunu görebilirsiniz. Öncelikle “App.xaml” dosyasına çift tıklayınız ve inceleyiniz.

16

```
<Application x:Class="FirstWPFApp.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
StartupUri="MainWindow.xaml">
<Application.Resources>
</Application.Resources>
</Application>
```

17

Uygulama ile ilgili tanımlamaların yer aldığı “App.xaml” dosyası “XAML” biçimli dosyadır. Uygulamayı çalıŞtırdığınızda, “App.xaml” içinde yer alan bildirimler derleyici tarafından “Application” nesnesine dönüŞtürülür. Application nesnesini iŞletim sistemi ile uygulama kodu arasındaki bir arayüz olarak da düŞünebilirsiniz.

18

“App.xaml” içinde tanımlanan “<Application>...</Application>” etiketinin “StartupUri” isimli parametresi ile uygulama nesnesi baŞlatıldığında baŞlangıç penceresi tasarımının hangi dosyada yer aldığı bilgisini iŞaret etmektedir. Bir WPF uygulaması birden fazla pencereden oluŞabilir. Farklı bir pencere ile uygulamanın baŞlatılmasını isterseniz, bu durumda yapmanız gereken “StartupUri” parametresini değiŞtirmek olacaktır.

19

"MainWindow1.xaml" dosyasına çift tıkladığınızda tasarım ekranınız resim 1.3„te görüldüğü üzere ekranınıza gelecektir. İlki arayüz tasarımının ön izlemesi olan "design" bölümü diğeri ise tasarımı betimleyen "XAML" kodlarınızın yer aldığı bölümdür.

"Xaml" tasarım kodlarını inceleyiniz.

20

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

21

"XAML" dili "XML"in bir türevi olduğundan XML yazım kuralları burada aynen geçerlidir. Tasarım kodları ile ilgili aŞağıda yazılı kurallara dikkat ederseniz, okumanız ve yazmanız daha kolay hale gelecektir.

- Açılan etiketler kapatılmalıdır (Örneğin; <Grid>.....</Grid>).
- Gereken Şemalar verilmelidir (xmlns...).

22

- Büyük-küçük harf ayrımına uyulmalıdır.
- HiyerarŞi korunmalıdır.
- Mutlaka bir tane kök düğüm bulunmalıdır (Örneğin; <Window>.....</Window>).

23

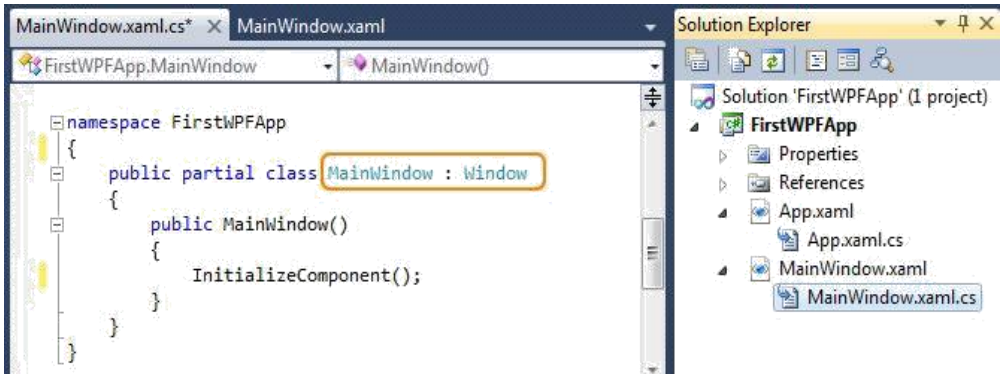
Uygulama Şablonu içinde bulunan “MainWindow.xaml” dosyası “<Window>...</Window>” kök etiketleri baŞlatılır. Bunun anlamı tıpkı “Aplication” nesnesinde olduğu gibi bir “Window” nesnesini baŞlatacak olmasıdır. İlk “Window” etiketi kapatılmadan yapılmıŞ “xmlns” deyimi ile “XAML” dosyası için kullanılacak hizmet sınıflarına ait isim alanı (namespace) bildirimleri yapılmaktadır. “x:” bildirimi, uygulamanın kendi sınıf bildirimi için yazılmıŞ bir önektir. Sahip olduğu sınıfı “Class” bildirimi ile tanımlamıŞtır. Bu bildirim program kodunuzun tasarım kodu ile birlikte derlenebilmesi için önemlidir.

24

WPF Form Özellikleri

WPF Şablonu üzerinde bulunan “MainWindow.xaml.cs” dosyasına ait resimde gösterilen sınıf tanımını incelediğinizde tasarımında bulunacağınız pencerenin (MainWindow) “Window” sınıfından kalıtılmıŞ olarak elde edildiğini görebilirsiniz.

25



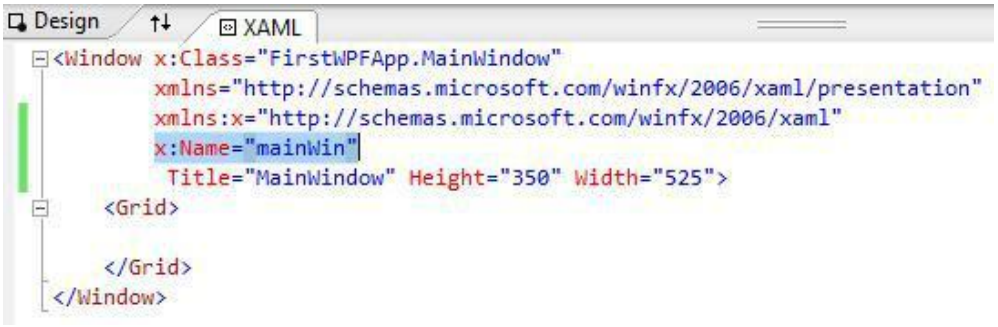
26

"MainWindow" ile oluŞturulmuŞ Şablon pencereniz "Window" sınıfının tüm özelliklerini taŞıyacaktır. Bu sayede gerek "XAML" tarafında gerekse kod sayfasında pencere ile ilgili iŞlemek istediğiniz özellikleri belirleyerek pencerenin durumlarını değiŞtirebilir, metotlarını kullanarak davranıŞları uygulayabilir, olaylarını iŞleyerek kullanıcı ile program kodunuzu iletiŞime geçirebilirsiniz.

Pencerelerin en sık kullanılan özelliklerini örnekleri üzerinden inceleyelim.

Name : Tüm WPF kullanıcı denetimlerinin ortak özellikler arasındadır. Söz konusu nesnenin hafızada sahip olacağı ismidir.

27



28

Title : Pencerenin baŞlık çubuğunda bulunan bilgi metnidir.

Width: Pencerenin sahip olduğu geniŞlik değeridir.

Height : Pencerenin sahip olduğu yükseklik değeridir.

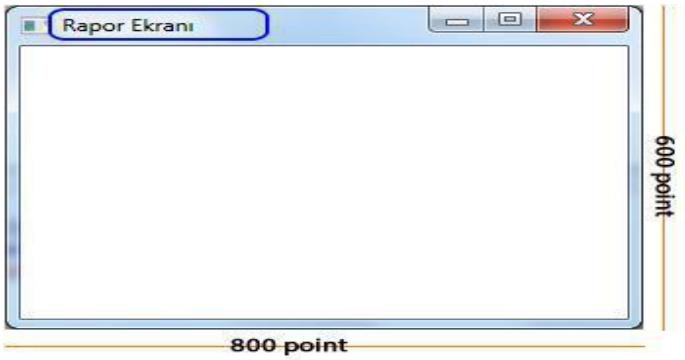
29

Örnek 1: Uygulama penceresinin baŞlık bilgisini “Rapor Ekranı”, yüksekliğini 600 point geniŞliğini 800 point olarak tasarlayınız.

30

```
<Window x:Class="FirstWPFApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="mainWin" Title="Rapor Ekranı"
Height="600" Width="800">
<Grid></Grid>
</Window>
```

Çıktı:



MinWidth : Pencerenin sahip olabileceği en az geniŞlik değeridir.

MinHeight : Pencerenin sahip olabileceği en az yükseklik değeridir.

MaxWidth : Pencerenin sahip olabileceği en fazla geniŞlik değeridir.

MaxHeight : Pencerenin sahip olabileceği en fazla yükseklik değeridir.

ResizeMode : Pencerenin yeniden boyutlandırılabilir olma özelliğini ayarlar.

33

Örnek 2: Uygulama penceresi, kullanıcı tarafından yeniden boyutlandırılmaya çalıŞıldığında en az “300 * 500” en fazla “600 * 800” değerlerini geçmemesi tasarlanmak istenmektedir.

34

```
<Window x:Class="FirstWPFApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="mainWin" Title="Rapor Ekranı"
Height="350" Width="400" ResizeMode="CanResize" MinWidth="500" MinHeight="300"
MaxWidth="800" MaxHeight="600"> <Grid></Grid>
```

35

Uygulamayı çalıŞtırınız ve ekrana gelen pencerenin sağ alt köŞesinden fare yardımıyla yeniden boyutlandırmayı deneyiniz.

Background : Pencerenin zemin rengini belirler.

36

Örnek 3: Uygulama pencere zemin rengini mavi olarak tasarlayınız.

<Window x:Class="FirstWPFApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="mainWin" Title="Rapor Ekranı"

Height="350" Width="400"

Background="Blue"> <Grid></Grid> </Window>

37

Örnek 1.4: Uygulama pencere zemin rengini maviden siyaha doğru renk geçiŞi olacak Şekilde tasarlayınız.

<Window x:Class="FirstWPFApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="mainWin" Title="Rapor Ekranı"

Height="350" Width="400">

<Window.Background>

38

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0"> <GradientStop Color="Blue" Offset="0" />

<GradientStop Color="Black" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid> </Grid> </Window>

39

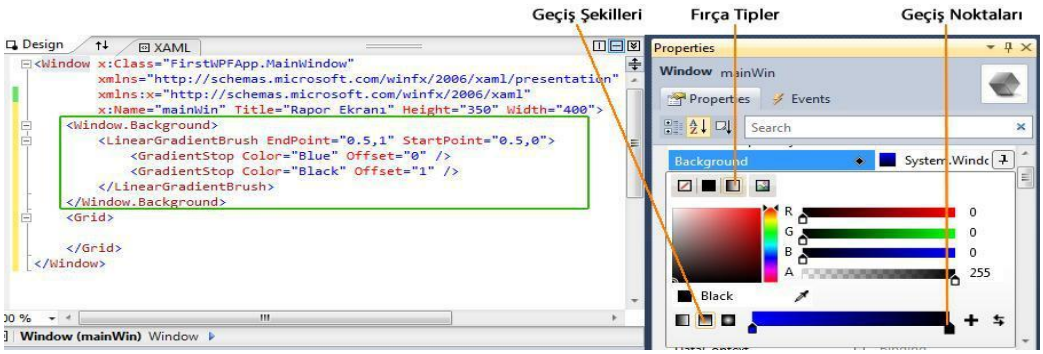
Yukarıda bulunan “XAML” kodunu incelediğinizde “Window” sınıfına ait “Background” özelliğinin <Window.Background>...</Window.Background> etiketleri arasında tanımlandığını görebilirsiniz. Nesnelere ait özelliklerin ayrı bir etiket ile düzenlenebiliyor olması “XAML” ile ne kadar esnek tasarımların yapılabileceğinin bir göstergesidir. Elde edilmek istenen geçiŞli renkler, “LinearGradientBrush” sınıfı altında belirlenir. Bu sınıf WPF için boyama yapabilen bir fırçadır ve renk geçiŞ noktaları olarak “GradientStop” sınıflarının “Color” özelliğine ihtiyaç duyar.

40

Arkaplan

Tek bir boyama iŞlemi için bu kadar fazla “XAML” kodu tanımlanıyor olması oldukça detaylı görülebilir. Burada “XAML” kodunun üretimini tasarım araçlarına bırakmak programcı için daha hızlı ve pratik bir yol olacaktır. Bunu gerçekleŞtirebilmek için tasarım aracınızın “Properties” panelini inceleyiniz.

41



42

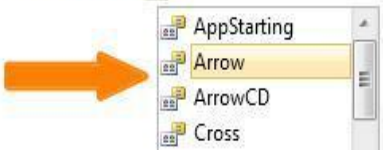
Seçmiş olduğunuz nesne ile ilgili özellikleri "Properties" panelinde bulabilirsiniz. Burada yapılan tüm değişiklikler tasarım aracınız tarafından "XAML" koduna dönüştürülecektir.

Cursor : Pencere üzerinde fare (mouse) görünümünü belirler. İşletim sistemine ait fare işaretçilerini tanımlama yaparken resimde olduğu gibi görebilir ve uygun olanı seçebilirsiniz.

43

```
<Window x:Class="FirstWPFApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        x:Name="mainWin" Title="Rapor Ekranı" Height="350" Width="400" Cursor="">
    <Grid>

    </Grid>
</Window>
```

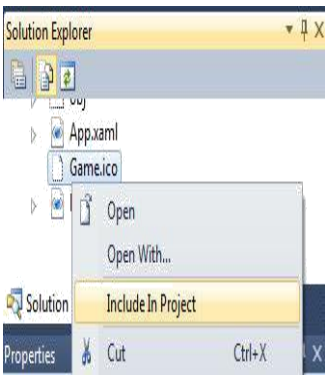


44

Icon : Pencerenin baŞlık çubuğu üzerinde ve iŞletim sisteminin görev çubuğunda görünen ikon resmini belirler.

Örnek 5 : Uygulamanıza ikon ekleyiniz.

Öncelikle uygulamanız için tasarlanan ikon dosyasını projenize dahil etmelisiniz. İkon dosyasını projenizin yer aldığı klasörün içine yerleŞtirin ardından "Solution Explorer" üzerinde dosyayı seçin ve fare sağ tuŞa basınız. Açılan menüden "Include Project" seçimini resimde görüldüğü gibi yapınız.



47

"Icon" özelliğini "Properties" panelinden bulunuz ve resim seç (Choose Image) butonuna tıklayınız. Ekrana gelen resim seçme iletiŞim kutusundan projenize dahil etmiŞ olduğunuz ikon dosyasını seçiniz ve onaylayınız. Pencerenizin "XAML" kodunda "Window" nesnesinin "icon" özelliğine dosya bilgisinin atandığını görebilirsiniz.

48

```
<Window x:Class="FirstWPFApp.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" Icon="/FirstWPFApp;component/Game.ico">
<Grid></Grid></Window>
```

Uygulamanızı çalıŞtırınız ve test ediniz.

•WindowStyle : Pencere stil tanımlamasıdır.

•WindowState : Pencerenin simge durumunda veya ekranı kaplamıŞ olarak boyutlamasını sağlar.

•WindowStartupLocation : Pencerenin ekran üzerindeki ilk açılıŞ konumunu belirler.

•Visibility : Pencerenin görünürlüğünü belirler.

Örnek 6: Uygulama penceresi ekranı kaplayan araç kutusu Şeklinde tasarlayınız.

```
<Window x:Class="FirstWPFApp.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" WindowStyle="ToolWindow" WindowState="Maximized">
<Grid></Grid></Window>
```

Örnek 7: Uygulama penceresi 300 *400 point büyüklüğünde ve ekranın tam orta noktasına konumlanacak Şekilde tasarlayınız.

```
<Window x:Class="FirstWPFApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="300"
Width="400" WindowStartupLocation="CenterScreen">
<Grid></Grid></Window>
```


ShowInTaskBar : Uygulama penceresi simgesinin iŞletim sistemi görev çubuğu üzerinde görüntülenmesini tayin eder.

Topmost : Uygulama penceresinin diğer uygulama pencerelerine göre daima önünde kalmasını sağlar.

Örnek 8: Uygulama penceresi simgesi görev çubuğunda görünmeden diğer uygulamalara göre en önde kalması istenmektedir. Gereken tasarımı yapınız.

53

<Window x:Class="FirstWPFApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="300"
Width="400" ShowInTaskbar="False" Topmost="True">
<Grid></Grid></Window>

Tooltip: Pencere üzerinde fare beklemesinin ardından açılan balon gösterimine ait metni belirler.

54

Örnek 9: Uygulama penceresinde açılacak balon bilgisi üzerinde uygulamanın adını göstermesi için gereken tasarımı yapınız.

<Window x:Class="FirstWPFApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="Benim Uygulamam" Height="300"
Width="400" ToolTip="Benim Uygulamam"> <Grid></Grid></Window>

55

Left: Pencerenin açıldığı ekran üzerinde soldan belirlenen değer kadar mesafede konumlanmasını sağlar.

Top: Pencerenin açıldığı ekran üzerinde üstten belirlenen değer kadar mesafede konumlanmasını sağlar.

BorderThickness: Pencereye verilen değer kalınlığında kenarlık oluŞturur.

BorderBrush: Pencereye çizdirilen kenarlığın rengini tayin eder.

AllowTransparency: Pencerenin Şeffaf olmasına izin verir.

Opacity: Pencerenin Şeffaflık oranını belirler. Alabileceği değer 0 ile 1 arasındadır.

56

Örnek 10: Uygulama penceresini %50 oranında ŞeffaflaŞtırıp, sistem kenarlıklarını kaldırılarak turuncu renk kenarlığa sahip özel bir pencere tasarlayınız.

<Window x:Class="FirstWPFApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="Benim Uygulamam" Height="300"

57

Width="400" BorderThickness="4" BorderBrush="Orange" WindowStyle="None“

AllowsTransparency="True" Opacity="0.5"> <Grid>

</Grid></Window>

1. **Tag:** Pencere üzerinde herhangi bir etkisi yoktur. “Object” tipinde bir özellik olup herhangi bir bilgiyi nesne ile birlikte tutabilme ihtiyacını karŞılamak için kullanılır.

58