

Gönülden Gönüle



Microsoft®
Silverlight™

“Bu kitabı Volkan Albayrak tarafından gönüllü olarak
Daron Yöndem'in blogundaki yazılarından derlenmiştir.”



<İÇİNDEKİLER>

- Colorful Expression ile birbiri ile uyumlu renkleri yakalamak
- Dinamik Assembly Üretimi (Kodla Dinamik DLL Compile Etmek)
- Java ve Silverlight kardeşliği
- Kendi Silverlight Yükle mesajınızı göstermeniz için ipuçları
- NET için De-Compile işlemleri ve Obfuscation
- Reflection nedir
- Silverlight 2.0 ve WCF Servisleri
- Silverlight 2.0 Beta 1 içerisinde DataGridView kullanımı
- Silverlight 2.0 Beta 1 içerisinde ToggleButton kullanımı
- Silverlight 2.0 Beta 1 ile Klasik ASMX Web Servisi kullanımı
- Silverlight 2.0 Beta 2 içerisinde dinamik resim yüklemek
- Silverlight 2.0 Beta 2 ile ASP.NET Forms Authentication kullanımı
- Silverlight 2.0 Beta 2 ile beraber gelen TabControl incelemesi
- Silverlight 2.0 Beta 2 ve PHP ile mailform uygulaması
- Silverlight 2.0 Calendar ve DatePicker kontrolleri
- Silverlight 2.0 Cross-Domain WebClient ile REST (GET) ve XSLT Kullanımı
- Silverlight 2.0 GridSplitter Kullanımı
- Silverlight 2.0 HyperlinkButton Kullanımı
- Silverlight 2.0 içerisinde asenkron font dosyası indirerek kullanmak
- Silverlight 2.0 içerisinde AutoCompleteBox kullanımı
- Silverlight 2.0 içerisinde Carousel kullanımı
- Silverlight 2.0 içerisinde ControlTemplating ve Style yapıları
- Silverlight 2.0 içerisinde createFromXaml alternatifisi ve alt seviyeli dinamik nesne üretimi
- Silverlight 2.0 içerisinde fare roller'ını kullanmak
- Silverlight 2.0 içerisinde farenin çift tıklamasını algılamanın yolu
- Silverlight 2.0 içerisinde harici Class Library yapılarının asenkron kullanımı
- Silverlight 2.0 içerisinde harici dinamik XAP (Silverlight 2.0) uygulamalarının asenkron yüklenmesi
- Silverlight 2.0 içerisinde Hue Saturation ve Lightness ile dinamik renk paletleri yaratmanın yolu
- Silverlight 2.0 içerisinde Isolated Storage kullanımı
- Silverlight 2.0 içerisinde Localization kullanımı
- Silverlight 2.0 içerisinde maskeleme (clipping)
- Silverlight 2.0 içerisinde MultiScaleImage kullanımı ve DeepZoom maceraları
- Silverlight 2.0 içerisinde Open File Dialog kullanımı
- Silverlight 2.0 içerisinde ProgressBar kullanımı
- Silverlight 2.0 içerisinde sağ tuş menüsünü değiştirmek
- Silverlight 2.0 içerisinde ScrollViewer kullanımı
- Silverlight 2.0 içerisinde Silverlight Toolkit ve TreeView kullanımı
- Silverlight 2.0 içerisinde Toolkit'den ViewBox kullanımı
- Silverlight 2.0 içerisinde Toolkit'ten Label kontrolünün kullanımı
- Silverlight 2.0 içerisinde ToolTip Kontrolü ve Tooltip şablonları

- Silverlight 2.0 içerisinde VisualStateManager kullanımı
- Silverlight 2.0 içerisinde Download penceresi açtırmak
- Silverlight 2.0 için özel ön yükleme ekranları geliştirmek. (PreLoader)
- Silverlight 2.0 ile Analog Saat Uygulaması
- Silverlight 2.0 ile Oyun Programlama'ya Giriş
- Silverlight 2.0 ile Powerpoint dosyalarının Thumbnail'lerini göstermek
- Silverlight 2.0 ile Twitter ve TwitXR Combo Widget
- Silverlight 2.0 ItemsControl ve ObservableCollection ile DataBinding
- Silverlight 2.0 kontrolleri yaratmanın yolu
- Silverlight 2.0 Plug-In Algılama ve OBJECT Tagı
- Silverlight 2.0 PopUp Kontrolü
- Silverlight 2.0 RCO içerisinde ComboBox kullanımı
- Silverlight 2.0 RCO içerisinde PasswordBox kullanımı
- Silverlight 2.0 Uygulamaları Parametre Gönderimi
- Silverlight 2.0 Uygulamalarında farklı XAML dosyaları kullanmak
- Silverlight 2.0 ve Adaptive Streaming
- Silverlight 2.0 ve ADO.NET Data Services
- Silverlight 2.0 ve JavaScript kardeşliği
- Silverlight 2.0 ve JSON Serialize DeSerialize İşlemleri
- Silverlight 2.0 ve Socket Programlama Mucizesi
- Silverlight 2.0 XAP Paketleri
- Silverlight 2.0 XAP Paketleri ve Kaynak Dosyalar
- Silverlight 2.0'da farenin hareket etmediğini anlamak
- Silverlight içerisinde ClipBoard kullanımı
- Silverlight içerisinde sayfa adresine ulaşmak
- Silverlight Runtime ve SDK DLL'leri ve açıklamaları
- Silverlight Toolkit içerisinde gelen hazır renk şablonları (Thema) inceliyoruz
- Silverlight Toolkit'ten WrapPanel'in kullanımı
- Silverlight uygulamaları ve IIS MIME Type ayarı
- Silverlight uygulamamızın tarayıcı tarafından tekrar boyutlandırıldığını nasıl algılarız
- Silverlight ve WPF'de Design Mode ve Init durumunda kodlar sorunsalı
- Silverligth 2.0'da uygulama fonunu şeffaf kullanmak
- SL 2.0'da ZIP içerisinde asenkron kaynak kullanımı
- Vista Gradientları XAML Kodları

Colorful Expression ile birbiri ile uyumlu renkleri yakalamak.

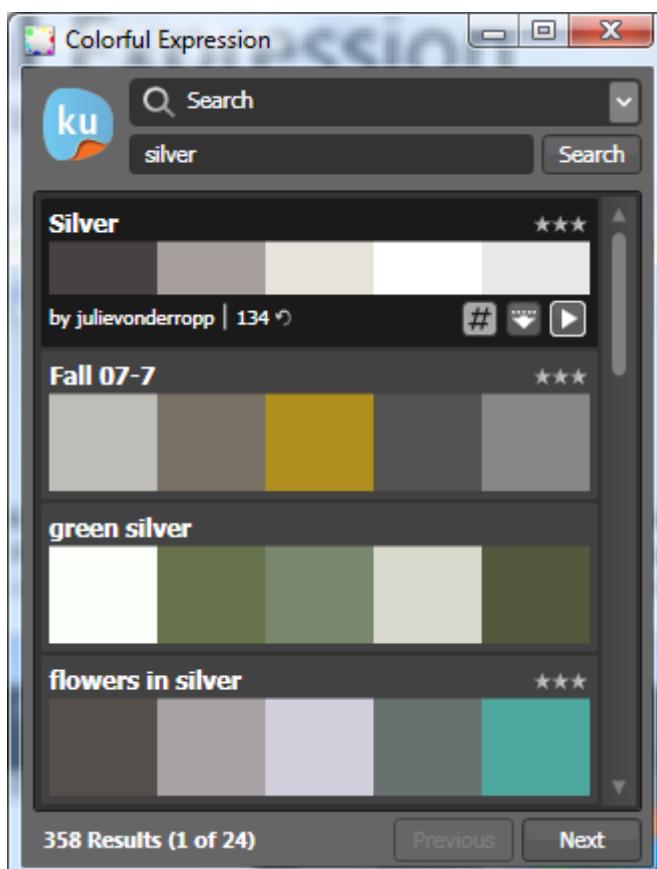
Renklerin birbirine uyumu özellikle biz yazılımcılar için pek anlaşılması zor bir sistemi tanımlar :) Kişisel olarak ben bir renk yiğimindaki renklerin birbirine uyumlu olup olmadığını anlayabilsem de "Buyur uyumlu 3 renk seç" derseniz pek de başarılı olamam. Belli ki bu durum genel geçer bir sorunu tanımlıyor ki RD, [Jonas Folloseo](#) birazdan sizlere detaylarından bahsedeceğim uygulamayı hazırlamış. Uygulama özünde Adobe'nin [Kuler](#) sitesinin API'larını kullanıyor. Kuler'dan hızlı bir şekilde bahsetmek gerekirse tasarımcıların birbirleri ile uyumlu renk şemalarını paylaştıkları bir Web 2.0 portalı diyebiliriz.

Colorful Expression

Aşağıdaki adresten indirebileceğiniz uygulama toplam 3 bölümden oluşuyor.

<http://www.codeplex.com/colorful>

Birincisi **Colorful WPF** adında tek başına çalışabilen bir WPF uygulaması. Bu uygulama içerisinde birbirleri ile uyumlu renk şemalarını inceleyebilir ve aramalar yapabilirsiniz. Unutmayın ki sistem Kuler'in API'larından faydalanyor yani programı ancak online durumdayken kullanabilirsiniz. Colorful WPF'in en güzel özelliği herhangi bir renk şemasının altındaki düğmeler aracılığı ile hızlı bir şekilde bu renkleri kullanabilmenizi sağlayacak XAML Brush kodlarını alabiliyor olmamız.



Colorful WPF içerisinde "silver" kelimesi aratıldığında çıkan birbiri ile uyumlu renklerin bir listesi.

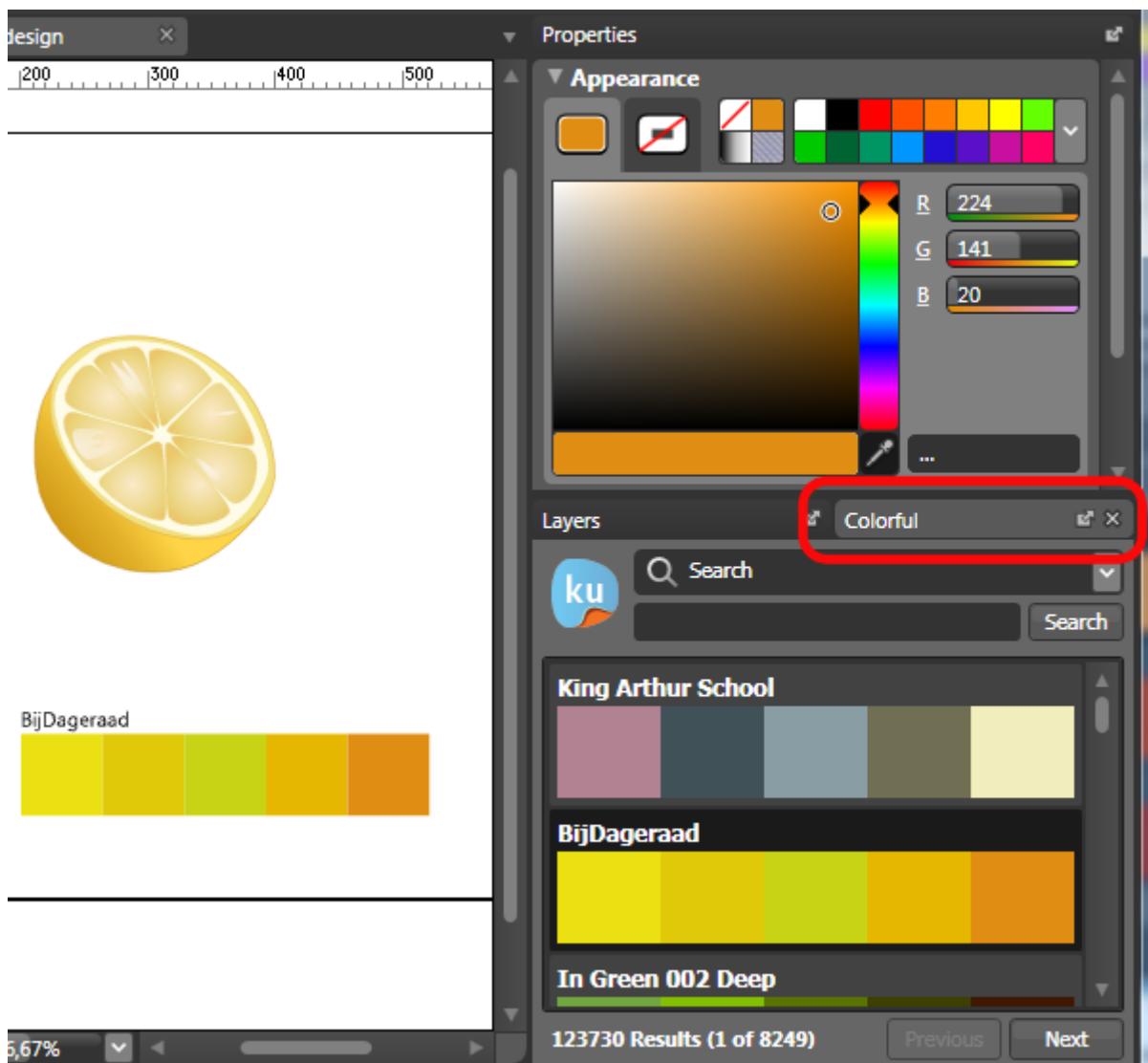
Yukarıdaki ekran görüntüsünde yer alan en üstteki "Silver" adındaki renk şemasının altındaki "Swatches" düğmesine tıkladığında doğrudan aşağıdaki XAML kodu panoya kopyalıyor ve rahatlıkla Silverlight veya WPF projelerinde kullanabiliyoruz.

```
<SolidColorBrush x:Key="SilverColor1" Color="#FF474143" />
<SolidColorBrush x:Key="SilverColor2" Color="#FFA69E9D" />
<SolidColorBrush x:Key="SilverColor3" Color="#FFE7E2DA" />
<SolidColorBrush x:Key="SilverColor4" Color="#FFFFFFFF" />
<SolidColorBrush x:Key="SilverColor5" Color="#FFE7E8E7" />
```

Expression Design ve Blend Add-In

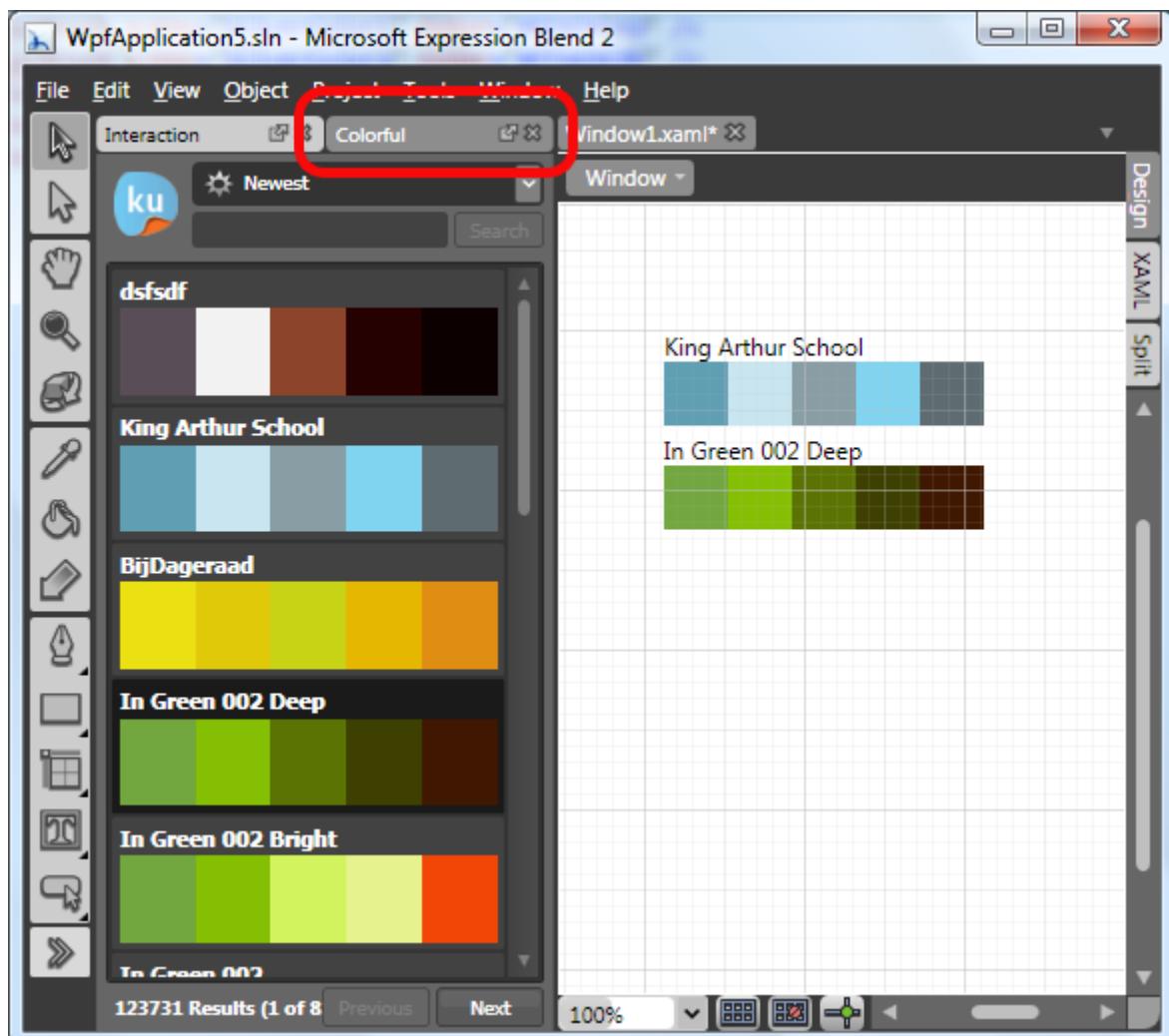
Colorful Expression içerisindeki renk şablonlarını isterseniz doğrudan Expression Design veya Blend içerisinde de kullanabiliyorsunuz. Bunun için download paketi içerisinde program adına uygun klasörün içindeki 2 DLL dosyasını programların bilgisayarlarınızda yüklü oldukları konumlara kopyalamanız gerek. Sonrasında aşağıdaki şekilde hem Blend hem de Design'ı çalıştırığınızda **Colorfull Expression**'ı doğrudan Blend veya Design içerisinde de kullanabilirsiniz.

```
Blend.exe –addin:ColorfulBlend.AddIn.dll
Design.exe –addin:ColorfulDesign.AddIn.dll
```



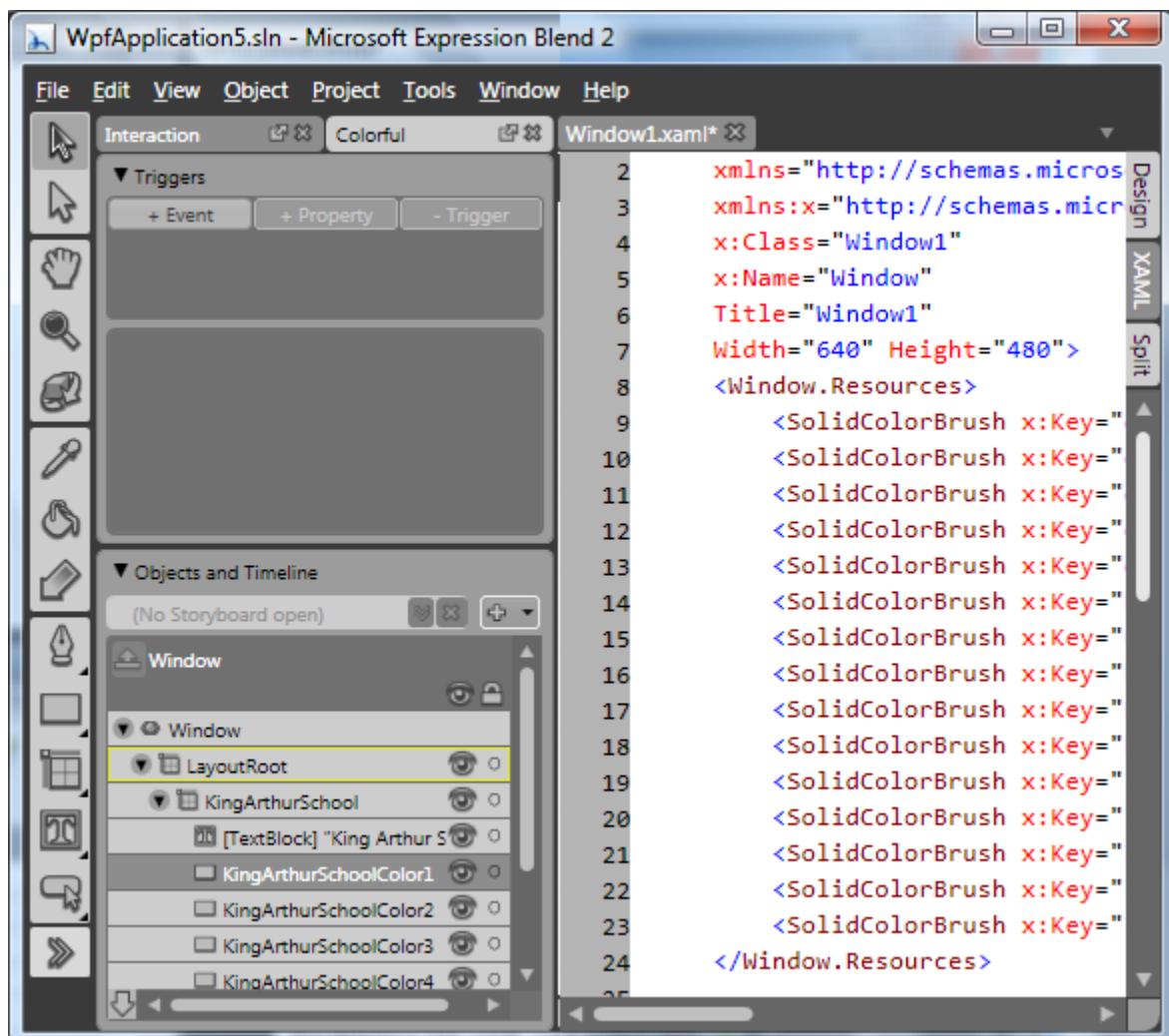
Expression Design içerisinde Colorful paneli.

Yukarıdaki ekran görüntüsünde Colurful Expression'in doğrudan Expression Design içerisinde kullanılabilğini görebiliyorsunuz. Aynı şekilde Blend 2 içerisinde de rahatlıkla **Colorful** paneline ulaşılabiliriyor.



Expression Blend 2 içerisinde Colorful paneli.

Blend içerisinde Colorful panelinin kullanımı ile ilgili Design'a kıyasla ek avantajlar da söz konusu. Sahneye sürükleyip bıraktığınız bir renk şablonu aslında arka planda birer **SolidColorBrush** olarak sayfanın Resource'larına ekleniyor. Böylece bu renkleri istediğiniz kadar farklı yerlerde rahatlıkla merkezi olarak kullanabiliyorsunuz.



Colorful'un yarattığı XAML kodları otomatik olarak karşımızda.

Hepinize kolay gelsin.

Daron YÖNDEM

Cross Domain Request için sunucu tarafı ASP.NET Proxy

İstemci tarafı programlama sistemleri AJAX ile karşımıza çıkmıştı, Silverlight ile beraber ise artık istemci tarafı programlama neredeyse "hayatımız" oluyor. Bu durumda karşılaştığımız en büyük sorun "Cross-Domain-Request" sınırlaması. Güvenlik nedenleriyle bir alan adından bir başka alan adına bağlanarak veri talebinde bulunamıyoruz. Eğer karşısındaki alan adının ihtiyac etiği siteye admin erişiminiz varsa tabii ki farklı teknikler kullanarak bu sorunu çözebilirsiniz. Bu konuda Silverlight 2.0 ile beraber [clientaccesspolicy.xml](#) dosyası geliyor.

Peki ya karşı siteye admin erişimimiz yoksa?

İşte o zaman kendi sitemizde sunucu tarafı bir proxy kullanmamız şart. ASP.NET ile sunucu tarafından istediğimiz siteye bağlanarak istediğimiz dosyası alabiliriz. Bu durumda bir ASPX sayfası yapsak bizim yerimize gidip kendisine hedef gösterdiğimiz adresden gerekli dosyayı alıp istemci tarafına, yani bize iletse hoş olmaz mı?

```
Dim Talep As New Net.WebClient
Dim GelenVeri As Byte() = Talep.DownloadData(Request.QueryString("Dosya"))
Response.ContentType = Talep.ResponseHeaders("Content-type").ToString
Response.OutputStream.Write(GelenVeri, 0, GelenVeri.GetLength(0))
Response.OutputStream.Close()
Response.End()
```

Yukarıdaki kod içerisinde doğrudan bir **WebClient** yaratarak farklı bir adresden veri indirme işlemi yapıyoruz. Kod içerisindeki en önemli nokta indirmek istediğimiz hedef veri ile istemciye göndereceğimiz verinin **ContentType** değerlerinin aynı olması gerektiği. Bunun için **Response.ContentType**'ı WebClient üzerinden aldığımız **Content-Type header** bilgisi ile eşleştiriyoruz. Böylece proxy'miz gerektiğinde video veya resim dosyalarını da rahatlıkla indirerek bize ulaştırabilir.

Performans?

Yukarıdaki örneğimiz çok basit bir yapıya sahip. Dosyayı sunucuya indirerek doğrudan istemciye gönderiyor. Yüksek sayıda istek oluşan projelerde veya büyük dosyalar indirecek olan uygulamalarında farklı performans senaryoları uygulamak gerekecektir. Aslında baktığımızda bu yapının herhangi bir Proxy programlamaktan pek farkı yok. Aklıma ilk aşamada gelen dikkat edilmesi gereken noktalar şöyle oldu;

- Büyük dosya indirirken istemcinin hala bağlı olup olmadığını **Response.IsClientConnected** ile kontrol etmek gereklidir.
- Büyük dosya indirme işlemlerinde bufferlamak ve kısım kısım indirerek istemciye göndermek daha mantıklı olabilir. Özellikle video dosyalarında.
- Kesinlikle bu dosyaya request yollayanın headerini kontrol etmek lazımdır. Kötü niyetli biri bu proxy'yi sadece sunucunun bant genişliğini harcamak için kullanabilir veya gereksiz yere sunucuya yorabilir.

Hepinize kolay gelsin.

Daron YÖNDEM

Dinamik Assembly Üretilimi (Kodla Dinamik DLL Compile Etme)

Bir uygulama düşünün kendini programlayabilen. Konumuz “Star Trek” veya “Geleceğe Dönüş” değil. Emin olun gerçek dünyadan ve yapılabileceklerden bahsediyorum. Uygulamalarınızın dış sistemlerle ciddi bir bağlantı içerisinde olduğu durumlarda bazen kendi içlerinde dış sistemlere uygun kodlar üretecek kullanmalari gerekebilir. Bunu bazen uygulamaların kendi içlerindeki yapay zeka ile yapabilecekleri gibi bazen ise başka bir dış kaynaktan aldıkları yeni parametrelerden yola çıkarak kendi kodlarında değişiklik yapabilirler. Eğer bunların hiçbiri size gerçekçi gelmiyorsa başka bir seçenek olarak da harici uygulamaların kullanabileceği DLL dosyaları yaratacak bir uygulama yazmak istediğinizde yapmanız gerekenlerden bahsedebiliriz. Aslında her ikisi de aynı kapıya çıkıyor.

Bize dinamik olarak uygulamalar tarafından kullanılabilen DLL dosyaları yaratacak bir kod lazım. Kullanacağımız nesnelerin çoğunun bulunduğu esas namespace **System.CodeDom.Compiler** olacak. Bunun haricinde C# veya VB için ayrı ayrı uygun namespace’leri kullanmamız gerek. Eğer VB kodu derleyecekseniz VB sınıflarını C# kodu derleyecekseniz tabi ki C# sınıflarını kullanmalısınız. Çapraz işlem yaparak C# kodunuz ile VB kodundan DLL üretme şansınız da var. Biz örneklerimizde C# ile C#’dan derleme, VB kodu ile de VB’den derleme yapacağız.

[VB]

```
Dim KodUretici As New Microsoft.VisualBasic.VBCodeProvider
Dim Derleyici As System.CodeDom.Compiler.CodeCompiler =
KodUretici.CreateCompiler()

Dim Referanslarim As String() = {"System.dll"}
Dim AssemblyAdi As String = "Ornek.dll"
```

[C#]

```
Microsoft.CSharp.CSharpCodeProvider KodUretici = new
Microsoft.CSharp.CSharpCodeProvider();
System.CodeDom.Compiler.ICodeCompiler Derleyici = KodUretici.CreateCompiler();

String[] Referanslarim = {"System.dll"};
String AssemblyAdi= "Ornek.dll";
```

Kodumuzun başlangıcında ilk olarak birer **CodeProvider** nesnesi yaratıyoruz. Elimizdeki hazır kodu derleyecek olan nesneler olarak bu sınıflar VB ve C# için farklılaşıyor. **CodeProvider**’lar üzerinden birer de derleyici nesnesi aldıktan sonra sıra geliyor derleyeceğimiz kodun referanslarına karar vermeye. Referansları DLL isimleri ile bir **String** dizisine aktarmanız şart. Windows uygulamalarında en azından **System.dll**’in web uygulamalarında da **System.Web.dll**’in referans alınmış olması gerekiyor. Son olarak üreteceğimiz DLL dosyasının adını da başka bir değişkene aktararak yolumuza devam edelim.

[VB]

```
Dim DerlemeParametreleri As New
System.CodeDom.Compiler.CompilerParameters(Referanslarim, AssemblyAdi)
```

DerlemeParametreleri.GenerateExecutable = **True**
 DerlemeParametreleri.GenerateInMemory = **False**

[C#]

```
System.CodeDom.Compiler.CompilerParameters DerlemeParametreleri = new  

  System.CodeDom.Compiler.CompilerParameters(Referanslarım, AssemblyAdı);  

  DerlemeParametreleri.GenerateExecutable = false;  

  DerlemeParametreleri.GenerateInMemory = false;
```

Derleme işlemini yaparken yapmamız gereken ayarlar var. Bu ayarları derleyicimize bir **CompilerParameters** nesnesi olarak aktaracağız. **DerlemeParametreleri** değişkenimizi yaratırken referanslarımızı ve DLL adını aktardıktan sonra özel olarak **GenerateExecutable** özelliğini **false** olarak ayarlıyoruz. Böylece derleyicimiz bize tek başına çalışabilir bir dosya yaratmaktansa bir DLL dosyası yaratacak. Bir sonraki adımda da **GenerateInMemory** özelliğini **false** yaparak yaratılacak dosyanın uygulamamız ile aynı konuma, diske yazdırılmasını sağlıyoruz. Aksi halde yaratılan **Assembly** sadece hafızada tutulacak ve diske yazılmayacaktır. Sıra geldi dinamik olarak derlemeyeceğimiz kodu bir değişkene aktarmaya.

[VB]

```
Dim Kodum As String = <Kod>Public Class Deneme  

  Function Metin() As String  

    Return "Çalışıyor"  

  End Function  

End Class</Kod>.Value
```

[C#]

```
System.IO.StreamReader Okuyucu = new System.IO.StreamReader("Class1.cs");  

  string Kodum = Okuyucu.ReadToEnd();  

  Okuyucu.Close();
```

Bu noktada VB ile C# arasında farklı işlemler yaptım. VB'de doğrudan yaratacağım kodu uygulamanın içeresine gömerken C#'da derleyeceğim C# kodunu harici bir **Class1.cs** dosyasından çektim. Siz kendi uygulamalarınızda ister bu kodları farklı dosyalardan çekin ister metin işlemleri ile dinamik kod yaratın. İhtiyaçlarınıza göre uygun çözümü üretmek tamamen size kalmış. Önemli olan tek nokta aslında bu kodlarda hiçbir hatanın olmaması gereği, aksi halde derleme işlemi yapılamayacaktır.

[VB]

```
Dim Sonuc As System.CodeDom.Compiler.CompilerResults =  

  KodUretici.CompileAssemblyFromSource(DerlemeParametreleri, Kodum)
```

[C#]

```
System.CodeDom.Compiler.CompilerResults Sonuc =  

  KodUretici.CompileAssemblyFromSource(DerlemeParametreleri, Kodum);
```

Tüm ayarlarımız tamamlandığında göre doğrudan **CodeProvider** nesnemizin **CompileAssemblyFromSource** metodunu kullanarak derleme işlemini başlatabiliriz. Tabi bu esnada daha önce hazırlamış olduğumuz **DerlemeParametrelerini** de metoda parametre olarak aktarıyoruz. Derleme işlemimizi baştan sona tamamlayan kodumuzu bir bütün olarak inceleyelim.

[VB]

```

Dim KodUretici As New Microsoft.VisualBasic.VBCodeProvider
Dim Derleyici As System.CodeDom.Compiler.CodeCompiler =
KodUretici.CreateCompiler()

Dim Referanslarim As String() = {"System.dll"}
Dim AssemblyAdi As String = "Ornek.dll"

Dim DerlemeParametreleri As New
System.CodeDom.Compiler.CompilerParameters(Referanslarim, AssemblyAdi)
    DerlemeParametreleri.GenerateExecutable = True
    DerlemeParametreleri.GenerateInMemory = False

Dim Kodum As String = <Kod>Public Class Deneme
    Function Metin() As String
        Return "Çalışıyor"
    End Function
End Class</Kod>.Value

Dim Sonuc As System.CodeDom.Compiler.CompilerResults =
KodUretici.CompileAssemblyFromSource(DerlemeParametreleri, Kodum)

```

[C#]

```

Microsoft.CSharp.CSharpCodeProvider KodUretici = new
Microsoft.CSharp.CSharpCodeProvider();
System.CodeDom.Compiler.ICodeCompiler Derleyici = KodUretici.CreateCompiler();

String[] Referanslarim = {"System.dll"};
String AssemblyAdi= "Ornek.dll";

System.CodeDom.Compiler.CompilerParameters DerlemeParametreleri = new
System.CodeDom.Compiler.CompilerParameters(Referanslarim, AssemblyAdi);
DerlemeParametreleri.GenerateExecutable = false;
DerlemeParametreleri.GenerateInMemory = false;

System.IO.StreamReader Okuyucu = new System.IO.StreamReader("Class1.cs");
string Kodum = Okuyucu.ReadToEnd();
Okuyucu.Close();

System.CodeDom.Compiler.CompilerResults Sonuc =
KodUretici.CompileAssemblyFromSource(DerlemeParametreleri, Kodum);

```

Dinamik olarak DLL dosyası derlemek işte bu kadar kolay. Dinamik kod yaratma araçları son dönemde çok popüler. Veritabanına bağlanarak veritabanındaki nesneleri algılayıp uygun “Veri Katmanı” kodunu dinamik olarak oluşturan hazır uygulamalar olduğu gibi bazı durumlarda özel kodlar yazmak da gerekebiliyor. Böyle bir durumda artık siz de uygulamalarınıza farklı kaynaklardaki şartlara uygun kodu dinamik olarak üretebilir ve bir DLL olarak farklı uygulamalara aktarabileceğiniz gibi kendi uygulamalarınızda da kullanabilirsiniz. Yarattığınız DLL dosyasını hemen uygulamanızda kullanmak isterseniz bu sefer dinamik olarak Assembly kullanımını ve Reflection konusuna eğilmenizde fayda var.

Hepinize kolay gelsin.

Daron YÖNDEM

Java ve Silverlight kardeşliği.

Silverlight'in sunucu tarafındaki programlama dillerinden ve sunucu platformundan tamamen bağımsız olduğundan sürekli bahsediyoruz. Bu çerçevede daha önceki yazılarımından birince [PHP ile Silverlight 2.0](#) kullanımına değinmiştim. Bu yazımızda da Java ile Silverlight kullanımına değineceğiz.

Örneğimizde Java tarafından hazırladığımız bir web servisini Silverlight 2.0 tarafında Visual Studio içerisinde kullanacağınız. Visual Studio ve .NET altyapısı ile rahatlıkla WSDL uyumlu web servislerini kullanabildiğimizi düşünürsek aynı standartlara uygun bir web servisinin Java ile hazırlanmış olması durumunda herhangi bir sorun yaşamayacağımıza kesin gözü ile bakabiliriz. İlk olarak Java tarafında aşağıdaki kodumuz ile basit bir web servisi hazırlayalım.

```
package com.daron.ws;

public class wsclass {

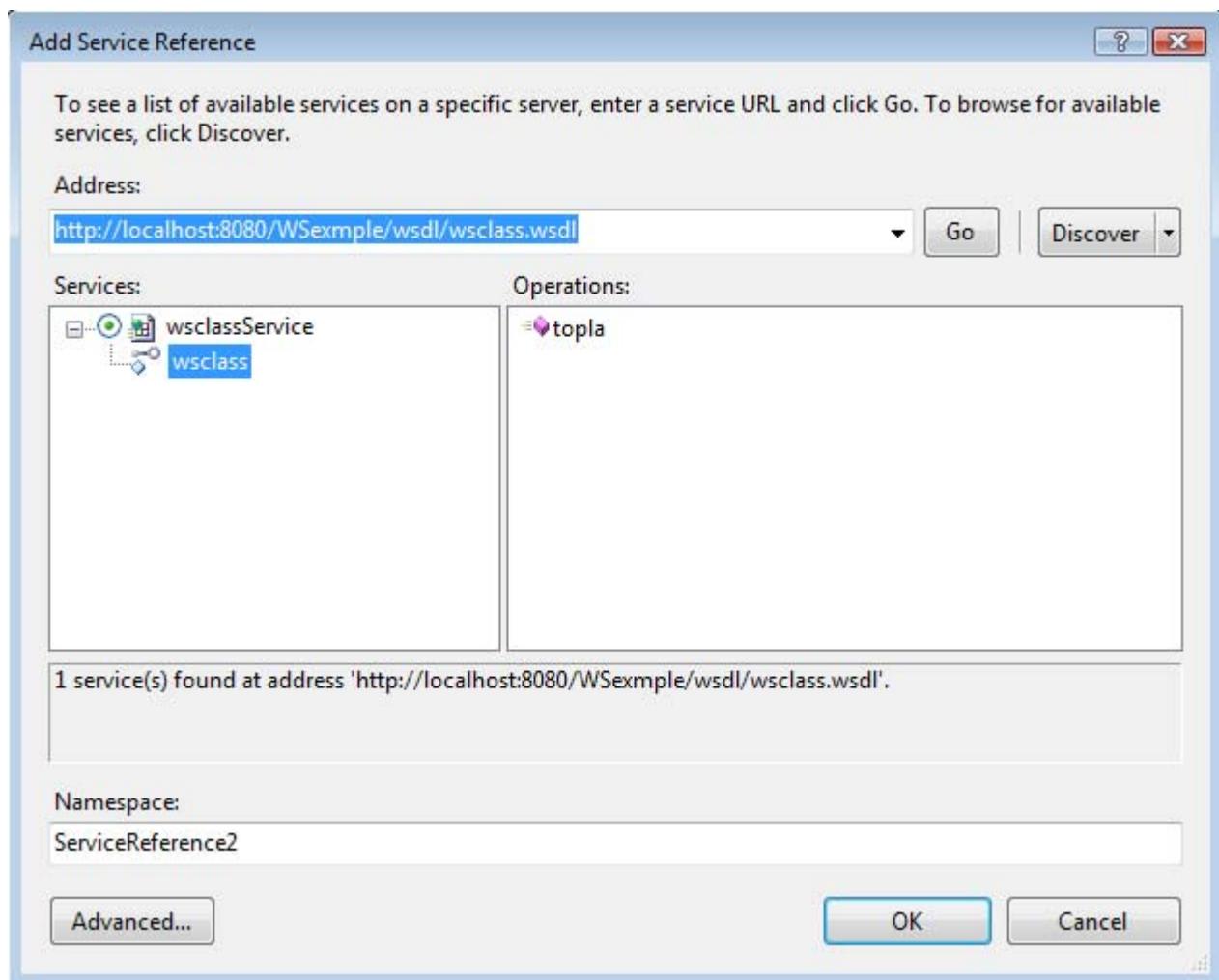
    public int topla(int x, int y)
    {
        return x+y;
    }
}
```

Yukarıdaki ufak kod ile aslında basit bir metod tanımlamış oluyoruz. Metodumuz aldığı iki integer parametreyi toplayıp geri döndürüyor. Bu parametreler ve metodun yapacağı işlemler sizin örneklerinizde çok daha farklı olabilir. Şu an için amacımız Silverlight tarafından Java'ya veri gönderip geriye sonuç alabiliyor olmak.

Eclipse üzerinden WSDL dosyasını da otomatik olarak yukarıdaki metod üzerinden yarattıktan sonra artık sıra geliyor bu servisi Silverlight tarafında kullanmaya. Silverlight 2.0 uygulamamızı yine Visual Studio içerisinde geliştireceğimiz için Java servisinin bulunduğu siteyi Visual Studio içerisinde de açmanız daha rahat bir çalışma ortamı yaratacaktır. Basit bir şekilde Visual Studio 2008 içerisinde "File / Open Web Site" dierek Java ile hazırlanmış siteyi açabilirsiniz. Tabi Java dosyalarını sadece birer dosya olarak göreceksiniz, düzenleme şansınız olmayacak. Siteyi açtıktan sonra "File / Add / New Project" dierek Silverlight projenizi sitenize ekleyebilirsiniz. Silverlight uygulamasını çalıştıracak olan örnek HTML dosyası otomatik olarak Java sitenize eklenecektir.

Web servisini referans alalım...

Web servisini referans olarak ekleyebilmeniz için tabii ki servisin çalışıyor olması gereklidir. Bunun için Eclipse üzerinden Tomcat'i kullanabilirsiniz. IIS yüklü bir makinede çalışıyorsanız 8080 gibi harici bir port vermeyi unutmayın. Web servisini tarayıcınızda çalıştırıldıktan sonra adresini kopyalayarak Silverlight projenize sağ tıklayarak "Add Service Reference" dierek referans ekleme işlemini tamamlayabilirisiniz.



Java web servisimizi referans olarak ekliyoruz.

Referans ekleme işlemi tamamlandıında artık Silverlight ile herhangi bir web servisini kullanır gibi Java web servisimizi de kullanabiliyoruz. Örneğimizi çalıştırabilmek için ilk olarak Silverlight ekranımıza iki metin kutusu ve bir de düğme yerleştirelim.

```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Height="48.411"
Margin="33.2830009460449,22.693000793457,115.582000732422,0"
VerticalAlignment="Top" Text="TextBox" TextWrapping="Wrap" x:Name="Kutu1"/>
        <TextBox Height="48.95"
Margin="33.2830009460449,75.1039962768555,115.582000732422,0"
VerticalAlignment="Top" Text="TextBox" TextWrapping="Wrap" x:Name="Kutu2"/>
        <Button Height="52.95" HorizontalAlignment="Stretch"
Margin="92.2839965820313,0,165.50700378418,88.1999969482422"
VerticalAlignment="Bottom" Content="Button" x:Name="Dugme"/>
    </Grid>
</UserControl>
```

Her şey hazır olduğuna göre artık web servislerimizi kodumuz ile tanımlayıp kullanabiliriz.

[VB]

```

Partial Public Class Page
    Inherits UserControl

    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme.Click
        Dim Servisim As New ServiceReference1.wsclassClient
        AddHandler Servisim.toplaCompleted, AddressOf Servisim_toplaCompleted
        Servisim.toplaAsync(Kutu1.Text, Kutu2.Text)
    End Sub

    Private Sub Servisim_toplaCompleted(ByVal sender As Object, ByVal e As
ServiceReference1.toplaCompletedEventArgs)
        Dugme.Content = e.Result
    End Sub
End Class

```

[C#]

```

namespace SilverlightApplication2
{
    public partial class Page : UserControl
    {
        ServiceReference1.wsclassClient Servisim = new
        SilverlightApplication2.ServiceReference1.wsclassClient();

        public Page()
        {
            InitializeComponent();
            this.Dugme.Click += new RoutedEventHandler(Dugme_Click);
            Servisim.toplaCompleted += new
EventHanlder<SilverlightApplication2.ServiceReference1.toplaCompletedEventArgs>(Servis
im_toplaCompleted);
        }

        void Dugme_Click(object sender, RoutedEventArgs e)
        {
            Servisim.toplaAsync(int.Parse(Kutu1.Text), int.Parse(Kutu2.Text));
        }

        void Servisim_toplaCompleted(object sender,
SilverlightApplication2.ServiceReference1.toplaCompletedEventArgs e)
        {

```

```
        Dugme.Content = e.Result.ToString();  
    }  
}  
}
```

Konumuz Silverlight ile web servisleri kullanımı olmadığı için yukarıdaki kodun detaylarına girmeyeceğim. Bu konuda detaylı bir yazıyı aşağıdaki adresten inceleyebilirsiniz.

<http://daron.yondem.com/tr/PermaLink.aspx?guid=19fe09b2-2987-4369-a5d5-58e0641c8d6b>

Kodları incelediğimizde yaptığımız şeyin aslında ASP.NET ile hazırlanmış bir web servisi kullanmaktan farklı olmadığını görüyoruz. Java ile yazılmış olan web servisimiz yine Silverlight tarafından asenkron olarak kullanılabilir.

Projenizi Visual Studio içerisinde Build ederek Silverlight XAP dosyasını oluşturuktan sonra siteyi Tomcat üzerinden çalıştırınak zorunda olduğunuzu unutmayın. Aksi halde web servisi çalışmayaçağı için Silverlight hata verecektir.

Sonuç

Silverlight'in güzelliklerinden faydalananmak için ASP.NET tarafında olmanız şart değil. İster Java ister PHP ister herhangi bir sunucu taraflı programlama dili kullanın Silverlight ile kullanıcı arayüzünüzü hazırlayabilirsiniz.

Java ile WSDL uyumlu web servisi hazırlayıp çalıştırabilme :) konusundaki yardımlarından dolayı sevgili **Bilge Başaltun**'a buradan çok teşekkür ediyorum.

Hepinize kolay gelsin.

Daron YÖNDEM

Kendi Silverlight Yükle mesajınızı göstermeniz için ipuçları

Silverlight kullanılan web sitelerin artış ile aslında kullanıcılar da Silverlight yükletme konusunda ısrar giderek artıyor :) Bu konunun bir ısrar olmaması ve kullanıcıların gönül rahatlığı ile Silverlight Runtime'ını yükleyemeleri için aslında yükleme sürecinin öncesindeki kullanıcı deneyimi çok önemli.

Eğer istemicide Silverlight yüklü değil ise sizin OBJECT tagları ile sayfaya yerleştirdiginiz uygulama gösterilmeyecektir. Bunun yerine OBJECT tagları arasındaki HTML kodu kullanıcıya gösterilir. Visual Studio ve Expression Blend ile yeni bir proje yarattığınızda söz konusu HTML kodu varsayılan şekli ile aşağıdaki gibi gelir.

[HTML]

```
<object data="data:application/x-silverlight-2," type="application/x-silverlight-2"
width="100%" height="100%">
<param name="source" value="ClientBin/Carousel.xaml" />
<param name="onerror" value="onSilverlightError" />
<param name="background" value="white" />
<param name="minRuntimeVersion" value="2.0.31005.0" />
<param name="autoUpgrade" value="true" />
<a href="http://go.microsoft.com/fwlink/?LinkId=124807" style="text-decoration:
none;">
    
</a>
</object>
```

Yukarıdaki OBJECT tagları arasında renkli olarak gördüğümüz kısım sadece Silverlight yüklü olmadığında gösterilecektir. Bu kısma istediğiniz HTML kodunu koyabilirsiniz. Buradaki standart link Silverlight'in Microsoft sitesinde yükleme sayfasına yönlendirirken diğer de standart "Install Silverlight" görselini gösterir.



Standart Silverlight Yükleme mesajı.

Kendi özel "Silverlight Yükle" görselinizi ve mesajınızı hazırlarken kullanıcıları korkutup kaçırılmamak adına sizlere birkaç tavsiyem olacak.

- İnsanlara "Silverlight Yükle" derken neden yüklemelerini istedığınızı belirtin.
- Kullanıcıların yüklemeden önce de sitenizin nasıl bir şey olduğunu görmeleri sağlayın, merak uyandırın. Özetle Silverlight yüklerlerse nasıl bir şeyle karşılaşacaklarını görsünler ki yüklemeye daha sıcak baksınlar.

Aşağıda bulabileceğiniz örnekte Silverlight ile çalışan bir sitenin Silverlight yüklü olmadığından da nasıl gösterildiğini inceleyebilirsiniz. Standart "Install Silverlight" görseli yerine böyle bir deneyim çok daha çekici olacaktır.



Örnek Silverlight Yükleme Ekranı

Hepinize kolay gelsin.

Daron YÖNDEM

NET için De-Compile işlemleri ve Obfuscation

İster VB olsun ister C#, ister web ister Windows uygulaması olsun yazdığımız tüm kodların derlenerek (Compile) bir EXE veya DLL haline dönüştürüldüğünü biliyoruz. Aslında .NET içerisinde yapılan işlem sizin yazdığınız herhangi bir .NET dilindeki kodun **MSIL** (Microsoft Intermediate Language)'a çevrilmesidir. İşte tam bu noktada akla gelen ilk soru; acaba bu çeviri işleminin tersini yapmak mümkün mü? Yani elimizdeki DLL veya EXE dosyasından yola çıkarak VB veya C# kodumuzu geri alabilir miyiz? Cevap: **Evet**.

Şu andan itibaren yapacaklarımız hedef olarak kullanacağınız uygulamanın lisans sözleşmesine göre yeri geldiğinde suç teşkil edebilir. O nedenle sizi özellikle uyarmak istiyorum. Çoğu zaman De-Compile işlemleri yaparkenki amacımız yazdığımız kodun nasıl derleyici tarafından MSIL'e çevrildiğini incelemek veya kaynak kodunu kaybettigimiz ve bize ait olan bir uygulamanın kodlarına ulaşmak olacaktır. Diğer yandan lisans sözleşmesi ile aykırı düşmediği sürece farklı uygulamaları da De-Compile ederek arka planda farklı işlemlerin nasıl yapıldığını inceleme şansınız da olabilir.

.NET tarafına geçtiğimizde herhangi bir DLL veya EXE'nin aslında MSIL kodları içerdiginden bahsetmiştik. Tabi ki bu **MSIL** kodları doğrudan bilgisayarlar tarafından çalıştırılabilir kodlar degiller. O nedenle içerisinde MSIL bulunan bir .NET yapısının çalışabilmesi için hedef makinede .NET Framework'ün yüklü olması gerekiyor. .NET Framework içerisindeki **CLR** (Common Language Runtime) bizim MSIL kodumuzu makine diline çevirerek çalışmasını sağlayacaktır. Kabaca baktığımızda De-Compile yolunda bizim ilk olarak elimizdeki DLL veya EXE içerisinde MSIL kodunu alarak çıkarmamız gerekecek. Bunun için doğrudan .NET Framework SDK paketi ile beraber gelen **MSIL DisAssemblers (ILDASM)** uygulamasını kullanabiliriz.

IL DASM Kullanımı

Bilgisayarınıza .NET Framework SDK paketini kurduktan sonra doğrudan “Başlat” menüsünden ulaşabileceğiniz ILDASM programını Visual Studio yükleme konumu içerisinde SDK klasörü altında da bulabilirsiniz. Programı açtıktan sonra “File / Open” menüsünden istediğiniz bir .NET DLL veya EXE dosyasını açma şansınız olacaktır. Deneme amaçlı olarak gelin mini bir Windows uygulaması yazalım ve ILDASM ile açarak alacağımız sonucu görelim. Uygulamamız içerisinde birer TextBox, Button ve Label bulunacak. Basit bir şekilde düğmeye basıldığında TextBox içerisindeki değeri Label içerisinde aktaracağız.

[VB]

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Label1.Text = TextBox1.Text
    End Sub
End Class
```

[C#]

```
namespace WindowsFormsApplication1
{
```

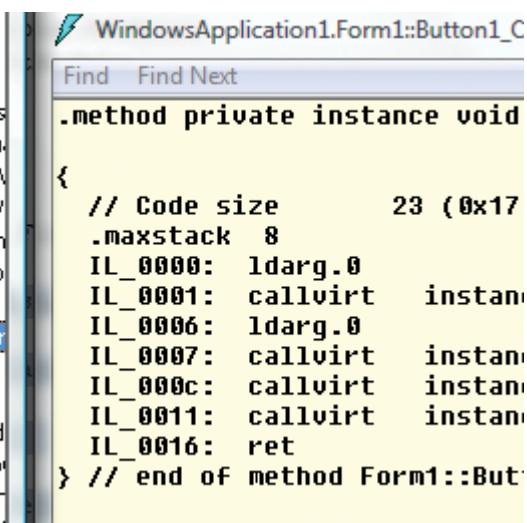
```

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        label1.Text = textBox1.Text;
    }
}

```

Yukarıda yazdığımız kodlar ile oluşturduğumuz uygulamayı ILDASM ile açarak sonucu inceleyelim. Uygulamanın ilk açılan penceresinde bizim EXE'ye ait tüm sınıflar ve namespace'ler gözükmek olacaktır. Eğer herhangi bir nesnenin tanımı veya metodu ile ilgili MSIL kodunu görmek isterseniz doğrudan çift tıklayarak yeni bir pencerede kodların açılmasını sağlayabilirsiniz.



The screenshot shows the ILDASM interface. On the left, there is a tree view of the assembly symbols, including classes like WindowsApplication1.Form1, methods like .ctor, and events like Button1_Click. The right pane displays the assembly code for the Button1_Click method. The code is:

```

WindowsApplication1.Form1::Button1_C
Find Find Next
.method private instance void
{
    // Code size      23 (0x17)
    .maxstack 8
    IL_0000: ldarg.0
    IL_0001: callvirt  instance
    IL_0006: ldarg.0
    IL_0007: callvirt  instance
    IL_000c: callvirt  instance
    IL_0011: callvirt  instance
    IL_0016: ret
} // end of method Form1::But

```

ILDASM içerisinde EXE'mizin MSIL kodları açıkça gözükmüyor

Hazırladığımız örnek uygulamanın **Button_Click** durumundaki MSIL kodunu bulduğumuzda aşağıdaki sonuç ile karşılaşıyoruz.

[MSIL]

```

.method private instance void Button1_Click(object sender,
                                             class [mscorlib]System.EventArgs e) cil managed
{
    // Code size      23 (0x17)
    .maxstack 8
    IL_0000: ldarg.0
    IL_0001: callvirt  instance class [System.Windows.Forms]System.Windows.Forms.Label
WindowsApplication1.Form1::get_Label1()
    IL_0006: ldarg.0

```

```

IL_0007: callvirt instance class
[System.Windows.Forms]System.Windows.Forms.TextBox
WindowsApplication1.Form1::get_TextBox1()
IL_000c: callvirt instance string
[System.Windows.Forms]System.Windows.Forms.TextBox::get_Text()
IL_0011: callvirt instance void
[System.Windows.Forms]System.Windows.Forms.Label::set_Text(string)
IL_0016: ret
} // end of method Form1::Button1_Click

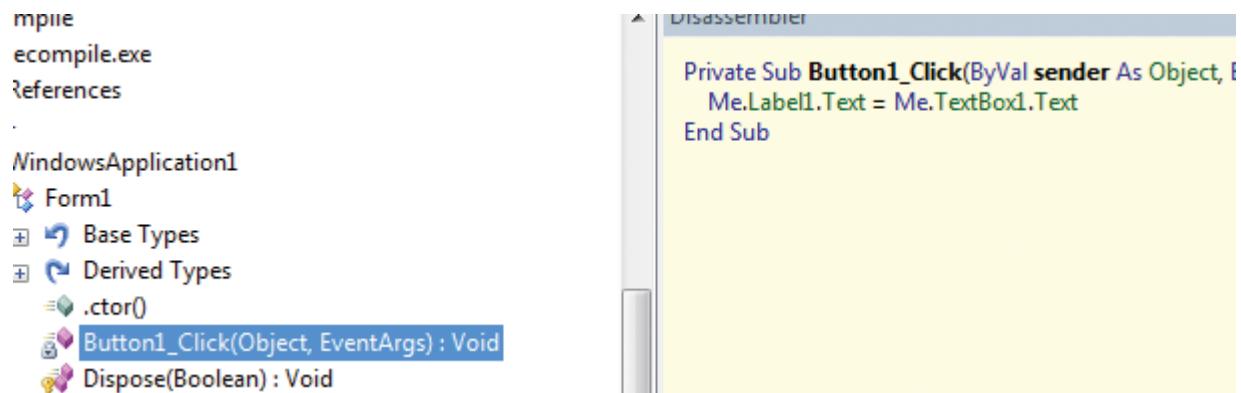
```

Yukarıdaki MSIL kodu normal şartlarda CLR tarafından makine koduna çevrilerek hedef ortamda çalıştırılıyor. Artık MSIL kodumuzu aldığıma göre bu kodu VB veya C# koduna çevirmemiz lazım. Tabi bu iş o kadar kolay değil ve tek tek elle yapılabilecek bir iş de değil. O nedenle bu sefer de farklı bir araç kullanacağız.

Reflector iş başında

Lutz Roeder tarafından yazılmış bir program olarak Reflector'ı <http://www.red-gate.com/products/reflector/> adresinden bilgisayarınıza indirebilirsiniz. Program aslında bir önceki adımda anlattığım MSIL çözme işlemini de kendi içinde yapabiliyor. Bununla kalmayıp çözdüğü MSIL kodunu istediğiniz .NET diline de çevirebiliyor.

Programı çalıştırdıktan sonra “File / Open” menüsünden istediğiniz bir EXE veya DLL dosyasını seçebilirsiniz. Uygulamanın ana penceresindeki sınıf listesine hemen seçtiğiniz program da gelecektir.



Reflector ile kaynak kodunu görebiliyoruz.

Ufak bir gezinti ile istediğiniz sınıfın veya metodun koduna doğrudan ulaşabilirsiniz. Reflector arayüzündeki “Programlama Dili” seçenekinde VB, C#, Delphi ve IL seçenekleri bulunuyor. Bir önceki bölümde hazırladığımız uygulamamızı açarak **Button.Click** durumundaki kodu farklı dillerde Reflector ile alıp inceleyelim.

[VB]

```

Private Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs)
    Me.Label1.Text = Me.TextBox1.Text
End Sub

```

[C#]

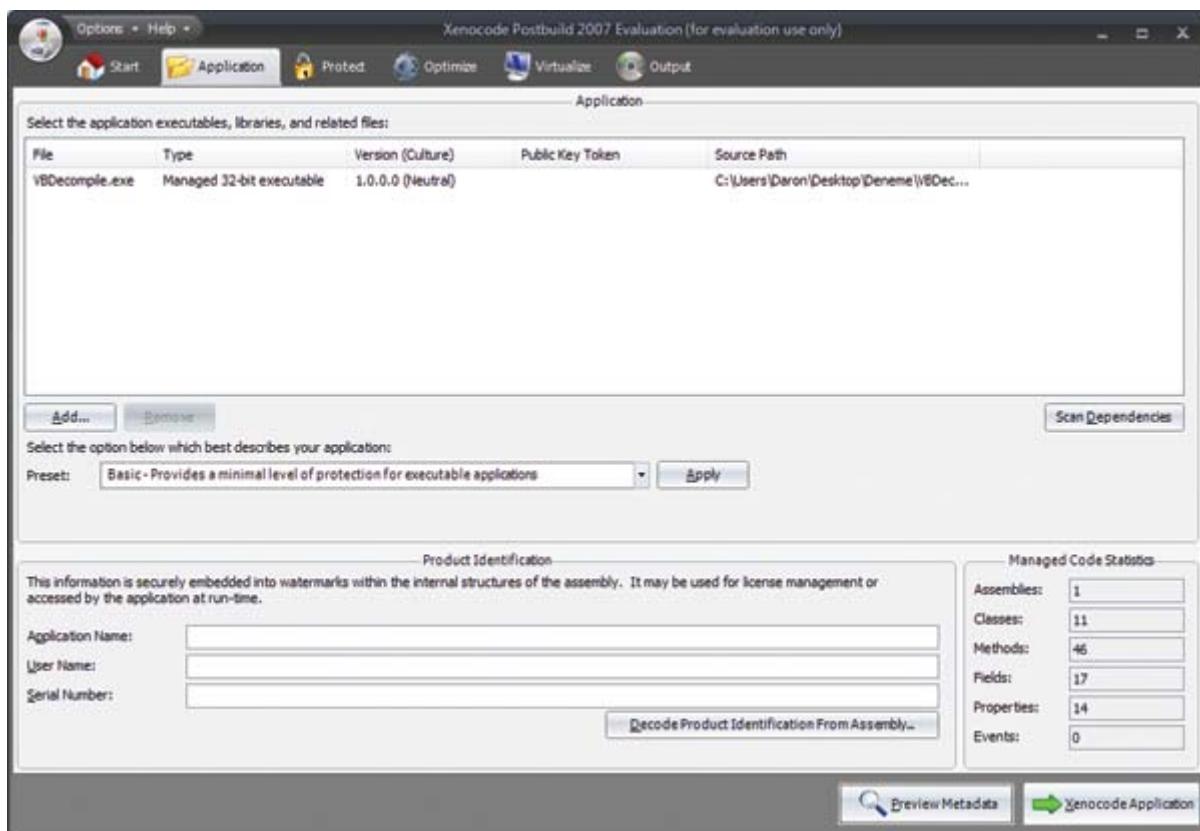
```
private void Button1_Click(object sender, EventArgs e)
{
    this.Label1.Text = this.TextBox1.Text;
}
```

Yazdığımız kodlar ile Reflector'ın bize verdiği kodlar tam olarak aynı değil. Bu durum zaten çok normal. Çünkü MSIL koduna çeviri esnasında aslında çoğu şey değişiyor. Örneğin tanımladığımız değişkenlerin bize özel olan isimlendirmeleri yok oluyor veya bizim kullandığımız bazı kısa metotlar uzun şekilleri ile yazılabilüyor. Hatta özellikle VB içerisindeki casting kolaylıklarını Compile esnasında farklı değişikliklere neden olabiliyor. Bu durumda De-Compile ile aldığımız kod da yazdığımız koddan biraz farklı oluyor. Yine de elimizde çalışır durumda bir kod olduğuna kesin gözü ile bakabiliriz.

Nasıl engelleriz? Obfuscation!

Herhalde çoğuınız “tüm kodlarımız gözler önünde” endişesi içerisindeiniz. Aslında durum gerçekten de öyle. Tabi bu durumun birçok faydası var. Kişisel olarak itiraf etmek gerekirse farklı yazılımları De-Compile ederek çok şey öğrendiğimi söyleyebilirim. Bir defasında da kendi ürettiğimiz bir yazılımı De-Compile etmemiz gerekmisti, gerçekten hayat kurtarmıştı. Peki bunu nasıl engelleyebiliriz? İlk olarak şunu açıkça belirtiyim, herhangi bir .NET uygulamasından MSIL kodunun alınmasını engellemenin hiçbir yolu yok. Yapabileceğimiz tek şey MSIL kodunun okunabilirliğini azaltmak için işlevsel olarak aynı işi gören fakat daha karışık bir MSIL kodu yaratmak. Bu işlem **obfuscation** olarak adlandırılıyor.

Obfuscation ile ilgili sektörde çok sayıda ücretli yazılım bulabilirsiniz. Biz bunlardan **Xenocode'aait Postbuild 2008** adındaki ticari yazılımı kullanarak obfuscation ile neler yapabildiğimize bakacağız. XenoCode'u ilk açtığımızda karşımıza hemen bir uygulama listesi geliyor. Bu listeye bir önceki adımda kendi hazırladığımız EXE dosyasını ekleyerek uygulamanın üst menüsünden “Protect” tabına geçiyoruz. Burada sadece Windows'da çalışacak EXE dosyalarına uygulanabilecek özel bir koruma yöntemi olan “**Surpress ILDASM**” seçeneğinin işaretini kaldırımız gerek. Bu seçenek DLL'lere zaten uygulanamayacaktır. Ekranın sağ tarafında korumak istediğimiz sınıfların ve metodların bir listesini işaretleyebiliyoruz. Tüm ayarları tamamladıktan sonra uygulamanın sağ altındaki “**XenoCode Application**” düğmesine basıyoruz.



Obfuscation işlemi için yollardayız

Obfuscation işlemini tamamladıktan sonra sıra geldi testlerimizi yapmaya. İlk olarak uygulamamızı ILDASM ile açarak bakalım MSIL kodumuz ne hale gelmiş.

[MSIL]

```
.method private instance void x44d0c0526a414989(object xe0292b9ed559da7d,
    class [mscorlib]System.EventArgs xfbf34718e704c6bc) cil
managed
{
    // Code size    23 (0x17)
    .maxstack 8
    IL_0000: ldarg.0
    IL_0001: callvirt instance class [System.Windows.Forms]System.Windows.Forms.Label
WindowsApplication1.xaa4f033827d75b4d::get_x029e304eb4c44750()
    IL_0006: ldarg.0
    IL_0007: callvirt instance class
[System.Windows.Forms]System.Windows.Forms.TextBox
WindowsApplication1.xaa4f033827d75b4d::get_x77691a2cfb8f8048()
    IL_000c: callvirt instance string
[System.Windows.Forms]System.Windows.Forms.TextBox::get_Text()
    IL_0011: callvirt instance void
[System.Windows.Forms]System.Windows.Forms.Label::set_Text(string)
    IL_0016: ret
} // end of method xaa4f033827d75b4d::x44d0c0526a414989
```

Gördüğünüz gibi aslında çok büyük bir değişiklik yok. Sadece sınıfların ve metodların isimleri değiştirilerek karışık isimler verilmiş. Aynı uygulamayı **Reflector** ile açtığımızda ise aşağıdaki kodları elde ediyoruz.

[VB]

```
Private Sub x44d0c0526a414989(ByVal xe0292b9ed559da7d As Object, ByVal
xbf34718e704c6bc As EventArgs)
    Me.x029e304eb4c44750.Text = Me.x77691a2cfb8f8048.Text
End Sub
```

[C#]

```
private void x44d0c0526a414989(object xe0292b9ed559da7d, EventArgs
xbf34718e704c6bc)
{
    this.x029e304eb4c44750.Text = this.x77691a2cfb8f8048.Text;
}
```

Kodlar epey okunurluluğunu kaybetmiş durumda. Bizim örneğimizde sadece tek bir satır kod bulunduğu için neyin ne olduğunu anlamak çok zor olmuyor. Fakat binlerde satırdan oluşan uygulamaların kodlarından anlaşılabilir bir sonuç çıkarmak neredeyse imkânsız olacaktır.

Hepinize kolay gelsin.

Daron YÖNDEM

Reflection nedir?

Başlık olarak “**Reflection**” yazdıktan sonra ardına sayfalarca açıklama ve örnek konulabilir. Hatta bu konuda ayrı bir kitap bile yazılabilir. Reflection’ın çok farklı kullanıcılar var. Özetleyerek hızlı bir şekilde tanımlamak istersek aslında Reflection bize hakkında bilgi sahibi olmadığını programatik nesnelerle ilgili çalışma zamanında (run-time) bilgi alabilmemize olanak tanıyan bir metottur. Peki böyle bir şeye neden ihtiyacımız olsun? En basit örnek gerçek zamanlı olarak uygulamalara farklı DLL dosyalarının bağlandığı durumları gösterebiliriz. Böyle bir durumda kaynak konumdaki sınıflar veya metodlar ile ilgili herhangi bir bilgi bulunmaz. Söz konusu bu bilgilerin program çalışırken elde edilerek kullanılması gereklidir. Gelin ilk olarak Reflection’ın yapısını ve sistemini tanımak adına tek bir uygulama içerisinde nasıl kullanılabileceğimize göz atalım. Örnek uygulamamızda aşağıdaki şekli ile tanımlanmış bir Urun sınıfı kullanacağımız.

[VB]

Public Class Urun

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

Sub New()

End Sub

```
Sub New(ByVal adi As String)
    Me.Adi = adi
End Sub
```

```
Function Uyari() As String
    Return "Ürünün adı: " & Me.Adi
End Function
```

End Class

[C#]

```
public class Urun
{
```

```
    private string PAdi;
    public string Adi
    {
        get { return PAdi; }
```

```

    set { PAdi = value; }

}

public Urun()
{
}

public Urun(string adi)
{
    this.Adi = adi;
}

public string Uyari()
{
    return "Ürünün adı: " + this.Adi;
}
}

```

Uygulamamız içerisinde iki adet düğme yer alacak ve kullanacağımız Windows penceresinde global olarak tanımlanmış bir de **Object** tipinde değişkenimiz bulunacak.

[VB]

```
Dim BirUrun As Object
```

[C#]

```
object BirUrun;
```

Uygulama içerisindeki düğmelerden birine basıldığında global **BirUrun** değişkenimiz yeni bir **Urun** değişkenine dönüştürülecek.

[VB]

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    BirUrun = New Urun
End Sub

```

[C#]

```

private void button1_Click_1(object sender, EventArgs e)
{
    BirUrun = new Urun();
}

```

Programımız içerisinde diğer düğmeye basıldığında **BirUrun** adındaki değişkenimizin **Adı** özelliğini değiştirek **Uyari** adındaki metodunu kullanmak istiyoruz. Fakat Visual Studio içerisinde maalesef ki **BirUrun** adındaki değişkenle beraber **Urun** tipine ait Intellisense

desteği gelmeyecektir. Aslında bu durumun haklı bir nedeni var. İkinci düğmeye basıldığında **BirUrun** adındaki değişkenin tipinin **Object** mi yoksa **Urun** mü olacağı belli değil. İşte tam da istediğimiz ortamı yaratmış olduk. Kullanacağımız nesnenin tipi belirsiz ve biz ona ait bazı özelliklerini kullanmak istiyoruz. Bu durumda ilk olarak ikinci düğmeye basıldığında gerçekten **BirUrun** değişkeninin tipi **Urun** mü yoksa değil mi sorusunu kontrol etmemiz lazım.

[VB]

```
If TypeOf (BirUrun) Is Urun Then
    End If
```

[C#]

```
if ((BirUrun) is Urun)
{
}
```

Buraya kadar her şey çok kolay. Bundan sonra eğer **IF** kontrollerimize olumlu sonuç dönüyorsa ilk olarak gidip nesnenin **Adı** özelliğini bulmamız ve ona bir değer aktarmamız gerekiyor.

[VB]

```
BirUrun.GetType.GetProperty("Adı").SetValue(BirUrun, "Daron", Nothing)
```

[C#]

```
BirUrun.GetType().GetProperty("Adı").SetValue(BirUrun, "Daron", null);
```

Yukarıdaki kod ile elimizdeki nesnenin tipini bilmeden onun **Adı** adındaki özelliğini (**property**) yakalayarak değerini **Daron** olarak değiştiriyoruz. Kodumuzu detaylı olarak adım adım bakacak olursak ilk aşamada nesnenin tipini **GetType** ile alıyoruz. Sonrasında ise tipini yakaladığımız nesnenin **GetProperty** ile **Adı** adındaki özelliğini alarak **SetValue** ile söz konusu özelliğin değerini değiştiriyoruz. **SetValue** metodu toplam üç parametre alıyor; bunlardan ilki değer değişikliği yapılacak nesnenin kendisi, ikincisi yeni atanacak olan değer, üçüncüsü ise eğer değiştirilecek olan özellik (**property**) indeksli ise söz konusu indeks değeri. Bizim örneğimizde indeksli bir özellik olmadığı için bu parametreyi boş geçiyoruz.

Değer atamamızı tamamladığımıza göre bu sefer de sıra geldi **BirUrun** değişkenimize ait **Uyarı** metodunu çalıştırımıya. Metodumuz bize bir **String** döndürecek biz de onu doğrudan bir mesaj kutusu ile kullanıcıya göstereceğiz.

[VB]

```
BirUrun.GetType.InvokeMember("Uyarı", Reflection.BindingFlags.InvokeMethod,
    Nothing, BirUrun, Nothing)
```

[C#]

```
BirUrun.GetType().InvokeMember("Uyari",
    System.Reflection.BindingFlags.InvokeMethod, null, BirUrun, null).ToString();
```

Reflection kullanarak türü bilinmeyen bir nesnenin bir metodunu çalıştırırmak için **InvokeMember** metodundan faydalananmamız gerekiyor. **InvokeMember** aslında çok geniş kullanımı olan bir metod, biz şimdilik sadece bir çeşit kullanımına değineceğiz. Örneğimizde **InvokeMember** bir metod çalıştıracağı için ilk parametresinde çalıştırılacak olan metodun adını ikincisinde **BindingFlags.InvokeMethod** ile bir **Metod** çalıştırılacağını belirtiyoruz. Üçüncü parametre bizim şimdilik kullanım alanımız dışında kalan Binding'lerle ilgili, aynı şekilde beşinci parametre de boş bırakılarak geçilecek. Dördüncü parametrede ise hedef nesnemiz olan **BirUrun** değişkenimizi atayacağız. Böylece metodumuzu da çalıştırılmış olduk.

Dinamik DLL Kullanımı

Kabaca Reflection'ın nasıl kullanılabildiğine dair bir örnek yaptıktan sonra artık sıra geldi harici bir DLL dosyasının çalışma anında programımıza ekleyerek içerisindeki yapıları kullanmaya. Bu çeşit bir işlevselligi özellikle gerçek zamanlı DLL derlemesi ile birleştirdiğinizde çok farklı bir dünyaya kapı açmış olacaksınız. Hedef olarak kullanacağımız DLL dosyasını aşağıdaki kodlardan yaratacağız.

[VB]

```
Public Class Deneme
    Function Metin() As String
        Return "Çalışıyor"
    End Function
End Class
```

[C#]

```
public class Deneme
{
    string Metin()
    {
        return "Çalışıyor";
    }
}
```

Yarattığımız DLL dosyasını uygulamamız ile aynı konuma yerleştirdikten sonra aşağıdaki kod ile DLL'imizi kullanmaya başlayabiliyoruz.

[VB]

```
Dim BirAssembly As Reflection.Assembly =
    Reflection.Assembly.LoadFrom("ornek2.dll")
```

[C#]

```
System.Reflection.Assembly BirAssembly =
    System.Reflection.Assembly.LoadFrom("ornek2.dll");
```

Artık yukarıda tanımladığımız Assembly üzerinden **Reflection** kullanarak ilerleyebiliriz. İlk olarak **Deneme** adında sınıfımızdan bir **instance** almamız gerekecek. Bunun için **Deneme** tipini bulmamız lazım.

[VB / C#]

```
BirAssembly.GetModule("Ornek2.dll").GetType("Deneme")
```

Assembly üzerinden modülüümüzü yakalıyor sonra da **Deneme** adındaki tipinizi buluyoruz. Tabi tipi bulmak yeterli değil, söz konusu tipte bir değişken yaratmamız gerekiyor. **Activator** sınıfını kullanarak bu tip üzerinden bir instance yaratarak **Sınıf** adında bir değişkene aktaracağız.

[VB]

```
Dim Sınıf =  
Activator.CreateInstance(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme"))
```

[C#]

```
object Sınıf =  
Activator.CreateInstance(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme"))  
;
```

Yarattığımız sınıfın maalesef özellikleri otomatik olarak gelmeyecek. O nedenle **Metin** adındaki metodumuzu da elle bularak çalıştırmak zorundayız.

[VB]

```
BirAssembly.GetModule("Ornek2.dll").GetType("Deneme").GetMethod("Metin").Invoke(Sınıf, Nothing)
```

[C#]

```
BirAssembly.GetModule("Ornek2.dll").GetType("Deneme").GetMethod("Metin").Invoke(Sınıf, null)
```

Yine **Assembly** üzerinden yola çıkarak bu sefer daha da ileri gidiyoruz. **Deneme** sınıfımızı bulduktan sonra içerisinde **Metin** adındaki metodumuzu buluyor ve doğrudan **Invoke** ile söz konusu методу çalıştırıyoruz. **Invoke** methodu bizden iki parametre istiyor; bunlardan ilki ana sınıfın kendisi. Bir önceki adımda yakaladığımız sınıfı buraya parametre olarak aktarıyoruz. Diğer ise bizim kullanmayıcağız Binding parametresi.

Metin metodumuz çalıştırıldığında geriye bir String değişkeni döndürüyor. Bu değişkeni de bir mesaj kutusu ile kullanıcıya göstermek istersek uygulamamızın tam kodunun aşağıdaki şekilde sonlanması gerekiyor.

[VB]

Public Class Form2

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim BirAssembly As Reflection.Assembly =
Reflection.Assembly.LoadFrom("ornek2.dll")

    Dim Sinif =
Activator.CreateInstance(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme"))

MsgBox(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme").GetMethod("Me
tin").Invoke(Sinif, Nothing))
End Sub
End Class

```

[C#]

```

namespace CSReflection
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Reflection.Assembly BirAssembly =
System.Reflection.Assembly.LoadFrom("ornek2.dll");

            object Sinif =
Activator.CreateInstance(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme"));

            MessageBox.Show(BirAssembly.GetModule("Ornek2.dll").GetType("Deneme").GetMethod(
"Metin").Invoke(Sinif, null).ToString());
        }
    }
}

```

Böylece harici bir DLL dosyasını yükleyerek istediğimiz metodu dinamik olarak kullanabildik. Farklı durumlarda isterseniz bir DLL içerisinde tüm metod, sınıf ve özelliklerin listelerini alabilir hatta bunları LINQ sorguları ile tarayabilirsiniz.

[VB]

```

Dim Metodlar = From Gelenler In BirAssembly.GetModule("Ornek2.dll").GetTypes
Where Gelenler.GetMethod("Metin") IsNot Nothing

```

[C#]

```
var Metodlar = from Gelenler in BirAssembly.GetModule("Ornek2.dll").GetTypes()
where Gelenler.GetMethod("Metin") != null select Gelenler;
```

Örneğin yukarıdaki LINQ sorgumuz ile harici DLL dosyası içerisinde **Metin** adında metodu olan tüm sınıfların bir listesini alıyoruz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ve WCF Servisleri

.NET Framework 3.0 ile beraber karşımıza çıkan WCF servisleri aslında çoktan klasik web servislerinin yerini de almış durumda. Tabi ki doğrudan bir karşılaştırma yapmak çok yanlış olacaktır, WCF çok daha geniş kapsamlı bir çerçevede değerlendirilmeli. Silverlight 2.0 tarafına baktığımızda ise istemci ile sunucu arasındaki veri trafigini klasik ASMX web servislerine bağlayabileceğimiz gibi istersek doğrudan WCF servislerini de kullanabiliyoruz. Bu yazında Silverlight 2.0 Beta 1 ile WCF servislerinin kullanımına değineceğiz.

WCF servisimizi hazırlayalım.

Visual Studio 2008 içerisinde yarattığımız yeni Silverlight projemize eşlik eden Web Project içerisinde yeni bir WCF servisi yaratıyoruz. Örneğimizde Silverlight tarafından gönderilen iki sayfa WCF servisi tarafından alınarak sunucu tarafından toplanacak ve geri döndürülecek. Bu çerçevede uygun bir WCF servisini hazırlarken aşağıdaki kodları yazmamız gerekiyor.

[IService.vb]

Imports System.ServiceModel

```
<ServiceContract()>
Public Interface IService

<OperationContract()>
Function Toplama(ByVal x As Integer, ByVal y As Integer) As Integer

End Interface
```

[Service.vb]

```
Public Class Service
    Implements IService

    Public Function Toplama(ByVal x As Integer, ByVal y As Integer) As Integer Implements
IService.Toplama
        Return x + y
    End Function

End Class
```

WCF servisimiz hazır olduğuna göre Silverlight tarafına geçiş yapabiliriz diye düşünüyorsunuz kesinlikle aldaniyorsunuz. Varsayılan ayarları ile Visual Studio içerisinde herhangi bir WCF servisi yarattığında **wsHttpBinding** kullanılır oysa bizim Silverlight tarafında **basicHttpBinding**'e ihtiyacımız var. O nedenle hemen projemizin Web.Config dosyasına ufak bir yolculuk yaparak aşağıdaki şekilde ayarlarda değişiklik yapmamız gerekiyor.

```
<services>
<service behaviorConfiguration="ServiceBehavior" name="Service">
    <endpoint address="" binding="basicHttpBinding" contract="IService">
```

```

<identity>
  <dns value="localhost" />
</identity>
</endpoint>
<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
</service>
</services>

```

Artık tüm ayarlarımız tamamlandı. Silverlight 2.0 tarafına rahatlıkla geçebiliriz.

Silverlight 2.0 ve WCF bağlantısı

WCF servisimiz ile Silverlight uygulamamızın aynı domain içerisinde olması şart. Güvenlik kuralları nedeniyle "cross-domain" yani alan adları arası veri trafigi oluşturma şansımız yok. Visual Studio içerisinde Silverlight projenize sağ tıklayarak gelen menüden **"Add Service Reference"** düğmesine basarak proje içerisinde WCF servisini Silverlight uygulamasına referans olarak ekleyebilirsiniz.

İlk olarak gelin uygulamamızın XAML koduna bir göz atalım.

```

<UserControl x:Class="SilverlightApplication3.Page"
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="400" Height="300">
  <Grid x:Name="LayoutRoot" Background="White">
    <TextBox Height="29" Margin="72,37,134,0" VerticalAlignment="Top" Text="TextBox"
    x:Name="Sayi1"/>
    <TextBox Height="39" Margin="72,81,134,0" VerticalAlignment="Top" Text="TextBox"
    x:Name="Sayi2"/>
    <Button HorizontalAlignment="Stretch" Margin="114,144,188,122"
    VerticalAlignment="Stretch" Content="Button" x:Name="Topla"/>
    <TextBlock Height="43" Margin="86,0,147,47" VerticalAlignment="Bottom"
    Text="TextBlock" TextWrapping="Wrap" x:Name="Sonuc"/>
  </Grid>
</UserControl>

```

Şimdi kod tarafına geçerek bir önceki adımda referans olarak projemize eklediğimiz WCF servisini kullanmaya başlayalım.

WithEvents Servis As New WCFServisim.ServiceClient

Yukarıdaki şekli ile servisimizi uygulama içerisinde global olarak tanımlıyoruz. WCF servisimi bu şekilde tanımlamamın aslında önemli bir nedeni var. Birazdan WCF servisi ile istemci tarafından suncuya bir veri talebi gönderdiğimizde, yani toplanacak olan sayıları gönderip toplamı istediğimizde aslında asenkron bir talepte bulunmuş olacağız. Klasik Windows uygulamalarından alıştığımız yapıdan farklı olarak Silverlight 2.0 içerisinde WCF servislerinin kullanımını tamamen asenkron olarak gerçekleştiriyor. Durum böyle olunca asenkron bir istek sonrasında sunucudan cevap (veri) geldiğinde bizim kodumuzun da durumdan haberdar edilmesi gerekecek. Söz konusu haber yine WCF servisimize özel olan bir başka event-handler'ın çalıştırılması ise bize ulaştırılacak. Aslında dinamik olarak servisimizi

yaratırken event-handler da bağlayabiliyoruz. Ama Visual Basic ile yukarıdaki gibi bir kullanım çok daha rahat oluyor. C# programcılar dinamik event-handler bağlamayı kullanabilirler.

Global değişkenimizde WCF servisimiz hazır olduğuna göre artık düğmemize bazıldığından söz konusu servisi rahatlıkla kullanabiliriz.

```
Private Sub Topla_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Topla.Click
    Servis.ToplamaAsync(Integer.Parse(Sayi1.Text), Integer.Parse(Sayi2.Text))
End Sub
```

Gördüğünüz gibi **Servis** değişkenim üzerinden **ToplamaAsync** metodunu çağırıyorum. Metodu çalıştırıldıktan sonra sunucudan veri geldiğinde bu metoda özel olan **ToplamaCompleted** event'i çalıştırılacak.

```
Private Sub Servis_ToplamaCompleted(ByVal sender As Object, ByVal e As
WCFServisim.ToplamaCompletedEventArgs) Handles Servis.ToplamaCompleted
    Sonuc.Text = e.Result.ToString
End Sub
```

ToplamaCompleted event'ına gelen parametrelerden ikincisinin tipine baktığımızda **ToplamaCompletedEventArgs** ile karşılaşıyoruz. Bu tamamen bizim metodumuza özel bir değişken tipi. Buradan yola çıkarak **e.Result** dediğimizde ise doğrudan bizim WCF metodumuzun döndürdüğü nesneyi yakalayabiliyoruz. Örneğimizde gelen sonucu uygulama içerisinde bir **TextBlock** içine yazdırıyoruz.

Kodumuzun tamamına baktığımızda aşağıdaki manzara ile karşılaşıyoruz.

```
Partial Public Class Page
    Inherits UserControl
```

```
Public Sub New()
    InitializeComponent()
End Sub
```

```
WithEvents Servis As New WCFServisim.ServiceClient
```

```
Private Sub Topla_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Topla.Click
    Servis.ToplamaAsync(Integer.Parse(Sayi1.Text), Integer.Parse(Sayi2.Text))
End Sub
```

```
Private Sub Servis_ToplamaCompleted(ByVal sender As Object, ByVal e As
WCFServisim.ToplamaCompletedEventArgs) Handles Servis.ToplamaCompleted
    Sonuc.Text = e.Result.ToString
End Sub
End Class
```

Peki ya C# olsaydı?

Visual Basic'e özel yapılar kullandığım için aynı kodun C# muadilini de sizlerle paylaşmak istiyorum. Böylece C# programcılar için anlaşılması çok daha kolay olacaktır.

```
namespace SilverlightApplication4
{
    public partial class Page : UserControl
    {
        public Page()
        {
            InitializeComponent();
            Topla.Click += new RoutedEventHandler(Topla_Click);
        }

        void Topla_Click(object sender, RoutedEventArgs e)
        {
            WCFServisim.ServiceClient Servis = new WCFServisim.ServiceClient();
            Servis.ToplamaCompleted += new
EventHandler<WCFServisim.ToplamaCompletedEventArgs>(Servis_ToplamaCompleted
);
            Servis.ToplamaAsync(int.Parse(Sayı1.Text), int.Parse(Sayı2.Text));
        }

        void Servis_ToplamaCompleted(object sender,
SilverlightApplication4.WCFServisim.ToplamaCompletedEventArgs e)
        {
            Sonuc.Text = e.Result.ToString();
        }
    }
}
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 1 içerisinde DataGrid kullanımı!

Silverlight 2.0 Beta 1 ile beraber gelen ilginç kontrollerden biri de DataGrid kontrolüdür. Aslında kontrolün kendisinde herhangi bir ilginçlik yok, ilginç olan WPF'in ilk sürümlerinde böyle bir kontrol yokken Silverlight'in ikinci sürümünde DataGrid'in geliyor olması. Bu yazımızda Silverlight 2.0 Beta 1 ile **DataGrid** kullanımına deyineceğiz.

Silverlight 2.0 projenizi Visual Studio 2008 ile yarattıktan sonra hemen araç çubüğunda **DataGrid** kontrolü ile karşılaşabilirsiniz. Expression Blend içerisinde ise varsayılan ayarlar ile gelmeyecektir. Bunun aslında basit bir nedeni var; DataGrid gibi veri kontrolleri Silverlight 2.0 için harici bir Control Library olan **System.Windows.Controls.Data** altında geliyor ve bu kütüphane normal şartlarda uygulamaları referans olarak eklenmiş olmuyor. Eğer uygulamanıza bu sınıfı referans olarak eklerseniz Blend içerisinde de gerekli seçenekler gelecektir. Visual Studio içerisinde sahneye bir DataGrid yerleştirdiğinizde gerekli referanslar otomatik olarak ekleniyor.

```
<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
x:Class="SilverlightApplication25.Page"
xmlns="http://schemas.microsoft.com/client/2007"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400" Height="300">
<Grid x:Name="LayoutRoot" Background="White">
<my:DataGrid></my:DataGrid>
</Grid>
</UserControl>
```

Yukarıdaki kodu incelediğinizde herhangi bir Silverlight uygulamasına DataGrid yerleştirildiğinde en üst satırındaki XML namespace tanımını görebilirsiniz. Söz konusu tanım veri kontrollerinin Assembly'lerine bağlı. Böylece artık uygulamamızda veri kontrollerini kullanabiliriz. Bunun bir sonucu olarak artık uygulamamızdan üretilecek XAP paketinde de **System.Windows.Controls.Data.dll** dosyası bulunacaktır.

İlk olarak istemci tarafındaki kodumuz ile DataGrid içerisinde gösterilmek üzere bir veri yığını yaratalım. Bu noktada siz uygulamalarınızda rahatlıkla farklı web servislerinden çektiğiniz verileri kullanabilirsiniz.

[VB]

Public Class Urun

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

```

Private PStok As Boolean
Public Property Stok() As Boolean
    Get
        Return PStok
    End Get
    Set(ByVal value As Boolean)
        PStok = value
    End Set
End Property

Sub New()
End Sub

Sub New(ByVal adi As String, ByVal stok As Boolean)
    Me.Adi = adi
    Me.Stok = stok
End Sub

End Class

```

[C#]

```

public class Urun
{
    private string PAdi;
    public string Adi
    {
        get { return PAdi; }
        set { PAdi = value; }
    }

    private bool PStok;
    public bool Stok
    {
        get { return PStok; }
        set { PStok = value; }
    }

    public Urun()
    {

    }

    public Urun(string adi, bool stok)
    {

```

```

        this.Adi = adi;
        this.Stok = stok;
    }

}

```

Yukarıdaki sınıf yapısını verimizi oluştururken kullanacağımız nesneler olarak hazırladık. Silverlight 2.0'daki DataBinding WPF ile büyük bir benzerliğe sahip. Özellikle LINQ ile beraber kullanıldığında nesneleri kontrollere bind edebiliyor olmak büyük avantaj sağlıyor. Şimdi gelelim bize geçici olarak veri yaratacak olan kodumuzu yazmaya.

[VB]

```

Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim liste As New System.Collections.Generic.List(Of Urun)
    For x As Integer = 0 To 9
        liste.Add(New Urun("Urun Adı" & x, (Math.Round(Rnd() * 1) - 1)))
    Next
    BirGrid.ItemsSource = liste
End Sub

```

[C#]

```

public Page()
{
    InitializeComponent();

    System.Collections.Generic.List<Urun> liste = new
    System.Collections.Generic.List<Urun>();
    Random RastGele = new Random();
    for (int x = 0; x <= 9; x++)
    {
        liste.Add(new Urun("Urun Adı" + x.ToString(),
Convert.ToBoolean(RastGele.Next(0, 1) - 1)));
    }
    BirGrid.ItemsSource = liste;
}

```

Kod içerisinde de gördüğünüz gibi elimizdeki veriyi doğrudan BirGrid adındaki DataGridimizin **ItemsSource** özelliğine bağlıyoruz. Böylece **DataBinding** işlemi tamamlanmış oldu. Fakat bağladığımız bu verinin Grid içerisinde kolonlara yerleşmesi için tabi bizim "kolon"lara ihtiyacımız var. Otomatik olarak veriye uygun kolon yaratılabilmesi için DataGrid'in **AutoGenerateColumns** özelliğinin **True** olarak ayarlanmış olması gerekiyor.

```

<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
x:Class="SilverlightApplication25.Page"
    xmlns="http://schemas.microsoft.com/client/2007">

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400" Height="300">
<Grid x:Name="LayoutRoot" Background="White">
<my:DataGrid x:Name="BirGrid" AutoGenerateColumns="True"></my:DataGrid>
</Grid>
</UserControl>

```

XAML kodumuzun son hali yukarıdaki gibi olmalı. Böylece uygulamamızı çalıştırduğımızda aşağıdaki manzara ile karşılaşabiliriz.

	Adi	Stok	
▶	Urun Adi0	<input type="checkbox"/>	
	Urun Adi1	<input type="checkbox"/>	
	Urun Adi2	<input type="checkbox"/>	
	Urun Adi3	<input checked="" type="checkbox"/>	
	Urun Adi4	<input checked="" type="checkbox"/>	
	Urun Adi5	<input type="checkbox"/>	
	Urun Adi6	<input checked="" type="checkbox"/>	
	Urun Adi7	<input type="checkbox"/>	
	Urun Adi8	<input type="checkbox"/>	
	Urun Adi9	<input type="checkbox"/>	

Silverlight 2.0 içerisinde DataGrid görüntüsü.

İsterseniz alternatif satırların fon renklerini hatta kolonlar arası çizgilerin renklerini bile tek tek belirleyebilirsiniz. Aşağıdaki kod yapabileceğiniz renk değişikliklerine dair bir ipucu olabilir.

```

<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
x:Class="SilverlightApplication25.Page"
xmlns="http://schemas.microsoft.com/client/2007"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400"
Height="300">
<Grid x:Name="LayoutRoot"
Background="White">
<my:DataGrid x:Name="BirGrid"
AutoGenerateColumns="True"
AlternatingRowBackground="#FFFFFF00"
HorizontalGridlinesBrush="#FFD4FF00"
RowBackground="#FFE3E3E3"></my:DataGrid>
</Grid>

```

</UserControl>

Kendi kolonlarımızı tanımlayalım!

Aslında **AutoGenerateColumns** özelliği bizim ASP.NET'teki GridView'den de alışık olduğumuz bir özellik. Kolay bir kullanım sağlasa da çoğu zaman bu özellik istediğimiz uygulamaları hazırlayabilmemiz için yeterli değil. O nedenle gelin şimdi beraber bir DataGrid içerisinde kolonları nasıl elle ayarlayabileceğimizi inceleyelim.

Eğer **AutoGenerateColumns** özelliğini **True** yapmazsanız hali hazırda varsayılan ayarı zaten **False** olarak geliyor. O nedenle bir önceki projemize devam edeceğimiz için ilk olarak ya **AutoGenerateColumns** özelliğini XAML kodunuzdan silin ya da **False** olarak ayarlayın.

Bir DataGrid'in üç çeşit kolonu olabilir;

- **DataGridViewTextBoxColumn**
- **DataGridViewCheckBoxColumn**
- **DataGridViewTemplateColumn**

Adlarından da anlaşılacağı üzere ikisi kendi isimlerindeki kontrolleri kolonlara yerleştirirken **TemplateColumn** ise bize daha esnek bir yapı sağlıyor. İlk olarak gelin **TextBoxColumn** ve **CheckBoxColumn** kullanarak bir önceki adımdaki örneğimizin kolonlarını elle tanımlayalım.

```
<my:DataGrid x:Name="BirGrid"
    AutoGenerateColumns="False"
    AlternatingRowBackground="#FFFFFF00"
    HorizontalGridlinesBrush="#FFD4FF00"
    RowBackground="#FFE3E3E3">
<my:DataGrid.Columns>
    <my:DataGridViewTextBoxColumn Header="Adı"
        DisplayMemberBinding="{Binding Adı}" />
    <my:DataGridViewCheckBoxColumn Header="Stokta Var?"
        DisplayMemberBinding="{Binding Stok}" />
</my:DataGrid.Columns>
</my:DataGrid>
```

Kolonlarımızı ekledikten sonra her kolonun **Header** özelliğini değiştirerek o kolonda gözükecek olan başlığı ayarlayabiliyoruz. Son olarak da veri kaynağından hangi **Property**'nin söz konusu kolonda gözükeceğini belirlemek için bir **Binding** kullanıyoruz. Görsel olarak sonuç bir önceki örneğimizdeki ile aynı olacak fakat bu sefer kolonları biz el ile tek tek ayarlamış olduk. Bunun getireceği esnekliği özellikle **TemplateColumn** ile çok daha rahat görebiliriz.

Özel kolonlar : **TemplateColumn**

Özel bir kolon tanımlarken yapmamız gereken iki şey var; ilk olarak kolonun normal görüntüsünü tanımlamak, ikincisi ise "edit" modundaki görüntüsünü tanımlamak. Eğer **ReadOnly** özelliklerini değiştirmezseniz normal şartlarda hem **TextBoxColumn** hem de **CheckBoxColumn** üzerlerine tıklandıklarında içlerindeki verinin değiştirilebilmesine olanak

tanırlar. Hatta **Binding Mode** olarak da **TwoWay** parametresini aktarırsanız arka planda Bind ettiğiniz List içerisinde gerekli değişiklikler de otomatik olarak yapılır. Şimdi biz tüm bunları bir **TemplateColumn** ile deneyeceğiz. Amacımız **Stok** bilgisi gösteren kolonu biraz değiştirerek normalde içerisinde True veya False yazmasını sağlamak. Yani normal şartlarda o kolonda bir **CheckBox** gözükmeyecek, fakat kullanıcına kolona çift tıklar ve değeri değiştirmek isterse karşınızda bu sefer bir **CheckBox** gelecek.

```
<my:DataGridTemplateColumn Header="Stokta Var?">
<my:DataGridTemplateColumn.CellTemplate>
<DataTemplate>
<TextBlock Text="{Binding Stok}" />
</DataTemplate>
</my:DataGridTemplateColumn.CellTemplate>
<my:DataGridTemplateColumn.CellEditingTemplate>
<DataTemplate>
<CheckBox IsChecked="{Binding Stok, Mode=TwoWay}" /></CheckBox>
</DataTemplate>
</my:DataGridTemplateColumn.CellEditingTemplate>
</my:DataGridTemplateColumn>
```

Yukarıdaki kodu detaylı olarak incelemekte fayda var. Yarattığımız **TemplateColumn**'un içerisinde bir **CellTemplate**, bir de **CellEditingTemplate** var. Bu kolonun normal şartlardaki görüntüsü **CellTemplate**, düzenleme modundaki görüntüsü ise **CellEditingTemplate** içerisindeki şablonla göre hazırlanacak. **CellTemplate** içerisinde **DataTemplate** içinde sadece bir **TextBlock** koyuyoruz ve söz konusu **TextBlock**'un da **Text** özelliğini **Stok** bilgisini bind ediyoruz. Böylece normalde Stok bilgisi String olarak bu **TextBlock** içerisinde gösterilecek. Gelelim **CellEditingTemplate** şablonunda; bu şablon içerisinde de bir **CheckBox** kullanarak söz konusu **CheckBox**'un **.IsChecked** özelliğini **Stok** Property'sine Bind ederken **Mode** olarak da **TwoWay**'ı seçiyoruz. Böylece bu **CheckBox** üzerinde yapılan değişiklikler elimizdeki List verimiz yansiyacak, yani kaydedilecek.

Uygulamamızın tam XAML kodunu aşağıda inceleyebilirsiniz.

```
<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
x:Class="SilverlightApplication25.Page"
xmlns="http://schemas.microsoft.com/client/2007"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400"
Height="300">
<Grid x:Name="LayoutRoot"
Background="White">
<my:DataGrid x:Name="BirGrid"
AutoGenerateColumns="False"
AlternatingRowBackground="#FFFFFF00"
HorizontalGridlinesBrush="#FFD4FF00"
RowBackground="#FFE3E3E3">
<my:DataGrid.Columns>
<my:DataGridTextBoxColumn Header="Adı"
DisplayMemberBinding="{Binding Adı}" />
```

```

<my:DataGridCheckBoxColumn Header="Stokta Var?">
    DisplayMemberBinding="{Binding Stok}" />
<my:DataGridTemplateColumn Header="Stokta Var?">
    <my:DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding Stok}" />
        </DataTemplate>
    </my:DataGridTemplateColumn.CellTemplate>
    <my:DataGridTemplateColumn.CellEditingTemplate>
        <DataTemplate>
            <CheckBox IsChecked="{Binding Stok, Mode=TwoWay}" /></CheckBox>
        </DataTemplate>
    </my:DataGridTemplateColumn.CellEditingTemplate>
</my:DataGridTemplateColumn>
</my:DataGrid.Columns>
</my:DataGrid>
</Grid>
</UserControl>

```

IValueConverter ile Binding'lere müdahale edin

Bir önceki örnek biraz saçma gelmiş olabilir. Kolon içerisinde doğrudan True veya False yazıyor olmak pek hoş değil. Stok bilgisinden bahsettiğimize göre True veya Flase yerine "Var" veya "Yok" yazdırsak belki çok daha mantıklı olabilirdi. Kullanıcı satırı tıkladığında ise yine karşısına düzenleme modunda bir CheckBox gelecektir. Bu işlemi yapabilmemiz için bizim **CellTemplate** içerisindeki TextBlock'un Binding'ine müdahale ederek "*Eğer True geliyorsa VAR yazdır, gelmiyorsa YOK yazdır*" diyebilmemiz gerekiyor. İşte tam da bu işlemi yapabilmek için **Silverlight 2.0 Beta 1** içerisinde ValueConverter yapılarını kullanabiliyoruz.

[VB]

```

Public Class StokCevirici
    Implements Data.IValueConverter

    Public Function Convert(ByVal value As Object, ByVal targetType As System.Type,
                           ByVal parameter As Object, ByVal culture As System.Globalization.CultureInfo) As Object
        Implements System.Windows.Data.IValueConverter.Convert
        If value Then
            Return "Var"
        Else
            Return "Yok"
        End If
    End Function

```

```

    Public Function ConvertBack(ByVal value As Object, ByVal targetType As System.Type,
                               ByVal parameter As Object, ByVal culture As System.Globalization.CultureInfo) As Object
        Implements System.Windows.Data.IValueConverter.ConvertBack
        If value = "Var" Then
            Return True
        Else

```

```

    Return False
End If
End Function
End Class

```

[C#]

```

public class StokCevirici : System.Windows.Data.IValueConverter
{
    object System.Windows.Data.IValueConverter.Convert(object value, System.Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        if(bool.Parse(value.ToString()))
        {
            return "Var";
        }
        else
        {
            return "Yok";
        }
    }

    object System.Windows.Data.IValueConverter.ConvertBack(object value, System.Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        if(value.ToString() == "Var")
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

İlk olarak yukarıdaki şekilde **System.Windows.Data.IValueConverter** sınıfını implemente etmemiz gerekiyor. Bu şekilde bir **Converter** yapısının her zaman bir **Convert** ve bir de **ConvertBack** metodlarının bulunması şart. Bu metodlar aslında bizim elimizdeki True veya False olan **Boolean** değerinin String'e çevireceğimiz ve Binding için DataGrid'e göndereceğimiz veriyi oluşturmamıza olanak tanıyorlar. Kod içerisinde de duruma göre parametre olarak gelen Boolean değeri String'e veya tam tersine işlemler yapıyoruz. Sıra geldi bu **Converter** yapısını XAML kodumuzda kullanarak Binding işlemine dahil etmeye.

```

<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
    x:Class="SilverlightApplication25.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

```
xmlns:local="clr-namespace:SilverlightApplication25"
Width="400"
Height="300">>
```

İlk olarak yukarıdaki şekilde XAML içerisinde kullanacağımız Assembly'mizi tanımlıyoruz. Böylece Converter sınıfımızı rahatlıkla kullanabileceğiz. Fakat işlemler bu kadarla bitmiyor. Tanımladığımız Assembly içerisinde Convertor'ımızı da alarak sayfada Resource olarak yerleştirmemiz şart.

```
<UserControl.Resources>
<local:StokCevirici x:Key="StokCevirici" />
</UserControl.Resources>
```

Tüm bu işlemlerde Visual Studio'nun Intellisense yapısı size yardımcı olacaktır. Artık XAML tarafında **StokCevirici** adını verdigimiz **Converter** yapımızı istediğimiz bir **Binding** için kullanmaya hazırız.

```
<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
x:Class="SilverlightApplication25.Page"
xmlns="http://schemas.microsoft.com/client/2007"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="clr-namespace:SilverlightApplication25"
Width="400"
Height="300">
<UserControl.Resources>
<local:StokCevirici x:Key="StokCevirici" />
</UserControl.Resources>
<Grid x:Name="LayoutRoot"
Background="White">
<my:DataGrid x:Name="BirGrid"
AutoGenerateColumns="False"
AlternatingRowBackground="#FFFFFF00"
HorizontalGridlinesBrush="#FFD4FF00"
RowBackground="#FFE3E3E3">
<my:DataGrid.Columns>
<my:DataGridTextBoxColumn Header="Adi"
DisplayMemberBinding="{Binding Adi}" />
<my:DataGridCheckBoxColumn Header="Stokta Var?"
DisplayMemberBinding="{Binding Stok}" />
<my:DataGridTemplateColumn Header="Stokta Var?">
<my:DataGridTemplateColumn.CellTemplate>
<DataTemplate>
<TextBlock Text="{Binding Stok, Converter={StaticResource StokCevirici}}" />
</DataTemplate>
</my:DataGridTemplateColumn.CellTemplate>
<my:DataGridTemplateColumn.CellEditingTemplate>
<DataTemplate>
<CheckBox IsChecked="{Binding Stok, Mode=TwoWay}"></CheckBox>
</DataTemplate>
```

```
</my:DataGridTemplateColumn.CellEditingTemplate>
</my:DataGridTemplateColumn>

</my:DataGrid.Columns>
</my:DataGrid>
</Grid>
</UserControl>
```

Uygulamanın son halinin tam kodunu yukarıdaki inceleyebilirsiniz. Özellikle yarattığımız Converter'ın kullanım şekline dikkat etmekte fayda var. Artık **TextBlock** içerisinde gösterilen veriler söz konusu **Converter'dan** geçtikten sonra gösterileceği için ekranda "Var" veya "Yok" yazıları yer alacak. Oysa kullanıcı çift tıklayarak değeri değiştirmek istediginde karşısına bir **CheckBox** çıkacak ve True veya False olabilecek **Boolean** değeri değiştiriyor olacak.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 1 içerisinde ToggleButton kullanımı

Checkbox ve RadioButton kontrolleri neredeyse her projede en az bir defa kullandığımız kontroller arasında yerlerini alırlar. Bu kontroller gibi farklı kontroller oluşturarak kullanıcıyı bir durumdan haberdar etmek veya kullanıcının bir durumu değiştirmesini sağlamak mümkün olabilir. Örneğin basit bir video oynatıcısı uygulamasında "Play" düğmesi ile "Pause" düğmesini aynı düğme içerisinde kullanabilirsiniz. Söz konusu düğme kendi içinde değişerek her tıklandığında "Play" veya "Pause" şeklinde üzerindeki yazıyı değiştirir ve videonun da durdurulmasını veya oynatılmasını sağlar. **Checkbox** veya **RadioButton** düğmeleri gibi bu gibi kontollere özünde **"ToggleButton"** denir ve **Silverlight 2 Beta 1** içerisinde **Checkbox** ve **RadioButton** da zaten hali hazırda adı **ToggleButton** olan bir kontrol yapısından türetilmiştir. Bu yazımızda **ToggleButton** kontrolünün detaylarına ve kullanımına değineceğiz.

```
<UserControl x:Class="SilverlightApplication21.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <ToggleButton Margin="104,109,177,138" Content="ToggleButton"
x:Name="ToggleDugme"/>
    </Grid>
</UserControl>
```

Yukarıdaki şekli ile standart bir **ToggleButton** kontrolünü ister Blend ister Visual Studio içerisinde uygulamanıza yerleştirebilirsiniz. Sonrasında arkaplanda **ToggleButton** kontrolünün yakalayabileceğimiz iki kendine özel event'i var; bunlardan ilki **Checked**, diğeri ise **Unchecked** durumları. Yarattığımız ToggleButton'ın görsel özelliklerine deðinmeden önce hemen bu event'lar ile neler yapabileceğimize bir göz atalım.

[VB]

```
Private Sub ToggleDugme_CheckedChanged(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles ToggleDugme.Checked
    ToggleDugme.Content = "İşaretli"
End Sub
```

```
Private Sub ToggleDugme_Unchecked(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles ToggleDugme.Unchecked
    ToggleDugme.Content = "İşaretsiz"
End Sub
```

[C#]

```
private void ToggleDugme_CheckedChanged(object sender, System.Windows.RoutedEventArgs e)
{
    ToggleDugme.Content = "İşaretli";
}

private void ToggleDugme_Unchecked(object sender, System.Windows.RoutedEventArgs e)
```

```
{
    ToggleDugme.Content = "İşretsiz";
}
```

Basit bir şekilde ToggleButton kontrolümüzün içerisinde yazılı metni değiştirdiğimiz örneğimizi çalıştırıldığımızda artık **Button** görünümündeki **ToggleButton** kontrolüne her bastığımızda içinde duruma göre "İşaretli" veya "İşretsiz" yazacak. Ayrıca isterseniz ToggleButton kontrolüne ait **.IsChecked** özelliğini de kullanarak ToggleButton'un o anki durumundan haberdar olabilirsiniz.

Belirsiz durumlara özel...

Bazı durumlarda sadece iki seçenek yetmez ve "belirsizlik" seçimi de yapmak gerekebilir. Bu durumda kullanıcıya sadece Evet veya Hayır şeklinde cevap vermenin yanı sıra isterse "Bilmiyorum" gibi bir seçenek de söylebilir. **ToggleButton** içerisinde böyle bir yapı da var. Eğer bir ToggleButton'un **IsThreeState** özelliğini **True** olarak ayarlaysanız artık iki değil üç seçenekli bir ToggleButton sahibi olmuş oluyorsunuz. Peki bu üçüncü seçeneği kod tarafında nasıl yakalıyoruz?

[VB]

```
Private Sub ToggleDugme_Checked(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles ToggleDugme.Checked
    ToggleDugme.Content = "İşaretli"
End Sub
```

```
Private Sub ToggleDugme_Indeterminate(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles ToggleDugme.Indeterminate
    ToggleDugme.Content = "Belirsiz"
End Sub
```

```
Private Sub ToggleDugme_Unchecked(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles ToggleDugme.Unchecked
    ToggleDugme.Content = "İşretsiz"
End Sub
```

[C#]

```
private void ToggleDugme_Checked(object sender, System.Windows.RoutedEventArgs e)
{
    ToggleDugme.Content = "İşaretli";
}

private void ToggleDugme_Indeterminate(object sender,
System.Windows.RoutedEventArgs e)
{
    ToggleDugme.Content = "Belirsiz";
}

private void ToggleDugme_Unchecked(object sender, System.Windows.RoutedEventArgs e)
```

```
{  
    ToggleDugme.Content = "İşretsiz";  
}
```

Gördüğünüz gibi ToggleButton'un ayrıca bir de **Indeterminate** adında bir event-handler'ı bulunuyor. Söz konusu durumu yakalayarak belirsizlik halinde de gerekli işlemlerin yapılmasını sağlayabilirsiniz. Böyle bir durumda **.IsChecked** özelliği geriye **null** / **nothing** döndürecektir.

Unutmayın ki herhangi bir kontrolün **Content** özelliği aslında içerisinde farklı Silverlight kontrolleri de alabilir hatta kontrollerin görsel yapısının tamamen değiştirebilirsiniz. Silverlight 2 Beta 1 içerisinde [Control Templating](#) ile ilgili yazıyı inceleyerek ToggleButton için de aynı teknikleri uygulayabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 1 ile Klasik ASMX Web Servisi kullanımı

Silverlight 2.0 ile beraber .NET dillerini kullanabilenler asında sunucu tarafındaki veriye ulaşma yapısı biraz değişiyor. Normal şartlarda istemci tarafından sunucuya bağlanabilmek için AJAX isteklerini kullanırken artık istemci tarafında elimizde .NET varken ne yapacağız?

İşte bu soruya bir cevap olarak klasik **ASMX** web servislerinin **Silverlight 2.0 Beta 1** ile kullanımına göz atarak hali hazırda elimizde web servisleri ile bulunan projelere nasıl Silverlight 2.0 Beta 1 uygulamalarını bağlayabileceğimizi göreceğiz.

Klasik ASMX servisimiz hazır

Örneğimizde kullanılmak üzere kendisine verilen iki sayıyı toplayarak geri döndüren bir method'u harici bir web servisi olarak hazırlayarak Silverlight uygulamamıza bağlayacağımız. Bunun için Silverlight uygulamamızı host edecek olan ASP.NET 3.5 sitesine bir web servisi ekleyerek içerisinde aşağıdaki kodu yazıyoruz.

Imports System.Web

Imports System.Web.Services

Imports System.Web.Services.Protocols

```
<WebService(Namespace:="http://tempuri.org")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class servisim
    Inherits System.Web.Services.WebService

    <WebMethod()>
    Public Function Toplama(ByVal x As Integer, ByVal y As Integer) As Integer
        Return x + y
    End Function

End Class
```

Bir sonraki adımda Silverlight projemize Visual Studio 2008 içerisinde sağ tıklayarak gelen menüden "Add Service Reference" komutunu vererek servisimizi Silverlight projesine referans olarak ekliyoruz.

Uygulamamızı tasarılıyoruz

Silverlight uygulamamız içerisinde iki adet TextBox, bir Button ve bir de TextBlock yer alacak. Bu TextBox'lardan alınan değerler web servisine gönderilecek. Değerlerin toplam web servisinden geri döndüğünde ise sonuç TextBlock içerisinde yazılacak. Hazırlayacağımız örnek bir Silverlight uygulamamısının XAML kodu aşağıdaki şekilde sonuçlanıyor.

```
<UserControl x:Class="SilverlightApplication2.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
```

```

<TextBox Height="31" HorizontalAlignment="Left" Margin="48,46,0,0"
VerticalAlignment="Top" Width="115" Text="" x:Name="Sayi1"/>
<TextBox Height="31" HorizontalAlignment="Right" Margin="0,46,79,0"
VerticalAlignment="Top" Width="116" Text="" x:Name="Sayi2"/>
<Button Height="33" HorizontalAlignment="Stretch" Margin="133,112,155,0"
VerticalAlignment="Top" Content="TOPLA" x:Name="Topla"/>
<TextBlock Height="42" Margin="87,0,103,72" VerticalAlignment="Bottom"
Text="TextBlock" TextWrapping="Wrap" x:Name="Sonuc"/>
</Grid>
</UserControl>

```

Servisimizi kullanalım...

Sıra geldi artık işin arkaplanın geçerek kodumuzu yazmaya. Projemize eklediğimiz servis referansına verdigimiz isim üzerinden servisimizin bir kopyasını yaratarak içerisindeki **Toplama** metodunu kullanmak istiyoruz.

```

Dim BirServis As New KlasikServisim.servisimSoapClient
BirServis.ToplamaAsync(Sayı1.Text, Sayı2.Text)

```

Yukarıdaki kod aslında alışık olduğumuz web servisi kullanımından pek farklı değil. Fakat arada ufak bir değişiklik var. Bizim adını **"Toplama"** olarak koyduğumuz metodun sonuna **Async** eklenmiş. Aslında bunun anlamı çok basit; Silverlight tarafında çağrıdığınız bir web servisi tamamen asenkron olarak çalıştırılıyor. Yani bizim AJAX tarafında alışık olduğumuz istemciden sunucuya bağlanarak veri çekme mantığı bire bir web servisleri için de uygulanmış. Oysa eskiden Windows programlarında web servisleri kullanırken içerisinde bulduğumuz Threat kesinlikle verinin gelmesini beklerdi.

Peki veri asenkron geliyorsa bizim verinin geldiğinden haberdar olmamız gerekmek mi? Çünkü bu şartlar altında verinin ne zaman sunucudan geleceğini bilemiyoruz. İşte tam bu noktada kodumuz içerisinde dinamik olarak bir event-handler tanımlamamız gerekiyor.

```

Dim BirServis As New KlasikServisim.servisimSoapClient
AddHandler BirServis.ToplamaCompleted, AddressOf Bitti
BirServis.ToplamaAsync(Sayı1.Text, Sayı2.Text)
Sonuc.Text = "Hesaplanıyor..."

```

Yukarıdaki kodumuzda **Bitti** adındaki bir event-handler'ı servisimizin **ToplamaCompleted** metoduna bağlıyoruz. Böylece Toplama işlemi tamamlandığında söz konusu event-handler çalıştırılacak. Şimdi bir de veri geldiğinde nasıl TextBlock içerisinde yazdıracağımıza göz atalım.

```

Public Sub Bitti(ByVal sender As Object, ByVal e As
KlasikServisim.ToplamaCompletedEventArgs)
    Sonuc.Text = e.Result
End Sub

```

Gördüğünüz gibi bir önceki adımda tanımladığımız **Bitti** event'inin aldığı ikinci parametre olan **ToplamaCompletedEventArgs** tipinde e değişkeni üzerinden **Result** yani sonuca ulaşabiliyoruz. Söz konusu parametre doğrudan **Toplama** metoduna özel olduğu için

îçerisinde **Result** özelliğinin tipi de zaten web servisimizdeki metodun dönüş tipi olan **Integer**.

Uygulamamızın tam kodu aşağıdaki şekilde sonlanıyor.

```
Partial Public Class Page
```

```
    Inherits UserControl
```

```
    Public Sub New()
```

```
        InitializeComponent()
```

```
    End Sub
```

```
    Private Sub Topla_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Topla.Click
```

```
        Dim BirServis As New KlasikServisim.servisimSoapClient
```

```
        AddHandler BirServis.ToplamaCompleted, AddressOf Bitti
```

```
        BirServis.ToplamaAsync(Sayi1.Text, Sayi2.Text)
```

```
        Sonuc.Text = "Hesaplanıyor..."
```

```
    End Sub
```

```
    Public Sub Bitti(ByVal sender As Object, ByVal e As
KlasikServisim.ToplamaCompletedEventArgs)
```

```
        Sonuc.Text = e.Result
```

```
    End Sub
```

```
End Class
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 2 içerisinde dinamik resim yüklemek

Silverlight 2.0 Beta 2 ile beraber gelen değişikliklerden biri de dinamik olarak Image nesneleri yaratmak ve bu nesnelere farklı resimler yüklemekle ilgili. Eskiden Beta 1 içerisinde sadece **Source** özelliğini değiştirerek resimleri dinamik olarak yükleyebiliyorken artık biraz daha uğraşmamız gerekiyor.

İlk olarak yeni bir Silverlight 2.0 Beta 2 projesi yaratarak projemize kullanacağımız resimleri ekleyelim. Sonrasında dinamik olarak bir **Image** nesnesi yaratıp içerisinde **ImageSource** ile dolduruyoruz.

[VB]

```
Dim Foto As New Image
Dim Adres As New Uri("garden.jpg", UriKind.Relative)
Dim FotoKaynak As ImageSource = New
System.Windows.Media.Imaging.BitmapImage(Adres)
Foto.SetValue(Image.SourceProperty, FotoKaynak)
Me.LayoutRoot.Children.Add(Foto)
```

[C#]

```
Image Foto = new Image();
Uri Adres = new Uri("garden.jpg", UriKind.Relative);
ImageSource FotoKaynak = new System.Windows.Media.Imaging.BitmapImage(Adres);
Foto.SetValue(Image.SourceProperty, FotoKaynak);
this.LayoutRoot.Children.Add(Foto);
```

Yukarıdaki kodumuzun ilk satırında yarattığım Silverlight Image nesnesini doldurmak için bir **ImageSource**'a ihtiyacımız var. Bunun için ilk olarak yükleyeceğimiz resim adresini **Uri** tipinde yaratıyoruz. Relative konumlandırma ile doğrudan uygulamanın içerisindeki resim dosyasını kullanacağız. Sonrasında elimizdeki Uri'den bir **ImageSource** yaratıyoruz ve bu **ImageSource**'u da Image nesnemizin **SourceProperty'sine** atıyoruz. Tüm bu işlemler tamamlandıktan sonra tabi ki eldeki Image Silverlight nesnesini de sahneye eklememiz lazım.

Peki ya proje harici bir resmi yüklemek istersek?

Harici resmin bulunduğu internet adresi üzerinden bir Uri yaratarak yukarıdaki kodun aynısını kullanabilirsiniz. Böylece uygulamanız söz konusu adresteği fotoğrafı yükleyecektir.

[VB]

```
Dim Adres As New Uri("www.alanadi.com/resim.jpg", UriKind.Absolute)
```

[C#]

```
Uri Adres = new Uri("www.alanadi.com/resim.jpg", UriKind.Absolute);
```

Hepinize kolay gelsin. Daron YÖNDEM

Silverlight 2.0 Beta 2 ile ASP.NET Forms Authentication kullanımı

ASP.NET Authentication mekanizmaları neredeyse tüm ASP.NET projelerinde kullandığımız pratik çözümlerden. Özellikle Forms Authentication belki de özellikle internet projelerinde en sık karşılaştığımız sistem. Peki nasıl yaparız da **Silverlight 2.0 Beta 2** uygulamalarımızda **ASP.NET Forms Authentication** yapısını kullanabiliriz?

WCF üzerinden Authentication servisine ulaşalım.

Silverlight tarafından sunucuya veri alışverişi için en uygun seçim WCF servisleri. Bu nedenle bize bir şekilde Authentication servisine ulaşabileceğimiz bir servis gerekiyor. Yeni bir Silverlight projesi yaratarak yanında gelen ASP.NET sitesiyle işlemlerimizi yapmaya başlayalım.

ASP.NET'in kendi WCF Authentication servis altyapısını kullanacağız. Tek ihtiyacımız olan bir Wrapper. Bunun için hemen ASP.NET sitesine sağ tuş ile tıklayarak gelen menüden "Add New Item" diyip normal bir Text File ekleyelim. Bu yeni dosyanın adını **Auth.svc** şeklinde düzenledikten sonra içini açarak aşağıdaki kodu yapıştıralım.

```
<%@ ServiceHost Language="VB"
Service="System.Web.ApplicationServices.AuthenticationService" %>
```

Böylece servisimizi tamamladık. Sıra geldi bu servisin çalışması için **Web.Config** içerisinde yapmamız gereken ayarlara. İlk olarak **Forms Authentication** yapımızı ayarlayalım.

```
<authentication mode="Forms">
<forms loginUrl="default.aspx" protection="All" timeout="30" path="/">
<credentials passwordFormat="Clear">
<user name="daron" password="123" />
</credentials>
</forms>
</authentication>
<authorization>
<deny users="?" />
</authorization>
```

Yukarıdaki kodumuz ile **default.aspx** haricindeki sitedeki her şeyi dışarıya kapadık. Çok uğraşmamak için hemen Web.Config içerisinde de bir kullanıcı tanımladım. Normalde bu tarz bir yapıyı kimseye tavsiye etmiyorum. Tabi tüm dosyaları dışarıya kapadığımız için Silverlight uygulaması da **Auth.svc** servisine de ulaşamayacak. O nedenle servisimizi Authentication dışında tutup herkese açmamız lazım.

```
<location path="Auth.svc">
<system.web>
<authorization>
<allow users="*"/>
</authorization>
</system.web>
</location>
```

Forms Authentication ayarlarınıza tamamladığımıza göre artık WCF servisimizle ilgili ayarları da yapabiliriz.

```
<system.serviceModel>
<services>
    <service name="System.Web.ApplicationServices.AuthenticationService"
        behaviorConfiguration="AuthenticationServiceTypeBehaviors">
        <endpoint contract="System.Web.ApplicationServices.AuthenticationService"
            binding="basicHttpBinding" bindingConfiguration="userHttp"
            bindingNamespace="http://asp.net/ApplicationServices/v200"/>
    </service>
</services>
<bindings>
    <basicHttpBinding>
        <binding name="userHttp">
            <security mode="None"/>
        </binding>
    </basicHttpBinding>
</bindings>
<behaviors>
    <serviceBehaviors>
        <behavior name="AuthenticationServiceTypeBehaviors">
            <serviceMetadata httpGetEnabled="true"/>
        </behavior>
    </serviceBehaviors>
</behaviors>
<!-- HTTP üzerinden servise ulaşımı sağlar. -->
<serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>
</system.serviceModel>
<!--Authentication servisini dışarıya açar-->
<system.web.extensions>
    <scripting>
        <webServices>
            <authenticationService enabled="true" requireSSL="false"/>
        </webServices>
    </scripting>
</system.web.extensions>
```

Her şey hazır. Internet tarayıcınızda Auth.svc adresini açtığınızda servisin çalışır halde olduğunu görebilirsiniz. Sıra geldi Silverlight ile bu servisi kullanmaya.

WCF Authentication Servisimiz Silverlight ile dille尼yor.

Herhangi bir WCF servisini Silverlight uygulamamıza linklermiş gibi yine projeye sağ tıklayarak "Add Service Reference" diyerek referansımızı yaratıyoruz. Sonrasında artık kod içerisinde tüm Authentication mekanizmalarını kullanabiliyoruz.

[VB]

WithEvents Servisim As New ServiceReference1.AuthenticationServiceClient

```

Private Sub Giris_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Giris.Click
    Servisim.LoginAsync(Kullanici.Text, Sifre.Text, "", True)
End Sub

Private Sub Servisim_LoginCompleted(ByVal sender As Object, ByVal e As
ServiceReference1.LoginCompletedEventArgs) Handles Servisim.LoginCompleted
    Giris.Content = e.Result
End Sub

```

[C#]

```

public Page()
{
    InitializeComponent();
    Giris.Click += new RoutedEventHandler(Giris_Click);
}

void Giris_Click(object sender, RoutedEventArgs e)
{
    ServiceReference1.AuthenticationServiceClient Servisim = new
SilverlightApplication2.ServiceReference1.AuthenticationServiceClient();
    Servisim.LoginCompleted += new
EventHandler<SilverlightApplication2.ServiceReference1.LoginCompletedEventArgs>(Serv
sim_LoginCompleted);
    Servisim.LoginAsync(Kullanici.Text, Sifre.Text, "", true);
}

void Servisim_LoginCompleted(object sender,
SilverlightApplication2.ServiceReference1.LoginCompletedEventArgs e)
{
    Giris.Content = e.Result.ToString();
}

```

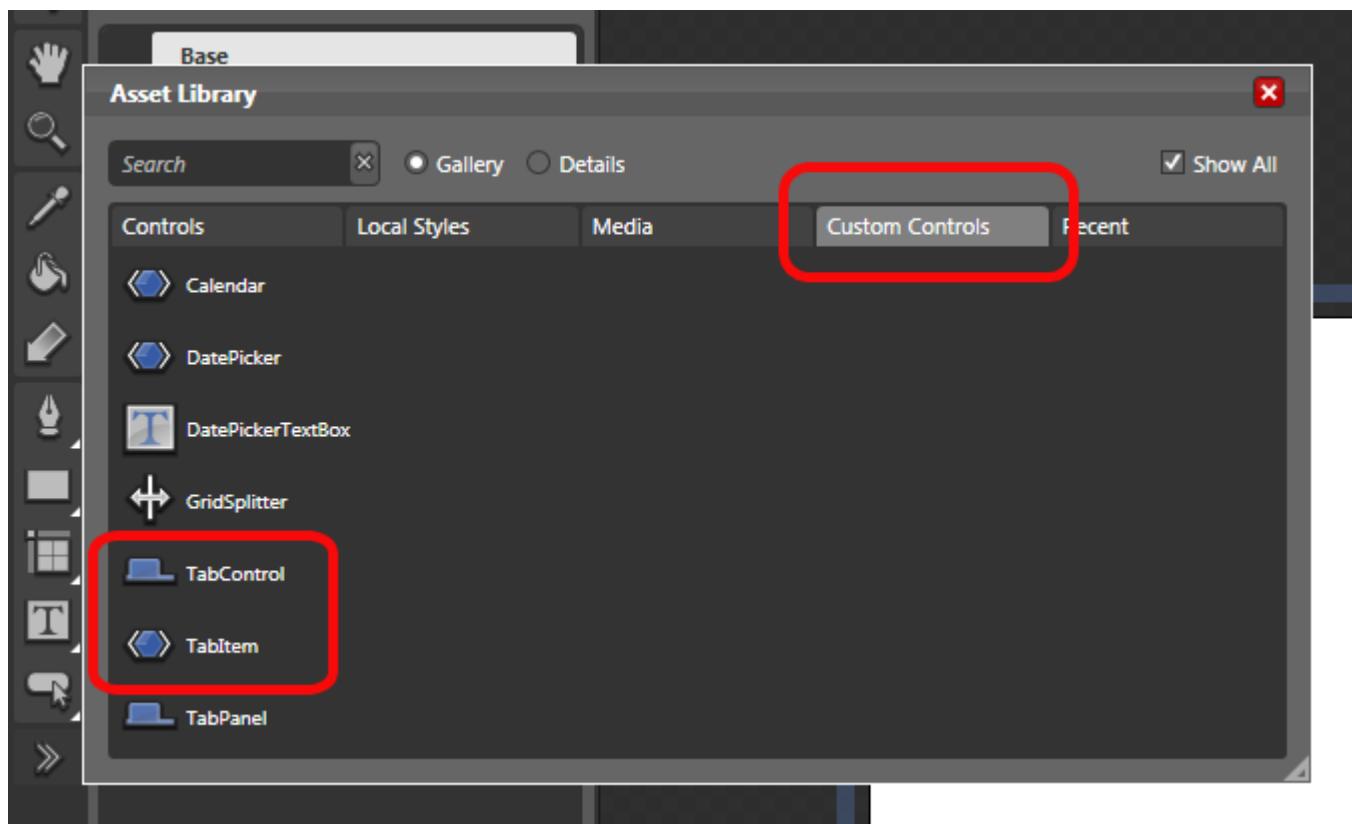
Yukarıdaki örnek kodlar içerisinde WCF servisimizin LoginAsync metodunu kullanarak bir Login işlemi yapmaya çalışıyoruz. Yine servis ile beraber gelen event'lardan biri olan LoginCompleted durumunda ise Login işleminin başarılı olup olmadığını ekran'a yazdırıyoruz. Bu şekilde servis içerisinde kullanabileceğiniz **Logout**, **IsLoggedIn** gibi metodlar da bulunuyor.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 2 ile beraber gelen TabControl incelemesi

Silverlight 2.0 Beta 2 ile beraber gelen yeni kontrollerden biri olan **TabControl** özellikle Windows uygulamalarından alışmış olduğumuz sayfali uygulama tasarımlı ekranları web ortamında da rahatlıkla oluşturabilmemizi sağlıyor. **System.Windows.Controls.Extended** sınıfı altında bulunan TabControl'u kullanabilmek için projenize söz konusu sınıfı reference olarak eklemiş olmanız gerekiyor. Visual Studio içerisinde Silverlight projenize sağ tuş tıklayarak "Add Reference" dedikten sonra gerekli eklemeleri yapabilirsiniz. Visual Studio içerisinde araç çubuğundan bir TabControl alarak sahneye yerleştirdiğinizde de işlem otomatik olarak gerçekleşecektir. Referanslama kısmı tamamlandıktan sonra Expression Blend içerisinde de Asset Library'de **Custom Controls** kısmında projenize referans olarak eklediğiniz sınıfların altındaki kontrolleri bulabilirsiniz.



Expression Blend 2 June Preview içerisinde TabControl ve TabItem

Expression Blend içerisinde sahneye bir TabControl yerleştirdikten sonra sıra geldi söz konusu **TabControl** içerisinde **TabItem** (sayfa) yerleştirmeye. Kolaylık olması açısından Blend içerisinde yerleştirmiş olduğunuz TabControl'a "Objects and Timeline" penceresinde çift tıklarsanız söz konusu kontrolün sarı bir çerçeve içerisinde alındığını göreceksiniz. Bu şekilde herhangi bir kontrol sarı bir çerçeve ile işaretlendiğinde o kontrol dışında ekranda bulunan her şey kilitlenmiş olacaktır. Böylece rahatlıkla ekrana yerlestireceğimiz yeni **TabItem** kontrollerinin kesinlikle **TabControl** içerisinde yerleştirileceğini garanti edebiliriz. Aksi halde fare ile kontrol eklerken özel olarak dikkat etmeniz gerekecektir.

```
<UserControl x:Class="SilverlightApplication3.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```

Width="400"
Height="300"
xmlns:System_Windows_Controls="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Extended"
xmlns:System_Windows_Controls_Primitives="clr-
namespace:System.Windows.Controls.Primitives;assembly=System.Windows.Controls.Exten-
ded">
<Grid x:Name="LayoutRoot"
      Background="White">
<System_Windows_Controls:TabControl HorizontalAlignment="Left"
      Margin="8,34,0,64"
      Width="184">
<System_Windows_Controls:TabItem Content="TabItem"
      Header="Tab1"/>
<System_Windows_Controls:TabItem Content="TabItem2"
      Header="Tab2"/>
</System_Windows_Controls:TabControl>
</Grid>
</UserControl>

```

Yukarıdaki kod içerisinde yerleştirdiğimiz TabControl ve TabItem'ların XML kodunda namespace olarak uzun uzun System_Windows_Controls adını görüyorsunuz. Aslında bu yapıyı değiştirebiliriz; eğer dokümanın üzerindeki namespace isimlerini değiştirirseniz aynı isimleri kodunuz içerisinde de rahatlıkla kullanabilirsiniz.

TabItem'ların iki önemli özelliği var; bunlardan ilki **Header** yani TabItem'im sayfa bilgisinin gözüktüğü yerde yazılacak olan yazı, diğeri ise **Content** yani TabItem'in temsil ettiği sayfada gösterilecek olan içerik. Şimdi örneğimizdeki hem namespace'leri değiştirerek daha okunaklı bir isim verelim hem de TabItem'larımızın içerisinde daha farklı içerikler yerlestirelim.

```

<UserControl x:Class="SilverlightApplication3.Page"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             Width="400"
             Height="300"
             xmlns:Ex="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Extended"
             xmlns:Ex="clr-
namespace:System.Windows.Controls.Primitives;assembly=System.Windows.Controls.Exten-
ded"
             xmlns:vsm="clr-namespace:System.Windows;assembly=System.Windows">
<Grid x:Name="LayoutRoot"
      Background="White">
<Ex:TabControl HorizontalAlignment="Left"
      Margin="8,34,0,64"
      Width="184">
<Ex:TabItem>
<Ex:TabItem.Header>
<Grid>
<Image HorizontalAlignment="Right"

```

```

    Width="76"
    Source="Dock.jpg"/>
<TextBlock>Bölüm 1</TextBlock>
</Grid>
</Ex:TabItem.Header>
<TextBlock>Deneme amaçlı metin</TextBlock>
</Ex:TabItem>
<Ex:TabItem Content="TabItem2"
    Header="Tab2"/>
</Ex:TabControl>
</Grid>
</UserControl>

```

Kodumuz içerisinde yer alan TabItem'in hem Header (başlık) kısmını hem de içeriğini özel olarak düzenliyoruz. **<Ex:TabItem.Header>** tagları arasında TabItem için header görseli olarak farklı Silverlight kontrolleri kullanabiliyoruz. Tek bir sınırlamamız var; **Header** içerisinde kök element sadece bir adet olabiliyor. Bu sorunu aşmak için Container Elementlerimizden Grid'i kullanabiliriz. Header içerisinde yerleştirdiğimiz bir Grid içerisinde istedigimiz kadar Silverlight kontrolü koyabiliyoruz. Header tagları haricinde doğrudan TabItem'in içerisinde de TabItem'in sayfa içerisinde gözükmescini istediğimiz kontrolleri koyabiliyoruz.



Özelleştirilmiş TabItem kontrolümüz karşımızda!

TabItem'ların Header'ları içerisinde farklı Silverlight kontrollerini koymakın yanı sıra istersek Header'in tamamen görsel şablonunu da değiştirebiliriz. Bunun için bir **ControlTemplate** hazırlayarak TabItem'ımıza bağlamamız gerekecek.

```

<UserControl x:Class="SilverlightApplication3.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300"
    xmlns:Ex="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Extended"
    xmlns:Prj="clr-
namespace:System.Windows.Controls.Primitives;assembly=System.Windows.Controls.Exten-
ded">

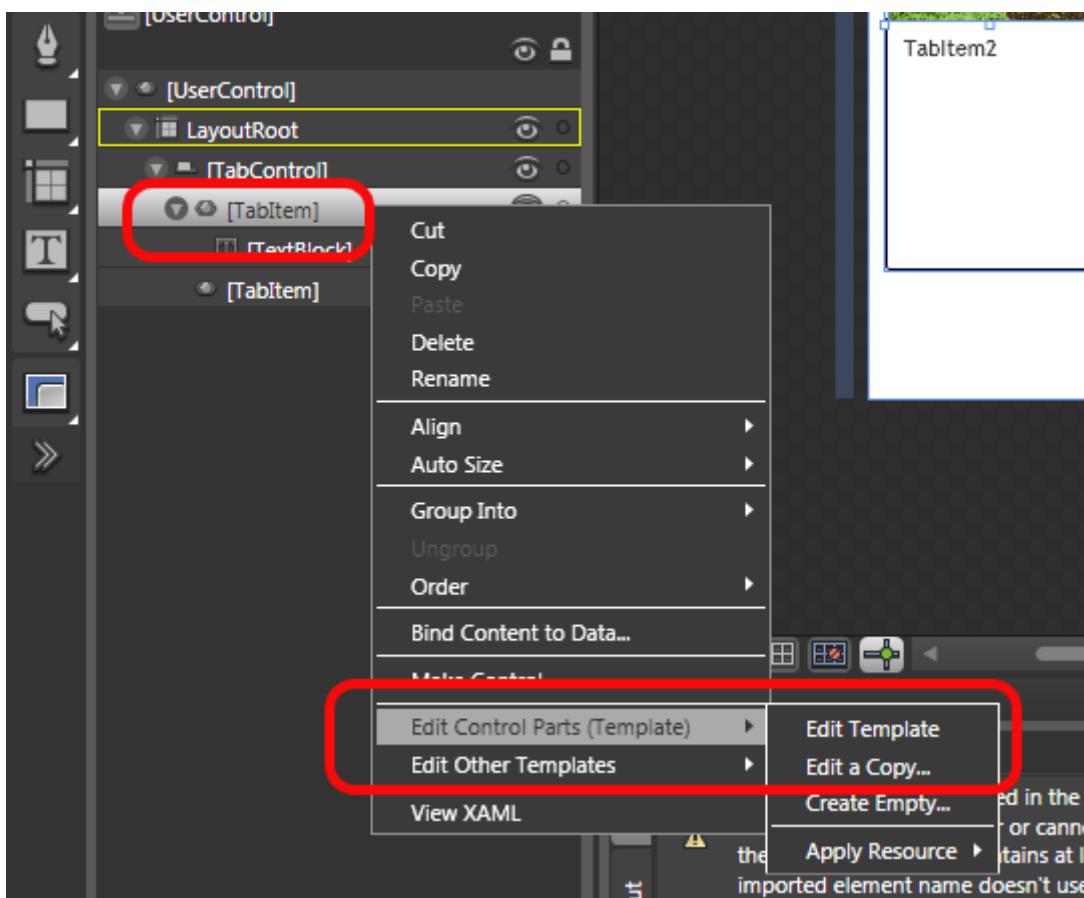
```

```

xmlns:vsm="clr-namespace:System.Windows;assembly=System.Windows">
<UserControl.Resources>
<ControlTemplate x:Key="TabItemControlTemplate1"
    TargetType="Ex:TabItem">
<Grid>
    <Image HorizontalAlignment="Left"
        Width="100"
        Source="Forest.jpg"/>
    <ContentPresenter Content="{TemplateBinding Header}" />
</Grid>
</ControlTemplate>
</UserControl.Resources>
<Grid x:Name="LayoutRoot"
    Background="White">
    <Ex:TabControl HorizontalAlignment="Left"
        Margin="8,34,0,64"
        Width="184">
        <Ex:TabItem Template="{StaticResource TabItemControlTemplate1}">
            <Ex:TabItem.Header>
                <Grid>
                    <Image HorizontalAlignment="Right"
                        Width="76"
                        Source="Dock.jpg"/>
                    <TextBlock>Bölüm 1</TextBlock>
                </Grid>
            </Ex:TabItem.Header>
            <TextBlock>Deneme amaçlı metin</TextBlock>
        </Ex:TabItem>
        <Ex:TabItem Content="TabItem2"
            Header="Tab2"/>
    </Ex:TabControl>
</Grid>
</UserControl>

```

Kodumuzda yarattığımız ControlTemplate içerisinde bir Grid ve onun içinde de bir Image ile **ContentPresenter** yer alıyor. Söz konusu ContentPresenter'in Content özelliğini Template'in uygulanacağı kontrolün Header özelliğine bağlanmış. Böylece bu şablonu bağlı bir TabItem'ın Header'ına yerleştirilen kontrollerin bu şablon uygulandığında şablon içerisindeki ContentPresenter'in içerisine yerleştirilecek. XAML kodunu çok uzatmamak adına örnekte sürekli Image nesneleri kullandığımız için ortaya çıkan örnek çok anlamlı olmayacağı fakat Expression Blend içerisinde biraz daha detaylı bir çalışma ile güzel sonuçlar alınabilir.



Expression Blend 2 July Preview içerisinde Silverlight ControlTemplate desteği.

Tüm bu yapıları tamamen XAML kodları yazarak oluşturabileceğiniz gibi Expression Blend içerisinde araçları kullanarak da yapabilirsiniz. Herhangi bir TabItem kontrolüne sağ tuş ile tıklayarak yukarıdaki şekilde "Edit Control Parts / Edit Template" diyerek TabItem'ların görselliklerini Blend içerisinde de ayarlayabilirsiniz. Her kontrol için ilk başta "Edit a Copy" diyerek var olan görsellikten bir şablon kopyası alarak veya "Create Empty" diyerek boş bir şablon yaratarak ilerleyebilirsiniz.

Son olarak her TabControl'ün bir de **TabStripPlacement** özelliği olduğundan bahsetmek gerek. Bu özelliğe verdığınız değerler ile TabItem'ların Header kısımlarının TabControl'ün üstünde, sağında, solunda veya altında gözükmesini sağlayabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 2 ve PHP ile mailform uygulaması

İster PHP olsun ister farklı sunucu taraflı programlama dilleri olsun hepsi de "adı üzerinde" sunucu tarafında çalışıyorlar. Biz ise Silverlight tarafından tamamen istemcide çalışıyor. Bu çerçevede Silverlight'in tamamen sunucudan bağımsız olduğunu düşünürsek aslında sunucu ile belirli standartları yakaladığımız sürece istediğimiz sunucu taraflı programlama altyapısı ile entegrasyon sağlayabiliriz. Bu standartlar WSDL kuralları çerçevesinde hazırlanmış bir web servisi olabileceği gibi bazen çok basit bir POST işlemi bile olabilir. Bu yazımда **Silverlight 2.0 Beta 2** ile beraber sunucu tarafında bir PHP kodu kullanarak mail gönderim işlemi yapacağız. Hazırladığımız Silverlight 2.0 uygulamasının XAP dosyasını sunucuya atmamız uygulamamızın çalışması için yeterli olacaktır.

Önce PHP tarafını çözelim!

PHP tarafında çok detaya girmeyeceğiz. Yapacağımız şey basit bir şekilde sayfaya POST ile gönderilen değişkenleri alıp uygun bir mail mesaj stringi haline çevirdikten sonra mail olarak istediğimiz kullanıcıya göndermek olacak.

```
<?php
$senderName = $_POST['Gonderen'];
$senderEmail = $_POST['Email'];
$emailMessage = $_POST['Mesaj'];
$recipient = "alici@domain.com";
$subject = "Mesaj Konusu";
$headers = "From: $Email";
$message = "Kimden: $Gonderen\nEposta Adresi: $Email\n\n Mesaj: $Mesaj";
$message = stripslashes($message);
mail($recipient, $subject, $message, $headers)
?>
```

Örneğimize devam ederken ufak bir uyarıda bulunmam gereklidir. Kesinlikle yukarıdaki gibi bir PHP dosyasını sitenize bu haliyle bırakmayın. Şu an yukarıdaki dosyada ne post eden arkadaşın kimliği, ne sender'in agent tipi hiçbir şey kontrol edilmiyor. Güvenlik açısından kesinlikle bu kodun geliştirilmesi gereklidir aksi halde önüne gelen buraya bilgileri POST ederek size milyonlarca mail yollayabilir.

Uygulamamızın tasarımını yapalım

Yine çok basit bir mailform hazırlayacağız. Blend 2.5 içerisinde Silverlight sayfamıza toplam üç adet TextBox ve bir de Button koyuyoruz. Site ziyaretçileri isimlerini, maillerini ve mesajlarını yazarak düğmeye basıp gize gönderebilecekler. Oluşturduğumuz uygulamanın XAML kodunu aşağıda inceleyebilirsiniz.

```
<UserControl x:Class="SilverlightApplication2.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Height="32" Margin="96,42,74,0" VerticalAlignment="Top" Text="Adınız"
            TextWrapping="Wrap" x:Name="txtAdi"/>
```

```

<TextBox Height="29" Margin="96,88,74,0" VerticalAlignment="Top"
Text="Mailiniz" TextWrapping="Wrap" x:Name="txtMaili"/>
<TextBox Margin="96,132,74,81" Text="Mesajınız" TextWrapping="Wrap"
x:Name="txtMesaji"/>
<Button Height="32" HorizontalAlignment="Stretch" Margin="180,0,146,31"
VerticalAlignment="Bottom" Content="Gönder" x:Name="btnGonder"/>
</Grid>
</UserControl>

```

Kodlamaya geçelim

Uygulamamızın kod kısmında bir WebClient nesnesi kullanacağız. WebClient nesnemize sahnedeki tüm bilgileri bir String olarak vererek POST metodu ile bilgileri kendisine parametre olarak vereceğimiz bir adrese göndermesini isteyeceğiz.

[VB]

```

Dim VeriGonder As New System.Net.WebClient
VeriGonder.Headers[HttpRequestHeader.ContentType] = "application/x-www-form-
urlencoded"

```

[C#]

```

System.Net.WebClient VeriGonder = new System.Net.WebClient();
VeriGonder.Headers[HttpRequestHeader.ContentType] = "application/x-www-form-
urlencoded";

```

VeriGonder adını verdigimiz **WebClient** nesnemin hemen **ContentType** bilgisini ayarlamam gerekiyor. Bunun için **WebClient'in Headers** dizisinden **ContentType'ı** bularak form-urlencoded olarak değiştirmiyorum. Böylece birazdan **URLEncode** tekniği ile hazırladığımız verileri bu WebClient ile rahatlıkla sunucuya gönderebileceğiz.

[VB]

```

Dim GonderilecekData As String = "Gonderen=" &
Browser.HttpUtility.UrlEncode(txtAdi.Text) & "&"
GonderilecekData &= "Email=" & Browser.HttpUtility.UrlEncode(txtMaili.Text) & "&" 
GonderilecekData &= "Mesaj=" & Browser.HttpUtility.UrlEncode(txtMesaji.Text)

```

[C#]

```

string GonderilecekData = "Gonderen=" +
System.Windows.Browser.HttpUtility.UrlEncode(txtAdi.Text) + "&";
GonderilecekData += "Email=" +
System.Windows.Browser.HttpUtility.UrlEncode(txtMaili.Text) + "&";
GonderilecekData += "Mesaj=" +
System.Windows.Browser.HttpUtility.UrlEncode(txtMesaji.Text);

```

Kodumuz içerisinde hemen göndereceğimiz verileri **Key/Value** çiftleri şeklinde birleştiriyoruz. Göndereceğimiz her verinin bir ismi ve tabi ki değeri olması gerekiyor.

Aslında yaptığımız şey normalde URL üzerinden göndereceğimiz veriyi **URLEncode** ile aynı şekilde oluşturmak. Eğer göndereceğiniz verilerin sayısı çok ise performans açısından standart **String** işlemleri yerine bir **StringBuilder** kullanmanızı tavsiye ederim.

[VB]

```
AddHandler VeriGonder.UploadStringCompleted, AddressOf
VeriGonder_UploadStringCompleted
    VeriGonder.UploadStringAsync(New
Uri("http://localhost:49424/SilverlightApplication2Web/mailgonder.php", UriKind.Absolute),
"POST", GonderilecekData)
```

[C#]

```
VeriGonder.UploadStringCompleted += VeriGonder_UploadStringCompleted;
    VeriGonder.UploadStringAsync(new
Uri("http://localhost:49424/SilverlightApplication2Web/mailgonder.php", UriKind.Absolute),
"POST", GonderilecekData);
```

Son olarak verimizi sunucuya göndermeden önce gönderme işlemi tamamlandığında çalıştırılmak üzere **VeriGonder** nesnemizin **UploadStringCompleted** event'ına da bir event-handler bağlıyoruz. Artık verimizi sunucuya göndermeye hazır olduğumuza göre hemen adresini vererek **POST** metodu ile veriyi yolculayabiliriz.

[VB]

```
Private Sub VeriGonder_UploadStringCompleted(ByVal sender As Object, ByVal e As
System.Net.UploadStringCompletedEventArgs)
    btnGonder.Content = "Tamam"
End Sub
```

[C#]

```
private void VeriGonder_UploadStringCompleted(object sender,
System.Net.UploadStringCompletedEventArgs e)
{
    btnGonder.Content = "Tamam";
}
```

Veri gönderme işlemi tamamlandığında ekrandaki kontrolleri kaldırıp bir teşekkür mesajı göstermek güzel olabilirdi. İşin o kısmını ben size bırakmış olyim. Simdilik **UploadStringCompleted** event'ında düğmeye "Tamam" yazdırarak örneğimizi çalıştırabiliriz.

@ İşaret Sorunu!

Ufak bir sorunumuz var. **Silverlight 2.0 Beta 2** ile beraber gelen bu sorun ufak gibi gözükmekte de aslında epey önemli :) Yukarıdaki örneği çalıştırıldığınızda göreceksiniz ki herhangi bir TextBox içerisinde **@** işaretini koyamıyorsunuz. Bunun basit bir nedeni var, aslında **AltGr** tuşu ile oluşturduğunuz hiçbir karakteri TextBox'lara yerleştiremeyeceksiniz. Neden mi?

Bilmiyorum, bu bir bug. Silverlight 2.0'in Beta 2 sonrasında sürümlerinde bu hata giderilecek. Simdilik aşağıdaki gibi bir çözüm uygulayabiliriz.

[VB]

```
Dim Oncekiler(1) As Integer
```

```
Private Sub txtMaili_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Input.KeyEventArgs) Handles txtMaili.KeyDown
    If e.PlatformKeyCode = 81 Then
        If Oncekiler(0) = 17 And Oncekiler(1) = 18 Then
            txtMaili.Text &= "@"
            txtMaili.SelectionStart = txtMaili.Text.Length
        End If
    End If
    Oncekiler(0) = Oncekiler(1)
    Oncekiler(1) = e.PlatformKeyCode
End Sub
```

[C#]

```
int[] Oncekiler = new int[2];

private void txtMaili_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)
{
    if (e.PlatformKeyCode == 81)
    {
        if (Oncekiler[0] == 17 & Oncekiler[1] == 18)
        {
            txtMaili.Text += "@";
            txtMaili.SelectionStart = txtMaili.Text.Length;
        }
    }
    Oncekiler[0] = Oncekiler[1];
    Oncekiler[1] = e.PlatformKeyCode;
}
```

İlk önce uygulamaya çalıştığımız mantığı kavrayalım. **txtMaili** adındaki **textBox** içerisinde herhangi bir tuşa basıldığında yukarıdaki gibi **KeyDown** event'i çalışacaktır. Bu event'a baktığımızda klavyede **AltGr** tuşuna basıldığında sırası ile iki tuşa basılmış gibi sistemin 17 ve sonrasında da 18 numaralı PlatformAnahtarları'nı döndürduğunu görebiliriz. Bu tuşların Macintosh dahil tüm sistemlerdeki **PlatformKeyCode** adında anahtarları vardır ve bu değerler sürekli aynıdır. Normalde biz @ işaretini koyabilmek için AltGr'ye bastıktan sonra bir de Q harfine basarız. O zaman kontrol etmemiz gereken durum şu; Q harfine basıldıysa acaba bir önceki basılan tuş **AltGr** miydi? Eğer öğleyse bana bir @ işaretini lazım. İşte kodumuz da bu kontrolü yapıyor. Sürekli olarak basılan son iki tuşu **Oncekiler** adındaki dizimizde saklıyor ve her tuşa basıldığında **KeyDown** içerisinde eğer Q harfine basılmış ise son basılan iki tuşun **KeyCode'larının** da 17 ve 18 olup olmadığını kontrol ediyoruz. Eğer durum buyusa txtMaili **TextBox**'ı içerisinde bir @ işaretini ekleyip imleci metnin en sonuna gönderiyoruz.

Sonuç

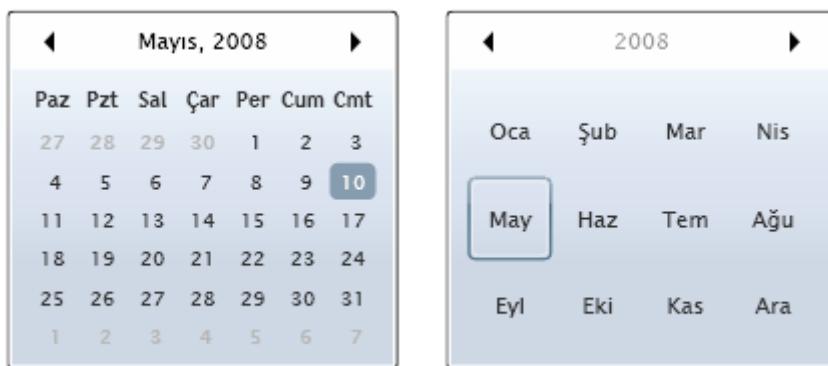
Makalemizde kullandığımız teknik aslında web programcılığının en ilkel zamanlarından bu güne kadar gelen ve yapı taşı diyebileceğimiz POST metodunun ta kendisi. Bu çerçevede sunucu taraklı programlama sistemlerinin hepsi bu şekilde veri trafiğine açık olduğu için aynı teknikler ile Silverlight'ı sunucu tarafı ile rahatlıkla konuşturabilir ve ister sunucunun işletim sistemi olsun, ister kullanılan teknoloji olsun her konuda tam bağımsızlığın tadını çıkartabilirisiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Calendar ve DatePicker kontrolleri.

Silverlight 2.0 ile beni en çok şaşırtan özelliklerden biri de WPF'in web sürümü diyeBILECEĞİMİZ ve WPF'e kıyasla bir çok eksiği olan bir torun olarak Silverlight ile beraber artık WPF'de bulunmayan bazı kontrollerin geliyor olması. Tahminen uzun vadede her iki taraf da birbirinden besleniyor olacaktır. Bugün baktığımızda ilk dikkati çeken kontrollerden biri de maalesef WPF tarafından olmayan **Calendar** ve **DateTimePicker** kontrolleri. Bu yazımızda bu iki kontrolü ve bu kontrollerle neler yapabileceğimizi inceleyeceğiz.



Calendar kontrolü şekilde şékilden şékilde girebiliyor.

Yukarıdaki şékli ile birer Calender kontrolü yaratmak için tek yapmanız gereken XAML kodunuza aşağıdaki şékilde düzenlemek.

```
<UserControl x:Class="Calendar.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Calendar Margin="21,20,145,57"/>
    </Grid>
</UserControl>
```

Tek yaptığımız bir Calendar tagı açarak kenarlardan olan uzaklığını belirtmek. "Çocuk oyuncası" denen bu olsa gerek. Peki daha neler yapabiliriz? Aslında bu aşamadan sonra bahsedeceğimiz tüm özellikler Calendar ve DatePicker kontrolleri için birebir aynı. O nedenle gelin öncesinde bir de DatePicker kontrolünün XAML koduna bakalım.

```
<DatePicker Height="20" Margin="42,0,156,23" VerticalAlignment="Bottom"/>
```

DatePicker kontrolünü de sahneye yerleştirmek en az Calendar kontrolü kadar basit. Bu durumda hızlıca programatik işlevselliklere göz atabiliriz. Aşağıdaki gibi bir Silverlight uygulaması hazırlayarak kodlamaya başlayalım.

```
<UserControl x:Class="Calendar.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
```

```
<Calendar Margin="21,20,145,57" x:Name="Takvim"/>
<DatePicker Height="20" Margin="42,0,156,23" VerticalAlignment="Bottom"
x:Name="TarihSecici"/>
</Grid>
</UserControl>
```

Uygulamamızda **Takvim** adında bir **Calendar** kontrolü ve **TarihSecici** adında bir **DateTimePicker** bulunuyor. Bu kontroller Silverlight uygulaması ile beraber ilk gösterildiklerinde içlerinde herhangi bir tarih seçili gelmiyor. Oysa aşağıdaki şekilde güncel tarihi seçili hale getirebilirsiniz.

```
Takvim.SelectedDate = Date.Now
TarihSecici.SelectedDate = Date.Now
```

Makalemizin en üstündeki görsele baktığınızda Calendar kontrolünün iki farklı görsel durumunun bulunduğulığını görebilirsiniz. Normal şartlarda Calendar kontrolü günleri gösterecek şekilde açılıyor, sonrasında eğer üstteki ay ismine tıklarsanız ayların seçilebileceği arayüz geliyor. Oysa isterseniz Calendar kontrolü sayfada ilk açıldığında da ayların seçilebileceği arayüzün otomatik gelmesini sağlayabilirsiniz.

```
Takvim.DisplayMode = CalendarMode.Year
```

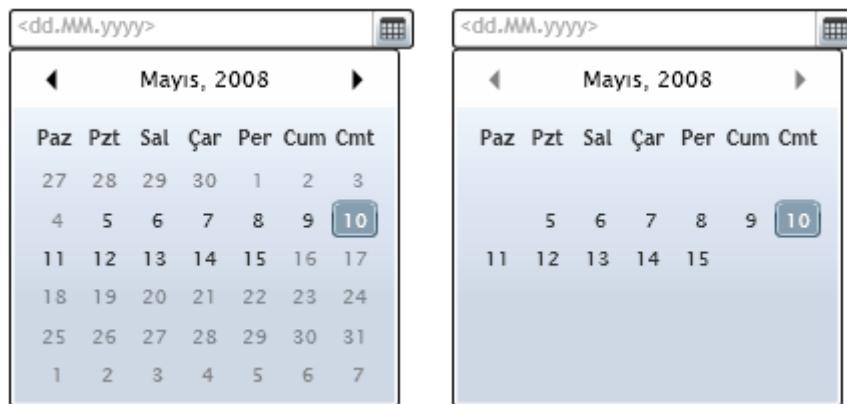
Eğer Calender nesnesinin **DisplayMode** özelliği **Month** olursa tam bir ayı gösteriyor, **Year** şeklinde düzenlendiğinde ise tüm yılı yani ayları gösteriyor. Bunun haricinde isterseniz her iki kontrolün de hangi tarih aralıklarını gösterebileceğini ek olarak düzenleyebilirsiniz.

```
TarihSecici.SelectableDateStart = Date.Today.Subtract(New TimeSpan(5, 0, 0, 0))
TarihSecici.SelectableDateEnd = Date.Today.AddDays(5)
```

Yukarıdaki kod içerisinde DatePicker kontrolümüzün **SelectableDateStart** ve **SelectableDateEnd** özelliklerine **DatePicker** içerisinde seçilebilecek başlangıç ve bitiş tarihlerini aktarıyoruz. Bunu yaparken söz konusu tarihlerin hesaplamalarını da tabii ki dinamik olarak yapabilirsiniz. Böylece bu örneğimizde kontrol sürekli olarak mevcut tarihden 5 gün öncesinin ve 5 gün sonrasının seçilebilmesine olanak tanıyacaktır.

```
TarihSecici.DisplayDateStart = Date.Today.Subtract(New TimeSpan(5, 0, 0, 0))
TarihSecici.DisplayDateEnd = Date.Today.AddDays(5)
```

Seçili tarihleri belirlemenin yanı sıra isterseniz belirli tarihleri seçilemez yapmanın yanı sıra tamamen o tarihlerin gösterilmemesini de sağlayabilirsiniz. Bunun için **DisplayDateStart** ve **DisplayDateEnd** özelliklerinden faydalananabilirsiniz.



SelectableDate ve DisplayDate arasında fark.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Cross-Domain WebClient ile REST (GET) ve XLINQ Kullanımı

Silverlight 2.0 ile beraber istemci tarafında .NET dillerini kullanabildiğimizi ilk duyduğumda aklıma gelen ilk şey **WebClient** sınıfını artık istemci tarafında da kullanıp kullanamayacağım olmuştu. Kesinlikle kullanabiliyoruz, hatta bununla kalmayıp istersek daha detaylı bir kullanım için **HttpWebRequest'i** de tercih edebiliriz. Tüm bu sınıflar bize REST kullanımında yardımcı oluyorlar. Normal şartlarda uygulamalar arasında veri transferi için WSDL tanımlarına sahip servislerin kullanımı tavsiye edilse de hala maaleshf herhangi bir kural tanımı olmayan veri kaynaklarını da kullanmak durumunda kalabiliyoruz. İşte tam bu noktada WebClient basit işlemler için imdadımıza yetişiyor. Eğer farklı HTTP Verb'lerini (GET, PUT, POST, DELETE) kullanacaksanız daha detaylı işlemler için **HttpWebRequest'i** tercih etmeniz gerekecektir. **WebClient** işin sadece **GET** kısmında yer alıyor.

Örneğimizdeki Silverlight 2.0 Beta 1 uygulamasında **System.Net.WebClient** sınıfını kullanarak sunucudaki bir xml dosyasını okuyacağız. İlk olarak okuduğumuz veriyi göstermek üzere uygulamamıza bir TextBlock ve veriyi çekme işlemini başlatmak üzere bir de Button ekleyerek aşağıdaki XAML kodunu yaratalım.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Height="47" Margin="57,45,151,0" VerticalAlignment="Top"
Text="TextBlock" TextWrapping="Wrap" x:Name="Metin"/>
        <Button HorizontalAlignment="Stretch" Margin="94,136,151,121"
VerticalAlignment="Stretch" Content="Button" x:Name="Dugme"/>
    </Grid>
</UserControl>
```

Sayfamız hazır olduğuna göre artık WebClient nesnemizi yaratacak olun kodu düğmemizin arkasına yazabilirim.

```
Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme.Click
    Dim Istek As New System.Net.WebClient
    AddHandler Istek.DownloadStringCompleted, AddressOf
istek_DownloadStringCompleted
    Istek.DownloadStringAsync(New Uri("http://www.alanadi.com/veri.xml"))
End Sub
```

Yukarıdaki kodumuz içerisinde ilk olarak **WebClient** nesnemizi yaratıyoruz. Bir sonraki adımda elimizdeki WebClient nesnesinin **DownloadStringCompleted** durumunu harici bir event-handler'a bağlıyoruz. Bunu yapmamızın nedeni WebClient sınıfının kendisine verilen bir adresen alacağı veriyi tamamen asenkron olacak alıyor olması. Yani sunucudan veri tam olarak geldiğinde bizi haberدار edecek olan event-handları tanımlamamız gerekiyor. Son adımda ise **DownloadStringAsync** komutuna hedef adresi de bir **Uri** değişkeni olarak aktararak veri talebimizi sunucuya göndermiş oluyoruz. Sıra geldi veri geldiğinde çalıştırılacak olan event-handler kodunu yazarak veriyi TextBlock içerisinde yazdırımıya. Fakat onun öncesinde ilk olarak hedef aldığımız XML dosyasının yapısına bir göz atalım.

```
<?xml version="1.0" encoding="utf-8" ?>
<Root>
  <Kayitlar>
    <Kayit>
      <Adi>Ahmet</Adi>
    </Kayit>
    <Kayit>
      <Adi>Daron</Adi>
    </Kayit>
    <Kayit>
      <Adi>Mehmet</Adi>
    </Kayit>
  </Kayitlar>
</Root>
```

Örnek XML dosyamızı sunucudan istemciye aldıktan sonra biz sadece adının içerisinde "Dar" geçen ilk kaydı bularak onun Adı'ni göstermek istiyoruz. Bu XML dosyası çok daha farklı olabilirdi, içerisinde ID ve Adı bilgileri olan bir ürün listesi olabilir ve istemci tarafına aldıktan sonra farklı şekillerde filtrelemek isteyebilirdiniz. Tüm bunları rahatlıkla yapabilmek için Silverlight 2.0 Beta 1 ile istemci tarafında **XLINQ** kullanacağız. Silverlight 2.0 içerisinde **XLINQ** kullanabilmek için Silverlight projenizde "Solution Explorer" içerisinde sağ tıklayarak gelen menüden "Add Reference" komutu vermeniz ve "System.Xml.Linq" sınıfını eklemeniz gerekiyor. Sonrasında artık normalde olduğu gibi **XDocument** ve tüm **XLINQ** özelliklerinden faydalanabiliriz.

```
Private Sub istek_DownloadStringCompleted(ByVal sender As Object, ByVal e As System.Net.DownloadStringCompletedEventArgs)
    Dim YeniDoc As Xml.Linq.XDocument = Xml.Linq.XDocument.Parse(e.Result)
    Metin.Text = (From Gelenler In YeniDoc.<Root>.<Kayitlar>.<Kayit> _
        Where Gelenler.<Adi>.Value.Contains("Dar") _
        Select Gelenler.<Adi>.Value).SingleOrDefault
End Sub
```

Bir önceki adımda düğmeye tıklandığında yarattığımız WebClient nesnesine aktardığımız event-handları burada tanımlıyoruz. Böylece sunucudan veri geldiğinde bu metod çalıştırılıyor olacak. Sunucudan gelen ham veriye **e.Result** ile ulaşabiliyoruz. Gelen veri özünde XML olacağı için hemen bir **XDocument** yaratarak XDocument sınıfının **Parse** özelliği ile verimizi işlenebilir hale getiriyoruz. Son adımda ise klasik bir **XLINQ** sorgusu yazarak adında "Dar" geçen kaydı bularak değeri **Metin** adındaki **TextBlock** nesnemize aktarıyoruz.

Böylece sunucudan farklı bir XML dosyasını alarak **XLINQ** ile rahatlıkla istemci tarafında işleyebildiğimizi gördük.

Peki ya başkan bir alan adından veri çekmek istersek?

Aslında bu bölümde bahsedeceğimiz sorun Silverlight'dan bağımsız olup tüm AJAX uygulamalarında geçerli bir sorun. Maalesef tarayıcılardaki uygulamalar güvenlik sebepleri ile kendi çalışıkları alan adı haricindeki konumlardan veri alamaz veya gönderemezler. Bu

nedenle maalesef Silverlight tarafından da yola çıkarak başka bir alan adından veri almak mümkün değil gibi gözükebilir. Oysa bir yol var.

İster harici klasik ASMX Web Servisleri, ister WCF servisleri veya ister doğrudan REST kullanmak isteyin, harici bir alan adına ulaşmak istiyorsanız aslında söz konusu alan adındaki veri kaynağının size ulaşım izni vermiş olması gerekiyor. Silverlight 2.0 karşı hedef alan adında **clientaccesspolicy.xml** adında bir dosya arar. Eğer bu dosyayı bulabiliyorsa içerisinde yazılı kurallar çerçevesinde sizin söz konusu alan adındaki içeriğe ulaşmanıza izin verir.

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from>
        <domain uri="*"/>
      </allow-from>
      <grant-to>
        <resource path="/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

Yukarıdaki gibi bir **clientaccesspolicy.xml** dosyası herhangi bir alan adından hedef alan adındaki her konuma ulaşabileceğin anlamına gelir. İsterseniz bu dosyayı değiştirerek farklı kurallar koyabilir, sadece belirli alan adlarında çalışan uygulamaların hedef konuma ulaşabilmesini sağlayabilirsiniz.

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from>
        <domain uri="http://daron.yondem.com"/>
      </allow-from>
      <grant-to>
        <resource path="/servisler/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

Örneğin yukarıdaki gibi bir policy dosyasında sadece **daron.yondem.com** adresinden xml dosyasının bulunduğu alan adında **servisler** klasörü içerisinde kaynaklara ulaşabileceğin tanımlanmış. Policy dosyaları ile ilgili detaylara [buradan](#) ulaşabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 GridSplitter Kullanımı

Silverlight 2.0 içerisinde **Grid** kullanımı HTML içerisinde alışık olduğumuz Table yapısından pek farklı değil. Kolonlar ve satırlar yaratarak görsel öğeleri ekranda konumlandırıbilmenizi sağlayan Grid kontrolü ile beraber kullanabileceğimiz kontrollerden biri de **GridSplitter** kontrolü. GridSplitter kontrolü bir Grid'in kolon veya satırlarının kullanıcı tarafından fare ile boyutlandırılabilmesini sağlıyor. Herhangi bir Grid yaratarak satır veya sütunlar oluşturduktan sonra istediğiniz **Grid** hücresinde **GridSplitter** kontrolü yerleştirebiliyorsunuz.

```
<UserControl x:Class="GridSplit.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.435*"/>
            <ColumnDefinition Width="0.07*"/>
            <ColumnDefinition Width="0.495*"/>
        </Grid.ColumnDefinitions>
        <GridSplitter Height="Auto" VerticalAlignment="Stretch" Grid.Column="1"
Background="#FF0092FF" Width="10" HorizontalAlignment="Center"/>
    </Grid>
</UserControl>
```

Yukarıdaki XAML kodunda yer alan Grid'in toplam üç kolonu var. Bu kolonlardan birinin içerisinde gözükecek şekilde bir de **GridSplitter** nesnesi yerleştirilmiş. GridSplitter nesnesinin **Grid.Column** özelliği 1 olduğu için içerisinde bulunduğu Grid'in 1 Index numaralı kolonunda gözükecek.



GridSplitter kontrolü 2 resmi boyutlandırıyor.

Yukarıdaki gibi bir örnek elde etmek için Grid'in diğer kolonlarına birer Image nesnesi yerleştirdim. GridSplitter kontrolünü fare ile tutup sürüklendiğimde her iki resim de içerisinde bulundukları kolonlara sığacak şekilde kendilerini boyutlandırdılar.

```
<UserControl x:Class="GridSplit.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="0.435*"/>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="0.495*"/>
    </Grid.ColumnDefinitions>
    <GridSplitter Height="Auto" VerticalAlignment="Stretch" Grid.Column="1"
Background="#FF0092FF" Width="10" HorizontalAlignment="Center"/>
        <Image Source="Creek.jpg"/>
        <Image Grid.Column="2" Source="Dock.jpg"/>
    </Grid>
</UserControl>
```

Yukarıdaki kod içerisinde dikkat etmemiz gereken nokta GridSplitter kontrolünü yerleştirdiğimiz Grid kolonunun **Width** özelliğinin **Auto** olması, böylece GridSplitter'in genişliği ne ise söz konusu kolonun genişliği de o olacaktır.

GridSplitter kontrolünü yukarıdaki taktikleri izleyerek sadece dikey olarak ekranı bölmek için değil yatay olarak Grid'in satırları arasında da kullanabilirsiniz. Ayrıca birden çok Grid kontrolünü iç içe kullanarak farklı ekranlar yaratmak da mümkün.

```
<UserControl x:Class="GridSplit.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.RowDefinitions>
        <RowDefinition Height="0.72*"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="0.237*"/>
    </Grid.RowDefinitions>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.435*"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="0.538*"/>
        </Grid.ColumnDefinitions>
        <GridSplitter HorizontalAlignment="Center" Width="10" Grid.Column="1"
Background="#FF5EFF00"/>
            <Image Source="Creek.jpg" Stretch="Uniform"/>
            <Image Grid.Column="2" Source="Dock.jpg"/>
        </Grid>
    </Grid>
</UserControl>
```

```
<GridSplitter HorizontalAlignment="Stretch" Width="Auto" Grid.Row="1"
VerticalAlignment="Center" Height="10" Background="#FF0092FF"/>
<Image Grid.Row="2" Source="Garden.jpg"/>
</Grid>
</UserControl>
```

Yukarıdaki örnekte toplamda üç satırı bulunan bir Grid kontrolünün ikinci satırında **GridSplitter** bulunuyor. Aynı Grid'in birinci satırında ise içerisinde üç kolun bulunan ayrı bir Grid var. İçteki bu Grid'in de ikinci kolonunda bir **GridSplitter** var. Böylece yatay GridSplitter kullanıldığında dikey olan ve iç Grid'de bulunan GridSplitter da otomatik olarak boyutlandırılmış oluyor.



2 Grid ve 2 GridSplitter'in kardeşliği.

Bu gibi arayüzler neredeyse çoğu yazılımda karşımıza çıkan sistemler içerisinde. Silverlight 2.0 ile beraber iş uygulamaları geliştirirken bu tarz kolaylıkların büyük bir iş yükünü omuzlarımızdan kaldıracağım kesin.

GridSplitter değişiklikleri algılamak?

GridSplitter'i kullanmak gerçekten çok kolay. Fakat istemci tarafında kullanıcı tüm GridSplitter'ları ayarladıkten sonra Silverlight uygulamasını başka bir zamanda tekrar açtığında tüm ayarları tekrar yapmak zorunda kalması hiç hoş olmaz. O nedenle GridSplitter ile yapılan ayarları bir şekilde saklamamız gereklidir.

Aslında GridSplitter'in yaptığı işlem içerisinde bulunduğu Grid'in kolonlarının boyutlarını değiştirmek öte değil. Bu durumda bizim Grid'in kolonlarında değişiklik olup olmadığı yakalamamız ve söz konusu değişiklikleri kaydetmemiz gerekiyor.

```
Private Sub IcGrid_LayoutUpdated(ByVal sender As Object, ByVal e As System.EventArgs)
Handles IcGrid.LayoutUpdated
    Metin.Text = IcGrid.ColumnDefinitions(0).Width.ToString
End Sub
```

Herhangi bir Grid'in LayoutUpdated metodunu yakaladığınızda aslında Grid'in görselliğindeki tüm boyut değişikliklerini de yakalamiş oluyorsunuz. GridSplitter Grid'in kolonlarının boyutunu değiştirdikçe LayoutUpdated metodu çalıştırılacaktır. Bizim yapacağımız da basit bir şekilde **IcGrid** adını verdigimiz Grid'in sıfırı kolonunun genişliğini alarak kaydetmek. Kaydetme işlemini bir [web servisi](#) ile sunucu tarafına yapabileceğiniz gibi doğrudan [Isolated Storage](#) kullanarak istemci tarafında da saklayabilirsiniz. Ben örnek içerisinde söz konusu değeri **Metin** adındaki bir **TextBlock** içeresine yazdırıldım.

Kaydettiğiniz genişlik ve yükseltik değerlerini Silverlight uygulaması tekrar açıldığında görsel arayüze uygulamak ise çok daha kolay.

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    IcGrid.ColumnDefinitions(0).Width = New System.Windows.GridLength(100)
End Sub
```

İstediğimiz herhangi bir Grid'in kolonunu yakalayarak Width özelliğini değiştirebiliyoruz.

Hepinize kolay gelsin ;)

Daron YÖNDEM

Silverlight 2.0 HyperlinkButton Kullanımı

Silverlight 1.0 içerisinde harici sayfalara linkler verecek düğmeler yaratmak için el ile kod yazmak gerekiyordu, eğer bir de gerçekten bir HTML linki gibi gözüken bir HyperLink yaratmak isterseniz epey bir uğraşmanız gerekecektir. Silverlight 2.0 Beta 1 ile bu soruna çok basit bir çözüm geliyor; **HyperlinkButton**.

```
<UserControl x:Class="SilverlightApplication19.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <HyperlinkButton Margin="175,101,92,0" Content="Tıkla Git"
            VerticalAlignment="Top" Height="Auto" NavigateUri="http://www.live.com"
            TargetName="_blank"/>
    </Grid>
</UserControl>
```

Yukarıda da görebileceğiniz gibi **HyperlinkButton** kısmen bizim ASP.NET tarafından alışılmış yapılardan farklı değil. **NavigateUri** özelliğine verdığınız adrese yönlendirme yaparken **TargetName** ile isterseniz hedef bir pencere veya frame de belirtebiliyorsunuz. Son olarak **HyperlinkButton** içerisinde gözükmeyen içeriği de **Content** özelliğine aktarabilirsiniz.

Özellikle ASP.NET ile bir karşılaştırma yaparsak ASP.NET içerisinde **HyperLink** kontrollerinin içine farklı kontroller koyabildiğimizi de hatırlayabiliriz. Örneğin rahatlıkla bir **Image** kontrolünü HyperLink içerisinde koymak kullanabiliriz. Aynı durum Silverlight için de geçerli.

```
<UserControl x:Class="SilverlightApplication19.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <HyperlinkButton VerticalAlignment="Center"
            Height="Auto"
            NavigateUri="http://www.live.com"
            TargetName="_blank" HorizontalAlignment="Center">
            <HyperlinkButton.Content>
                <Image Height="54"
                    Source="Waterfall.jpg" />
            </HyperlinkButton.Content>
        </HyperlinkButton>
    </Grid>
</UserControl>
```

Herhangi bir **HyperlinkButton'ın Content** özelliği aslında sadece metin içeriği almak için düzenlenmiş değil. İsterseniz herhangi bir Silverlight kontrolünü de **Content** içerisine yerleştirebilirsiniz. Bunun için tek yapmanız gereken **Content** özelliğini Inline olarak değil de **HyperlinkButton** içerisinde ek taglar içinde düzenlemeniz. İsterseniz birden çok kontrolü de

Content içerişine yerleştirebilirsiniz, tek şart elinizdeki tüm kontrolleri **Canvas** gibi bir container element içerişine toplamış olmak.

HyperlinkButton için Template Kullanımı

Bir uygulamada birden çok **HyperlinkButton** kullanılması olası. Bu gibi durumlarda tüm bu linklerin görsel özelliklerinin ortak bir noktada tutuluyor olması gerekiyor. Hali hazırda bir **HyperlinkButton**'un görsel özelliklerinin değiştirilebilmesi için **Content** özelliğine farklı içerikler aktarmaktansa doğrudan **HyperlinkButton**'un **Template** özelliği düzenlenebilir.

```
<HyperlinkButton VerticalAlignment="Center"
    Height="Auto"
    NavigateUri="http://www.live.com"
    TargetName="_blank"
    HorizontalAlignment="Center"
    Content="TIKLA"
    Template="{StaticResource LinkTemplate}" />
```

Yukarıdaki **HyperlinkButton**'un **Template** özelliği uygulama içerisindeki kaynaklardan birine bağlanmış. Birazdan **LinkTemplate** adındaki **HyperlinkButton** şablonumuzu hazırlayacağız. Böylece aynı görsel şablonu birden çok link kullanabilecek.

Örneğimizde ulaşmak istediğimiz noktayı belirleyelim. **HyperlinkButton**'un **Content** özelliğine verilen içeriğin doğrudan görsel şablonun ortasında gözükmemesini istiyoruz. Bunun için bir **ContentPresenter** kullanacağız. Bu **ContentPresenter**'in arkasında ise kenarları yuvarlatılmış bir dikdörtgen kullanalım. Böylece **HyperlinkButton**'umuz normal bir düğme gibi gözüksün.

```
<ControlTemplate x:Key="LinkTemplate" TargetType="HyperlinkButton">
<Grid>
    <Rectangle Stroke="#FF000000"
        RadiusY="16"
        RadiusX="16">
        <Rectangle.Fill>
            <RadialGradientBrush>
                <GradientStop Color="#FFFF0000"
                    Offset="1" />
                <GradientStop Color="#FFFFFF00"
                    Offset="0" />
            </RadialGradientBrush>
        </Rectangle.Fill>
    </Rectangle>
    <ContentPresenter Margin="10,10,10,10" Content="{TemplateBinding Content}"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" />
</Grid>
</ControlTemplate>
```

Yukarıdaki kod tüm isteklerimizi yerine getiriyor. Şimdi önemli noktalara tek tek degeinelim. İlk olarak **ControlTemplate** içerişine iki kontrol koyacağımız için bunları bir container

element içerisine almamız gerekiyor. Biz örneğimizde **Grid** kullandık. ContentPresenter kontrolümüzün **Content** özelliğini şablonun hedefi olacak HyperlinkButton'un **Content** özelliğine bağladıktan sonra geriye sadece ufak iki ayar kalıyor. Birincisi **ContentPresenter** ile **Rectangle** arasında mesafe kalması için ContentPresenter'a elle **Margin** vermek. Böylece **HyperlinkButton** içerisinde ne konulursa konulsun her zaman arkasında Rectangle'ın kenarlarından 10'a piksel uzak kalacak. İkinci ufak detay ise **ControlTemplate'in** **TargetType** özelliğinin kesinlikle ayarlanmış olması gerektiği.

Artık birden çok **HyperlinkButton** kullanarak aynı görsel özellikleri kullanabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde asenkron font dosyası indirerek kullanmak

Silverlight 1.0 ile beraber istemci tarafında herhangi bir font yüklemesi olmadan istediğimiz fontları TextBlock ve TextBox gibi kontrollerde kullanabilir hale gelmiştık. Silverlight 2.0 tarafında gelen yeniliklerle beraber artık her şeyi VB veya C# kodumuz ile halletmemiz gerekiyor. Bu yazımızda Silverlight 2.0 ile beraber özel font kullanımını inceleyeceğiz.

Sunucudan asenkron font indirerek kullanmak

Sunucudaki istediğimiz bir fontu asenkron olarak istemci tarafına indirip istediğimiz kontrollere bağlayabiliriz. Böylece gerekli fontları sadece gerekli olduğunda istemci tarafına taşımış oluruz. Bunun için Silverlight 2.0 tarafında bir **WebClient** nesnesi kullanarak ilk olarak font dosyasını istemci tarafına indirmemiz gerekiyor.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Yukleme As New WebClient
    Yukleme.OpenReadCompleted += AddressOf Yukleme_OpenReadCompleted
    Yukleme.OpenReadAsync(New Uri("/SilverlightApplication33Web/deeload.ttf",
    UriKind.Relative))
End Sub
```

[C#]

```
public Page()
{
    InitializeComponent();

    WebClient Yukleme = new WebClient();
    Yukleme.OpenReadCompleted += new
    OpenReadCompletedEventHandler(Yukleme_OpenReadCompleted);
    Yukleme.OpenReadAsync(new Uri("/SilverlightApplication34Web/deeload.ttf",
    UriKind.Relative));
}
```

Kodumuzda Silverlight uygulaması yükleniği gibi hemen bir **WebClient** nesnesi yaratarak ardından **OpenReadCompleted** event'ı için dinamik bir event handler bağlıyoruz. Söz konusu event handler WebClient nesnesinin sunucudan aldığı dosyanın download işlemi bittiğinde çalıştırılacak ve böylece biz de veri geldiğinde istediğimiz kontrole bağlayabileceğiz. Event-handler bağlantılarını da tamamladıktan sonra **OpenReadAsync** ile sunucudan verimizi almaya başlıyoruz.

[VB]

```
Private Sub Yukleme_OpenReadCompleted(ByVal sender As Object, ByVal e As System.Net.OpenReadCompletedEventArgs)
    Label1.FontSource = New FontSource(e.Result)
    Label1.FontFamily = New FontFamily("DEVELOAD")
```

End Sub

[C#]

```
void Yukleme_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    Label1.FontSource = new FontSource(e.Result);
    Label1.FontFamily = new FontFamily("DEVELOAD");
}
```

Veri sunucudan geldiğinde, download işlemi bittiğinde çalışan event-handler içerisinde hemen elimizdeki TextBlock olan **Label1** adındaki kontrolün **FontSource** özelliğini değiştiriyoruz. Tanımladığımız yeni **FontSource** veri kaynağını **e.Result** ile alıyor, yani WebClient'in indirdiği **Stream**'i doğrudan FontSource'a çevirerek TextBlock'a aktarıyoruz. Son olarak tabi ki TextBlock'un **FontFamily** özelliğini de uygun şekilde değiştirmemiz gereklidir.

Isterseniz **WebClient** nesnesinin **DownloadProgressChanged** durumuna da bir event-handlar bağlayarak Fontların yüklenmesi esnasında yüklemenin ne kadarının tamamlandığını da takip edebilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde AutoCompleteBox kullanımı.

AutoComplete işlevselligi AJAX günlerinden alışık olduğumuz bir sistem. Herhangi bir TextBox'a kullanıcı yazı yazarken aynı anda uygun alternatifleri göstermek ve aslında arka planda bir arama sistemi kurmak gibi işlemleri uzun zamandır farklı arayüz araçları kullanıksa da bir şekilde programcılar olarak hazırlayabiliyoruz. Silverlight tarafında ise Silverlight'in görsel gücünden de faydalananarak çok ilginç çözümler üretmek mümkün. Silverlight dünyasında AutoComplete altyapılarını incelerken Silverlight Toolkit içerisindeki **AutoCompleteBox** kontrolünü kullanacağız.

Not: Silverlight Toolkit'i kullanabilmeniz için [CodePlex](#) üzerindeki adresten kütüphaneyi indirerek içerisindeki Microsoft.Windows.Controls.dll dosyasını projenize referans olarak eklemelisiniz.

En hızlı şekilde AutoCompleteBox kullanımı..

AutoCompleteBox'in kullanımı aslında çok basit. Hemen bir **String Array** veya **List** yaratarak AutoCompleteBox'a bind etmeniz yeterli olacaktır. Tüm filtreleme ve AutoComplete işlemleri otomatik olarak yapılacaktır.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Liste As New List(Of String)
    Liste.Add("ASP.NET")
    Liste.Add("AJAX")
    Liste.Add("Silverlight")
    Liste.Add("WPF")
    AutoComplete1.ItemsSource = Liste
End Sub
```

[C#]

```
void Page_Loaded(object sender, RoutedEventArgs e)
{
    List<String> Liste = new List<string>();
    Liste.Add("ASP.NET");
    Liste.Add("AJAX");
    Liste.Add("Silverlight");
    Liste.Add("WPF");
    AutoComplete1.ItemsSource = Liste;
}
```

Yukarıdaki kod ile hazırladığımız basit bir uygulamanın aşağıda da XAML kodunu inceleyebilirsiniz.

[XAML]

```
<UserControl x:Class="SilverlightApplication3.Page"
```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400" Height="300" xmlns:controls="clr-
namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
<Grid x:Name="LayoutRoot" Background="White">
<controls:AutoCompleteBox Height="24" Margin="24,24,144,0"
VerticalAlignment="Top" x:Name="AutoComplete1"/>
</Grid>
</UserControl>

```



Basit bir AutoComplete örneği.

AutoCompleteBox'ın özellikleri.

IsTextCompletionEnabled - Bu özellik açık olduğunda kullanıcı yazı yazdığında sadece alternatifler gösterilmez, ek olarak en yakın alternatif sanki yazılmış gibi TextBox içerisinde de gösterilir. Varsayılan değeri True şeklindedir.

SelectedItem - Eğer kullanıcı AutoCompleteBox'ın açılan ListBox kısmından bir öğe seçerse SelectedIndexChanged çalışır ve SelectedItem geriye bir Item döndürür. Kullanıcı ListBox içerisinde yer alan bir Item'in metnini elle yazmışsa SelectedItem geriye değer döndürmeyecektir.

SearchMode - AutoComplete işlemi yapılrken kaynak veride ne şekilde arama yapılacağına karar veren SearchMode özelliği varsayılan değeri olan **StartsWith** ile gelir. İsterseniz **Contains** seçeneğini seçerek doğrudan kaynak verinin içindeki tüm metinlerin içinde arama yapılmasını da sağlayabilirsiniz. Eğer kendi arama sisteminizi entegre edecekseniz **None** seçeneğini seçmeniz gerekecektir.

MinimumPopulateDelay - Kullanıcı yazı yazarken ne kadar süre sonra alternatiflerin gösterileceğini belirler. Eğer veri kaynağı istemci tarafında bir Silverlight değişkeni ise varsayılan değer olan 0 ile herhangi bir sorun yaşamazsınız. Fakat alternatifleri sunucudan her seferinde çekiyorsanız buradaki bekleme süresini uzatmakta büyük fayda olacaktır.

MinimumPrefixLength - Alternatifler gösterilmeden önce kaç karakterlik verinin girilmiş olması gerekiğine dair ayar bu özellik üzerinden yapılabılır.

Kendi filtreleme mekanizmamızı yazalım.

Bir AutoCompleteBox'ı aslında iki şekilde veri bağlamış olabiliriz. Bunlardan birincisi yazımızın başındaki gibi basit bir metin dizisini AutoCompleteBox'a aktarmak ikincisi ise kendi tanımladığımız nesnelerin bir dizisini bağlamak. Gelin her iki durumda da filtreleme işlemlerini nasıl özelleştirebileceğimize bakalım.

Bir önceki örneğimizin üzerinden yola devam edersek zaten hali hazırda bir String listesini alıp AutoCompleteBox'ımıza bağlamıştık. AutoCompleteBox'ın **TextFilter** özelliğini değiştirerek filtreleme işleminin tam olarak nasıl yapılacağına karar verebiliriz. Bizim yapacağımız örnekte kullanıcının yazdığı metin ile başlayan değil de biten öğeleri göstermeye çalışacağız.

[VB]

```
Function Filtreleme(ByVal Search As String, ByVal item As String)
    If item.ToString().EndsWith(Search) Then
        Return True
    Else
        Return False
    End If
End Function
```

[C#]

```
public bool Filtreleme(string Search, string item)
{
    if (item.ToString().EndsWith(Search)) {
        return true;
    }
    else {
        return false;
    }
}
```

Yukarıdaki fonksiyonumuzu filtreleme işlemlerini yapmak için kullanacağız. Gelen iki parametreden ilki kullanıcının TextBox içerisinde yazdığı metin, ikincisi ise o an fonksiyonumuza aktarılan ana kaynak veriden gelen bir Item. Burada kafalar biraz karışabilir o nedenle biraz daha detaya inelim. AutoCompleteBox filtreleme işlemini yaparken elindeki verinin içindeki her bir öğeyi tek tek bizim filtreleme fonksiyonumuza verecek ve söz konusu öğenin gösterilip gösterilmeyeceğine dair bir cevap bekleyecek. Yani eğer veri kaynağında 10 adet öğe varsa bu fonksiyon 10 defa çağrılaracak.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Liste As New List(Of String)
    Liste.Add("ASP.NET")
    Liste.Add("AJAX")
    Liste.Add("Silverlight")
    Liste.Add("WPF")

    AutoComplete1.TextFilter = New
Microsoft.Windows.Controls.AutoCompleteSearchPredicate(Of String)(AddressOf
Filtreleme)
    AutoComplete1.ItemsSource = Liste
```

End Sub

[C#]

```
void Page_Loaded(object sender, RoutedEventArgs e)
{
    List<String> Liste = new List<string>();
    Liste.Add("ASP.NET");
    Liste.Add("AJAX");
    Liste.Add("Silverlight");
    Liste.Add("WPF");

    AutoComplete1.TextFilter = new
Microsoft.Windows.Controls.AutoCompleteSearchPredicate<string>(Filtreleme);
    AutoComplete1.ItemsSource = Liste;
}
```

Hazırladığımız filtreleme fonksiyonunu tabi ki AutoCompleteBox'in **TextFilter'ına** da aktarmamız gereklidir. Yukarıdaki kodlardaki kalın satırlarda söz konusu aktarma işleminin nasıl yapıldığını inceleyebilirsiniz. **TextFilter** bizden bir **AutoCompleteSearchPredicate** istiyor, biz de kendisine istediğimizi veriyoruz :)

Peki ya AutoCompleteBox'a kendi yarattığımız nesne türlerinden bir liste bağlamış olsaydık bu filtreleme işlemini nasıl yapacaktık. Böyle bir durumda **TextFilter** yerine **ItemFilter'ı** kullanmak zorunda kalacaktık. Gelin ilk olarak örneğimizde ilerleyebilmek için **Kitap** adında kendi sınıfımızı tanımlayalım.

[VB]

Public Class Kitap

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

```
Private PFiyat As Double
Public Property Fiyat() As Double
    Get
        Return PFiyat
    End Get
    Set(ByVal value As Double)
        PFiyat = value
    End Set
End Property
```

End Class

[C#]

```
public class Kitap
{
    public string Adi { get; set; }
    public double Fiyat { get; set; }
}
```

Şimdi bu sınıflar üzerinden yola çıkararak Silverlight tarafında birkaç kitap yaratıp AutoCompleteBox'lارımıza DataBind edeceğiz.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Liste As New List(Of Kitap)
    Liste.Add(New Kitap With {.Adi = "ASP.NET AJAX", .Fiyat = 25})
    Liste.Add(New Kitap With {.Adi = "ADO.NET", .Fiyat = 35})
    Liste.Add(New Kitap With {.Adi = "WPF", .Fiyat = 15})
    Liste.Add(New Kitap With {.Adi = "Silverlight", .Fiyat = 25})

    AutoComplete1.ItemFilter = New
Microsoft.Windows.Controls.AutoCompleteSearchPredicate(Of Object)(AddressOf
Filtreleme)

    AutoComplete1.ItemsSource = Liste
End Sub
```

[C#]

```
void Page_Loaded(object sender, RoutedEventArgs e)
{
    List<Kitap> Liste = new List<Kitap>();
    Liste.Add(new Kitap {Adi = "ASP.NET AJAX", Fiyat = 25});
    Liste.Add(new Kitap {Adi = "ADO.NET", Fiyat = 35});
    Liste.Add(new Kitap {Adi = "WPF", Fiyat = 15});
    Liste.Add(new Kitap {Adi = "Silverlight", Fiyat = 25});

    AutoComplete1.ItemFilter = new
Microsoft.Windows.Controls.AutoCompleteSearchPredicate<Object>(Filtreleme);

    AutoComplete1.ItemsSource = Liste;
}
```

Kodumuz içerisinde çok büyük değişiklikler yok. Bu sefer bir **String** listesi yerine **Kitap** listesi yaratıyor ve AutoCompleteBox'in **ItemsSource'una** eşitliyoruz. Filtreleme işlemi için

yne **Filtreleme** adında bir metod kullanacağımız için **ItemFilter** özelliğine de söz konusu metodу bağlıyoruz. Aradaki en önemli fark elimizdeki **Predicate'in** artık **Object** türünü taşıyor olması.

[VB]

```
Function Filtreleme(ByVal Search As String, ByVal item As Object)
    If CType(item, Kitap).Fiyat < CInt(Search) Then
        Return True
    Else
        Return False
    End If
End Function
```

[C#]

```
public bool Filtreleme(string Search, object item)
{
    if (((Kitap)item).Fiyat < int.Parse(Search)) {
        return true;
    }
    else {
        return false;
    }
}
```

Filtreleme metodumuzun ikinci parametresi artık **Object** tipinde. Biz aslında buradaki değişkenin bir **Kitap** nesnesi olduğunu biliyoruz çünkü **AutoCompleteBox** tek tek kendisine verilen öğeleri bufiltreleme fonksiyonuna ileticektir. Bu nedenle elimizde gelen objeyi **Kitap** tipine cast edip bu sefer de kitapların fiyatları üzerindenfiltreleme yapıyoruz. Kullanıcıların **TextBox** içerisinde sayısal bir fiyat gireceğini ve bu fiyatın altındaki kitapların listeleneceğini varsayıyalım.

Bu şekilde uygulamamızı çalıştırırsak ufak bir karışıklık olacaktır. AutoCompleteBox'a bağladığımız veri Kitap'lardan oluşuyor. Peki **AutoCompleteBox** bu Kitap nesnelerinin hangi özelliğini ekrana nasıl getirecek? Örneğin kitabı adını mı yoksa fiyatını mı gösterecek **AutoComplete** esnasında? İşte bu noktada da bizim XAML tarafına geçip birkaç işlem yapmamız gereklidir.

[XAML]

```
<UserControl x:Class="SilverlightApplication3.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
    namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <UserControl.Resources>
        <DataTemplate x:Key="DataTemplate1">
            <Grid>
```

```

<TextBlock HorizontalAlignment="Left" VerticalAlignment="Top" Text="{Binding Path=Fiyat}" TextWrapping="Wrap"/>
</Grid>
</DataTemplate>
</UserControl.Resources>
<Grid x:Name="LayoutRoot" Background="White">
    <controls:AutoCompleteBox Height="24" Margin="24,24,144,0"
VerticalAlignment="Top" x:Name="AutoCompletel" ItemTemplate="{StaticResource DataTemplate1}" />
</Grid>
</UserControl>

```

Artık bizim AutoCompleteBox'ımızın ItemTemplate'ini değiştirmemiz ve özelleştirmemiz gerekiyor. Böylece tam olarak gelen veriden hangi öğelerin nereelerde nasıl gösterileceğine karar verebiliriz. Yukarıdaki kod içerisinde AutoCompleteBox'in **ItemTemplate** özelliğinin değiştirildiğini görebilirsiniz. DataTemplate1 adında yarattığımız DataTemplate'i **UserControl.Resources** içeresine koyup kullanabiliyoruz. **DataTemplate** içerisinde ise bir Grid ve TextBlock bulunuyor. Söz konusu **TextBlock** doğrudan **Fiyat** adında bir Field'e DataBind edilmiş durumda. Hatırlarsanız bizim **Kitap** sınıfımızın **Fiyat** adında bir **Property**'si vardı. Böylece AutoCompleteBox'a kendisine verilen verilerden her birinin **Fiyat** özelliğinin **DataTemplate** içerisindeki bu TextBlock'un **Text'ine** bağlanması sağladık. Tüm bu işlemleri Visual Studio içerisinde XAML yazarak yapabileceğiniz gibi isterseniz Expression Blend'in arayüzünden de tabi ki daha rahatlıkla yapabilirsiniz. Tek yapmanız gereken AutoCompleteBox'a sağ tuş tıklayıp gelen menüden "Edit Other Templates / Edit ItemTemplate" demek. Böylece tasarım modunda Blend içerisinde de DataTemplate'in görselliğini değiştirebilirsiniz.

Artık filtreleme işlemleri bitirdik, filtreleme esnasından Kitapların sadece fiyatlarının gözükmesini de sağladık. Fakat neden sadece fiyatları gözüksün ki? Kullanıcı TextBox içerisinde 25 yazdığında 25YTL'den ucuz kitapların hem adı hem fiyatı yazsa? Ve bunların arasından bir kitap seçilince fiyatı otomatik olarak TextBlock içerisinde yerleşme daha güzel olmaz mı? AutoCompleteBox'in açılan kısmının görsellliğini ne de olsa yukarıda değiştirmiştik, gelin şimdi ikinci bir TextBlock ekleyerek onu da **Kitap** nesnelerinin **Adı** özelliğine bağlayalım.

[XAML]

```

<DataTemplate x:Key="DataTemplate1">
    <StackPanel HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
Background="{x:Null}" Orientation="Horizontal">
        <TextBlock Text="{Binding Path=Adi}" Foreground="#FFFF0000" Height="Auto"
Width="Auto"/>
        <TextBlock Text="{Binding Path=Fiyat}" Foreground="#FF838383" Height="Auto"
Width="Auto"/>
        <TextBlock Text="YTL" TextWrapping="Wrap" Foreground="#FF838383"
Height="20" Width="20"/>
    </StackPanel>
</DataTemplate>

```

Gördüğünüz gibi DataTemplate'i değiştirerek aslında işimizi çözmüş olduk. Geriye tek bir sorun kalıyor. Kullanıcı herhangi bir Item'ı ListBox içerisinde seçtiğinde TextBox'ın içine ne yazılacak? **Kitap** nesnesinin **Adı** mı? yoksa **Fiyatı** mı? Tabi ki bizim örneğimizde **Kitap** nesnesinin fiyatı yazılacak aksi halde filtreleme mekanizmamızla çakışacaktır.

[VB]

Public Class Kitap

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

```
Private PFiyat As Double
Public Property Fiyat() As Double
    Get
        Return PFiyat
    End Get
    Set(ByVal value As Double)
        PFiyat = value
    End Set
End Property
```

```
Public Overrides Function ToString() As String
    Return Me.Fiyat
End Function
```

End Class

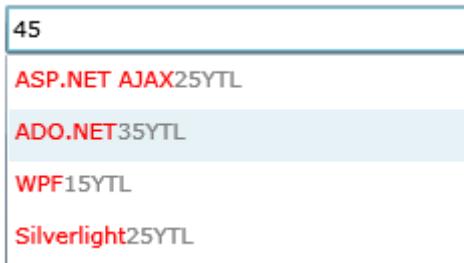
[C#]

```
public class Kitap
{
    public string Adi { get; set; }
    public double Fiyat { get; set; }

    public override string ToString()
    {
        return this.Fiyat.ToString();
    }
}
```

Bu da ne şimdi? dediğinizi duyar gibiyim. Maalesef AutoCompleteBox'ın **ListBox'tan** seçili öğelerin hangi özelliklerinin alınacağına dair bir ayarı yok. Aslında arka planda

AutoCompleteBox kendi içerisinde bir öğe seçildiğinde o öğeyi **ToString()** metodu ile alıp **TextBox** içerisinde yerleştiriyor. Bu durumda bizim de **ToString** metodunu **Override** etmemiz her şeyi çözecektir. Artık herhangi bir **Kitap** nesnesi üzerinden **ToString** çalıştırılırsa kitabın fiyat bilgisi verilecek.



Özelleştirilmiş bir AutoCompleteBox örneği.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Carousel kullanımı.

Carousel kontrolleri son dönemin moda diyebiliriz. Çoğu yazılımın arayüzünde Carousel kontrolleri görmeye başladık. Özellikle web sitelerinde de neredeyse RIA denildiği anda bir yere bir Carousel konulması gibi bir moda da mevcut. Bu çerçevede Silverlight 2 uygulamalarınızda Carousel yapılarından faydalananız isterseniz herşeyi sıfırdan yazmanıza gerek yok. Bu yazımmda sizlere açık kaynak kodu ile dağıtılan hazır bir Carousel kontrolünü tanıtacağım.

Coolmenu Carousel kontrolü

İlk olarak gelin kullanacağımız kontrolü kendi web sitesinden bir bilgisayarımıza indirelim. Aşağıdaki adresten indirebileceğin kontrolün tüm kaynak kodları ile alıp inceleme şansınız var. Biz şimdilik RC0 için hazırlanmış olan paketi alarak içerisinde **Coolmenu.DLL** dosyasını kullanacağız. Yani kaynak kodları ile uğraşmayacak doğrudan kontrolün Compile edilmiş halini projelerimize entegre edeceğiz.

<http://pagebrooks.com/archive/2008/08/21/coolmenu-a-silverlight-menu-control.aspx>

Projemize Carousel ekleyelim!

Projemizde Coolmenu Carousel kontrolünü kullanabilmek için ilk olarak download paketinden Coolmenu.Dll dosyasına Silverlight projemize referans olarak eklemeliyiz. Sonrasında XAML tarafında söz konusu kontrolü sayfaya koyabilmemiz için gerekli namespace tanımlarını yapmamız şart.

[XAML]

```
<UserControl x:Class="SilverlightApplication8.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300"
    xmlns:SilverlightContrib_Controls="clr-
    namespace:SilverlightContrib.Controls;assembly=CoolMenu">
    <Grid x:Name="LayoutRoot" Background="White">
        <SilverlightContrib_Controls:CoolMenu x:Name="Carousel"/>
    </Grid>
</UserControl>
```

Yukarıdaki XAML kodunu incelediğiniz özellikle dikkat etmemiz gereken aslında iki nokta var. Bunlardan ilki xmlns tanımımız. **SilverlightContrib_Controls** adında tanımladığımız yeni XML namespace'ımız doğrudan **Coolmenu** assembly'sini hedefliyor. Böylece söz konusu assembly içerisindeki tüm kontrollü XAML içerisinde kullanabileceğiz. Bir sonraki adımda da tanımladığımız NameSpace'i kullanarak **CoolMenu** kontrolünden bir adet ekranaya yerleştirerek adını da **Carousel** olarak tanımlıyoruz. Bu aşamadan sonrası için kod tarafına geçmemiz ve bu Carousel içerisinde gösterilecek öğeleri tanımlamamız gereklidir.

[VB]

Dim Foto As New Image

```

Foto.Source = New Imaging.BitmapImage(New
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute))
Carousel.Items.Add(New SilverlightContrib.Controls.CoolMenuItem() With {.Content
= Foto})
Foto = New Image
Foto.Source = New Imaging.BitmapImage(New
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute))
Carousel.Items.Add(New SilverlightContrib.Controls.CoolMenuItem() With {.Content
= Foto})
Foto = New Image
Foto.Source = New Imaging.BitmapImage(New
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute))
Carousel.Items.Add(New SilverlightContrib.Controls.CoolMenuItem() With {.Content
= Foto})

```

[C#]

```

Image Foto = new Image();
Foto.Source = new Imaging.BitmapImage(new
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute));
Carousel.Items.Add(new SilverlightContrib.Controls.CoolMenuItem { Content = Foto
});
Foto = new Image();
Foto.Source = new Imaging.BitmapImage(new
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute));
Carousel.Items.Add(new SilverlightContrib.Controls.CoolMenuItem { Content = Foto
});
Foto = new Image();
Foto.Source = new Imaging.BitmapImage(new
Uri("http://daron.yondem.com/tr/images/vesikalik2.png", UriKind.Absolute));
Carousel.Items.Add(new SilverlightContrib.Controls.CoolMenuItem { Content = Foto
});

```

Örnek kodlarımız içerisinde sürekli yeni Image nesneleri yaratarak bunları tek tek birer öğe (**CoolMenuItem**) olarak Carousel kontrolümüze ekliyoruz. Her bir CoolMenuItem'in **Content** özelliğine herhangi bir Silverlight nesnesi atayabilirsiniz. Bu ister bizim örneğimizdeki gibi bir resim ister bir video, yani MediaElement olabilir.

Örnek Carousel Kontrolü (Tiklamayı unutmayın :))

Hepinize kolay gelsin...

Daron YÖNDEM

Silverlight 2.0 içerisinde ControlTemplating ve Style yapıları.

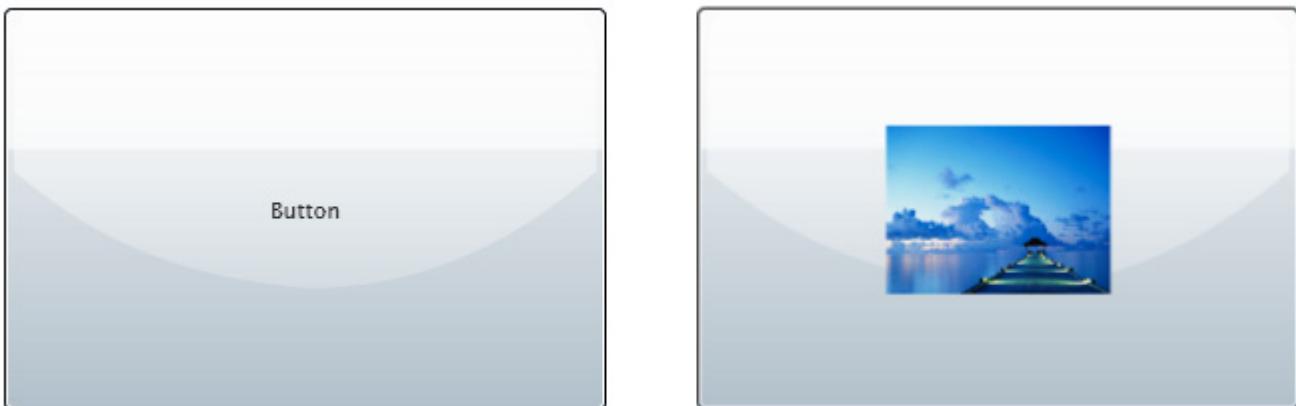
Silverlight 2.0 Beta 1 ile beraber gelen görsel özellikler içerisinde özellikle tasarımcıların en çok begenecekleri ve bildiğimiz teknolojiler arasında CSS'e benzetebileceğimiz "**Resource**" yapısı çok önemli bir yere sahip. WPF'de hali hazırda var olan ve Silverlight tarafına da (ciddi farklılıklar ile) taşınan bu özellikler sayesinde Silverlight uygulamaları içerisinde kontrollerin programatik işlevselliklerinden bağımsız olarak görsel özellikleri ayarlanabildiği gibi merkezi bir yönetim de sağlanabiliyor. Gelin hızlı bir örnek ile konumuza giriş yapalım.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
VerticalAlignment="Stretch" Content="Button"/>
    </Grid>
</UserControl>
```

İlk olarak yukarıdaki gibi yaratacağımız yeni bir Silverlight 2.0 uygulamasına basit bir **Button** yerleştirelim ve Button içerisinde yazının (**Content**) yerine farklı birşeyler koymayı deneyelim. Yukarıdaki kod içerisinde **Content** değeri doğrudan bit metne eşitlenmiş durumda. Oysa bu değerin içerisinde başka bir Silverlight kontrolü yerlestirebiliriz.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
VerticalAlignment="Stretch">
            <Button.Content>
                <Image Height="84" HorizontalAlignment="Center" VerticalAlignment="Center"
Width="139" Source="Dock.jpg"/>
            </Button.Content>
        </Button>
    </Grid>
</UserControl>
```

Yukarıdaki kod içerisinde **Button** nesnesinin **Content** özelliğini belirlemek için **Button** tagları arasında ayrıca bir **Button.Content** tagları açtık.



Button.Content özelliğini değiştiren Button nesnesi sağda.

Bu noktada Button.Content içerisinde istediğiniz Silverlight kontrolünü yerleştirebilirsiniz fakat dikkat etmeniz gereken bir nokta var; maalesef bu taglar arasında sadece bir Silverlight kontrolü yerleştirilebilir. Aslında cümle olarak yanlış bir cümle oldu, daha doğru tabiri ile Button.Content içerisinde XAML kodunun her zaman tek bir RootElement'inin olması gerekiyor. Yani eğer bu bölgeye iki kontrol yerleştirmek istiyorsanız kontrollerinizi bir Container Element içerisinde groplayarak yerleştirmek zorundasınız. Aşağıdaki örneğimizde **Container Element** olarak bir **StackPanel** kullanacağız.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
VerticalAlignment="Stretch">
            <Button.Content>
                <StackPanel Orientation="Horizontal">
                    <Image Height="84" HorizontalAlignment="Center" VerticalAlignment="Center"
Width="139" Source="Dock.jpg"/>
                    <Image Height="84" HorizontalAlignment="Center" VerticalAlignment="Center"
Width="139" Source="Dock.jpg"/>
                </StackPanel>
            </Button.Content>
        </Button>
    </Grid>
</UserControl>
```

Gördüğünüz gibi Root Element olarak Button.Content içerisinde bir StackPanel kullandıkten sonra artık StackPanel içerisinde istediğimiz kadar kontrol yerleştirebiliyoruz.



Button.Content içerisinde birden çok Silverlight kontrolü.

Bu yapı ile farklı kontrollerinin farklı görsel özelliklerini tanımlamak mümkün.

Bir kontrolün tüm görsel yapısını nasıl değiştiririz?

Bu noktaya kadar elimizdeki bir düğmenin içerisinde farklı Silverlight kontrolleri yerleştirdik. Oysa söz konusu düğmenin tüm görsel yapısını değiştirmek de isteyebilirdiniz. Örneğin bir Resim nesneninin bir Button olarak tanımlanması mümkün müdür?

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
VerticalAlignment="Stretch">
            <Button.Template>
                <ControlTemplate>
                    <Image Height="84" HorizontalAlignment="Center" VerticalAlignment="Center"
Width="139" Source="Dock.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
    </Grid>
</UserControl>
```

Yukarıdaki kodumuz içerisinde bir Button nesnesinin Template özelliğini başka bir ControlTemplate atayarak değiştiyoruz. Tanımladığımız ControlTemplate içerisinde sadece bir adet Image nesnesi var. Böylece artık düğmemiz sadece bir Image nesnesinden oluşacak oysa programcımız için bu nesne hala bir Button.



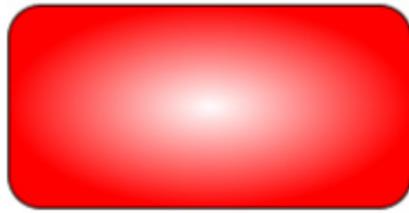
Bir Button nesnesine benzemiyor değil mi? Ama öyle.

Peki tüm bunların anlamı nedir?

Yukarıdaki iki yapıyı Button.Content ve Button.Template beraber kullanmanız halinde istediğimiz gibi düğmeler oluşturabilirsiniz. Gelin şimdi güzel bir dikdörtgen çizelim ve bunu yeni yarattığımız bir Button nesnesinin Template özelliğine aktaralım.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
    VerticalAlignment="Stretch">
            <Button.Template>
                <ControlTemplate>
                    <Rectangle Stroke="#FF000000" RadiusY="16" RadiusX="16"
Margin="{TemplateBinding Margin}">
                        <Rectangle.Fill>
                            <RadialGradientBrush>
                                <GradientStop Color="#FFFF0000" Offset="1"/>
                                <GradientStop Color="#FFFFFF00" Offset="0"/>
                            </RadialGradientBrush>
                        </Rectangle.Fill>
                    </Rectangle>
                </ControlTemplate>
            </Button.Template>
        </Button>
    </Grid>
</UserControl>
```

Yukarıdaki kod içerisinde en önemli nokta Button'umuzun **ControlTemplate**'ı içerisindeki **Rectangle** nesnesinin **Margin** özelliklerinin **TemplateBinding** ile içerisinde olduğu kontrolün **Margin** özelliğine bağlanmış olması. Böylece Button nesnesi büyündükçe otomatik olarak ControlTemplate içerisinde tanımlanmış Rectangle da büyüyecek. Her zaman olduğu gibi eğer ControlTemplate içerisinde birden çok Silverlight kontrolü kullanarak görsel özellikler tanımlamak isterseniz bir Container Element kullanmanız gerekektir. Bizim örneğimizde tek nesne bulunduğu için gerek olmadı.



Yeni Silverlight Button nesnemiz karşınızda!

Gördüğünüz gibi düğmemizin görsel özellikleri tamamlandı. Peki ya bu düğmenin içerisinde yazılacak olan yazı nerede? Şu anda kontrolün bu hali ile **Button.Content** özelliğine farklı değerler verdiginiz görsel anlamda herhangi bir değişiklik görmeyeceksiniz. Çünkü kontrolün **ControlTemplate** tanımı içerisinde herhangi bir şekilde **Content** içeriğinin konacağı bir yer ayarlanmış değil. Yani **Button.Content** içerisinde konacak nesnelerin veya yazının **ControlTemplate** içerisinde nereye konacağını tanımlamamız gerekiyor.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button HorizontalAlignment="Stretch" Margin="44,54,55,45"
VerticalAlignment="Stretch">
            <Button.Content>
                <TextBlock VerticalAlignment="Center" Text="TextBlock" TextWrapping="Wrap"/>
            </Button.Content>
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <Grid>
                        <Rectangle Stroke="#FF000000" RadiusY="16" RadiusX="16"
Margin="{TemplateBinding Margin}">
                            <Rectangle.Fill>
                                <RadialGradientBrush>
                                    <GradientStop Color="#FFFF0000" Offset="1"/>
                                    <GradientStop Color="#FFFFFF00" Offset="0"/>
                                </RadialGradientBrush>
                            </Rectangle.Fill>
                        </Rectangle>
                    <ContentPresenter Content="{TemplateBinding Content}"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                </Grid>
            </ControlTemplate>
        </Button.Template>
    </Button>
</Grid>
</UserControl>
```

Yukarıdaki kod içerisinde tanımladığımız Button'in ControlTemplate'i içerisinde bir **ContentPresenter** nesnesi bulunuyor. Bu ContentPresenter'in **Content** özelliği ise

TemplateBinding ile ana Button nesnesinin Content'ına bağlanmış durumda. Böylece Button'un **Button.Content**'ı içerisindeki tüm Silverlight kontrolleri otomatik olarak **ControlTemplate** içerisindeki **ContentPresenter** içerisine yerleştirilmiş olacaktır. Bu noktada dikkat etmemiz gereken önemli bir ayar var, ControlTemplate'in **TargetType'ının** özel olarak ayarlanmış olması gerekiyor, aksi halde söz konusu ControlTemplate bir Button içerisinde bulunmasına rağmen doğru olarak çalışmıyor. Umarım Beta sürümü sonrasında sürümlerde bu yapı düzelttilir, şu anki çalışma yapısı pek mantıklı değil.



Button'umuzun son hali.

Peki neden doğrudan herşeyi ControlTemplate içerisinde koymuyoruz?

Çok mantıklı bir soru. Çünkü bu noktaya kadar yaptığımız tüm görsel ayarların üzerinden bir merkezi yönetim sistemi kurmak istiyoruz. Bir önceki adımda Button nesnemizin Template özelliğini ayarlamıştık. Şimdi sıra geldi söz konusu Template'i birden çok kontrol tarafından kullanılabilir hale getirmeye.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
<UserControl.Resources>
    <Style x:Key="Dugme" TargetType="Button">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Grid>
                        <Rectangle Stroke="#FF000000" RadiusY="16" RadiusX="16"
Margin="{TemplateBinding Margin}">
                            <Rectangle.Fill>
                                <RadialGradientBrush>
                                    <GradientStop Color="#FFFF0000" Offset="1"/>
                                    <GradientStop Color="#FFFFFF00" Offset="0"/>
                                </RadialGradientBrush>
                            </Rectangle.Fill>
                        </Rectangle>
                        <ContentPresenter Content="{TemplateBinding Content}">
                            <ContentPresenter.HorizontalContentAlignment>Center</ContentPresenter.HorizontalContentAlignment>
                            <ContentPresenter.VerticalContentAlignment>Center</ContentPresenter.VerticalContentAlignment>
                        </ContentPresenter>
                    </Grid>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</UserControl.Resources>
<Grid>
    <Button Style="Dugme" Content="Merhaba Silverlight!"></Button>
</Grid>
```

```

</Style>
</UserControl.Resources>
<Grid x:Name="LayoutRoot" Background="White">
    <Button Style="{StaticResource Dugme}" HorizontalAlignment="Stretch"
Margin="44,54,55,45" VerticalAlignment="Stretch">
        <Button.Content>
            <TextBlock VerticalAlignment="Center" Text="TextBlock" TextWrapping="Wrap"/>
        </Button.Content>
    </Button>
</Grid>
</UserControl>

```

Template tanımımızı Button'un içerisinde keserek doğrudan **UserControl.Resources** altına alıyoruz. Böylece artık tüm Silverlight XAML sayfası içerisinde kullanabileceğiz. Tabi bunu yaparken bir **Style** tanımlıyoruz ve **TargetType** özelliğini de **Button** olarak düzenliyoruz. Style içerisinde bir **Setter** yerleştirerek hedef kontrolün **Template** özelliğini değiştirmek istediğimizi de belirttikten sonra **Setter.Value** içerisinde daha önce hazırladığımız Template yapısını yerleştiriyoruz.

Son olarak adını **Dugme** koyduğumuz bu Style'ı kullanmak istediğimiz tüm **Button** kontrollerinin **Style** özelliğini **StaticResource** olarak stilimize bağlayarak kullanabiliyoruz. Böylece tüm düğmelerimiz aynı ControlTemplate'i kullanacaklar ve hepsi de kendi içerisindeki **Content**'ı hedef Style içerisindeki **ContentPresenter**'a yerleştirerek gösterecekler. Herhangi bir görsel değişiklik gerektiğinde ise merkezi olarak Style'ımızı değiştirerek ilerleyebileceğiz.

İsterseniz bu Style'ımızı Silverlight uygulamasının **App.xaml** dosyası içerisinde koyarak Silverlight uygulamanızdaki tüm XAML dosyaları içerisinde kullanılabilir hale de getirebilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde createFromXaml alternatifi ve alt seviyeli dinamik nesne üretimi

Silverlight 1.0 içerisinde JavaScript tarafından sahip olduğumuz **createFromXaml** metodu çoğu durumda işimizi kolaylaştırıyordu. Dinamik olarak yaratmak istediğimiz nesnelerin XAML kodunu bir kereliğine Blend ile yaratarak sonrasında söz konusu XAML kodunu programatik olarak String tipinde değiştirip doğrudan **createFromXaml** metoduna verdığımızde istediğimiz tüm Silverlight nesnelerini gerekli özelliklerile beraber almış oluyorduk. **Silverlight 2.0** tarafında ise tüm nesneler artık birer .NET nesnesi olduğuna göre aslında bu nesnelerin yaratılarak sahneye yerleştirilmesi çok daha kolay gibi gözüküyor. Gelin ufak bir örnek yaparak durumu inceleyelim.

```
<Ellipse Height="57" HorizontalAlignment="Left" Margin="20,22,0,0"
VerticalAlignment="Top" Width="64" Stroke="#FF000000">
<Ellipse.Fill>
<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
<GradientStop Color="#FF000000"/>
<GradientStop Color="#FFFFFF" Offset="1"/>
</LinearGradientBrush>
</Ellipse.Fill>
</Ellipse>
```

Tasarımcımız Expression Blend ile yukarıdaki gibi özünde çok basit olan bir **Ellipse** tasarladığını düşünelim ve biz de bu Ellipse'in yükseklik ve genişlik özelliklerini elimizdeki herhangi bir veriye bağlı olmak şartı ile değiştirek birden çok Ellipse yaratmaya çalışacağız. Bu noktada Silverlight 1.0 içerisinde **createFromXaml** kullanırken SL 2.0 içerisinde .NET nesneleri şeklinde gerekli işlemleri yaparak Ellipse'ler yaratmaya karar verirsek aşağıdaki kodu yazmamız gerekiyor.

[VB]

```
Dim Daire As New Ellipse
Daire.Height = 57
Daire.HorizontalAlignment = HorizontalAlignment.Left
Daire.Margin = New Thickness(20, 22, 0, 0)
Daire.VerticalAlignment = VerticalAlignment.Top
Daire.Width = 64
Daire.Stroke = New SolidColorBrush(Color.FromArgb(100, 0, 0, 0))
Dim GradientFirca As New LinearGradientBrush
GradientFirca.StartPoint = New Point(0.5, 0)
GradientFirca.EndPoint = New Point(0.5, 1)
Dim GradStop As New GradientStop
GradStop.Color = Color.FromArgb(100, 0, 0, 0)
GradientFirca.GradientStops.Add(GradStop)
GradStop = New GradientStop
GradStop.Color = Color.FromArgb(100, 255, 255, 255)
GradStop.Offset = 1
GradientFirca.GradientStops.Add(GradStop)
Daire.Fill = GradientFirca
Me.LayoutRoot.Children.Add(Daire)
```

[C#]

```

Ellipse Daire = new Ellipse();
Daire.Height = 57;
Daire.HorizontalAlignment = HorizontalAlignment.Left;
Daire.Margin = new Thickness(20, 22, 0, 0);
Daire.VerticalAlignment = VerticalAlignment.Top;
Daire.Width = 64;
Daire.Stroke = new SolidColorBrush(Color.FromArgb(100, 0, 0, 0));
LinearGradientBrush GradientFirca = new LinearGradientBrush();
GradientFirca.StartPoint = new Point(0.5, 0);
GradientFirca.EndPoint = new Point(0.5, 1);
GradientStop GradStop = new GradientStop();
GradStop.Color = Color.FromArgb(100, 0, 0, 0);
GradientFirca.GradientStops.Add(GradStop);
GradStop = new GradientStop();
GradStop.Color = Color.FromArgb(100, 255, 255, 255);
GradStop.Offset = 1;
GradientFirca.GradientStops.Add(GradStop);
Daire.Fill = GradientFirca;
this.LayoutRoot.Children.Add(Daire);

```

Gördüğünüz gibi aslında her şeyi .NET kodumuz ile de yapabiliyoruz fakat tasarımcının Blend içerisinde yapmış olduğu tasarımı yukarıdaki koda çevirmek ciddi "işkence" kıvamında. Tabi ki tasarımcının sahneye koyduğu nesnelerin özelliklerine tek tek erişerek .NET kodumuz ile değiştirebiliriz fakat şu anda yapmak istediğim bu nesnelerden programatik olarak birden çok yaratıp özelliklerini de ayarlayarak sahneye yerleştirmek.

Çözümlerden ilki UserControl yapısı; eğer yaratacağınız nesnelerin yapıları sabit ise yani büyük değişiklikler yoksa söz konusu yapıyı bir UserControl olarak tasarlayarak belirli özelliklerini değiştirip sahneye yerleştirebilirsiniz. Fakat bazı durumlardan UserControl yapısı da gerekli esnekliği sağlayamayabiliyor. İşte böyle durumlarda yardımımıza **System.Windows.Markup.XamlReader.Load** metodu yetiyor. Bu metod Silverlight 1.0'da alışık olduğumuz **createFromXaml** metodu ile birebir aynı mantıkta çalışıyor. Parametre olarak aldığı String XAML kodundan bize Silverlight nesneleri yaratarak geri döndürüyor. Şimdi de aşağıdaki şekilde XAML kodumuzun dinamik olarak yaratıp Silverlight nesnemizi oluşturalım.

[VB]

```

Dim EllipseXAML = <Ellipse
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Height="57"
    HorizontalAlignment="Left" Margin="20,22,0,0" VerticalAlignment="Top" Width="64"
    Stroke="#FF000000">
    <Ellipse.Fill>
        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#FF000000"/>
            <GradientStop Color="#FFFFFF" Offset="1"/>
        </LinearGradientBrush>
    </Ellipse.Fill>

```

</Ellipse>

[C#]

```
string EllipseXAML = "<Ellipse  
xmlns=\"http://schemas.microsoft.com/winfx/2006/xaml/presentation\" Height=\"57\"\nVisibility=\"Collapsed\" HorizontalAlignment=\"Left\" Margin=\"20,22,0,0\"\nVerticalAlignment=\"Top\" Width=\"64\" Stroke=\"#FF000000\">" +  
    "<Ellipse.Fill>" +  
        "<LinearGradientBrushEndPoint=\"0.5,1\" StartPoint=\"0.5,0\">" +  
            "<GradientStop Color=\"#FF000000\"/>" +  
            "<GradientStop Color=\"#FFFFFF\" Offset=\"1\"/>" +  
        "</LinearGradientBrush>" +  
    "</Ellipse.Fill>" +  
</Ellipse>;
```

XAML kodumuzu VB veya C# kod tarafına alıp **System.Windows.Markup.XamlReader.Load** ile kullanacağımız zaman kopyaladığımız XAML kodu içerisinde kullanılan tüm XML namespace tanımlamalarını ana XAML dosyasından kopyalamış olmamız gerekiyor. Yukarıdaki kodlarda kalın yazılı kısımlar bu şekilde kopyalanmıştır.

LINQ ile beraber VB içerisinde "inline XML" yapısı geliyor, bu yapıyı Silverlight içerisinde de kullanabiliyoruz. Bunun için tek yapmamız gereken Silverlight projemize sağ tuş tıklayarak "Add Reference" diyip **System.XML.Linq** kütüphanesini projemize eklemek. Böylece artık tasarımcının Blend içerisinde düzenlenmiş olduğu XAML kodunu doğrudan Silverlight içerisinde VB dosyamıza kopyalayabiliriz. C# içerisinde ise XAML kodumuzu bir string değişkene aktarabilmek için bolca escape char kullanmamız gerekiyor.

Sıra geldi yarattığımız XAML kodu içerisinde istedigimiz özelliklerini değiştirmeye. VB içerisinde yarattığımız değişken doğrudan XAML aldığı ve bir XElement nesnesi oluşturduğu için XML içerisindeki özelliklere rahatlıkla ulaşabiliyoruz.

[VB]

EllipseXAML.Width = 300

Maalesef C# içerisinde böyle bir şansımız yok. Bir seçenek olarak C# içerisinde tüm XAML kodunu harici bir dosyada tutup söz konusu harici dosyayı bir XML dosyasımış gibi XDocument olarak yükleyerek kullanmak olabilir. Yukarıdaki şekliyle özelliklere doğrudan ulaşmanın yanı sıra istersek XAML kodunun içerisine ilk tanımlama esnasında da yerleştirebiliriz.

[VB]

```
Dim Yukseklik As Integer = 57  
Dim EllipseXAML = <Ellipse  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Height=<%=  
    Yukseklik %> HorizontalAlignment="Left" Margin="20,22,0,0" VerticalAlignment="Top"  
    Width="64" Stroke="#FF000000">
```

```
<Ellipse.Fill>
  <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
    <GradientStop Color="#FF000000"/>
    <GradientStop Color="#FFFFFF" Offset="1"/>
  </LinearGradientBrush>
</Ellipse.Fill>
</Ellipse>
```

[C#]

```
int Yukseklik = 57;
string EllipseXAML = "<Ellipse
  xmlns=\"http://schemas.microsoft.com/winfx/2006/xaml/presentation\" Height="" +
Yukseklik + "\" HorizontalAlignment=\"Left\" Margin=\"20,22,0,0\" +
VerticalAlignment=\"Top\" Width=\"64\" Stroke=\"#FF000000\">" +
  "<Ellipse.Fill>" +
    "<LinearGradientBrush EndPoint=\"0.5,1\" StartPoint=\"0.5,0\">" +
      "<GradientStop Color=\"#FF000000\"/>" +
      "<GradientStop Color=\"#FFFFFF\" Offset=\"1\"/>" +
    "</LinearGradientBrush>" +
  "</Ellipse.Fill>" +
"</Ellipse>";
```

Yukarıdaki kod içerisinde yarattığımız bir Integer değişkenin değerini kendi XAML kodumuzdaki Elipse'in yüksekliğine atıyoruz. VB içerisinde bu işlemi yine **Inline XML** tekniğini kullanarak yaparken C# içerisinde standart **string** işlemi olarak ilerlememiz gerekiyor.

Son olarak sıra geldi elimizdeki bu XAML kodundan nesnemizi yaratarak sahneye yerleştirmeye.

[VB]

```
Dim BirElips = System.Windows.Markup.XamlReader.Load(EllipseXAML.ToString)
Me.LayoutRoot.Children.Add(BirElips)
```

[C#]

```
var BirElips = System.Windows.Markup.XamlReader.Load(EllipseXAML);
this.LayoutRoot.Children.Add((UIElement)BirElips);
```

Örneğimizi tamamladık. Fakat eğer bir veriye bağlı olarak birden çok nesne yaratacak olsak nasıl olurdu? Hemen minik bir örnek ile onu da inceleyelim. Farklı yüksekliklerden birden çok Ellipse yaratıbmek için elimizdeki yüksekliklerin bir listesinin dizi olarak bulunduğu varsayıyalım.

[VB]

```
Dim Yukseklik() As Integer = {57, 67, 80}
```

```

Dim EllipseXAML = <Canvas
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
        <%= From GelenVeri In Yukseklik Select <Ellipse
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Height=<%= GelenVeri %> HorizontalAlignment="Left" Margin="20,22,0,0" VerticalAlignment="Top" Width="64" Stroke="#FF000000">
                <Ellipse.Fill>
                    <LinearGradientBrush EndPoint="0.5,1"
                        StartPoint="0.5,0">
                            <GradientStop Color="#FF000000"/>
                            <GradientStop Color="#FFFFFF" Offset="1"/>
                    </LinearGradientBrush>
                </Ellipse.Fill>
            </Ellipse> %>
        </Canvas>
Dim BirElips = System.Windows.Markup.XamlReader.Load(EllipseXAML.ToString)
    Me.LayoutRoot.Children.Add(BirElips)

```

VB programcılar için biraz yeni bir yapı olduğu için ilk önce yukarıdaki VB kodumuzu inceleyelim. Her zamanki gibi yine VB içerisinde inline XML özelliklerinden faydalaniyoruz. Birden çok Ellipse'i doğrudan **XamlReader.Load**'a verme şansımız yok. XamlReader aynı SL 1.0'daki **createFromXaml**'da olduğu gibi sadece tek bir nesne geri döndürebiliyor. O nedenle tüm Ellipse'lerimizi bir Canvas içerisinde yerleştiriyoruz. XamlReader bize içerisinde Ellipse'lerin bulunduğu **Canvas** nesnesini geri döndürecek. VB kodumuzda Canvas XML tagları arasında <%=%> işaretleri içerisinde bir LINQ sorgusu görüyorsunuz. Bu sorgu kodumuzun en üstündeki veri kaynağımız olan **Yukseklik** dizisini sorgulayarak kayıtları alıyor. Alınan her kayıt için yapılan **Select** işleminde ise geriye bir XML döndürülüyor. Doğal olarak bu XML bizim Ellipse'in XAML kodu olacak. Tüm bu işlemleri yaparken **Select** esnasında yarattığımız XAML'in yüksekliğine de sorgunun çalıştığı satırındaki değeri, yani dizinin içerisindeki değeri yükseklik olarak atıyoruz. Böylece sorgumuz geriye yükseklikleri veri kaynağından alınarak ayarlanmış bir Ellipse serisini XML olarak döndürüyor. Bu dönen değeri Canvas tagları arasına alarak doğrudan EllipseXAML değişkenimize aktarıyoruz. İşlem tamam.

[C#]

```

int[] Yukseklik = {57, 67, 87};

string EllipseXAML = "<Canvas
    xmlns=\"http://schemas.microsoft.com/winfx/2006/xaml/presentation\">" +
    string.Join("", (from GelenVeri in Yukseklik
                    select "<Ellipse
            xmlns=\"http://schemas.microsoft.com/winfx/2006/xaml/presentation\" Height=\"" +
            GelenVeri + "\" HorizontalAlignment=\"Left\" Margin=\"20,22,0,0\""
            VerticalAlignment=\"Top\" Width=\"64\" Stroke=\"#FF000000\">" +
                "<Ellipse.Fill>" +
                "<LinearGradientBrush EndPoint=\"0.5,1\""
            StartPoint=\"0.5,0\">" +
                "<GradientStop Color=\"#FF000000\"/>" +

```

```

    "<GradientStop Color=\"#FFFFFF\" Offset=\"1\"/>" +
    "</LinearGradientBrush>" +
    "</Ellipse.Fill>" +
    "</Ellipse>").ToArray() +
    "</Canvas>";
var BirElips = System.Windows.Markup.XamlReader.Load(EllipseXAML);
this.LayoutRoot.Children.Add((UIElement)BirElips);

```

C# kodumuza baktığımızda ise "inline XML" desteği olmadığı için yine String üzerinden gitmek durumunda kalıyoruz. XamlReader'a verilmek üzere tüm Ellipse'lerimizi ana bir Canvas içerisinde alırken Ellipse üretimi için LINQ'in sorgularının yardımını istiyoruz. C# içerisinde LINQ sorgumuz gerekli değerleri de XML içerisinde yerleştirerek bir string dizisi üretiyor. Son olarak elde ettigimiz XAML kodundan nesnelerimizi yaratıp sahneye yerleştiriyoruz.

Sonuç

Yukarıdaki teknikler ile farklı uygulamalarında tüm Silverlight nesnelerini en alt seviyeden müdahale ederek XAML üzerinden üretebilirisiniz. Fakat unutmamakta fayda var ki bu tarz yazılmış uygulamaların sonradan değiştirilmesi, yönetilmesi çok zor olacaktır. Örneğin tasarımcı yukarıdaki örneklerdeki Ellipse'in tasarımını değiştirmeye karar vermesi programcı sanırım intihara epeyce yaklaşacaktır. O nedenle XamlReader.Load tekniği olabildiğince en zor durumlarda, en sıkıştığınız anlarda kullanmakta fayda var. Normal şartlarda UserControl yapıları çoğu zaman gerekli çözümü sunacak ve daha rahat bir çalışma ortamı sağlayacaktır.

Hepinize kolay gelsin...

Daron YÖNDEM

Silverlight 2.0 içerisinde fare roller'ını kullanmak...

Silverlight 2.0 içerisinde .NET programlama dillerini kullanırken istemci tarafında bir tarayıcıda olduğumuzu sürekli hatırlamak gerek. Bu bilgiyi aklımızda tutarsak aslında Silverlight içerisinde yapılamayan bazı şeyleri tarayıcının özelliklerinden faydalananarak yapma şansımız olabiliyor. Bu konuya örneklerden biri Silverlight 2.0 içerisinde farenin roller'ını yakalayarak zoom-in veya zoom-out efekti yaratmak. Maalesef şimdilik Silverlight içerisinde farenin roller'ını yakalayabileceğimiz herhangi bir event-handler yok. Bu durumda biz de tarayıcının özelliklerinden faydalanaçğız.

İlk olarak hazırlayacağımız **Silverlight 2.0 Beta 1** uygulamasının XAML kodunu inceleyelim. **Page.xaml** içerisinde basit bir **Image** nesnesi yer alıyor. Farenin rollerini kullanarak söz konusu Image nesnesini zoom yapabileceğiniz, yani özünde Image nesnesinin boyutlarını büyüterek küçültleneceğiz.

```
<UserControl x:Class="ScrollWheel.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Image Margin="147,112,156,126" Source="Forest.jpg"
        RenderTransformOrigin="0.5,0.5" x:Name="Foto">
            <Image.RenderTransform>
                <TransformGroup>
                    <ScaleTransform x:Name="ZoomCarpani" ScaleX="1" ScaleY="1"/>
                    <SkewTransform/>
                    <RotateTransform/>
                    <TranslateTransform/>
                </TransformGroup>
            </Image.RenderTransform>
        </Image>
    </Grid>
</UserControl>
```

Yukarıdaki kod içerisinde özellikle dikkat edilmesi gereken bir nokta var. Normal şartlarda Expression Blend içerisinde bir Image nesnesi yerleştirirseniz **Image.RenderTransform** tagı ve içerisindekiler gelmeyecektir. Eğer söz konusu Image nesnesini sadece bir kere arayüz içerisinde boyutlandırırsanız hemen bu taglar eklenir. Sonrasında **RenderTransform** içerisindeki **ScaleTransform** kısmıyla ilgileneceğiz. **Image** nesnemizin büyülüüğünü değiştirecek olan tag bu. ScaleTransform'un **ScaleX** ve **ScaleY** adında X ve Y doğrultusunda boyutlandırma yapabilen çarpanları var. Bu çarpanlara ve **ScaleTransform** nesnesine programatik olarak ulaşabilmek için hemen ScaleTransform'a **ZoomCarpani** adını veriyoruz. Böylece artık rahatlıkla kod tarafından bu resmin boyutu ile oynayabiliriz.

Gelelim code-behind sayfamıza; ilk olarak internet tarayıcımızın farenin roller'ını algıladığı event-handlerları yakalayarak kendi .NET event handlerlarımızı bağlamalıyız.

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
```

```

System.Windows.Browser.HtmlPage.Window.AttachEvent("DOMMouseScroll",
AddressOf FareTekerlekDondu)
System.Windows.Browser.HtmlPage.Window.AttachEvent("onmousewheel", AddressOf
FareTekerlekDondu)
System.Windows.Browser.HtmlPage.Document.AttachEvent("onmousewheel", AddressOf
FareTekerlekDondu)
End Sub

```

Silverlight uygulamamız sayfaya ilk yüklendiğinde hemen mevcut HtmlPage (HTML sayfa) üzerinden tarayıcı penceresini (Window) yakalayarak **DOMMouseScroll** ve **onmousewheel** durumlarına kendi **FareTekerlekDondu** kodumuzu atıyoruz. Ayrıca sayfada yüklü dokümanın da (Document) **onmousewheel** event'ını aynı şekilde yakalamamız gerekiyor. Tüm bu farklı event-handllerlar aslında hep bizim tek event-handllerimiz olan **FareTekerlekDondu'ye** yönlendiriliyor. Bunun nedeni farklı tarayıcıların farklı event-handllerlar ile çalışıyor olması. Bizim örneğimiz rahatlıkla Opera, Safari, FireFox ve IE'de çalışacak. Sıra geldi **FareTekerlekDondu'yü** kodlamaya.

```

Private Sub FareTekerlekDondu(ByVal sender As Object, ByVal e As
System.Windows.Browser.HtmlEventArgs)
...
End Sub

```

Yukarıdaki şekli ile kodumuzu yazarken dikkat etmemiz gereken nokta e parametresinin tipi. **HtmlEventArgs** bize tarayıcının döndürdüğü değerleri aktaracak.

```

Dim DonMiktar As Integer = 0
Dim Gelen As System.Windows.Browser.ScriptObject = e.EventObject

```

Kodumuzda iki adet değişken tanımlıyoruz. Bunlardan ilki olan **DonMiktar** tarayıcından gelen farenin roller'ını dönüş miktarını alarak yönünü belirleyecek. Tarayıcılarından dönüş miktarı + veya - değer olarak gelebilir ve maalesef bu durum tarayıcı tipine göre farklı yönlerde dönüşleri tanımlıyor. Tanimladığımız bir diğer değişken ise **Gelen** adında; bu değişken doğrudan e parametresinden **EventObject'i** alarak bize dönüş miktarını ulaştıracak.

```

'IE ve OPERA
If Not Gelen.GetProperty("wheelDelta") Is Nothing Then
    'OPERA'da ters!
    If Not Gelen.GetProperty("opera") Is Nothing Then
        DonMiktar = -DonMiktar
    End If
    DonMiktar = Gelen.GetProperty("wheelDelta")
    'Mozilla ve Safari
ElseIf Not Gelen.GetProperty("detail") Is Nothing Then
    DonMiktar = -Gelen.GetProperty("detail")
End If

```

Yukarıdaki kod biraz karışık gözükebilir fakat aslında yapmaya çalıştığımız şey çok basit. Tarayıcıdan farenin roller'ının dönüş miktarını almaya çalışıyoruz. Maalesef farklı tarayıcılarda bu sistemin farklı çalışması gerekiyor. Opera ve IE bu değeri **wheelDelta** özelliği üzerinden verirken Mozilla ve Safari **detail** adında bir property kullanıyor. Ayrıca IE

dışında tüm tarayıcılarda değerler ters, veya belki de IE'de ters :) Ama duruma göre gelen değeri eksi ile çarpmak zorunda kalıyoruz.

```
If DonMiktar > 0 Then
    ZoomCarpani.ScaleX += 0.1
    ZoomCarpani.ScaleY += 0.1
Else
    ZoomCarpani.ScaleX -= 0.1
    ZoomCarpani.ScaleY -= 0.1
End If
```

Artık elimizde **DonMiktar** değişkeni hazır olarak bulunduğuunda göre DonMiktar'ın eksi veya artı değer almasına göre elimizdeki resmi büyütüp küçültebiliriz. Bu noktada büyütme ve küçültme miktarını el ile ayarlamakta fayda var. Aslında **DonMiktar** içerisinde farenin roller'inin bir defada ne kadar döndürüldüğüne dair bir değer geliyor fakat bunu kullanmak farklı sistemlerde çok ilginç sonuçlar verebiliyor. En iyisi sabit bir yol izlemek.

```
If DonMiktar <> 0 Then
    e.PreventDefault()
    Gelen SetProperty("returnValue", False)
End If
```

Son olarak tarayıcıya farenin roller değişimi ile ilgili işlemlerini bizim yaptığımızı belirtmemiz gereklidir. Böylece tarayıcı içerisinde sayfa scroll etmeyecektir.

Uygulamamızın tam kodu aşağıdaki şekilde sonuçlanıyor....

```
Private Sub FareTekerlekDondu(ByVal sender As Object, ByVal e As System.Windows.Browser.HtmlEventArgs)
    Dim DonMiktar As Integer = 0
    Dim Gelen As System.Windows.Browser.ScriptObject = e.EventObject
    'IE ve OPERA
    If Not Gelen.GetProperty("wheelDelta") Is Nothing Then
        'OPERA'da ters!
        If Not Gelen.GetProperty("opera") Is Nothing Then
            DonMiktar = -DonMiktar
        End If
        DonMiktar = Gelen.GetProperty("wheelDelta")
        'Mozilla ve Safari
    ElseIf Not Gelen.GetProperty("detail") Is Nothing Then
        DonMiktar = -Gelen.GetProperty("detail")
    End If

    If DonMiktar > 0 Then
        ZoomCarpani.ScaleX += 0.1
        ZoomCarpani.ScaleY += 0.1
    Else
        ZoomCarpani.ScaleX -= 0.1
        ZoomCarpani.ScaleY -= 0.1
    End If
```

```
If DonMiktar <> 0 Then  
    e.PreventDefault()  
    Gelen SetProperty("returnValue", False)  
End If  
End Sub
```

Hepinize kolay gelsin...

Daron YÖNDEM

Silverlight 2.0 içerisinde farenin çift tıklamasını algılamanın yolu.

Silverlight içerisinde otomatik olarak farenin çift tıklamasını algılayacak bir sistem bulunmuyor. Çok ciddi bir eksik gibi gözükmesse de aslında özellikle iş uygulamaları hazırlarken bu eksik can sıkabiliyor. Aslında bu eksiği gidermenin çok kolay bir yolu var. Çift tıklama sistemi entegre etmek istediğiniz bir kontrol normal tıklama durumunu kontrol ederek bir önceki normal tıklama ile aradaki süreyi ve bir önceki tıklama ile şu anki tıklamanın pozisyonlarını kontrol etmeniz yeterli olacaktır.

[VB]

```
Dim SonKonum As Point
```

```
Dim SonTik As Date
```

```
Private Sub Page_MouseLeftButtonDown(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Me.MouseLeftButtonDown
    If e.GetPosition(Me).X < SonKonum.X + 10 And e.GetPosition(Me).X > SonKonum.X - 10 Then
        If e.GetPosition(Me).Y < SonKonum.Y + 10 And e.GetPosition(Me).Y > SonKonum.Y - 10 Then
            If DateDiff(DateInterval.Second, SonTik, Now) < 1 Then
                MessageBox.Show("HO")
            End If
        End If
    End If
    SonKonum = e.GetPosition(Me)
    SonTik = Now
End Sub
```

Yukarıdaki kod içerisinde de görebileceğiniz üzere ilk olarak bir önceki tıklama bilgilerini saklamak üzere bir **Date** bir de **Point** değişkenimizi global olarak tanımlıyoruz. Bu değişkenlerin sürekli en son tıklamaya ait konum ve zaman bilgilerini saklayacak. Sonrasında MouseLeftButtonDown event-listener'i içerisinde bir önceki tıklama ile şu anki tıklamanın koordinatlarını karşılaştırıyoruz. Kullanıcı tabi ki biraz fareyi kaydırılmış olabilir o nedenle yaklaşık 20 piksellik bir kayma payı verebiliriz. Eğer yeni gelen tıklamanın koordinatları bir önceki ile aynı ise bu sefer de ikinci tıklama anı ile bir önceki tıklama arasında geçen süreyi hesaplıyoruz. Süre bir saniyenin altında ise büyük ihtimal ile bir çift tıklama gerçekleşmiş demektir. Bizim örneğimizde basit bir MessageBox gösteriyoruz, sizin kodlarınızda tabi ki farklı işlemler yapılacaktır.

Tüm bu kontrollerin sonucu olumlu ve olumsuz olsun, en sonunda da Son Tıklama'ya ait bilgileri saklayacak olan değişkenlerimize yeni tıklamanın bilgilerini aktarmayı unutmayın ki bir sonraki tıklamada bu bilgileri "bir önceki" tıklama bilgileri başlığı ile incelenebilsin.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde harici Class Library yapılarının asenkron kullanımı.

Silverlight projeleriniz büyükçe projenin bazı bölümlerini sonradan istemci tarafına aktarmayı daha uygun bir seçenek olarak görebilirsiniz. Bu gibi durumlarda acaba ayrı bir XAP dosyası yapsak da onu haricen istemciye yüklesek diye düşünürseniz maalesef söz konusu XAP dosyasını kendi kodlarınız ile ZIP şeklinde açmanız ve içerisindeki Manifest.xml'i yine kendi kodunuz ile okuyup tek tek DLL'leri yüklemeniz gerekecektir. Bu konuda detaylı bir makaleye [buradan](#) ulaşabilirsiniz.

Bu zorluklarla uğraşmadan hızlı bir şekilde belki de sadece bir UserControl'ü haricen sonradan yüklemek istiyorsanız aslında çok daha pratik ve hızlı bir yöntem de kullanılabilir. Bu yönteme sadece UserControl'ler değil harici olarak yazılan sınıflar da dahil. Gelin daha fazla teorik konuşma yerine bir örnek üzerinden ilerleyelim.

Haricen yüklenecek içeriği hazırlayalım....

İlk olarak ana Silverlight uygulamamıza sonradan yüklenecek olan içeriği hazırlayalım. Bunun için Visual Studio içerisinde "File / New Project" dedikten sonra "Silverlight" seçenekleri altındaki "**Silverlight Class Library**" proje tipini seçiyoruz. Bu proje tipinde doğrudan tüm proje içeriği bir DLL içerisinde konacak fakat bu DLL ayrıca bir XAP dosyası içerisinde sıkıştırılmayacak. Böylece biz de Silverlight ile istemci tarafında bir XAP dosyası açmak veya Manifest ile uğraşmak zorunda kalmayacağız.

Normal şartlarda Silverlight Class Library projesi yarattığınızda proje içerisinde sadece bir CS veya VB dosyası görebilirsiniz. Oysa bu projelere de isterseniz XAML dosyaları ile beraber UserControl'ler eklenebilir. Projenize sağ tuş tıklayarak Solution Explorer içerisinde "Add New Item" demeniz ve gelen seçeneklerden de "Silverlight User Control"ü seçmeniz yeterli olacaktır. Artık isterseniz bu projeyi Blend içerisinde de açıp normal bir Silverlight projesindeki gibi animasyonlar vs kullanabilirsiniz.

Örnek olarak projemize bir resim dosyası ekleyerek UserControl'ümüz içerisinde de onu gösterebilir. Unutmayın ki resim dosyasını projeye "Add Existing Item" diyerek eklerseniz artık bu resim de DLL'inizin içerisinde dahil edilecektir.

[XAML]

```
<UserControl x:Class="SilverlightClassLibrary1.SilverlightControl1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="400" Height="300">
  <Grid x:Name="LayoutRoot" Background="White">
    <Image Margin="71,47,97,122" Source="Forest.jpg"/>
  </Grid>
</UserControl>
```

Yukarıdaki şekli ile UserControl'ümüz hazır olduktan sonra projemizi Build ederek DLL'imizi yaratmış oluyoruz. Bu DLL'i bir sonraki adımda yaratacağımız Silverlight projesinin XAP dosyası ile aynı konuma koyabilirsiniz. Silverlight projemiz içerisinde bu DLL'i istemciye asenkron olarak download ederek sahneye DLL içerisindeki UserControl'ü yükleyeceğiz.

Gelelim Silverlight projemize...

Tertemiz bir Silverlight projesi yarattıktan sonra proje ile beraber gelen ASP.NET sitesi içerisinde ClientBin klasörüne bir önceki adımda yarattığımız DLL dosyasını kopyalayalım. Böylece projeyi Build ettiğimiz aynı konuma otomatik olarak kopyalanacak olan XAP dosyası üzerinden DLL'e de rahatlıkla ulaşabiliriz.

Yeni Silverlight projemizin ana Page.XAML dosyasına bir Button ve bir de Canvas ekleyelim. Böylece düğmeye basıldığında harici DLL'i yükleyecek ve DLL içerisindeki UserControl'ümüzü de Canvas içerisine yerleştireceğiz.

[XAML]

```
<UserControl x:Class="SilverlightApplication5.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button x:Name="btnTikla" Height="19" HorizontalAlignment="Right"
Margin="0,0,18,17" VerticalAlignment="Bottom" Width="89" Content="Button"/>
        <Canvas x:Name="Icerik" Margin="13,13,18,54"/>
    </Grid>
</UserControl>
```

Düğmeye tıklandığı anda hemen bir WebClient yaratarak download işlemimizi başlatalım.

[VB]

```
Private BirAssembly As System.Reflection.Assembly

Public Sub New()
    InitializeComponent()
    AddHandler Me.btnAdd.Click, AddressOf btnTikla_Click
End Sub

Private Sub btnTikla_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    Dim Yukleyici As New WebClient()
    AddHandler Yukleyici.OpenReadCompleted, AddressOf
    Yukleyici_OpenReadCompleted
    Dim Yol As String =
    System.Windows.Application.Current.Host.Source.AbsoluteUri.Replace("SilverlightA
    pplication5.xap", "SilverlightClassLibrary1.dll")
    Yukleyici.OpenReadAsync(New Uri(Yol, UriKind.Absolute))
End Sub
```

[C#]

```
System.Reflection.Assembly BirAssembly;
public Page()
```

```

{
    InitializeComponent();
    this.btnTikla.Click += new RoutedEventHandler(btnTikla_Click);
}

void btnTikla_Click(object sender, RoutedEventArgs e)
{
    WebClient Yukleyici = new WebClient();
    Yukleyici.OpenReadCompleted += new
    OpenReadCompletedEventHandler(Yukleyici_OpenReadCompleted);
    string Yol =
System.Windows.Application.Current.Host.Source.AbsoluteUri.Replace("SilverlightApplicat
ion5.xap", "SilverlightClassLibrary1.dll");
    Yukleyici.OpenReadAsync(new Uri(Yol, UriKind.Absolute));
}

```

Yukarıdaki kodu dikkatli incelemek gerekirse ilk adımda en üstteki Assembly tipindeki BirAssembly adındaki değişkenimizi açıklamak gerekecek. Kodumuz sunucudan bir DLL indirecek ve içindeki UserControl'ü sahneye koyacak. Aslında DLL'i indirdikten sonra içerisinde UserControl sınıfından bir instance olarak sahneye koyacağz. Eğer bu işlemi yaptıktan sonra başka instance'lara da ihtiyacımız olursa tekrar DLL'i indirmemek için eldeki Assembly'yi bir değişken olarak tutmak daha mantıklı olacaktır. O nedenle en üstteki BirAssembly değişkenimiz şimdiden yerini almış durumda.

Button'umuzun Click koduna baktığımızda bir WebClient yarattığımızı ve OpenReadCompleted event listener'ını da başka bir koda bağladığımızı görebilirsiniz. Sunucudan bir dosya indireceği ve indirme işlemi bittiğinde de başka işler yapacağız. O nedenle bu event'ları yakalayabiliyor olmak çok önemli. İsteyenler WebClient'in **DownloadProgressChanged** event'ini da yakalayarak download durumu ile ilgili yüzde üzerinden ne kadarının indirildiğine dair bilgileri de ekranда gösterebilirler.

Sunucudan indireceğimiz dosyanın tam yolunu verebilmek için şu anki XAP dosyasının tam yolunu alıp sadece dosya adını değiştiriyoruz. Bizim örneğimizde saten her şeyin yeri ve dosya adları belli olduğu için herhangi bir sorun olmayacağı.

Son olarak OpenReadAsync metoduna da indirilecek olan dosyanın yolunu verip download işlemini başlatıyoruz. Peki ya bu işler bitince çalışacak olan **Yukleyici_OpenReadCompleted** metodunda neler yapacağız?

[VB]

```

Private Sub Yukleyici_OpenReadCompleted(ByVal sender As Object, ByVal e As
OpenReadCompletedEventArgs)
    Dim GelenAssembly As New AssemblyPart()
    BirAssembly = GelenAssembly.Load(e.Result)
    Dim Kontrol As UserControl =
DirectCast(BirAssembly.CreateInstance("SilverlightClassLibrary1.SilverlightControl1"
), UserControl)
    Me.Icerik.Children.Add(Kontrol)
End Sub

```

[C#]

```
AssemblyPart GelenAssembly = new AssemblyPart();
BirAssembly = GelenAssembly.Load(e.Result);
UserControl Kontrol =
(UserControl)BirAssembly.CreateInstance("SilverlightClassLibrary1.SilverlightControl1");
this.Icerik.Children.Add(Kontrol);
```

Download işlemi bittiği anda bir **AssemblyPart** değişkeni yaratarak onun da **Load** metodunu kullanıyoruz. **Load** metoduna **e.result** ile aslında **Yukleyici_OpenReadCompleted** event-listener'ına gelen argüman üzerindeki datayı almış oluyoruz. Yani özünde sunucudan indirdiğimiz DLL'in **Stream'i** **e.result** içerisinde saklanıyor ve biz de bu **AssemblyStream'i** doğrudan bir **AssemblyPart** üzerinden **Load** ederek **Assembly** tipindeki **BirAssembly** değişkenimize yükliyoruz. Hatırlarsanız zaten bu değişkenimiz de global anlamda sürekli hafızada tuttuğumuz bir değişkendi. Bir sonraki adımda bir UserControl değişkeni tanımlayarak bunu da Assembly'mız içerisinde SilverlightControl1'e eşitlememiz gerekiyor.

Assembly üzerinden CreateInstance metodu bizden yaratılacak nesnenin TypeName'ini istiyor. Silverlight Class Library projesinin içerisindeki UserControlümüzün tipinin adını full path olarak veriyoruz. Bunu zaten UserControl'ün XAML dosyasının en üstünden de bulabilirsiniz. Artık elimizdeki **Kontrol** değişkeni yine elimizdeki **BirAssembly'nin** içerisinde SilverlightControl1'in bir instance'ıdır. Herhangi bir UserControl gibi bu da alıp sahnedede istediğimiz yere yerleştirebiliriz.

Hepinize kolay gelsin.

Daron YÖNDEM

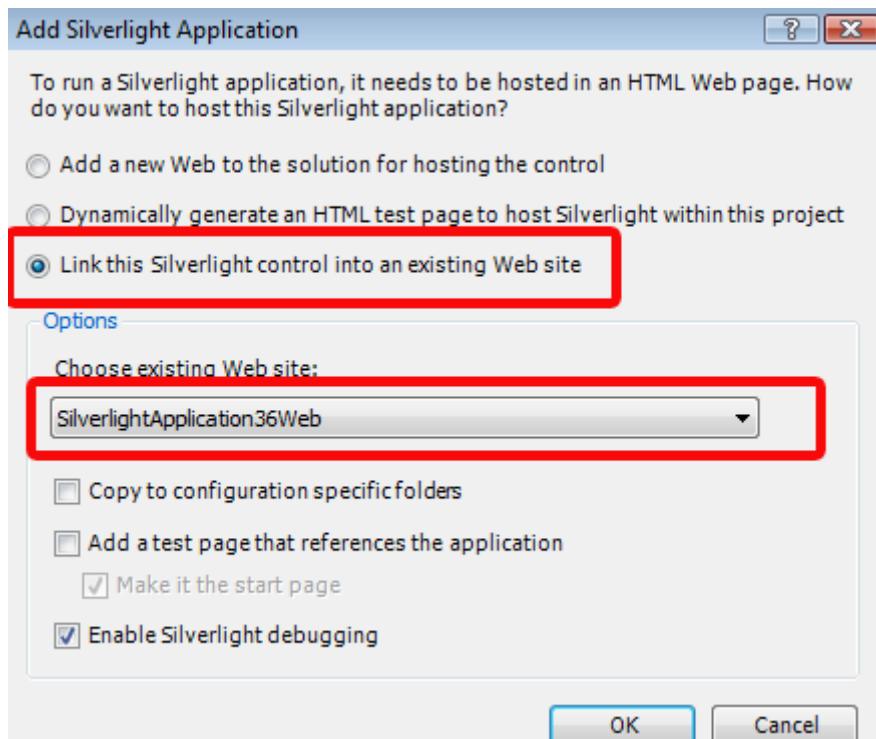
Silverlight 2.0 içerisinde harici dinamik XAP (Silverlight 2.0) uygulamalarının asenkron yüklenmesi.

Dinamik olarak Silverlight 2.0 uygulamalarında farklı XAP dosyalarını istemci tarafına yükleyerek ana XAP içerisinde gösterebiliyor olmak kullanıcı deneyimi açısından büyük önem taşıyor. İçerisinde yoğun animasyonların ve belki de harici kontrollerin bulunduğu bir arayüzü Silverlight uygulaması istemciye ilk gönderildiğinde topluca göndermek doğru olmayabilir. Bu durum hem ana Silverlight uygulamasının istemcide açılma süresini uzatacak hem de belki kullanıcının hiç kullanmayacağı uygulama bölümlerinin gereksiz yere kullanıcıya gönderilmesine neden olacaktır. Tüm bu nedenlerle dinamik olarak harici XAP dosyalarını, yani Silverlight uygulamalarını başka bir Silverlight uygulaması içerisinde yükleyerek çalıştırabilmek büyük önem taşıyor. Sonraki örneğimizde dinamik XAP yükleme işlemini nasıl yapabileceğimizi inceleyeceğiz.

Projemizi hazırlayalım...

İlk olarak Visual Studio içerisinde yeni bir Silverlight projesi yaratarak beraberinde bir de ASP.NET uygulaması yaratılması için ilk açılışta gelen uyarı kutusunda gerekli işaretlemeyi yapalım. Bu standart prosesi atlattıktan sonra artık Visual Studio içerisindeki Solution'ımıza yeni bir Silverlight projesi daha eklememiz gerekiyor. Toplam olarak Solution içerisinde bir ASP.NET ve iki Silverlight projesi bulunacak.

Solution'a Solution Explorer içerisinde sağ tuş ile tıklayarak gelen menüden "Add / New Project" yeni bir Silverlight projesi seçerek ekleyebilirsiniz. Ekleme işlemini yaparken size aşağıdaki şekilde bir pencere ile eklenen Silverlight uygulamasının Solution içerisinde bir ASP.NET sitesi ile ilişkilendirip ilişkilendirilmeyeceği sorulacaktır. Bu noktada var olan uygulama ile yeni Silverlight'ımızı ilişkilendirmemiz gerekiyor, böylece Visual Studio içerisinde F5'e bastığımızda bu Silverlight uygulaması da compile edilerek ASP.NET sitesi içerisinde kopyalanacaktır.



İkinci Silverlight uygulamamızı Solution içerisinde eklerken...

Sonradan yüklenecek Silverlight uygulamamızı hazırlayalım...

İlk olarak uzaktaki (remote) Silverlight uygulamamızı hazırlayalım. Örnek olması ve konudan çok uzaklaşmamak adına uygulamamız çok basit olacak fakat unutmayın ki uzaktaki Silverlight uygulaması video gösteren veya belki de harici kontroller (DataGrid) kullanan bir uygulama da olabilirdi. Böyle bir durumda karşılıkla yüklenen uygulamamız çok daha büyük bir boyuta sahip olurdu.

Biz şimdilik uygulama içerisinde toplama işlemi yapan iki TextBox ve bir de Button bulunsun. Böylece ana uygulamada toplama işlemi yapılmayı zaman bu harici uygulamayı yükleyerek kullanıcının istediğini yapmasını sağlayalım.

[XAML]

```
<UserControl x:Class="SilverlightRemote.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Height="40" HorizontalAlignment="Left" Margin="40,40,0,0"
            VerticalAlignment="Top" Width="120" Text="TextBox" TextWrapping="Wrap"
            x:Name="Kutu1"/>
        <TextBox Height="40" HorizontalAlignment="Right" Margin="0,40,40,0"
            VerticalAlignment="Top" Width="120" Text="TextBox" TextWrapping="Wrap"
            x:Name="Kutu2"/>
        <Button HorizontalAlignment="Stretch" Margin="160,120,160,140"
            VerticalAlignment="Stretch" Content="Button" x:Name="Dugme"/>
    </Grid>
```

</UserControl>

Uygulamamızın XAML kodu yukarıdaki şekilde sonlanıyor. Şimdi geçelim arkaplarda çalışan ve basit bir şekilde toplama işlemi yaparak sonucu düğmenin üzerine yazdırın koda.

[VB]

```
Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme.Click
    Dugme.Content = CInt(Kutu1.Text) + Kutu2.Text
End Sub
```

[C#]

```
public Page()
{
    InitializeComponent();
    Dugme.Click += new RoutedEventHandler(Dugme_Click);
}

void Dugme_Click(object sender, RoutedEventArgs e)
{
    Dugme.Content = int.Parse(Kutu1.Text) + int.Parse(Kutu2.Text);
}
```

Ana Silverlight uygulamamızı hazırlayalım...

Sıra geldi gerektiğinde uzaktaki Silverlight uygulamamızı yükleyerek kullanıcının kullanımını sağlayacak olan ana uygulamamızı geliştirmeye. Bunun için yine basit bir örnek olarak uygulamamız içerisinde bir **StackPanel** yerleştirelim. Söz konusu StackPanel içerisinde şimdilik uzaktaki uygulamamızın yüklenmesini sağlayacak olan bir düğme yer alacak.

[XAML]

```
<UserControl x:Class="SilverlightApplication36.Page"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400" Height="300">
    <StackPanel x:Name="LayoutRoot" Background="White">
        <Button Content="TOPLAMA işlemi için tıkla" x:Name="Dugme"/>
    </StackPanel>
</UserControl>
```

Uygulamamızın tasarımını da hazır olduğuna göre artık düğmeye basıldığında çalışacak kod kısmına geçebiliriz. Düğmemize tıklandığında bir WebClient kullanarak sunucudan diğer uygulamanın XAP dosyasını istemci tarafına yüklememiz gerekiyor.

[VB]

```

Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme.Click
    Dim Yukleme As New WebClient
    AddHandler Yukleme.OpenReadCompleted, AddressOf
Yukleme_OpenReadCompleted

    Yukleme.OpenReadAsync(New Uri("SilverlightRemote.xap", UriKind.Relative))
End Sub

```

[C#]

```

public Page()
{
    InitializeComponent();
    Dugme.Click += new RoutedEventHandler(Dugme_Click);
}

void Dugme_Click(object sender, RoutedEventArgs e)
{
    WebClient Yukleme = new WebClient();
    Yukleme.OpenReadCompleted += new
OpenReadCompletedEventHandler(Yukleme_OpenReadCompleted);
    Yukleme.OpenReadAsync(new Uri("SilverlightRemote.xap", UriKind.Relative));
}

```

Kodumuzun ilk satırında bir **WebClient** yarattıktan sonra **WebClient**'in **OpenReadCompleted** event'ını da bir event-handler'a bağlıyoruz. Sonrasında da **OpenReadAsync** metodu ile uzaktaki XAP dosyasını istemciye indirmeye başlıyoruz. İndirme işlemi tamamlandığında event-handler kodumuz çalışacak. Şimdi esas işlemleri yapacağımız, XAP dosyası istemciye indiğinde çalışacak olan event-handler kodumuza geçelim.

[VB]

```

Dim UygulamaManifesti As String = New
IO.StreamReader(Application.GetResourceStream(New
Windows.Resources.StreamResourceInfo(e.Result, Nothing), New
Uri("AppManifest.xaml", UriKind.Relative)).Stream).ReadToEnd()

```

[C#]

```

string UygulamaManifesti = new
System.IO.StreamReader(Application.GetResourceStream(new
System.Windows.Resources.StreamResourceInfo(e.Result, null), new
Uri("AppManifest.xaml", UriKind.Relative)).Stream).ReadToEnd();

```

Buradaki kod ilk bakışta biraz karışık gelebilir fakat aslında çok basit. Her Silverlight uygulaması içerisinde (XAP dosyası içerisinde) bir **AppManifest.xaml** bulunur. Bu dosya içerisinde tek tek söz konusu uygulamada kullanılan Assembly'lerin adları ve ilişkili DLL dosyalarının adları bulunur. Bizim de hedefimiz uzaktaki uygulamada kullanılmış tüm

Assembly'leri bularak istemci tarafında belleğe yüklemek. Aslında bizim örneğimizde tek bir Assembly olduğunu biliyoruz fakat eğer örnek farklı olsaydı ve harici kontroller kullanılmış olsaydı doğal olarak birden çok Assembly olacaktı. O nedenle biz daha genel geçer bir taktik kullanarak esnek olalım.

Yukarıdaki kod içerisinde bir **StreamResourceInfo** nesnesine **e.Result** ile indirdiğimiz XAP dosyasını aktarıyoruz. Bu **StreamResource** içerisinde de Application.**GetResourceStream** ile **AppManifest.xaml** dosyasını istiyoruz. Dosyayı aldığımızda **Stream'ini** de bir **StreamReader** ile sonuna kadar **ReadToEnd** ile okuyoruz. Böylece artık elimizde **AppManifest.xaml** var.

[VB]

```
Dim Dagitim As Deployment = CType(Markup.XamlReader.Load(UygulamaManifesti), Deployment)
```

[C#]

```
Deployment Dagitim = System.Windows.Markup.XamlReader.Load(UygulamaManifesti) as Deployment;
```

Bir sonraki adımda **AppManifest.xaml** içerisinde tanımlı olan **Deployment** şeklini bir **Deployment** nesnesine eşitlememiz gerekiyor. Bunu aslında elimizde String olarak var olan bir XAML'ı .NET nesnesine çevirme olarak değerlendirebilirsiniz.

Markup.XamlReader.Load metodu ile elimizdeki dosyayı okutarak gelen nesneyi de **Deployment'a** Cast ediyoruz. Zaten uzaktaki (remote) uygulamamızı Compile ettiğimizde oluşan XAP dosyasının içerisinde girerek AppManifest.xaml'ına baktığımızda da aşağıdaki XAML ile karşılaşıyoruz.

[XAML] AppManifest.xaml

```
<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
EntryPointAssembly="SilverlightRemote" EntryPointType="SilverlightRemote.App"
RuntimeVersion="2.0.30523.6">
<Deployment.Parts>
<AssemblyPart x:Name="SilverlightRemote" Source="SilverlightRemote.dll" />
</Deployment.Parts>
</Deployment>
```

Sıra geldi artık Deployment içerisindeki tüm Assembly'leri gezerek hepsini istemci tarafında hafızaya yüklemeye. Tüm bu yüklemeleri yaparken aralarından SilverlightRemote.dll adında olanı bir kenara çekerek referansını ayrı bir değişkende tutacağız. Bunun yapmamızın nedeni ise bu DLL içerisindeki **Page** sınıfından bir adet türeterek sahneye almak zorunda olmamız. Assembly'leri hafızaya yüklesek de işin görsel kısmını sahneye almamız lazım. Bunun detaylarına birazdan değineceğiz, önce bir Assembly'leri tek tek yükleyelim.

[VB]

```
Dim GorselAssembly As System.Reflection.Assembly = Nothing
```

```

For Each BirAssembly As AssemblyPart In Dagitim.Parts
    Dim AssemblyDLLAdi As String = BirAssembly.Source
    Dim StreamBilgi As Windows.Resources.StreamResourceInfo =
        Application.GetResourceStream(New Windows.Resources.StreamResourceInfo(e.Result,
            "application/binary"), New Uri(AssemblyDLLAdi, UriKind.Relative))

    If AssemblyDLLAdi = "SilverlightRemote.dll" Then
        GorselAssembly = BirAssembly.Load(StreamBilgi.Stream)
    Else
        BirAssembly.Load(StreamBilgi.Stream)
    End If

```

[Next](#)

[C#]

```

System.Reflection.Assembly GorselAssembly = null;

foreach (AssemblyPart BirAssembly in Dagitim.Parts)
{
    string AssemblyDLLAdi = BirAssembly.Source;
    System.Windows.Resources.StreamResourceInfo StreamBilgi =
        Application.GetResourceStream(new
            System.Windows.Resources.StreamResourceInfo(e.Result, "application/binary"), new
            Uri(AssemblyDLLAdi, UriKind.Relative));

    if (AssemblyDLLAdi == "SilverlightRemote.dll")
    {
        GorselAssembly = BirAssembly.Load(StreamBilgi.Stream);
    }
    else
    {
        BirAssembly.Load(StreamBilgi.Stream);
    }
}

```

Satır satır yukarıdaki kodumuzu inceleyelim. İlk satırda **Assembly** tipinde bir değişken yaratıyoruz. Bu değişken gerektiğinde bizim uzaktaki uygulama içerisinde görSEL arayüzü tanımlayan **Page** sınıfının bulunduğu DLL'in referansını taşıyacak. Sonrasında hemen **Deployment** nesnemiz olan **Dagitim'in Parts** dizisinde bir **ForEach** döngüsü başlatarak tüm Assembly'leri gezmeye başlıyoruz. Her Assembly'nin **Source'unu** yani tam DLL dosyasının adını alarak bu dosyaları tek tek **StreamBilgi** adındaki değişkenimize yükliyoruz. XAP dosyası içerisindeki DLL'leri alırken aynı **AppManifest.Xaml**'ı alırkenki teknigi kullanıyoruz. Son olarak hemen o an için üzerinde çalıştığımız Assembly'nin görSEL arayüzümüzün bulunduğu Assembly olup olmadığını kontrole yapıyoruz. Bunun için doğrudan Assembly'e ait DLL adını karşılaştırıyoruz.

Burada hemen bir dipnot geçelim. Bir Silverlight uygulaması içerisinde harici kontrolleri içeren veya farklı sınıfları ve kodları barındıran DLL'ler bulunabilir. Bunların haricinde bir de

her XAML (görsel) dosya arkasındaki UserControl tipindeki sınıflar vardır. Biz yukarıdaki kodumuzda içerisinde UserControl bulunan DLL'i ayırarak bir kenara referansını kaydediyoruz çünkü bu UserControl'ları sahneye almamız gerekiyor. Oysa diğer Assembly'leri kullanabilmek için sadece belleğe yüklememiz yeterli.

İF koşulumuz içerisinde gerekli kontrolleri de yaptıktan sonra normal Assembly'leri Load metodu ile Stream'ini vererek belleğe yüklerken içerisinde UserControl'lerimizin bulunduğu Assembly'i yüklerken **GorselAssembly** değişkenine de bir referansını alıyoruz.

[VB]

```
Dim Arayuz As UIElement =
 CType(GorselAssembly.CreateInstance("SilverlightRemote.Page"), UIElement)
 LayoutRoot.Children.Add(Arayuz)
```

[C#]

```
UIElement Arayuz = GorselAssembly.CreateInstance("SilverlightRemote.Page") as
UIElement;
this.LayoutRoot.Children.Add(Arayuz);
```

Sıra geldi **GorselAssembly** içerisindeki ana UserControl'ümüz olan **Page** sınıfından bir instance olarak sahneye yerleştirmeye. Bunun için Assembly'nin **CreateInstance** metoduna sınıfımızın tam yolunu vererek bir kopyasını alıyor ve **UIElement** tipine cast ediyoruz. Sonrasında da elimizdeki nesneyi uygulamamızdaki StackPanel içerisinde ekliyoruz.

Artık uygulamamızı çalıştırabiliriz. İlk Silverlight uygulaması istemciye yüklenikten sonra düğmeye bastığınızda ikinci uygulama sunucudan alınarak içerisinde UserControl sahneye yerleştirilecektir. Böylece rahatlıkla bir Silverlight projesini parçalar şeklinde geliştirebilir ve uygulamanın bölümlerini gereklük istemci tarafına aktarabilirisiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

[Silverlight 2.0 içerisinde Hue / Saturation ve Lightness ile dinamik renk paletleri yaratmanın yolu.](#)

Silverlight 2.0 ile beraber istemci taraflı CLR altyapısı ile aslında hayal bile edemeyeceğimiz bir sürü işlemi bildiğimiz .NET dilleri ile yapabiliyoruz. Birazdan yapacağımız örneği Silverlight 2.0 öncesi herhangi bir teknoloji ile uygulamaya kattığımızda bir .NET yazılım geliştiricisi için çok daha acı verici bir süreç söz konusu olabilirdi. Oysa doğrudan istemci tarafında .NET kullanımını ile çok daha rahat bir platform sağlanabiliyor.

Silverlight 2.0 Beta 1 içerisinde ister farklı User Control yapıları olsun veya ister farklı görsel stiller kullanın belirli noktalarda dinamik olarak **Gradient** yapıları kurmanız gerekebiliyor. İşte tam da bu noktada bir rengin geçiş yapabileceği başka uygun bir rengi programatik olarak bulmak ciddi sıkıntı verebilir. Aslında en basit çözüm Expression Design gibi programlarda yapabildiğimiz; bir rengin **Hue / Saturation / Lightness** değerlerini değiştirmektir. Böylece rengin ana yapısı değişmese de parlaklık ve ışık miktarı değiştirilerek farklı geçişler sağlanabilir ve bu farklı renklerden Gradient'lar dinamik olarak yaratılarak rahatlıkla kullanılabilir. Fakat maalesef Silverlight ile beraber gelen yapıya baktığımızda biz renklerin RGB (Red, Green, Blue) olarak geldiğini görüyoruz. Peki **Hue**, **Lightness** ve **Saturation** nasıl hesaplanıyor? Bu konuda live.com'da ufak bir araştırma HSL ile RGB arasında çeviri işlemlerinin nasıl yapılacağını öğrenmemiz için yeterli. Maalesef internette bu işi hazır yapan bir kod bulamadım. Ben matematik hesaplamaların ve sisteminin mantığının detayına girmeden sizinle yazmış olduğum kodu paylaşacağım.

Namespace HSLveRGB

 Public Structure HslRenk

 Public Alpha As Double

 Public Hue As Double

 Public Saturation As Double

 Public Lightness As Double

 Private Function Normal(ByVal gelen As Double) As Double

 If gelen < 0 Then

 gelen += 1

 End If

 If gelen > 1 Then

 gelen -= 1

 End If

 Return gelen

 End Function

 Private Shared Function B2P(ByVal gelen As Byte) As Double

 Dim giden As Double = gelen

 giden = giden / 255

 Return giden

 End Function

 Private Shared Function P2B(ByVal gelen As Double) As Byte

 gelen *= 255

 gelen += 0.5

 If gelen > 255 Then

```

gelen = 255
End If
If gelen < 0 Then
    gelen = 0
End If
Return CByte(gelen)
End Function

```

```

Public Shared Function FromColor(ByVal BirRenk As Color) As HslRenk
    Return HslRenk.FromArgb(BirRenk.A, BirRenk.R, BirRenk.G, BirRenk.B)
End Function

```

```

Public Function RengiAc(ByVal x As Double) As HslRenk
    Dim BirRenk As New HslRenk()
    BirRenk.Alpha = Me.Alpha
    BirRenk.Hue = Me.Hue
    BirRenk.Saturation = Me.Saturation
    BirRenk.Lightness = Math.Min(Math.Max(Me.Lightness + x, 0), 1)
    Return BirRenk
End Function

```

```

Public Shared Function FromArgb(ByVal Alpha As Byte, ByVal Kirmizi As Byte,
ByVal Yesil As Byte, ByVal Mavi As Byte) As HslRenk
    Dim BirRenk As HslRenk = FromRgb(Kirmizi, Yesil, Mavi)
    BirRenk.Alpha = B2P(Alpha)
    Return BirRenk
End Function

```

```

Public Shared Function FromRgb(ByVal Kirmizi As Byte, ByVal Yesil As Byte, ByVal
Mavi As Byte) As HslRenk
    Dim BirRenk As New HslRenk()
    BirRenk.Alpha = 1
    Dim red As Double = B2P(Kirmizi)
    Dim green As Double = B2P(Yesil)
    Dim blue As Double = B2P(Mavi)
    Dim max As Double = Math.Max(blue, Math.Max(red, green))
    Dim min As Double = Math.Min(blue, Math.Min(red, green))
    If max = min Then
        BirRenk.Hue = 0
    ElseIf max = red AndAlso green >= blue Then
        BirRenk.Hue = 60 * ((green - blue) / (max - min))
    ElseIf max = red AndAlso green < blue Then
        BirRenk.Hue = 60 * ((green - blue) / (max - min)) + 360
    ElseIf max = green Then
        BirRenk.Hue = 60 * ((blue - red) / (max - min)) + 120
    ElseIf max = blue Then
        BirRenk.Hue = 60 * ((red - green) / (max - min)) + 240
    End If

```

BirRenk.Lightness = 0.5 * (max + min)

```
If max = min Then
    BirRenk.Saturation = 0
ElseIf BirRenk.Lightness <= 0.5 Then
    BirRenk.Saturation = (max - min) / (2 * BirRenk.Lightness)
ElseIf BirRenk.Lightness > 0.5 Then
    BirRenk.Saturation = (max - min) / (2 - 2 * BirRenk.Lightness)
End If
Return BirRenk
End Function
```

```
Public Function RengiKoyulastir(ByVal x As Double) As HslRenk
    Return RengiAc(-x)
End Function
```

```
Private Function Hesap(ByVal Bir As Double, ByVal Iki As Double, ByVal Uc As Double) As Double
If Bir < (1 / 6) Then
    Return Iki + ((Uc - Iki) * 6 * Bir)
End If
If Bir < 0.5 Then
    Return Uc
End If
If Bir < (2 / 3) Then
    Return Iki + ((Uc - Iki) * 6 * ((2 / 3) - Bir))
End If
Return Iki
End Function
```

```
Public Function ToColor() As Color
Dim Bir As Double = 0
If Lightness < 0.5 Then
    Bir = Lightness * (1 + Saturation)
Else
    Bir = Lightness + Saturation - (Lightness * Saturation)
End If
Dim Iki As Double = (2 * Lightness) - Bir
Dim Key As Double = Hue / 360
Dim red As Double = Hesap(Normal(Key + (1 / 3)), Iki, Bir)
Dim green As Double = Hesap(Normal(Key), Iki, Bir)
Dim blue As Double = Hesap(Normal(Key - (1 / 3)), Iki, Bir)
Return Color.FromArgb(P2B(Alpha), P2B(red), P2B(green), P2B(blue))
End Function
End Structure
End Namespace
```

Yukarıdaki kodu isterseniz harici bir DLL olarak derleyerek tüm projelerinizde kullanabilirsiniz. Silverlight içerisindeki kullanımına da ufak bir örnek ile göz atalım. Aşağıdaki şekilde Silverlight 2.0 uygulamamıza bir dikdörtgen ve Slider yerleştirerek Slider ile dikdörtgen içerisindeki rengi değiştireceğiz.

```
<UserControl x:Class="HSL2RGB.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Rectangle HorizontalAlignment="Stretch" Margin="43,23,47,124"
            VerticalAlignment="Stretch" Fill="#FFD05D5D" Stroke="#FF000000" x:Name="Kutu"/>
        <Slider Height="24" Margin="43,0,47,82" VerticalAlignment="Bottom"
            x:Name="Slider" Maximum="1" LargeChange="0.1" SmallChange="0.01"/>
    </Grid>
</UserControl>
```

Özellikle Slider'in alabildiği maksimum değere dikkat etmekte fayda var. Bu değer üzerinden bizim daha önceki HSL nesnesini kullanarak **RengiAc** ve **RengiKoyulastir** metodlarını çalıştıracağız. Geçelim uygulamanın kod kısmına.

Partial Public Class Page

Inherits UserControl

Public Sub New()

 InitializeComponent()

End Sub

Dim AnaRenk As Color = System.Windows.Media.Color.FromArgb(100, 255, 50, 50)

Private Sub Slider_ValueChanged(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Slider.ValueChanged

 CType(Kutu.Fill, SolidColorBrush).Color =

 HSLveRGB.HslRenk.FromColor(ArenaRenk).RengiAc(e.NewValue - 0.5).ToColor

End Sub

End Class

Yukarıdaki kod içerisinde ilk olarak dikkat edilmesi gereken nokta bizim global **AnaRenk** değişkenimiz. Bu değişken içerisinde sürekli bizim ana rengimiz duruyor ve bu renk üzerinden gerekli işlemleri yaparak yarattığımız yeni rengi **Kutu** nesnesinin **Fill** özelliğine atanmış **SolidColorBrush**'in **Color** özelliğine aktarıyoruz. **RengiAc** metodunu kullanırken de Slider'in mevcut değerine göre -0.5 ile 0.5 arasında bir değer gelmesini sağlıyoruz. Zaten aynı metod kendisine eksi değer verildiğinde rengi açmak yerine kapatıyordu.

Böylece rahatlıkla renklerin Hue / Saturation ve Lightness özelliklerini Silverlight tarafından dinamik olarak değiştirebiliyor. Bu özellikleri kullanarak sadece tek bir renk üzerinden giderek başka renkler de yaratıp güzel Gradient yapıları kurabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Isolated Storage kullanımı

Web uygulamalarında Cookie kullanımı alışık olduğumuz bir yapıdır. Bu yapının bir benzeri Silverlight 2.0 Beta 1 ile beraber de karşımıza geliyor. "**Isolated Storage**" olarak adlandırılan alan sadece Silverlight uygulamanıza özel olarak varsayılan ayarları ile 100KB'luk bir alanı istemci tarafında programcının kullanımına sunuyor. İlk olarak gelin bu alana veri yazma ve okuma işlemlerinin nasıl yapıldığını bir göz atalım.

Örneğimizde uygulamamız içerisinde bir TextBox ve iki Button yer alacak. Buttonlardan birine basıldığında TextBox içerisindeki veri **Isolated Storage** içerisinde kaydedilecek diğeri ise veriyi silecek. Isolated Storage içerisinde doğrudan dosyalar ve klasörler saklayabiliyoruz. O nedenle biz de örneğimizde saklamak istediğimiz metni bir TXT dosyası şeklinde diske kaydedeceğiz. Gelin ilk olarak uygulamamızın arayüzü aşağıdaki şekilde hazırlayalım.

```
<UserControl x:Class="SilverlightApplication12.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Margin="38,18,51,105" Text="TextBox" x:Name="txtMetin"/>
        <Button Height="41" HorizontalAlignment="Right" Margin="0,0,51,35"
VerticalAlignment="Bottom" Width="127" Content="KAYDET" x:Name="DgmKaydet"/>
        <Button Height="41" HorizontalAlignment="Left" Margin="104,0,0,35"
VerticalAlignment="Bottom" Width="91" Content="KAYDI SİL" x:Name="DgmSil"/>
    </Grid>
</UserControl>
```

Uygulamamız ilk açıldığında Isolated Storage içerisinde daha önce kaydedilmiş "deneme.txt" adında bir dosyanın olup olmadığını kontrol edeceğiz. Eğer böyle bir dosya varsa içeriğini okuyarak doğrudan TextBox içerisinde göstereceğiz.

```
Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
    If DEPO.FileExists("deneme.txt") Then
        Dim Dosya As IO.IsolatedStorage.IsolatedStorageFileStream =
DEPO.OpenFile("deneme.txt", IO FileMode.Open, IO FileAccess.Read)
        Dim Okuyucu As IO.StreamReader = New IO.StreamReader(Dosya)
            txtMetin.Text = Okuyucu.ReadToEnd
            Okuyucu.Close()
        Else
            txtMetin.Text = "Dosya yok"
        End If
    End Using
```

Isolated Storage ile ilgili yapacağımız tüm işlemleri

IsolatedStorageFile.GetUserStoreForApplication() ile mevcut kullanıcının alanını ele alarak yapacağız. O an için istemci işlem yapan kullanıcının bizim uygulamamız için ayrılmış olan alanına ulaştıktan sonra hemen hedef konumda deneme.txt adında bir dosya olup olmadığını **FileExists** metodu ile kontrol edebiliyoruz. Eğer söz konusu dosya varsa **OpenFile** metodu ile dosyamızı açarak bir **StreamReader**'a kaynak olarak veriyoruz.

Bundan sonrası aslında alışık olduğumuz dosya okuma metodu yok ise TextBox içerisinde doğrudan "Dosya Yok" yazdırıyoruz. Eğer dosyamız herhangi bir klasör içerisinde olsaydı gerekli metodlara sadece dosya ismini değil klasör ismi ile beraber bir yol adresini vermek durumunda kalacaktık. Unutmayın tüm bu klasörler ve dosyalar bizim uygulamamıza ait Isolated Storage alanına saklanıyor olacak. Klasör yaratma konusunda özellikle bir uyarıda bulunmam gereklidir. Normal şartlarda Windows'ta herhangi bir klasör boş ise diskte yer kaplamaz. Isolated Storage içerisinde her klasör 1KB alan kaplıyor. Bunun aslında mantıklı bir açıklaması var; kötü niyetli Silverlight programcının istemci tarafında milyonlarca klasör yaratmasını engellemek :)

Şimdi geçelim bir sonraki adıma ve elimizdeki metin kutusuna yazılan herhangi bir değeri TXT dosyası olarak Isolated Storage içerisinde nasıl kaydedeceğimizi inceleyelim.

```
Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
Using Dosya As IO.IsolatedStorage.IsolatedStorageFileStream =
DEPO.CreateFile("deneme.txt")
Dim Yazici As IO.StreamWriter = New IO.StreamWriter(Dosya)
    Yazici.WriteLine(txtMetin.Text)
    Yazici.Close()
    Istatistik()
End Using
End Using
```

Yukarıdaki kod içerisinde yine **GetUserStoreForApplication** dierek mevcut kullanıcının Isolated Storage alanını alıyoruz ve sonrasında **CreateFile** metodu ile yeni bir dosya yaratıyoruz. Yarattığımız dosyanın içerisinde ise bir **StreamWriter** ile elimizdeki metni yazdırıyoruz. Kodun en sonunda **Istatistik** denen bir kodu çalıştırduğumu göreceksiniz. Söz konusu kodu ileriki adımlarda yazacağız. Amacımız Isolated Storage içerisinde kullanılan ve kalan alanı kullanıcıya göstermek olacak.

Artık dosyamızı da kaydettiğimize göre sıra geldi ikinci düğmeye basıldığında söz konusu dosyayı Isolated Storage alanından silmeye.

```
Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
If DEPO.FileExists("deneme.txt") Then DEPO.DeleteFile("deneme.txt")
End Using
```

Kodumuz gerçekten çok basit. Yine mevcut Isolated Storage alanından yola çekerek **FileExists** ile dosyanın varlığını kontrol ettikten sonra **DeleteFile** ile söz konusu dosyayı istemciden siliyoruz. Uygulamamızın tam kodunu incelemeden önce bir de **Istatistik** adındaki kodumuzu yazalım.

```
Sub Istatistik()
    Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
        DgmKaydet.Content = "Kaydet " & vbCrLf & "(Kalan Alan:" &
Math.Round(DEPO.AvailableFreeSpace / 1024) & "/" & Math.Round(DEPO.Quota / 1024)
& ")"
    End Using
End Sub
```

```
End Using
End Sub
```

Kullanıcının Isolated Storage alanını bir değişkene aktardıktan sonra doğrudan **Quota** ile mevcut kotayı, **AvailableFreeSpace** ile de boş alanının byte olarak alabiliyoruz. Örneğimizde bu sayıları 1024'e bölgerek kullanıcıya KB biriminde bir istatistik gösteriyoruz. Şimdi uygulamamızın son halini inceleyebiliriz.

```
Partial Public Class Page
    Inherits UserControl
```

```
    Public Sub New()
        InitializeComponent()
    End Sub
```

```
    Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
        Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
            If DEPO.FileExists("deneme.txt") Then
                Dim Dosya As IO.IsolatedStorage.IsolatedStorageFileStream =
DEPO.OpenFile("deneme.txt", IO FileMode.Open, IO FileAccess.Read)
                Dim Okuyucu As IO.StreamReader = New IO.StreamReader(Dosya)
                txtMetin.Text = Okuyucu.ReadToEnd
                Okuyucu.Close()
            Else
                txtMetin.Text = "Dosya yok"
            End If
        End Using
    End Sub
```

```
    Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles DgmKaydet.Click
        Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
            Using Dosya As IO.IsolatedStorage.IsolatedStorageFileStream =
DEPO.CreateFile("deneme.txt")
                Dim Yazici As IO.StreamWriter = New IO.StreamWriter(Dosya)
                Yazici.WriteLine(txtMetin.Text)
                Yazici.Close()
                Istatistik()
            End Using
        End Using
    End Sub
```

```
Sub Istatistik()
    Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
```

```
DgmKaydet.Content = "Kaydet " & vbCrLf & "(Kalan Alan:" &
Math.Round(DEPO.AvailableFreeSpace / 1024) & "/" & Math.Round(DEPO.Quota / 1024)
& ")"
End Using
```

```
End Sub
```

```
Private Sub DgmSil_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles DgmSil.Click
Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
If DEPO.FileExists("deneme.txt") Then DEPO.DeleteFile("deneme.txt")
End Using
Istatistik()
Page_Loaded(sender, e)
End Sub
End Class
```

Örneğimizdeki gibi farklı dosyalar yaratarak Isolated Storage alanı içerisinde saklayabilirsiniz. Ayrıca isterseniz **CreateDirectory**, **DeleteDirectory** metodlarını kullanarak istemci tarafında farklı klasörler yaratabilir, gerektiğinde **GetDirectoryName** ve **GetFileNames** ile daha önce kaydedilmiş dosya ve klasörlerin isimlerini de birer liste olarak alabilirsiniz.

Peki ya 100KB bize yetmezse?

Eğer 100KB size yetmiyorsa hedef istemcideki kullanıcının iznini alarak söz konusu alanı artıtabilirisiniz. Bunun için aşağıdaki gibi bir kod yeterli olacaktır.

```
Using DEPO As IO.IsolatedStorage.IsolatedStorageFile =
IO.IsolatedStorage.IsolatedStorageFile.GetUserStoreForApplication()
DEPO.TryIncreaseQuotaTo(1000000)
End Using
```

TryIncreaseQuotaTo metoduna parametre olarak istediğiniz alanın byte miktarnı aktarmanız gerekiyor. Böylece kullanıcıya uygulamanın daha fazla alan istediği dair bir uyarı gösterilecek onayı isteniyor. Eğer kullandı onay verirse **TryIncreaseQuotaTo** metodu geriye **True** döndürüyor, aksi halde ise **False** Boolean değeri geliyor.

Daha kolay kullanımı birşey yok mu?

Isolated Storage gerçekten bize istemci tarafında mini bir sabit disk vermişcesine olanaklar sağlıyor. Oysa bazı durumlarda sadece ufak bir değeri, uygulamaya ilgili bir ayarı istemci tarafında saklamak gerekebilir. Bunun için tek tek gidip dosyalar yaratmak ve verileri dosyalara kaydetmek uğraştırıcı gelebilir. İşte böyle bir durumda özel olarak hazırlanmış olan **System.IO.IsolatedStorage.ApplicationSettings** sınıfından faydalanaabiliyoruz.

'Mevcut AppSettings nesnesini alalım.

Dim Ayarlar As System.IO.IsolatedStorage.ApplicationSettings =

System.IO.IsolatedStorage.ApplicationSettings.Default

'Yeni bir ayar ekleyelim

```
Ayarlar.Add("RenkSecimi", "Kirmizi")
'Var olan ayarı değiştirelim
    Ayarlar("RenkSecimi") = txtMetin.Text
'Var olan bir ayarin değerini alalım
    txtMetin.Text = CType(Ayarlar("RenkSecimi"), String)
'Var olan bir ayarı silelim
    Ayarlar.Remove("RenkSecimi")
```

System.IO.IsolatedStorage.ApplicationSettings sınıfı üzerinden varsayılan ayarları bir değişkene aktardıktan sonra yukarıdaki örnek kod içerisindeki metodları kullanarak rahatlıkla farklı ayarları Isolated Storage içerisinde kaydedebiliyor, değiştirebiliyor ve silebiliyoruz.

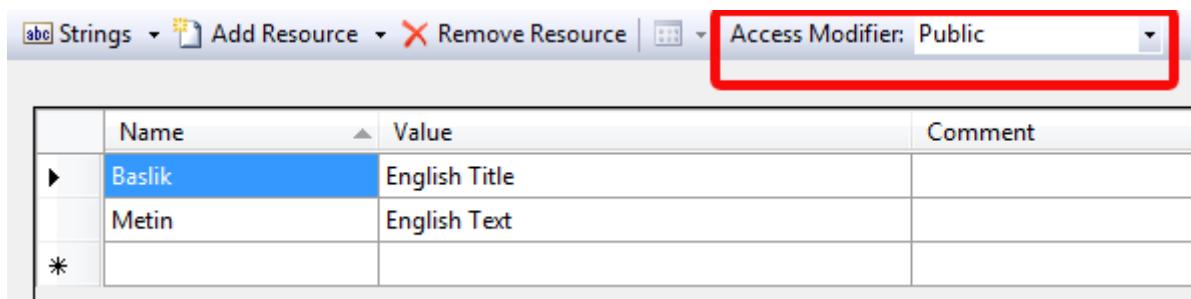
Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Localization kullanımı

Birden çok dil kullanılan projelerde .NET ile beraber gelen dahili Localization özellikleri hızlı çözümler yaratmak adına ciddi birer kurtarıcı olarak değerlendirilebilir. Kişisel olarak her zaman daha özelleştirilebilir altyapılara sahip olmak adına el yapımı altyapıları tercih etsem de her zaman bunun için zaman bulunamayabiliyor. **Silverlight 2.0** ile beraber de artık .NET dillerini ve altyapısını kullanabildiğimize göre bize yardımcı olacak bir Localization sistemi olsa gerek diyerek beraberce yola çıkalım. Ürün **Beta 2** aşamasında olduğu için bazı sorunları radikal çözümlerle atlatacağız fakat sonunda tabi ki çalışır bir örneğimiz olacak.

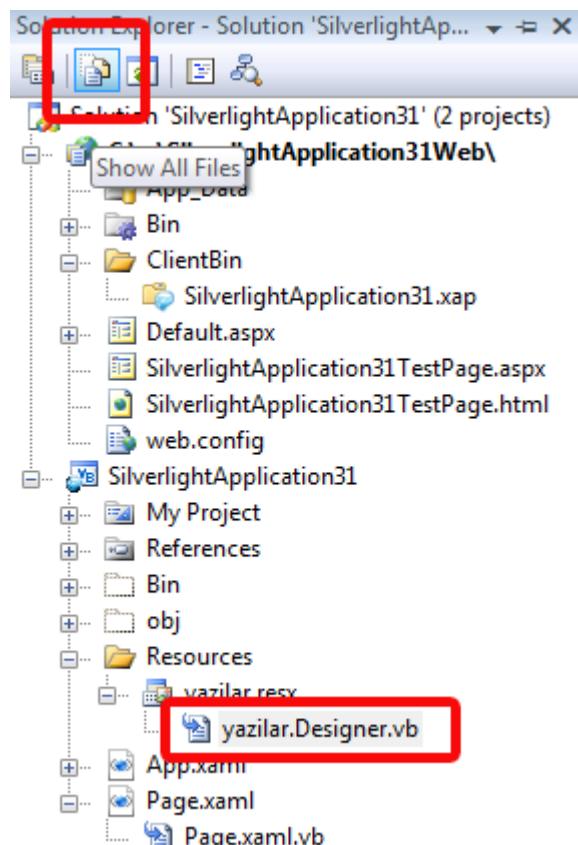
Yeni bir Silverlight 2.0 Beta 2 projesi yarattıktan sonra hemen Visual Studio içerisindeki Solution Explorer penceresinde Silverlight projemize sağ tuş tıklayarak "Add / New Folder" diyelim ve **Resources** adında bir klasör ekleyelim. Uygulamamızda olmasını istediğimiz dillere ait bilgileri bu klasör içerisine yerleştireceğiz. **Resources** klasörüne sağ tuş ile tıklayarak gelen menünden "Add / New Item" komutunu vererek **yazilar.resx** adında bir "Resources File" ekleyelim. Karşınıza çıkan ekranda **Resource** dosyası içerisinde istediğiniz "Name / Value" çiftini girebilirsiniz. Şimdilik örnek olması amacıyla aşağıdaki şekilde düzenlememizi yapalım.



	Name	Value	Comment
▶	Baslik	English Title	
Metin	English Text		
*			

İngilizce Localization Resource dosyamız.

Unutmamanız gereken detaylardan biri dosya ile ilgili **Access Modifiers** ayarsını **Public** olarak değiştirmek. Bu ayarı yaptığımızda aslında planda otomatik olarak değişmesi gereken bir kod daha var. Söz konusu kod Resource dosyamızla beraber otomatik olarak生成的 **Designer.vb** veya **Designer.cs** dosyası içerisinde yer alıyor.



Arkaplandaki sinsi kod dosyası!

Yukarıdaki ekran görüntüsünde de görebileceğiniz üzere Visual Studio içerisinde Solution Explorer'da "Show All Files" düğmesine tıkladığınızda karşınıza ek dosyalar çıkacaktır. Bu dosyalardan Resource dosyamızın arkasında durdan VB/CS dosyasını açarak aşağıdaki kodları değiştirmemiz gerekiyor.

Eskisi

[VB]

```
Friend Sub New()
    MyBase.New()
End Sub
```

[C#]

```
internal yazilar() {
}
```

Yenisi

[VB]

```
Public Sub New()
    MyBase.New()
End Sub
```

[C#]

```
public Resource10 {
}
```

Artık sıra geldi bu Resource dosyası içerisindeki verileri Silverlight XAML sayfalarımızda kullanmaya. Bunun için Page.XAML dosyamızı açarak hemen Root XML elementimize yeni bir XML NameSpace eklememiz şart.

```
<UserControl x:Class="SilverlightApplication31.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:yerel="clr-namespace:SilverlightApplication31.My.Resources"
```

```
Width="400" Height="300">
```

Yukarıdaki şekli ile eklediğimiz NameSpace aslında Intellisense desteği sayesinde otomatik olarak karşınıza çıkacaktır. Eğer herhangi bir hata alırsanız projenizi bir kereliğine Build ederek sonra tekrar deneyin. NameSpace'in ismini tanımlamak tamamen size kalmış ben örnekte "yerel" anahtar kelimesini kullanmayı tercih ettim. Son olarak aşağıdaki kod ile Assembly içerisindeki kullanacağımız kaynağımızı da sayfanın Resource'ları içerisinde ekleyelim.

```
<UserControl.Resources>
    <yerel:yazilar x:Name="LocStrings" />
</UserControl.Resources>
```

Dillere göre metinleri görebilmek için uygulamamızın arayüzüne iki adet TextBlock koymakta fayda var. TextBlock'ları da koyduğumuzda uygulamanın XAML kodu aşağıdaki şekilde sonuçlanıyor.

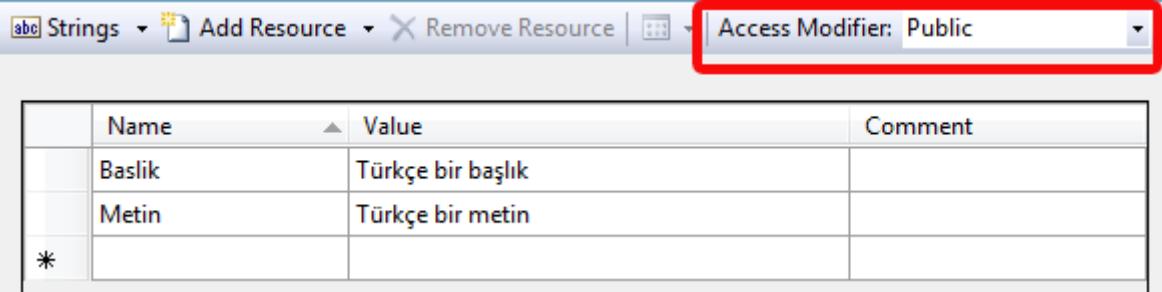
```
<UserControl x:Class="SilverlightApplication31.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:yerel="clr-namespace:SilverlightApplication31.My.Resources"
    Width="400" Height="300">
    <UserControl.Resources>
        <yerel:yazilar x:Name="Kaynak" />
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Height="40" HorizontalAlignment="Left" Margin="40,40,0,0"
VerticalAlignment="Top" Width="160" Text="TextBlock" TextWrapping="Wrap"
x:Name="Label1"/>
        <TextBlock HorizontalAlignment="Left" Margin="40,120,0,140" Width="160"
Text="TextBlock" TextWrapping="Wrap" x:Name="Label2"/>
    </Grid>
</UserControl>
```

Resource'lar içerisindeki bilgileri bu TextBlock'lara bağlamamız gereklidir. Bunun için klasik bir Binding kullanırken veri kaynağı olarak da sayfanın Resource'larına eklediğimiz **Kaynak** adındaki veri kaynağımızı kullanacağız.

```
<Grid x:Name="LayoutRoot" Background="White">
    <TextBlock Height="40" HorizontalAlignment="Left" Margin="40,40,0,0"
VerticalAlignment="Top" Width="160" Text="{Binding Baslik, Source={StaticResource
Kaynak}}" TextWrapping="Wrap" x:Name="Label1"/>
    <TextBlock HorizontalAlignment="Left" Margin="40,120,0,140" Width="160"
Text="{Binding Metin, Source={StaticResource Kaynak}}" TextWrapping="Wrap"
x:Name="Label2"/>
</Grid>
```

Bağlama işlemini de tamamladığımıza göre geri kaldı ikinci bir dil için Resource dosyası oluşturmaya. Elde olan **yazilar.resx** dosyasından bir kopya alarak **yaziler.tr.resx** adında bir

dosya yaratabilirsiniz. Bu dosya içerisinde de yine **Baslik** ve **Metin** kaynaklarının Türkçe karşılıklarını yazmamız gerekecek.



	Name	Value	Comment
	Baslik	Türkçe bir başlık	
	Metin	Türkçe bir metin	
*			

Türkçe Localization Resource dosyamız.

Artık neredeyse her şey hazır. Fakat ufak bir sorunumuz daha var. Visual Studio bizim için Silverlight projemizin Bin/Debug klasöründe gerekli dillere özel klasörleri yaratarak DLL'leri yaratırsa da maalesef bunları Silverlight'in XAP dosyasına kopyalamıyor. Bu ufak hata Silverlight'in Beta 2 sonrası sürümlerinde düzeltilene kadar bizim Silverlight projemizin .proj uzantılı dosyasını NotePad ile açarak elle düzenlememiz gerekiyor.

<SupportedCultures>tr</SupportedCultures>

Yukarıda gördüğünüz şekilde PROJ dosyası içerisinde normalde içi boş olan bu tagların arasında kullanacağımız dillerin dil kodlarını sıralamamız şart. Tüm işlemlerimizi tamamladık, artık Silverlight uygulamasını sayfaya yerleştirirken hangi dil ayarını parametre olarak aktarırsanız o dile uygun kaynaklar yüklenecektir.

```
<object data="data:application/x-silverlight," type="application/x-silverlight-2-b2"
width="100%" height="100%">
  <param name="source" value="ClientBin/SilverlightApplication31.xap"/>
  <param name="background" value="white" />
  <param name="culture" value="tr" />
  <param name="uiculture" value="tr" />
</object>
```

Yukarıdaki kod içerisinde uygulamanın Türkçe ayarlar ile açılmasını sağlıyoruz.

Hepinize kolay gelsin.

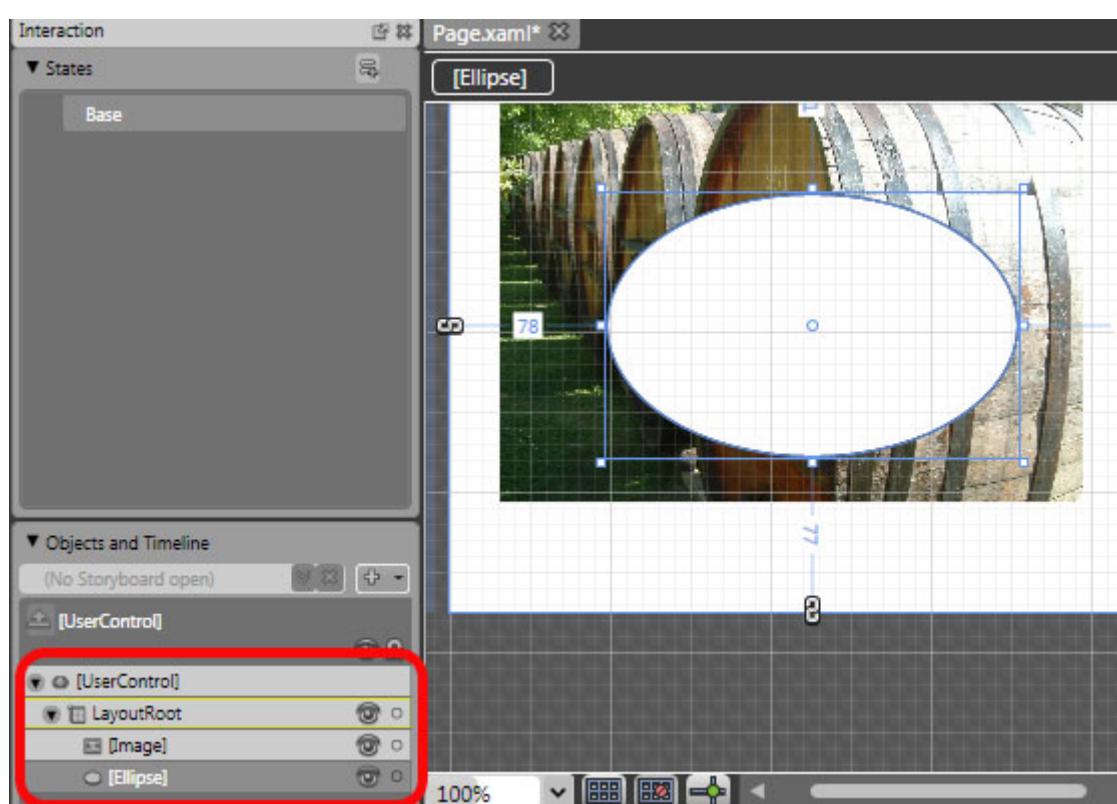
Daron YÖNDEM

Silverlight 2.0 içerisinde maskeleme (clipping)

Bu yazımızda Silverlight 2.0 içerisinde maskeleme (clip) işlemlerine göz atacağız. İlk olarak basit bir maskeleme işleminin XAML içerisinde nasıl yapıldığına baktıktan sonra bu maskeleri nasıl anime edebileceğimize ve programatik yoldan ulaşımına değineceğiz.

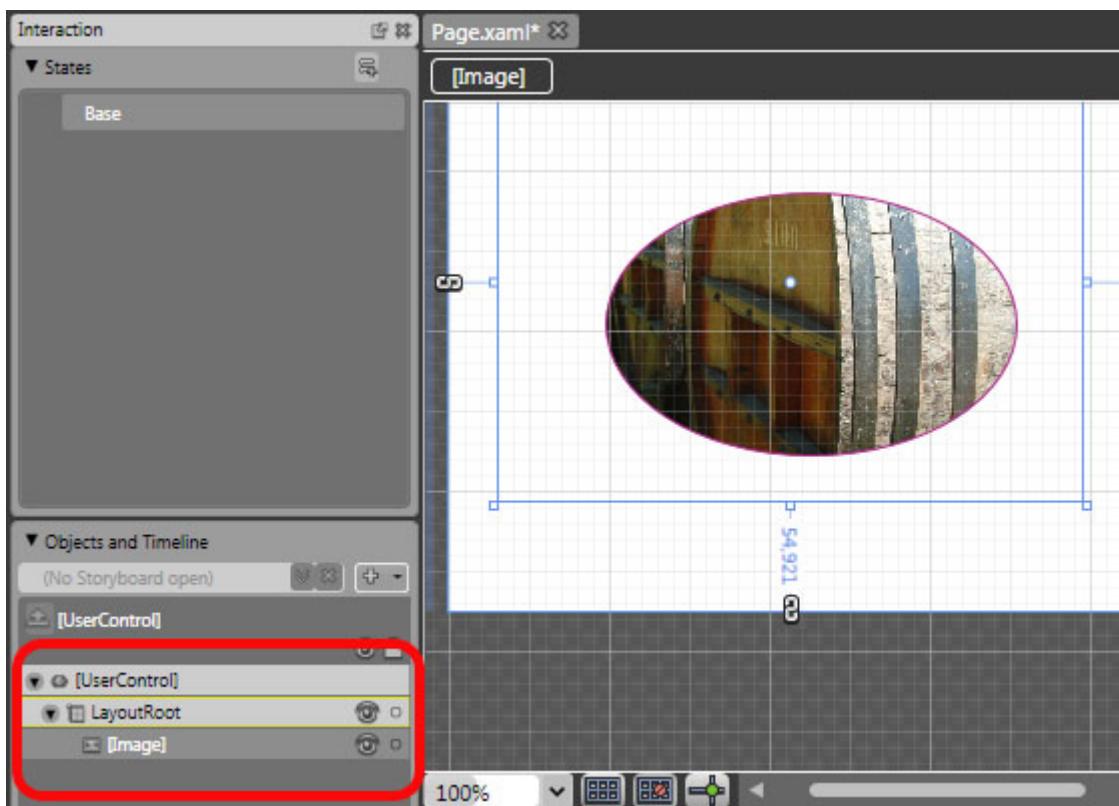
Bir maske yaratalım!

Herhangi bir nesneye maske aktarmak demek aslında o nesnenin Clip özelliğine uygun bir şekil aktarmak demektir. Elinizde var olan bir geometri nesnesini alarak bir element'in Clip özelliğine verdığınız anda artık söz konusu Geometri nesnesi bir maske görevi görür. Expression Blend içerisinde baktığınızda maskeler nesnelerin birer Property'sine atandığı için ayrı birer obje olarak arayüzde gözükmeyecektir.



Expression Blend içerisinde maskelenmeye hazır kontroller.

Yukarıdaki ekran görüntüsünde de görebileceğiniz üzere sahnede bir Image ve bir de Ellipse nesnesi bulunuyor. Bir sonraki adımda amacımız bu Ellipse nesnesini Image için bir maske haline getirmek. Yapacağımız işlem bu iki nesneyi fare ile seçip "Objects and Timeline" kısmında sağ tıklayıp "**Path / Make Clipping Path**" komutunu vermek. Böylece söz konusu Ellipse artık Image'in Clip özelliğine bir geometri olarak aktarılacak ve ortada Ellipse diye bir nesne kalmayacak.



Maskelenmiş Image kontrolümüz karşınızda.

Artık kontrolümüzü maskeledik ve Ellipse diye bir nesne kalmadı peki arkaplanda XAML tarafında neler oldu? Gelin Blend'in bizim için yarattığı XAML kodunu bir detaylıca inceleyelim.

[XAML]

```
<Image Margin="25.032,27,84.032,54.842" Source="1080366_88011245.jpg"
Stretch="Fill" Clip="M258.5,130 C258.5,166.17465 212.60919,195.5 156,195.5
C99.390816,195.5 53.5,166.17465 53.5,130 C53.5,93.825348 99.390816,64.5 156,64.5
C212.60919,64.5 258.5,93.825348 258.5,130 z"/>
```

Gördüğünüz gibi Image nesnesinin uzun bir Clip datası var. Bu data bizim bir önceki adımda yarattığımız Ellipse'in ta kendisi. Aslında bu kodu bu şekilde yazmak yerine daha okunaklı bir şekilde de yazabilirdik. Nasıl mı?

[XAML]

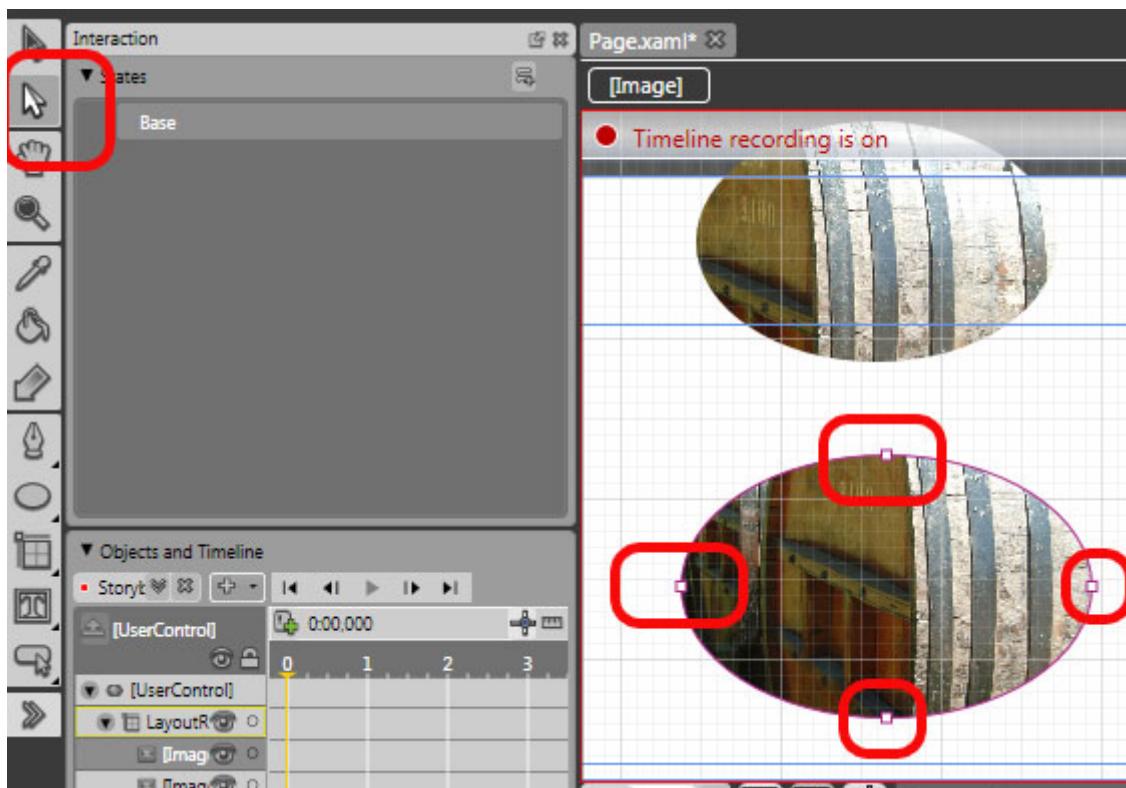
```
<Image Margin="7.968,-68,101.032,0" Source="1080366_88011245.jpg" Stretch="Fill"
VerticalAlignment="Top" Height="218">
<Image.Clip>
<EllipseGeometry Center="200,100" RadiusX="90" RadiusY="60" />
</Image.Clip>
</Image>
```

Peki ne değişti? Nesne basında **Clip** vermiş olduk. Image nesnesinin Clip özelliğine doğrudan koordinatları girmek yerine ayrı bir element vermeye karar verdik ve bu nedenle de **Image.Clip** tagları açarak içerisine bir Geometry nesnesi yerleştirdik. **EllipseGeometry**

nesnesi gibi **GeometryGroup**, **LineGeometry**, **PathGeometry** nesneleri de mevcut. Bizdeki örnekte **EllipseGeometry**'nin **Center** özelliği maskelenen **Image** nesnesinin merkez noktasına göre maskenin ne kadar uzaklıkta olacağına dair X ve Y değerlerini verirken **RadiusX** ve **RadiusY**'de yatay ve dikey olarak Ellipse'in yarıçapını tanımlıyor. Peki bu iki metod arasındaki diğer farklar nelerdir? Gelin olayın animasyon kısmına bir bakalım.

Maskelere animasyon katalım....

Herhangi bir nesnenin maskesine animasyon verebilmek için Expression Blend içerisinde animasyon modunda sol taraftaki araç çubuğundan **Selection** aracı yerine "**Direct Selection**" aracını kullanmalısınız. Söz konusu aracı seçtiğiniz anda seçili nesnenin maskesine ait tanımlı noktaları ekranda görebilirsiniz. Böylece rahatlıkla KeyFrame'ler yaratarak noktaların pozisyonlarını değiştirebilir ve maskeyi anime edebilirsiniz.



Maskemize animasyon verirken.

Biz örneğimizde maskedeki tüm noktaları toplu seçerek bir Ellipse şekländen tüm maskenin pozisyonunu değiştiren bir animasyon hazırlıyoruz. Böylece sanki bir ışık ile resme bakılıyormuş gibi resmin üzerinde geziliyor görüntüsü yaratıyoruz. Gelin Blend'in bizim için yarattığı XAML koduna göz atalım.

[XAML]

```
<Storyboard x:Name="Storyboard1">
    <PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.StartPoint)">
        <SplinePointKeyFrame KeyTime="00:00:00" Value="258.5,130"/>
```

```

<SplinePointKeyFrame KeyTime="00:00:01"
Value="218.937328102718,129.064332728402"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[0].(BezierSegment.Point1)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="258.5,166.174652099609"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="218.937328102718,165.238984828011"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[0].(BezierSegment.Point2)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="212.609191894531,195.5"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="173.046519997249,194.564332728402"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[0].(BezierSegment.Point3)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="156,195.5"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="116.437328102718,194.564332728402"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[1].(BezierSegment.Point1)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="99.3908157348633,195.5"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="59.8281438375813,194.564332728402"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[1].(BezierSegment.Point2)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="53.5,166.174652099609"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="13.937328102718,165.238984828011"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[1].(BezierSegment.Point3)">
<SplinePointKeyFrame KeyTime="00:00:00" Value="53.5,130"/>
<SplinePointKeyFrame KeyTime="00:00:01"
Value="13.937328102718,129.064332728402"/>

```

```

</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[2].(BezierSegment.Point1)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="53.5,93.8253479003906"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="13.937328102718,92.8896806287922"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[2].(BezierSegment.Point2)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="99.3908157348633,64.5"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="59.8281438375813,63.5643327284016"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[2].(BezierSegment.Point3)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="156,64.5"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="116.437328102718,63.5643327284016"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[3].(BezierSegment.Point1)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="212.609191894531,64.5"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="173.046519997249,63.5643327284016"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[3].(BezierSegment.Point2)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="258.5,93.8253479003906"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="218.937328102718,92.8896806287922"/>
</PointAnimationUsingKeyFrames>
<PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="image"
Storyboard.TargetProperty="(UIElement.Clip).(PathGeometry.Figures)[0].(PathFigure.Segments)[3].(BezierSegment.Point3)">
    <SplinePointKeyFrame KeyTime="00:00:00" Value="258.5,130"/>
    <SplinePointKeyFrame KeyTime="00:00:01"
Value="218.937328102718,129.064332728402"/>
</PointAnimationUsingKeyFrames>
</Storyboard>

```

Animasyonun kodu gördüğünüz gibi epey uzun. Aslında Blend kendi Clip'ini noktalardan yarattığı için mecburen bu noktaları tek tek anime ederek noktaların pozisyonlarını değiştirmek zorunda kalıyor. Bu esnada bizim Image'in XAML kodlarına bakarsan zaten minik bir değişiklik de dikkatimizi çekiyor.

[XAML]

```
<Image Margin="57,73.921,52,8.079" Source="1080366_88011245.jpg" Stretch="Fill"
x:Name="image">
  <Image.Clip>
    <PathGeometry>
      <PathFigure IsClosed="True" StartPoint="258.5,130">
        <BezierSegment Point1="258.5,166.174652099609"
Point2="212.609191894531,195.5" Point3="156,195.5"/>
        <BezierSegment Point1="99.3908157348633,195.5"
Point2="53.5,166.174652099609" Point3="53.5,130"/>
        <BezierSegment Point1="53.5,93.8253479003906"
Point2="99.3908157348633,64.5" Point3="156,64.5"/>
        <BezierSegment Point1="212.609191894531,64.5"
Point2="258.5,93.8253479003906" Point3="258.5,130"/>
      </PathFigure>
    </PathGeometry>
  </Image.Clip>
</Image>
```

Söz konusu Clip'in içindeki noktaları anime edebilmek için Blend de kısmen bizim taktiği geri dönmüş ve bir PathGeometry yerleştirmiştir. Oysa biz zamanında Ellipse koymuştuk :) Neden doğrudan o esnada **EllipseGeometry** koymadın? diye Blend'e sorarsak eminim cevabı basit olacaktır. "Nereden bilebilirim ki bu noktaları ayrı ayrı anime etmeyeceğiniz?" Aslında Blend haklı. Blend her ihtimali düşünmek zorunda çünkü o bir GUI :) Oysa biz kendi örneğimizde maske olarak Ellipse'in şeklini değiştirmeyeceğiz, sadece konumunu değiştireceğiz o nedenle bu sistem bize çok da uygun değil.

Yukarıdaki **PointAnimation** taglarına bakarsan tek tek Image nesnesinin **Clip** özelliğindeki değerin alıp bu değerdeki noktaların Index numarası verilerek bulduğunu ve pozisyonlarının değiştirildiğini görebilirsiniz. Performans açısından bir sıkıntısı olmasa da oluşan kodun okunabilirliği çok zayıf olmakla beraber bu gibi bir **Clip** nesnesinin programatik olarak anime edilmesi de neredeyse imkansız. C# veya VB kodu ile tek tek bu noktaları bulup anime etmek iğkenceden farksız olacaktır.

Peki ya bizim teknikle yaparsak?

Bizim esas istediğimiz maskenin konumunun değişmesiydi. Daha önceki kodlarımızda bir EllipseGeometry'yi maske olarak verebiliyoruz. Bu EllipseGeometry nesnesinin Center özelliği maskenin konumunu belirliyor. Bu durumda bizim kodumuzda bu özelliklerin anime edilmesi yeterli olacaktır. Fakat eğer Blend içerisinde bu animasyonu yapacak olursanız bizim EllipseGeometry'yi israrlı bir şekilde yukarıdaki gibi bir **PathGeometry**'ye çevirecek ve yine aynı animasyon kodunu üretecektir. Bu durumda bizim de programatik olarak bu maskeyi anime etmemiz yine zorlaşacaktır.

Sonuç olarak eğer bir nesnenin maskesinin pozisyonunu çok uğraşmadan kod ile anime edebilmek istiyorsanız Blend'in arayüzünden anime etmemeniz gerekiyor.

Programatik olarak maskeye erişmek...

Bir nesnenin maskesine programatik olarak erişmek için söz konusu nesnenin Clip özelliğini alıp uygun Geometry tipine cast edebilirsiniz. Sonrasında elinizde Geometry nesnesi ile ilgilenmeniz yeterli olacaktır. Oysa bir diğer seçenek de XAML içerisinde bu Geometry nesnesine el ile bir isim vermektedir.

[XAML]

```
<Image Margin="7.968,-68,101.032,0" Source="1080366_88011245.jpg" Stretch="Fill"
VerticalAlignment="Top" Height="218">
    <Image.Clip>
        <EllipseGeometry x:Name="Maskesi" Center="200,100" RadiusX="90"
RadiusY="60" />
    </Image.Clip>
</Image>
```

Gördüğünüz gibi basit bir şekilde bizim EllipseGeometry nesnesine bir isim verdik. Artık kod tarafında bu isim ile **EllipseGeometry**'ye ulaşabilir ve **Center** veya **RadiusX** ve **RadiusY** özelliklerini değiştirebiliriz.

[VB]

```
Maskesi.Center = New Point(200, 200)
```

Böylece tamamen kod ile bu maskeyi anime etmek istediğinizde de rahatlıkla bu noktayı anime ederek maskenin pozisyonunun değiştiği bir animasyon üretebilirsiniz. Örnek bir kodu aşağıda inceleyebilirsiniz.

[VB]

```
Dim DBL As New PointAnimation
DBL.From = New Point(100, 100)
DBL.To = New Point(200, 200)
DBL.Duration = New TimeSpan(0, 0, 2)
Storyboard.SetTarget(DBL, Maskesi)
Storyboard.SetTargetProperty(DBL, New
PropertyPath(EllipseGeometry.CenterProperty))
Dim SB As New Storyboard
SB.Children.Add(DBL)
SB.Begin()
```

[C#]

```
PointAnimation DBL = new PointAnimation();
DBL.From = new Point(100, 100);
DBL.To = new Point(200, 200);
```

```
DBL.Duration = new TimeSpan(0, 0, 2);
Storyboard.SetTarget(DBL, Maskesi);
Storyboard.SetTargetProperty(DBL, new
PropertyPath(EllipseGeometry.CenterProperty));
Storyboard SB = new Storyboard();
SB.Children.Add(DBL);
SB.Begin();
```

Eğer bu animasyonu XAML tarafında temiz olarak yazmak isterseniz aslında pratik bir şekilde elle de yazabilirsiniz.

[XAML]

```
<Storyboard x:Name="Storyboard1">
  <PointAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Maskesi"
Storyboard.TargetProperty="(EllipseGeometry.Center)">
  <SplinePointKeyFrame KeyTime="00:00:00" Value="100,100"/>
  <SplinePointKeyFrame KeyTime="00:00:01" Value="200,200"/>
</PointAnimationUsingKeyFrames>
</Storyboard>
```

Animasyonumuz doğrudan Maskesi adındaki **EllipseGeometry'yi** hedef alarak onun **Center** property'sini anime ediyor.

Unutmayın ki bizim örneğimizde böyle bir optimizasyon yapabilmemizin nedeni maskenin sadece pozisyonunu değiştirmek istememiz. Eğer maskedeki her bir noktanın konumunu ayrı ayrı birbirinden bağımsız olarak anime etmek isterseniz Blend'in yaptığı teknikten farklı bir seçenekiniz zaten yok.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde MultiScaleImage kullanımı ve DeepZoom maceraları

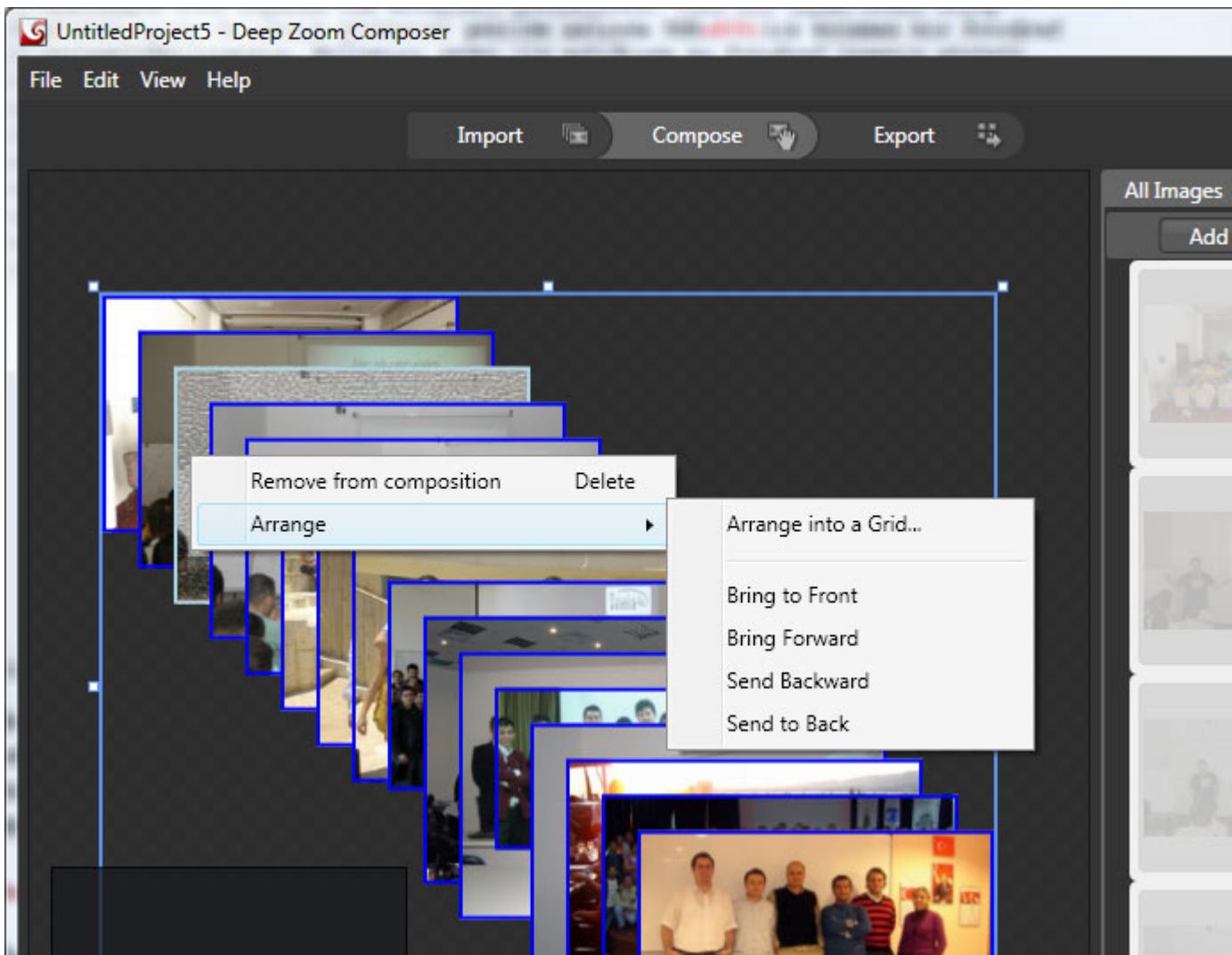
Silverlight 2.0 ile beraber gelen kontrollerden biri de **MultiScaleImage** kontrolü. Bu kontrolün yapabildikleri arasında gerçekten çok ilginç uygulamalar var. Genel itibarı ile çok büyük boyutta resimlerin istemci tarafında bant genişliğinin en uygun performans ile kullılarak gösterilmesini sağladığını söyleyebiliriz. Örneğin elinizde çok yüksek çözünürlükte 2GB'lık bir fotoğraf var ve bu fotoğrafı istemci tarafında göstermek istiyorsunuz veya elinizde toplam 3GB'lık bir fotoğraf arşivi var ve bunu güzel bir Silverlight galerisi şeklinde kullanıcılarla paylaşmak istiyorsunuz. Fakat uygulamanın çalışırken doğal olarak tüm resimleri yüklememesi hatta sadece o an ekranda gözüken kısımları yüklemesi gerekiyor. Tüm bunları elle tek tek kodlayarak yapabiliriz fakat **MultiScaleImage** varken aslında hersey çok daha kolay ilerliyor.

Deep Zoom Composer

MultiScaleImage kontrolü içerisinde gösterilecek resimlerin farklı detaylarda gösterilebilmesi ve kullanıcı resme zoom yaptıkça yeni detayların yüklenerek ekranada gösterilebilmesi için arkaplanda resimlerin biraz detaylı bir yapıda hazırlanmış olması gerekiyor. Bu işlemleri otomatik olarak yapabilecek bir uygulama olan Deep Zoom Composer'i aşağıdaki adresten bilgisayarınıza indirip kurabilirsınız.

<http://www.microsoft.com/downloads/details.aspx?FamilyID=457b17b7-52bf-4bda-87a3-fa8a4673f8bf&DisplayLang=en>

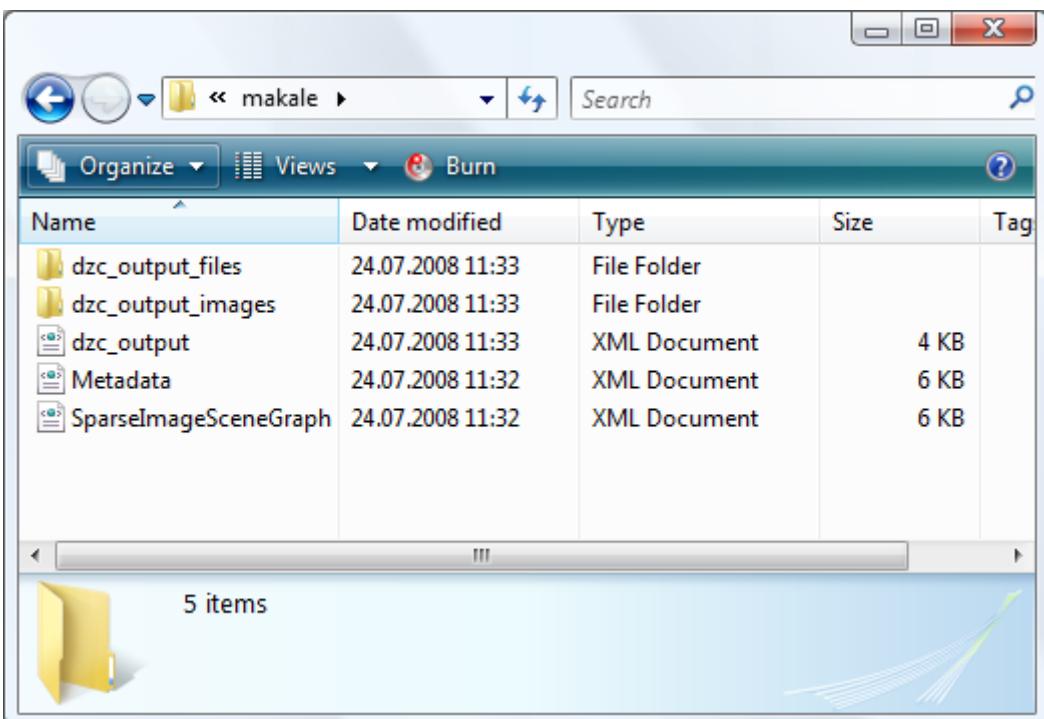
Yüklemeyi yapıp yeni bir DeepZoom projesi yarattığınızda programın arayüzündeki "Add Image" düğmesi ile projeye istediğiniz kadar resim ekleyebilirsınız. Eklediğiniz bu resimleri "Compose" sekmesinde sahneye sürükleyerek istediğiniz şekilde resimleri iç içe veya üst üste koyabilirsınız. Resimleri ufaltabilir veya büyütübilirsınız. Unutmayın ki burada bir resmi diğerine göre çok ufak yerleştirseniz de kullanıcı birazdan hazırlayacağımız uygulamada zoom yaparak tüm detayları görebilecek. Yani bir insan resmi koyup gözünün içine de ufak görünenecek şekilde aslında 5MB'luk kocaman bir fotoğraf yerleştirebilirsınız. Kullanıcı resmi ilk açtığında bu fotoğraf insanın gözünün içinde ufak olduğu için tamamen yüklenmeyecek ve sadece gözüktüğü kadarı istemciye gönderilecektir. Oysa kullanıcı zoom yaparak gözün içine girip büyük fotoğrafı görmeye başladığında ise detaylar yüklenerek fotoğrafınız tüm çözünürlüğü ile net bir şekilde gözükecektir. DeepZoom Composer'in kullanımı ile ilgili detaylı bilgiyi sevgili [Turhal Temizer'in yazısından](#) edinebilirsınız.



DeepZoom Composer içerisinde fotoğraflarımız.

Biz uygulamamızda yukarıdaki gibi fotoğrafları sahneye ekleyip sağ tuş ile gelen menüden de tüm fotoğrafları **"Arrange to Grid"** diyerek bir tablo içerisindeyim gibi hizalatalım. Böylece tüm fotoğraflar yan yana ve alt alta sıralanacaktır. Daha önce de bahsettiğim gibi isterseniz fotoğrafları üst üste yerleştirme ve farklı tasarımlar yapma şansınız da var. Örneğin bir mekanın duvarındaki reklamı ayrı bir detaylı fotoğraf olarak koyabilirsiniz. Böylece kullanıcı mekan fotoğrafındaki duvarda yer alan reklama zoom yaptığında aslında daha detaylı farklı bir görsel yüklenmeye başlayacak ve normalde elde edilemeyen bir detay gösterimi yapılabilicektir.

Son olarak DeepZoom Composer içerisinde **"Export"** bölümüne geçerek artık fotoğraflarımızın gerekli çıktılarını almak istiyoruz. Export esnasında özellikle **"Output Type"** olarak **"Export Images"** seçeneğini işaretlemeyi unutmayın. Normalde DeepZoomComposer isterseniz size hazır bir Silverlight projesi de yaratıyor fakat bu yazımızda biz kendi uygulamamızı hazırlayacağımız için sadece fotoğrafların düzenlenerek gerekli arşivin oluşturulmasını istiyoruz.



DeepZoom Composer arşivimiz.

Yukarıda da gördüğünüz üzere gerekli dosyalar bizim için hazırlanmış. Buradaki XML dosyalarını tek tek inceleyebilirsiniz. Aslında her bir dosya ayrı birer XML dosyasına yönlendiriyor bizi. Arşive eklediğimiz her resim için ayrıca birer XML dosyası yaratılıyor ve bu dosyalar içerisinde resimlerle ilgili konum bilgisi gibi detaylar yer alıyor. Ayrıca her resmimizin farklı boyutlarda kopyaları da klasörler içerisinde kopyalanmış durumda. Hatta çok büyük resimler parçalara bölünerek ayrı ayrı dosyalar olarak da kaydedilmiş.

MultiScaleImage kontrolünün de gücü zaten buradan geliyor. Kaynaktaki farklı boyuttaki ve parçalardaki fotoğrafları gerçek zamanlı olarak birleştirebiliyor ve otomatik olarak ekranda gözüken detaya uygun hedef dosyaları istemciye yükliyor. Tüm geçiş efektlerini de tabii ki otomatik olarak yapıyor. Peki artık kaynak dosyalarımız hazır olduğuna göre uygulamamızı da hazırlamaya başlayalım.

Silverlight projemizi yaratalım...

Yeni bir Silverlight projesi yaratarak hemen içerisinde bir **MultiScaleImage** kontrolü yerleştiriyoruz. Şimdilik tasarım anlamında farklı birşey yapmayacağız. Uygulamamızın XAML kodu aşağıdaki gibi basit bir şekilde sonlanıyor.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="1024" Height="768">
    <Grid x:Name="LayoutRoot" Background="White">
        <MultiScaleImage x:Name="DeepZoom"/>
    </Grid>
</UserControl>
```

DeepZoom adını verdığım **MultiScaleImage** kontrolümüzü programlamaya başlayacağız. İlk olarak yapmak istediğimiz işlemleri bir sıralayalım;

- Fotoğraf arşivimizi kontrole yükleyeceğiz.
- Fare ile fotoğraf arşivi içerisinde gezilebilmesini sağlayacağız
- Farenin roller'i ile zoom ve zoom out yapılabilmesini sağlayacağız
- Artistik hareketler yapacağız

Listemizdeki tek tek yaparak ilerleyelim. İlk olarak fotoğraf arşivimizi doğrudan kontrolümüze bağlayalım.

[VB]

```
DeepZoom.Source = New DeepZoomImageTileSource(New
Uri("GeneratedImages/dzc_output.xml", UriKind.Relative))
```

[C#]

```
DeepZoom.Source = new DeepZoomImageTileSource(new
Uri("GeneratedImages/dzc_output.xml", UriKind.Relative));
```

Gördüğünüz üzere doğrudan **DeepZoom** kontrolümün **Source** özelliğini değiştiriyyorum ve yarattığımız yeni bir **DeepZoomImageTileSource** nesnesini kendisine atıyoruz. Bu esnada bir önceki adımda DeepZoom Composer'in bizim için yarattığı arşivden **dzc_output.xml** dosyasını kaynak olarak gösteriyoruz. DeepZoom Composer'in yarattığı **GeneratedImages** klasörünü doğru olarak çalışabilmesi için XAP dosyanız ile aynı konuma kopyalamanız gerekecektir.

Fotoğraf arşivini gezelim.

Sıra geldi fare ile fotoğrafların arasındaki gezintimize başlamaya. Aslında yapacağımız işlem klasik bir Sürükle&Bırak işleminden farklı değil. Fakat bu sefer sürükleyip bırakıracığımız şey aslında **MultiScaleImage** kontrolünün **ViewPortOrigin'i**. Yani bizim bu arşive **MultiScaleImage** aracılığı ile bakış noktamızı değiştireceğiz.

[VB]

```
Dim Tasinıyor = False
Dim FareKonum As Point
Dim DeepZoomOrigin As Point
```

[C#]

```
bool Tasinıyor = false;
Point FareKonum;
Point DeepZoomOrigin;
```

Sürükle ve bırak işlemi öncesinde ihtiyacımız olacak global değişkenlerimizi yukarıdaki gibi yaratalım. Sonrasında toplam üç farklı event için kod yazmamız gerekiyor.

MouseLeftButtonDown, **MouseMove** ve **MouseLeftButtonUp** durumlarında sürükleme işlemini başlatıp durdurmaya karar vereceğiz.

[VB]

```
Private Sub Page_MouseLeftButtonDown(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Me.MouseLeftButtonDown
    Tasiniyor = True
    FareKonum = e.GetPosition(Me)
    DeepZoomOrigin = DeepZoom.ViewportOrigin
End Sub
```

[C#]

```
void Page_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    Tasiniyor = true;
    FareKonum = e.GetPosition(this);
    DeepZoomOrigin = DeepZoom.ViewportOrigin;
}
```

Kullanıcı fare ile sahneye tıkladığında hemen taşınma işlemini başlattığımız için global **Tasiniyor** değişkenini True yapıyoruz. Böylece ileride yazacağımız **MouseMove** kodu durumdan haberdar olabilecek. Bir sonraki adımda farenin pozisyonunu ve sonrasında da DeepZoom kontrolünün o anki orijin noktasını kenara not alıyoruz. Bu bilgileri ileriki hesaplamalarımızda kullanacağız.

[VB]

```
Private Sub Page_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventHandler) Handles Me.MouseMove
    If Tasiniyor Then
        Dim YeniDeepZoomOrigin As New Point
        YeniDeepZoomOrigin.X = DeepZoomOrigin.X - (((e.GetPosition(DeepZoom).X - FareKonum.X) / DeepZoom.ActualWidth) * DeepZoom.ViewportWidth)
        YeniDeepZoomOrigin.Y = DeepZoomOrigin.Y - (((e.GetPosition(DeepZoom).Y - FareKonum.Y) / DeepZoom.ActualHeight) * DeepZoom.ViewportWidth)
        DeepZoom.ViewportOrigin = YeniDeepZoomOrigin
    Else
        FareKonum = e.GetPosition(Me)
    End If
End Sub
```

[C#]

```
void Page_MouseMove(object sender, MouseEventArgs e)
{
    if (Tasiniyor)
    {
        Point YeniDeepZoomOrigin = new Point();
        YeniDeepZoomOrigin.X = DeepZoomOrigin.X - (((e.GetPosition(DeepZoom).X - FareKonum.X) / DeepZoom.ActualWidth) * DeepZoom.ViewportWidth);
        YeniDeepZoomOrigin.Y = DeepZoomOrigin.Y - (((e.GetPosition(DeepZoom).Y - FareKonum.Y) / DeepZoom.ActualHeight) * DeepZoom.ViewportWidth);
        DeepZoom.ViewportOrigin = YeniDeepZoomOrigin;
```

```
        }  
    else  
    {  
        FareKonum = e.GetPosition(this);  
    }  
}
```

Sahnenin **MouseMove** durumunda ilk olarak bir taşıma işlemi olup olmadığını kontrol ediyoruz. Eğer taşıma işlemi yoksa **FareKonum** adındaki ve farenin mevcut konumunu saklayan değişkeni yeniliyoruz. Bunu yapmamızın nedeni ileride yazacağımız zoom işlemleri. Zoom işlemini yaparken farenin konumuna hep ihtiyacımız olacak. Burada sürekli güncel konumu **FareKonum'a** aktarmak gerekiyor. Eğer taşıma işlemi yapılıyorsa bu sefer de DeepZoom kontrolümüzün ViewPortOrigin'ini uygun şekilde ayarlamamız gereklidir. Bu amaçla kodumuzda yeni bir **Point** değişkeni tanımlayarak farenin mevcut konumu, eski konumu ve eski orijin noktası arasında koordinat hesaplamaları yaparak ilerliyoruz.

[VB]

```
Private Sub Page_MouseLeftButtonUp(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Me.MouseLeftButtonUp
    Tasiniyor = False
End Sub
```

[C#]

```
void Page_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    Tasiniyor = false;
}
```

Son olarak farenin sol tuşu bırakıldığında da taşıma işlemini sonlandırıyoruz. Böylece artık fotoğraf arşivi içerisinde kullanıcılar rahatlıkla gezebilecektir. Sıra geldi zoom meselesine.

Fare ile Zoom-In ve Zoom-Out

Fare ile zoom işlemleri için farenin roller'ını kullanacağız. Detaylar için Silverlight 2.0 içerisinde fare roller'i kullanmayla ilgili [buradaki](#) makleyi inceleyebilirsiniz. İlk olarak DOM üzerinden gerekli eventları Silverlight ile yakalamamız gerekiyor.

[VB]

```
System.Windows.Browser.HtmlPage.Window.AttachEvent("DOMMouseScroll",  
AddressOf FareTekerlekDondu)  
System.Windows.Browser.HtmlPage.Window.AttachEvent("onmousewheel",  
AddressOf FareTekerlekDondu)  
System.Windows.Browser.HtmlPage.Document.AttachEvent("onmousewheel",  
AddressOf FareTekerlekDondu)
```

[C#]

```

System.Windows.Browser.HtmlPage.Window.AttachEvent("DOMMouseScroll",
FareTekerlekDondu);
System.Windows.Browser.HtmlPage.Window.AttachEvent("onmousewheel",
FareTekerlekDondu);
System.Windows.Browser.HtmlPage.Document.AttachEvent("onmousewheel",
FareTekerlekDondu);

```

FareTekerlekDondu eventi içerisinde aşağıda kodları yazmamız gerekiyor. Bu kodların çok detayına inmeyeceğiz çünkü bir önceki paragrafta bahsettiğim makalede zaten bu konu detayları ile inceleniyor. Bizim için bu kodda önemli olan kısım Zoom işleminin yapıldığı satırlar.

[VB]

```

Private Sub FareTekerlekDondu(ByVal sender As Object, ByVal e As
System.Windows.Browser.HtmlEventArgs)
    Dim DonMiktar As Integer = 0
    Dim Gelen As System.Windows.Browser.ScriptObject = e.EventObject
    'IE ve OPERA
    If Not Gelen.GetProperty("wheelDelta") Is Nothing Then
        'OPERA'da ters!
        If Not Gelen.GetProperty("opera") Is Nothing Then
            DonMiktar = -DonMiktar
        End If
        DonMiktar = Gelen.GetProperty("wheelDelta")
        'Mozilla ve Safari
    ElseIf Not Gelen.GetProperty("detail") Is Nothing Then
        DonMiktar = -Gelen.GetProperty("detail")
    End If
    If DonMiktar > 0 Then
        'Zoom yap
        Dim ZoomlananNokta As Point =
DeepZoom.ElementToLogicalPoint(FareKonum)
        DeepZoom.ZoomAboutLogicalPoint(2, ZoomlananNokta.X, ZoomlananNokta.Y)
    Else
        'Uzaklaş
        Dim ZoomlananNokta As Point =
DeepZoom.ElementToLogicalPoint(FareKonum)
        DeepZoom.ZoomAboutLogicalPoint(0.5, ZoomlananNokta.X,
ZoomlananNokta.Y)
    End If
    If DonMiktar <> 0 Then
        e.PreventDefault()
        Gelen SetProperty("returnValue", False)
    End If
End Sub

```

[C#]

```

private void FareTekerlekDondu(object sender,
System.Windows.Browser.HtmlEventArgs e)
{
    int DonMiktar = 0;
    System.Windows.Browser.ScriptObject Gelen = e.EventObject;
    //IE ve OPERA
    if ((Gelen.GetProperty("wheelDelta") != null)) {
        //OPERA'da ters!
        if ((Gelen.GetProperty("opera") != null)) {
            DonMiktar = -DonMiktar;
        }
        DonMiktar = (int)Gelen.GetProperty("wheelDelta");
    }
    //Mozilla ve Safari
    else if ((Gelen.GetProperty("detail") != null)) {
        DonMiktar = -1 * (int)Gelen.GetProperty("detail");
    }
    if (DonMiktar > 0) {
        //Zoom yap
        Point ZoomlananNokta = DeepZoom.ElementToLogicalPoint(FareKonum);
        DeepZoom.ZoomAboutLogicalPoint(2, ZoomlananNokta.X,
ZoomlananNokta.Y);
    }
    else {
        //Uzaklaş
        Point ZoomlananNokta = DeepZoom.ElementToLogicalPoint(FareKonum);
        DeepZoom.ZoomAboutLogicalPoint(0.5, ZoomlananNokta.X,
ZoomlananNokta.Y);
    }

    if (DonMiktar != 0) {
        e.PreventDefault();
        Gelen SetProperty("returnValue", false);
    }
}

```

İlk olarak **ElementToLogicalPoint** metodu ile elimizdeki fare konumunun **DeepZoom** içerisinde tam olarak hangi koordinatlara geldiğini buluyoruz. Sonrasında da **ZoomAboutLogicalPoint** metodu ile zoom işlemini yapıyoruz. Zoom işlemi esnasında bizden üç parametre isteniyor, zoom miktarı, zoomlanacak noktanın X ve Y koordinatları. Böylece basit bir şekilde zoom işlemini de çözmüş olduk.

Atraksyon zamanı!

Yapılacaklar listemizde son bir öğe kaldı :) "Atraksyon". Şimdi biraz hareketli birşeyler yapalım. Eğer uygulamanın başından beridir benimle aynı adımları takip ediyorsunuz şu an DeepZoom kontrolü içerisinde yan yana ve alt alta sıralı onlarca fotoğrafınız var demektir. Peki bu fotoğrafları rastgele karıştırın bir düğme eklesek? Animasyonlarla fotoğraf düğmeye her basıldığında rastgele olarak yerlerini değiştirseler hoş olmaz mıydı? Belki de sizin

istediğiniz farklı sıralara bile gelebilirler. Hemen kolları sıvayalım ve XAML'ımıza basit bir düğme ekleyelim.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="1024" Height="768">
    <Grid x:Name="LayoutRoot" Background="White">
        <MultiScaleImage x:Name="DeepZoom"/>
        <Button Height="65" HorizontalAlignment="Right" Margin="0,0,30,19"
    VerticalAlignment="Bottom" Width="174" Content="Karistis" x:Name="Karistir"/>
    </Grid>
</UserControl>
```

Artık uygulamamızın XAML kodu yukarıdaki şekilde olacak. Düğmeye tıklandığında DeepZoom kontrolü içerisindeki tüm resimleri alarak sırasını karıştırmamız sonra da animasyonlarla yeni konumlara yerleştirmemiz gerek. İlk olarak resimleri karıştıracak olan kodu yazalım.

[VB]

```
Dim FotoList As New List(Of MultiScaleSubImage)
FotoList = DeepZoom.SubImages.ToList()

For x As Integer = 0 To FotoList.Count - 1
    Dim Simdiki As MultiScaleSubImage = FotoList(x)
    FotoList.RemoveAt(x)
    FotoList.Insert(Rnd() * FotoList.Count, Simdiki)
Next
```

[C#]

```
List<MultiScaleSubImage> FotoList = new List<MultiScaleSubImage>();
FotoList = DeepZoom.SubImages.ToList();
Random Rastgele = new Random();

for (int x = 0; x <= FotoList.Count - 1; x++)
{
    MultiScaleSubImage Simdiki = FotoList[x];
    FotoList.RemoveAt(x);
    FotoList.Insert(Rastgele.Next(FotoList.Count), Simdiki);
}
```

Bir **DeepZoom** içerisinde tüm resimler birer **MultiScaleSubImage** olarak bulunuyor. DeepZoom içerisinde tüm listeyi alıp içerisindeki her resmi listeden çıkartıp yeni bir rastgele indeks ile ekliyoruz. Böylece her seferinde sıralamayı rastgele değiştirmiş olduk. Sıra geldi bu sıralama ile resimleri sahnedeki yeni konumlarına birer animasyon ile göndermeye.

[VB]

```

Dim KolonSayisi As Integer = 5
Dim SatirSayisi As Integer = 5
Dim ToplamEklenen As Integer = 0

For Satir As Integer = 0 To SatirSayisi
    For Kolon As Integer = 0 To KolonSayisi
        If ToplamEklenen <> FotoList.Count Then
            .....
            ToplamEklenen += 1
        Else
            Exit Sub
        End If
    Next
Next

```

[C#]

```

int KolonSayisi = 5;
int SatirSayisi = 5;
int ToplamEklenen = 0;

for (int Satir = 0; Satir <= SatirSayisi; Satir++)
{
    for (int Kolon = 0; Kolon <= KolonSayisi; Kolon++)
    {
        if (ToplamEklenen != FotoList.Count)
        {
            .....
            ToplamEklenen += 1;
        }
        else
        {
            break;
        }
    }
}

```

Yukarıdaki kodumuzun orta kısmında birazdan her resmi farklı bir konuma animasyon ile gönderen kodu yazacağız. Fakat onun öncesinde kodun ana yapısını bir inceleyelim. Resimler yan yana ve alt alta sıralayacağımız için aslında bir Grid yapısında görsellik yaratmış olacağız. Kodumuzda **KolonSayısı** ve **SatırSayısı** değişkenleri kaç kolon ve satırlık bir sıralama yapılacağını belirliyor. ToplamEklenen değişkeni ise o ana kadar kaç fotoğraf eklediğimizi hafızada tutacak, böylece döngü içerisinde eklenen toplam fotoğraf sayısı elimizdeki fotoğraf sayısına ulaşrsa döngüden çıkışcağız. Şimdi gelelim fotoğrafları yeni konumlarına gönderecek animasyonları yaratacak kodumuza.

[VB]

```

Dim Foto As MultiScaleSubImage = FotoList(ToplamEklenen)
Dim MevcutKonum As Point = Foto.ViewportOrigin
Dim HedefKonum As Point = New Point(-1.14 * Kolon, -0.8 * Satir)

'Animasyonu yaratalım
Dim Anim As New Storyboard
Dim NoktaAnim As New PointAnimationUsingKeyFrames
Dim KeyFrame As New SplinePointKeyFrame
KeyFrame.Value = HedefKonum
KeyFrame.KeyTime = KeyTime.FromTimeSpan(TimeSpan.FromSeconds(1))

Dim Ivme As New KeySpline
Ivme.ControlPoint1 = New Point(0, 1)
Ivme.ControlPoint2 = New Point(1, 1)
KeyFrame.KeySpline = Ivme
NoktaAnim.KeyFrames.Add(KeyFrame)

Storyboard.SetTarget(NoktaAnim, Foto)
Storyboard.SetTargetProperty(NoktaAnim, New PropertyPath("ViewportOrigin"))

Anim.Children.Add(NoktaAnim)

Anim.Begin()

```

[C#]

```

MultiScaleSubImage Foto = FotoList[ToplamEklenen];
Point MevcutKonum = Foto.ViewportOrigin;
Point HedefKonum = new Point(-1.14 * Kolon, -0.8 * Satir);

//Animasyonu yaratalım
Storyboard Anim = new Storyboard();
PointAnimationUsingKeyFrames NoktaAnim = new PointAnimationUsingKeyFrames();
SplinePointKeyFrame KeyFrame = new SplinePointKeyFrame();
KeyFrame.Value = HedefKonum;
KeyFrame.KeyTime = KeyTime.FromTimeSpan(TimeSpan.FromSeconds(1));

KeySpline Ivme = new KeySpline();
Ivme.ControlPoint1 = new Point(0, 1);
Ivme.ControlPoint2 = new Point(1, 1);
KeyFrame.KeySpline = Ivme;
NoktaAnim.KeyFrames.Add(KeyFrame);

Storyboard.SetTarget(NoktaAnim, Foto);
Storyboard.SetTargetProperty(NoktaAnim, new PropertyPath("ViewportOrigin"));

Anim.Children.Add(NoktaAnim);

```

```
Anim.Begin();
```

Kodumuz biraz uzun gibi gözükse de aslında özünde yaptığımız şey çok basit. İlk olarak döngümüzle oluşturduğumuz **ToplamEklenen** sayısı üzerinden o anki resmi **FotoList** içerisinde bir değişkene alıyoruz. **MevcutKonum** ve **HedefKonum** değişkenlerimiz ise fotoğrafın şu anki ve animasyonun sonundaki konumlarını saklıyor. Yeni konum hesaplarken kolon ve satır arası mesafelerle sayıları çarparak yeni konumu rahatlıkla hesaplayabiliyoruz. Artık hedeflediğimiz konum da belli olduğuna göre geriye kaldı eldeki fotoğrafı hedef konuma taşıyacak animasyonu yaratmak. Animasyon yapacağımız şey fotoğrafın **ViewportOrigin** özelliği ve bu özelliği **Point** tipinde. Bu nedenle **PointAnimationUsingKeyFrame** kullanarak hedef **KeyFrame'i** yaratacağız. Kodumuzda baktığımızda **KeyFrame** adındaki değişkenimiz 1 saniye sonra **ViewPortOrigin** değerini değiştiriyor. Ayrıca animasyonu ivme vermek için bir de KeySpline kullanıyoruz. Tüm bu nesnelerin birbirlerine eklenme şekilleri XAML'da bir StoryBoard yaratmaktan farklı değil. Kodumuzun en sonunda artık animasyonumuza çalıştırmak için Begin metodunu çağırabiliriz.

Herşey bitti

Uygulamamız bitti. Artık istediğimiz kadar fotoğraflarımız arasında geçebilir, zoom yapabilir hatta resimleri bir düğme ile animasyonlu bir şekilde karıştırabiliriz. Uygulamanın tam kodunu aşağıda inceleyebilirsiniz.

[VB]

```
Partial Public Class Page
```

```
Inherits UserControl
```

```
Public Sub New()
```

```
    InitializeComponent()
```

```
End Sub
```

```
Dim Tasiniyor = False
```

```
Dim FareKonum As Point
```

```
Dim DeepZoomOrigin As Point
```

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
```

```
'Kaynağımızı bağlayalım
```

```
    DeepZoom.Source = New DeepZoomImageTileSource(New Uri("NewFolder1/dzc_output.xml", UriKind.Relative))
```

```
'MouseWheel bağlantısı yapalım
```

```
    System.Windows.Browser.HtmlPage.Window.AttachEvent("DOMMouseScroll", AddressOf FareTekerlekDondu)
```

```
    System.Windows.Browser.HtmlPage.Window.AttachEvent("onmousewheel", AddressOf FareTekerlekDondu)
```

```
    System.Windows.Browser.HtmlPage.Document.AttachEvent("onmousewheel", AddressOf FareTekerlekDondu)
```

```
End Sub
```

```

Private Sub FareTekerlekDondu(ByVal sender As Object, ByVal e As
System.Windows.Browser.HtmlEventArgs)
    Dim DonMiktar As Integer = 0
    Dim Gelen As System.Windows.Browser.ScriptObject = e.EventObject
    'IE ve OPERA
    If Not Gelen.GetProperty("wheelDelta") Is Nothing Then
        'OPERA'da ters!
        If Not Gelen.GetProperty("opera") Is Nothing Then
            DonMiktar = -DonMiktar
        End If
        DonMiktar = Gelen.GetProperty("wheelDelta")
        'Mozilla ve Safari
    ElseIf Not Gelen.GetProperty("detail") Is Nothing Then
        DonMiktar = -Gelen.GetProperty("detail")
    End If
    If DonMiktar > 0 Then
        'Zoom yap
        Dim ZoomlananNokta As Point = DeepZoom.ElementToLogicalPoint(FareKonum)
        DeepZoom.ZoomAboutLogicalPoint(2, ZoomlananNokta.X, ZoomlananNokta.Y)
    Else
        'Uzaklaş
        Dim ZoomlananNokta As Point = DeepZoom.ElementToLogicalPoint(FareKonum)
        DeepZoom.ZoomAboutLogicalPoint(0.5, ZoomlananNokta.X, ZoomlananNokta.Y)
    End If
    If DonMiktar <> 0 Then
        e.PreventDefault()
        Gelen SetProperty("returnValue", False)
    End If
End Sub

Private Sub Page_MouseLeftButtonDown(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseButtonEventArgs) Handles Me.MouseLeftButtonDown
    Tasiniyor = True
    FareKonum = eGetPosition(Me)
    DeepZoomOrigin = DeepZoom.ViewportOrigin
End Sub

Private Sub Page_MouseLeftButtonUp(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseButtonEventArgs) Handles Me.MouseLeftButtonUp
    Tasiniyor = False
End Sub

Private Sub PageMouseMove(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventHandler) Handles Me.MouseMove
    If Tasiniyor Then
        Dim YeniDeepZoomOrigin As New Point
        YeniDeepZoomOrigin.X = DeepZoomOrigin.X - (((eGetPosition(DeepZoom).X -
FareKonum.X) / DeepZoom.ActualWidth) * DeepZoom.ViewportWidth)
    End If
End Sub

```

```

YeniDeepZoomOrigin.Y = DeepZoomOrigin.Y - (((e.GetPosition(DeepZoom).Y -
FareKonum.Y) / DeepZoom.ActualHeight) * DeepZoom.ViewportWidth)
DeepZoom.ViewportOrigin = YeniDeepZoomOrigin
Else
    FareKonum = e.GetPosition(Me)
End If
End Sub

Private Sub Karistir_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Karistir.Click
    Dim FotoList As New List(Of MultiScaleSubImage)
    FotoList = DeepZoom.SubImages.ToList()

    For x As Integer = 0 To FotoList.Count - 1
        Dim Simdiki As MultiScaleSubImage = FotoList(x)
        FotoList.RemoveAt(x)
        FotoList.Insert(Rnd() * FotoList.Count, Simdiki)
    Next

    Dim KolonSayisi As Integer = 5
    Dim SatirSayisi As Integer = 5
    Dim ToplamEklenen As Integer = 0

    For Satir As Integer = 0 To SatirSayisi
        For Kolon As Integer = 0 To KolonSayisi
            If ToplamEklenen <> FotoList.Count Then
                Dim Foto As MultiScaleSubImage = FotoList(ToplamaEklenen)
                Dim MevcutKonum As Point = Foto.ViewportOrigin
                Dim HedefKonum As Point = New Point(-1.14 * Kolon, -0.8 * Satir)

                'Animasyonu yaratalım
                Dim Anim As New Storyboard
                Dim NoktaAnim As New PointAnimationUsingKeyFrames
                Dim KeyFrame As New SplinePointKeyFrame
                KeyFrame.Value = HedefKonum
                KeyFrame.KeyTime = KeyTime.FromTimeSpan(TimeSpan.FromSeconds(1))

                Dim Ivme As New KeySpline
                Ivme.ControlPoint1 = New Point(0, 1)
                Ivme.ControlPoint2 = New Point(1, 1)
                KeyFrame.KeySpline = Ivme
                NoktaAnim.KeyFrames.Add(KeyFrame)

                Storyboard.SetTarget(NoktaAnim, Foto)
                Storyboard.SetTargetProperty(NoktaAnim, New
PropertyPath("ViewportOrigin"))

                Anim.Children.Add(NoktaAnim)

                Anim.Begin()
            End If
        Next
    Next
End Sub

```

```

    ToplamEklenen += 1
Else
    Exit Sub
End If
Next
Next
End Sub
End Class

```

[C#]

```

namespace SilverlightApplication1
{
    public partial class Page : UserControl
    {
        bool Tasiniyor = false;
        Point FareKonum;
        Point DeepZoomOrigin;

        public Page()
        {
            InitializeComponent();
            this.Loaded += new RoutedEventHandler(Page_Loaded);
            this.MouseLeftButtonDown += new
MouseEventHandler(Page_MouseLeftButtonDown);
            this.MouseLeftButtonUp += new
MouseEventHandler(Page_MouseLeftButtonUp);
            this.MouseMove += new MouseEventHandler(Page_MouseMove);
            this.Karistir.Click += new RoutedEventHandler(Karistir_Click);
        }

        void Karistir_Click(object sender, RoutedEventArgs e)
        {
            List<MultiScaleSubImage> FotoList = new List<MultiScaleSubImage>();
            FotoList = DeepZoom.SubImages.ToList();
            Random Rastgele = new Random();

            for (int x = 0; x <= FotoList.Count - 1; x++)
            {
                MultiScaleSubImage Simdiki = FotoList[x];
                FotoList.RemoveAt(x);
                FotoList.Insert(Rastgele.Next(FotoList.Count), Simdiki);
            }

            int KolonSayisi = 5;
            int SatirSayisi = 5;
            int ToplamEklenen = 0;

            for (int Satir = 0; Satir <= SatirSayisi; Satir++)

```

```

{
    for (int Kolon = 0; Kolon <= KolonSayisi; Kolon++)
    {
        if (ToplamEklenen != FotoList.Count)
        {
            MultiScaleSubImage Foto = FotoList[ToplamEklenen];
            Point MevcutKonum = Foto.ViewportOrigin;
            Point HedefKonum = new Point(-1.14 * Kolon, -0.8 * Satir);

            //Animasyonu yaratalım
            Storyboard Anim = new Storyboard();
            PointAnimationUsingKeyFrames NoktaAnim = new
            PointAnimationUsingKeyFrames();
            SplinePointKeyFrame KeyFrame = new SplinePointKeyFrame();
            KeyFrame.Value = HedefKonum;
            KeyFrame.KeyTime = KeyTime.FromTimeSpan(TimeSpan.FromSeconds(1));

            KeySpline Ivme = new KeySpline();
            Ivme.ControlPoint1 = new Point(0, 1);
            Ivme.ControlPoint2 = new Point(1, 1);
            KeyFrame.KeySpline = Ivme;
            NoktaAnim.KeyFrames.Add(KeyFrame);

            Storyboard.SetTarget(NoktaAnim, Foto);
            Storyboard.SetTargetProperty(NoktaAnim, new
            PropertyPath("ViewportOrigin"));

            Anim.Children.Add(NoktaAnim);

            Anim.Begin();

            ToplamEklenen += 1;
        }
        else
        {
            break;
        }
    }
}

void Page_MouseMove(object sender, MouseEventArgs e)
{
    if (Tasiniyor)
    {
        Point YeniDeepZoomOrigin = new Point();
        YeniDeepZoomOrigin.X = DeepZoomOrigin.X - (((e.GetPosition(DeepZoom).X -
FareKonum.X) / DeepZoom.ActualWidth) * DeepZoom.ViewportWidth);
    }
}

```

```

YeniDeepZoomOrigin.Y = DeepZoomOrigin.Y - (((e.GetPosition(DeepZoom).Y -
FareKonum.Y) / DeepZoom.ActualHeight) * DeepZoom.ViewportWidth);
DeepZoom.ViewportOrigin = YeniDeepZoomOrigin;
}

else
{
    FareKonum = e.GetPosition(this);
}
}

void Page_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    Tasiniyor = false;
}

void Page_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    Tasiniyor = true;
    FareKonum = e.GetPosition(this);
    DeepZoomOrigin = DeepZoom.ViewportOrigin;
}

void Page_Loaded(object sender, RoutedEventArgs e)
{
    //Kaynağımızı bağlayalım
    DeepZoom.Source = new DeepZoomImageTileSource(new
Uri("NewFolder1/dzc_output.xml", UriKind.Relative));

    //MouseWheel bağlantı yapalım
    System.Windows.Browser.HtmlPage.Window.AttachEvent("DOMMouseScroll",
FareTekerlekDondu);
    System.Windows.Browser.HtmlPage.Window.AttachEvent("onmousewheel",
FareTekerlekDondu);
    System.Windows.Browser.HtmlPage.Document.AttachEvent("onmousewheel",
FareTekerlekDondu);
}

private void FareTekerlekDondu(object sender,
System.Windows.Browser.HtmlEventArgs e)
{
    int DonMiktar = 0;
    System.Windows.Browser.ScriptObject Gelen = e.EventObject;
    //IE ve OPERA
    if ((Gelen.GetProperty("wheelDelta") != null)) {
        //OPERA'da ters!
        if ((Gelen.GetProperty("opera") != null)) {
            DonMiktar = -DonMiktar;
        }
        DonMiktar = (int)Gelen.GetProperty("wheelDelta");
    }
}

```

```
//Mozilla ve Safari
else if ((Gelen.GetProperty("detail") != null)) {
    DonMiktar = -1 * (int)Gelen.GetProperty("detail");
}
if (DonMiktar > 0) {
    //Zoom yap
    Point ZoomlananNokta = DeepZoom.ElementToLogicalPoint(FareKonum);
    DeepZoom.ZoomAboutLogicalPoint(2, ZoomlananNokta.X, ZoomlananNokta.Y);
}
else {
    //Uzaklaş
    Point ZoomlananNokta = DeepZoom.ElementToLogicalPoint(FareKonum);
    DeepZoom.ZoomAboutLogicalPoint(0.5, ZoomlananNokta.X,
ZoomlananNokta.Y);
}

if (DonMiktar != 0) {
    e.PreventDefault();
    Gelen SetProperty("returnValue", false);
}
}
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Open File Dialog kullanımı

Silverlight 2.0 Beta 2 içerisinde normal şartlarda uygulamanın çalıştığı bilgisayardaki dosyalara ulaşma şansınız olmaz. Bu durum aslında çok normal bir durum çünkü Silverlight tamamen internet tarayıcısi içerisinde çalışıyor ve dışarıya çıkarak sisteme ulaşması büyük bir güvenlik açığı olurdu. Oysa bizim bazı durumlarda Silverlight tarafına sistemimizdeki bir dosyayı aktarmak isteyebiliriz. Örneğin sistemdeki bir video dosyasını kullanıcının web sitesindeki Silverlight ile açarak oynatmasını isteyebiliriz. Özellikle bu Silverlight uygulamasının Macintosh tarafında da çalışacağını düşünürsek aslında Mac için basit bir WMV Player bile yapmış oluyoruz. Peki sistemdeki dosyalara nasıl ulaşacağız? Tabi ki "Open File Dialog" aracılığı ile.

Uygulamamızı tasarlayalım.

İlk olarak istemci tarafından video dosyalarını oynatacak olan uygulamamızın ana ekran tasarımını yapalım. Sahnede adı Video olan bir MediaElement ve bir de Button bulunacak. Düğmeye basıldığında Open File Dialog aracılığı ile istemcide video dosyalarından birini alacağız.

```
<UserControl x:Class="SilverlightApplication55.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <MediaElement Margin="24,8,128,84" x:Name="Video"/>
        <Button Height="24" HorizontalAlignment="Stretch" Margin="192,0,128,44"
    VerticalAlignment="Bottom" Content="Dosya Aç" x:Name="btnAc"
    Click="btnAc_Click"/>
    </Grid>
</UserControl>
```

Kodlama zamanı

Düğmemize basıldığında ilk olarak yeni bir Open File Dialog yaratacağız sonrasında da nesnemizin özelliklerini tanımlayacağız.

[VB]

```
Dim DosyaAc As New OpenFileDialog
DosyaAc.Filter = "WMV Dosyaları (*.wmv)|*.wmv"
DosyaAc.Multiselect = False
DosyaAc.ShowDialog()
```

[C#]

```
 OpenFileDialog DosyaAc = new OpenFileDialog();
DosyaAc.Filter = "WMV Dosyaları (*.wmv)|*.wmv";
DosyaAc.Multiselect = false;
```

```
DosyaAc.ShowDialog();
```

Kod içerisinde Open File Dialog'un ilk olarak **Filter** özelliğini tanımlıyoruz. Aslında buradaki kullanım Windows Forms'daki kullanım ile neredeyse aynı. Bizim örneğimizdeki filtrede kullanıcıya sadece WMV dosyalarını seçtiyoruz. Bir sonraki adımda Open File Dialog nesnemizin **Multiselect** özelliğinde **False** yaparak bir defada sadece bir dosya seçilebilmesini de sağladıkten sonra artık Dialog'u ekranda göstermek için **ShowDialog** metodunu çağırabiliriz.

[VB]

```
If DosyaAc.SelectedFile IsNot Nothing Then
    Video.AutoPlay = True
    Video.SetSource(DosyaAc.SelectedFile.OpenRead)
End If
```

[C#]

```
if (DosyaAc.SelectedFile != null) {
    Video.AutoPlay = true;
    Video.SetSource(DosyaAc.SelectedFile.OpenRead);
}
```

Kullanıcı Open File Dialog'u kapattıktan sonra herhangi bir dosyanın seçilip seçilmediğini **SelectedFile** özelliği üzerinden kontrol ederek eğer dosya seçilmiş ise **OpenRead** metodu ile dosyayı okuyabiliyoruz. Biz okuduğumuz dosyayı MediaElement'in **Source**'una aktaracağımız için ilk olarak aktarma sonrası hemen video oynatılsın diye MediaElement'in **AutoPlay** özelliğini **True** yapıyoruz. Son olarak da **SetSource** metoduna elimizdeki dosyayı aktarıyoruz. Böylece artık video dosyası **MediaElement** içerisinde oynatılıyor olacak. Unutmayın videoyu sunucuya almadık, hala tamamen istemci tarafında çalışıyoruz.

Birden çok dosya okuyalım

Open File Dialog'un **MultiSelect** özelliği **True** olduğunda birden çok dosya seçilerek işlem yapılabiliyor. Bu konudaki örneğimizde sahnede bir TextBox ve bir düğme bulunacak. Düğmeye tıklandığında sistemden kullanıcıların istedikleri kadar fazla sayıda TXT dosyası seçebilecekler. Bu dosyaların hepsinin içeriğini arka arkaya TextBox içerisinde yerlestireceğiz.

[VB]

```
Dim DosyaAc As New OpenFileDialog
DosyaAc.Filter = "TXT Dosyaları (*.txt)|*.txt"
DosyaAc.Multiselect = True
DosyaAc.ShowDialog()
If DosyaAc.SelectedFiles IsNot Nothing Then
    For Each Dosya In DosyaAc.SelectedFiles
        txtMetin.Text += Dosya.OpenText.ReadToEnd
    Next
End If
```

[C#]

```
 OpenFileDialog DosyaAc = new OpenFileDialog();
DosyaAc.Filter = "TXT Dosyaları (*.txt)|*.txt";
DosyaAc.Multiselect = true;
DosyaAc.ShowDialog();
if (DosyaAc.SelectedFiles != null) {
    foreach ( Dosya in DosyaAc.SelectedFiles) {
        txtMetin.Text += Dosya.OpenText.ReadToEnd();
    }
}
```

Yukarıdaki kod içerisinde her zamanki gibi Open File Dialog nesnemizi oluşturarak uygun filtreyi tanımlıyoruz. Dikkat etmemiz gereken nokta **Multiselect** özelliğinin **True** olarak ayarlanmış olması. Dosyalar seçildikten sonra **SelectedFiles** listesinden her bir dosyayı tek tek alarak **OpenText** metoduyla dosya içerisindeki yazıyı okuyarak **txtMetin** adındaki TextBox nesnemize aktarıyoruz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde ProgressBar kullanımı.

.NET Framework'ün ufak bir paketini içерse de Silverlight ile web tarayıcısı içerisinde büyük mucizeler yaratmak mümkün. .NET'in JavaScript gibi scripting dillerine kıyasla işlemci kullanımındaki hakimiyetini de düşündüğümüzde istemci tarafında yapacağımız işlemlerin yoğunluğunun artacağı kesin. Tüm bu işlemler süresince tabi ki kullanıcıları da bilgilendirmemiz gerekiyor. AJAX ile az çok web tarafında da alıştığımız "Loading" GIF'lerinin yerine Silverlight tarafında RC0 ile beraber artık **ProgressBar** kontrolümüz var.

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
    <Grid x:Name="LayoutRoot"
        Background="White">
        <ProgressBar Height="28"
            VerticalAlignment="Top"
            Margin="40,58,29,0"/>
    </Grid>
</UserControl>
```

Standart bir ProgressBar'ı yukarıdaki XAML kodu ile uygulamanıza ekleyebilirsiniz. Aslında Silverlight içerisinde ProgressBar'ın Winforms'daki muadilinden ilk bakışta pek bir farkı yok. ProgressBar'ın **Value** özelliği o anki değeri, **Maximum** özelliği ise alabileceği en yüksek değeri tanımlıyor.

Gelin ufak bir örnek yapmak için bir **DispatcherTimer** kullanalım. Timer'ımızın iki saniyede bir ProgressBar'ı biraz ilerletsin. Böylece işlem yapılmış gibi ProgressBar'ı izleyebilelim.

Partial Public Class Page

Inherits UserControl

WithEvents Timer As New System.Windows.Threading.DispatcherTimer

Public Sub New()

InitializeComponent()

End Sub

Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded

Progress.Maximum = 20

Timer.Interval = New TimeSpan(0, 0, 2)

Timer.Start()

End Sub

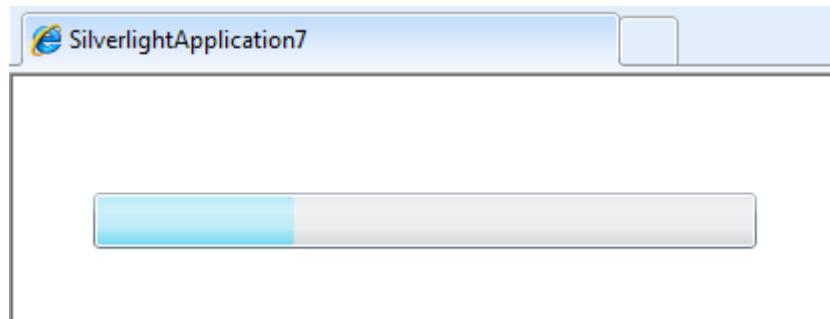
Private Sub Timer_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer.Tick

Progress.Value += 1

End Sub

End Class

Yukarıdaki kod ile Progress adındaki ProgressBar'ımızı oynattığımızda aşağıdaki görüntü ile karşılaşıyoruz.

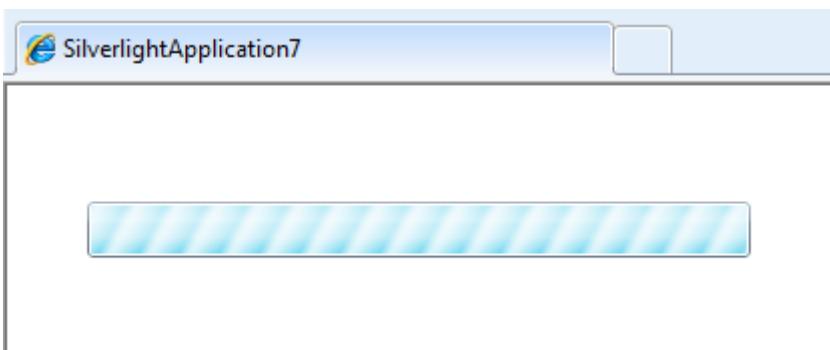


ProgressBar iş başında.

Eğer gerçekten yapılacak işlemin ne kadar süreceği bilmiyorsanız ve bir şekilde bu bilgiyi kullanıcıya gösterme şansınız yoksa bu sefer de işlemin sürdüğüne dair bir gösterge olarak ProgressBar'ı kullanmak için **IsIndeterminate** özelliğini **True** olarak ayarlayabilirsiniz.

```
<ProgressBar Height="28"
    VerticalAlignment="Top"
    Margin="40,58,29,0"
    x:Name="Progress"
    IsIndeterminate="True"/>
```

Böyle bir durumda ProgressBar içerisinde sürekli olarak aynı animasyon çalıştırılacaktır.



ProgressBar sonsuz döngüde!

ProgressBar'ın görsellliğini değiştirmek için **Foreground** ve **Background** özelliklerinden faydalanabilirsiniz. **Foreground** özelliği **ProgressBar** içerisinde dolu kısımları etkilerken **Background** ise ProgressBar'ın boş olan kısımlarını etkileyecektir.

```
<ProgressBar
    x:Name="Progress">
    <ProgressBar.Background>
        <ImageBrush
            ImageSource="Garden.jpg"/>
    </ProgressBar.Background>
</ProgressBar>
```

Örneğin yukarıdaki kod ile aşağıdaki şekilde bir ProgressBar'ın fonuna bir fotoğraf yerleştirebilirsiniz. Hatta **Background** için bir **ImageBrush** yerine **VideoBrush** kullanarak farklı uygulamalar da oluşturulabilir. Burada özellikle **Indeterminate** ile ilgili dikkat etmek gerek. **Indeterminate** aktif hale geldiğinde Silverlight içerisinde ProgressBar'ın ForeGround'u ile tüm kontrol kaplanıyor ve üzerinden beyaz dikey Gradient'lar animasyon ile geçiriliyor. Bu sistem ancak ProgressBar'in ControlTemplate'i özelleştirilebilirse değiştirilebiliyor aksi halde tüm standart ProgressBar kontrolleri bu şekilde çalışıyor.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde sağ tuş menüsünü değiştirmek.

Silverlight uygulamalarında sağ tuş uygulamaya tıkladığımızda karşımıza Silverlight'a ait özel "Silverlight Configuration" menüsü gelir. Bu menüden kullanıcıların Silverlight uygulaması ile ilgili ayarları yapabilecekleri bir arayüze ulaşılır. Fakat bazı durumlarda farenin sağ tuşunu biz de kullanmak isteyebiliriz. Bu ister global anlamda tüm sayfayı kapsayacak şekilde olsun ister belirli Silverlight kontrolleri için olsun istersek sağ tuş ile gelen menüyü değiştirmeye şansımız var.

Menümüzü ve animasyonları hazırlayalım.

Silverlight uygulamamızda sağ tuş ile tıklandığında gösterilmek üzere hızlıca bir menü hazırlayalım. Menümüzde birden çok düğme bulunacak. Düğmeleri dikey olarak sahnede sıralamak için Menu'ye ait kök element olarak bir StackPanel kullanacağız.

```
<StackPanel Height="248" Width="160" Background="#FFFF0000" x:Name="Menu1"
Canvas.Top="24" Canvas.Left="48" Opacity="0">
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
</StackPanel>
```

Yukarıdaki XAML kodu içerisinde **Menu1** adında bir StackPanel bulunuyor. StackPanel içerisinde tüm düğmeler dikey olarak hizalanacak. Bu StackPanel'ı duruma göre sahnede gösterecek veya saklayacak olan animasyonlarımızı da hazırlayalım. Animasyonlar basit bir şekilde StackPanel'in **Opacity** özelliğini değiştirecekler.

```
<Storyboard x:Name="MenuGel">
    <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Menu1" Storyboard.TargetProperty="(UIElement.Opacity)">
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="1"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
<Storyboard x:Name="MenuGit">
    <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Menu1" Storyboard.TargetProperty="(UIElement.Opacity)">
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="0"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

Sıra kodlamada...

Uygulayacağımız taktik aslında çok basit; Silverlight uygulamasının içerisinde bulunduğu sayfanın **oncontextmenu** event'ini yakalayarak normal çalışma şeklini iptal edip menü gösterme işlemini biz üstleneceğiz. İlk olarak söz konusu event'i uygulama açılışında yakalayalım.

[VB]

```

Public Sub New()
    InitializeComponent()
    System.Windows.Browser.HtmlPage.Document.AttachEvent("oncontextmenu",
AddressOf SagTus)
End Sub

```

[C#]

```

public Page()
{
    InitializeComponent();
    System.Windows.Browser.HtmlPage.Document.AttachEvent("oncontextmenu",
SagTus);
    this.MouseLeftButtonDown += new
MouseButtonEventHandler(Page_MouseLeftButtonDown);
}

```

Kodumuz içerisindeki **SagTus** adındaki event-handlerimizi da hemen yazalım.

[VB]

```

Private Sub SagTus(ByVal sender As Object, ByVal args As Browser.HtmlEventArgs)
    Menu1.SetValue(Canvas.LeftProperty, CDbl(args.OffsetX))
    Menu1.SetValue(Canvas.TopProperty, CDbl(args.OffsetY))
    MenuGel.Begin()
    args.PreventDefault()
End Sub

```

[C#]

```

void SagTus(object sender, System.Windows.Browser.HtmlEventArgs args)
{
    Menu1.SetValue(Canvas.LeftProperty, (double)args.OffsetX);
    Menu1.SetValue(Canvas.TopProperty, (double)args.OffsetY);
    MenuGel.Begin();
    args.PreventDefault();
}

```

Kodumuzda ilk olarak **args** üzerinden farenin konumuzu alıyoruz ve söz konusu konuma **Menu1** adındaki StackPanel'ımızı taşıyoruz. Bizim örneğimizdeki Silverlight uygulamasının kök elementi bir Canvas olduğu için taşıma işlemini StackPanel'in **Canvas.Left** ve **Canvas.Top** özelliklerini değiştirerek halledebiliriz. Bir sonraki adımda menüyü görünür hale getirecek olan **MenuGel** animasyonunu çalıştırıp, **args** üzerinden **PreventDefault** metodunu çalıştırarak söz konusu event-handler'ın tarayıcı tarafından değerlendirilmesini engelliyoruz. Böylece tarayıcı bu event çalışlığında herhangi bir menü göstermeyecek.

[VB]

```

Private Sub Page_MouseLeftButtonDown(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseButtonEventArgs) Handles Me.MouseLeftButtonDown

```

```
    MenuGit.Begin()  
End Sub
```

[C#]

```
void Page_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)  
{  
    MenuGit.Begin();  
}
```

Son olarak uygulamanın herhangi bir yerine tıklandığında menüyü saklayacak olan MenuGit animasyonunu çalıştırıyoruz. Böylece istediğimiz gibi sağ tuş ile gelecek olan menüyü ayarlayabilir, tasarımını değiştirebiliriz.

Hepinize kolay gelsin.

Daron YÖNDEM

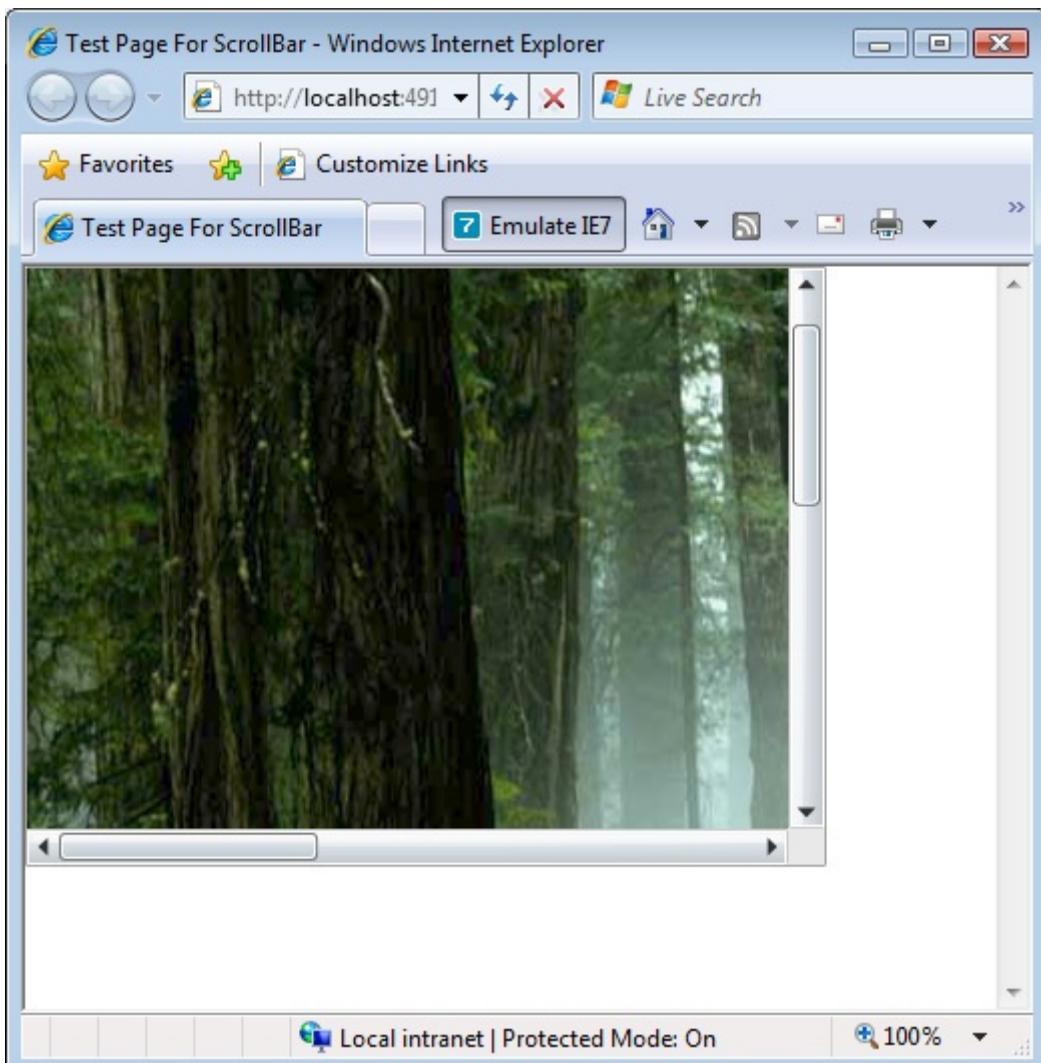
Silverlight 2.0 içerisinde ScrollViewer kullanımı.

Herhangi bir içeriği Silverlight 2.0 arayüzlerinde göstermek istediğinizde özellikle veri bağlanabilir çoğu kontrolün kendi içerisinde "ScrollBar" (Kaydırma Çubukları) içerdiğini görebilirsiniz. Fakat bazı durumlarda bu hazır kontrolleri kullanmadığınızda veya scrollbar özelliği bulunmayan bazı yapı taşı niteliğinde kontrolleri beraber kullanmak istediğiniz ayrıca bir ScrollBar'a ihtiyacınız olabilir. Bu gibi durumlarda bize scrollbar özellikleri eklemeye konusunda **ScrollViewer** kontrolü yardımcı oluyor.

Yapacağımız ilk örnekte 1024*768 piksel büyülüğünde bir resmi uygulamamıza ekleyeceğiz. Fakat biz bu resim nesnesini tam ekran göstermek istemiyoruz. Uygulamamız içerisinde ufak bir karede göstererek insanların istiyorlarsa ScrollBar'lar aracılığı ile resmi gezmesini istiyoruz. Bu durumda aslında yapmamız gereken çok basit. Aşağıdaki XAML kodunu yaratacak şekilde **Image** nesnemizi bir **ScrollViewer** içerisinde yerleştirmemiz yeterli.

```
<UserControl x:Class="ScrollBar.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid x:Name="LayoutRoot" Background="White">
        <ScrollViewer Margin="0,0,0,0" HorizontalScrollBarVisibility="Auto"
        VerticalScrollBarVisibility="Auto">
            <Image Height="768" Width="1024" Source="Forest.jpg"/>
        </ScrollViewer>
    </Grid>
</UserControl>
```

Gördüğünüz gibi ScrollViewer nesnemizin içerisinde kocaman bir resim var. ScrollViewer içerisinde programatik olarak farklı nesnelerin de yerleştirilebiliyor olabilirdi. O nedenle biz ScrollViewer'a ait **HorizontalScrollBarVisibility** ve **VerticalScrollBarVisibility** özelliklerini de Auto yaparak ScrollBar'ların sadece gerektiğiinde gözükmeyi sağladık. Zaten varsayılan ayarları ile maalesef yatay ScrollBar gösterilmiyor o nedenle her halükarda bu ayarları değiştirmek şart.



ScrollViewer kontrolü iş başında.

Kullanımın ne kadar basit olduğunu sanırım daha da anlatmaya gerek yok. Gelin biraz daha karışık bir örneğe doğru yola çıkalım. Varsayılm ki **ScrollViewer** ile beraber gelen kaydırma çubukları yerine kendi oluşturduğunuz bazı düğmeleri kullanarak kaydırma işlemi yaptırmak istiyorsunuz, bu durumda ne yapabilirdik?

İlk olarak örneğimizin görsel kısmını hazırlayarak uygulamamıza iki düğme ekleyelim. Bu düğmeler rahatlıkla farklı görsellikler atanarak daha anlamlı hale getirilebilir. Ben odak noktamızı kaybetmemə adına düğmelerin görsel özellikleri ile ilgilenmeyeceğim.

```
<UserControl x:Class="ScrollBar.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid x:Name="LayoutRoot" Background="White">
        <ScrollViewer Margin="0,0,63,0" HorizontalScrollBarVisibility="Hidden"
        VerticalScrollBarVisibility="Hidden" x:Name="Scroll">
```

```

<Image Height="768" Width="1024" Source="Forest.jpg"/>
</ScrollViewer>
<Button Height="57" HorizontalAlignment="Right" Margin="0,8,8,0"
VerticalAlignment="Top" Width="51" Content="Yukarı" x:Name="Yukarı"/>
<Button Height="46" HorizontalAlignment="Right" Margin="0,0,8,8"
VerticalAlignment="Bottom" Width="51" Content="Aşağı" x:Name="Asagi"/>
</Grid>
</UserControl>

```

Kodumuz içerisinde önemli birkaç nokta var. Bunlardan ilki ScrollViewer kontrolümüzün **VerticalScrollBarVisibility** ve **HorizontalScrollBarVisibility** özelliklerinin **Hidden** olarak ayarlanmış olması gerektiği. Eğer bu özellikleri **Disable** olarak ayarlırsanız maalesef birazdan yapacağımız şekilde ScrollViewer'in kaydırma özelliklerinden faydalananamayız. Oysa biz işimizi olabildiğince basite indirmek ve kolaylaştmak istiyoruz. O nedenle bu özellikler **Hidden** olması ve ScrollViewer'dan faydalananmamız şart. Bu haldeyken zaten ScrollBar'lar hiçbir şekilde gözükmeyecektir.

Sahnemizde ayrıca iki adet de düğme var. Bu düğmelere her basıldığında bir miktar scroll yapmak istiyoruz. Aslında bizim örneğimizde hem yatay hem de dikey kaydırma çubukları gerektiği için toplam dört düğme gerekirdi. Fakat ben şimdilik sadece dikey kaydırma çubuğunu simüle edeceğim, aynı sistemi yatay için kullanmak size kalıyor.

```

Private Sub Asagi_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Asagi.Click
    If Scroll.ScrollableHeight > 0 Then
        Scroll.ScrollToVerticalOffset(Scroll.VerticalOffset + 10)
    End If
End Sub

```

```

Private Sub Yukari_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Yukari.Click
    If Scroll.ScrollableHeight > 0 Then
        Scroll.ScrollToVerticalOffset(Scroll.VerticalOffset - 10)
    End If
End Sub

```

Hem **Yukarı** hem de **Aşağı** adındaki düğmelerimize yukarıdaki şekilde kodlamızı yazdığımızda düğmelere her basıldığında ScrollViewer içerisindeki resim 10 piksel yukarı veya aşağıya doğru kayıyor. Kodumuzu incelediğimizde basit bir IF kontrolü ile ScrollViewer'in yükseklik olarak kaydırılabilir kaydırılamayacağını öğreniyoruz eğer **ScrollableHeight** sıfırdan büyük ise demek ki kaydırma işlemi yapabiliyoruz. Kaydırma işlemini yapabilmek için mevcut **VerticalOffset** üzerinden konumu alarak üzerine 10 piksel ekleyip veya çıkartıp bu konuma scroll edilebilmesi için de **ScrollToVerticalOffset** metodunu kullanıyoruz. Bu metod almış olduğu Offset değerine scroll ediyor.

Daha kaygan bir Scroll olmaz mı?

Yukarıdaki örneğimizi denediğinizde düğmeye her bastığınızda 10 piksellik bir kayma göreceksiniz. Bu durum siz de benim gibi rahatsız ettiyse daha hoş bir çözüme doğru ilerleyebiliriz. Rahatsızlık yaratan aslında iki konu var, birincisi kullanıcı kaydırma işlemine

devam etmek için milyonlarca kez düğmeye tıklamak zorunda. Bu hiç de hoş bir durum değil. Oysa biz düğmeye tıklandığı anı yakalayıp kullanıcı düğmeyi bırakana kadar kaydirmaya devam etsek süper olurdu. Düğmeye fare ile tıklandığı ve bırakıldığı anları yakalamak için rahatlıkla **MouseLeftButtonDown** ve **MouseLeftButtonUp** eventlarını kullanabiliriz. Tek yapmamız gereken bu arada sürekli kaydırma işlemi yapmak. Hatta bu kaydırma işlemini de 20 milisaniyede 1 piksel şeklinde yaparsak aslında çok daha hoş bir kaydırma efekti yaratmış oluruz.

Peki tüm bunları nasıl yapacağız. Silverlight 2.0 Beta 1 ile beraber gelen **DispatcherTimer** nesnesini kullanacağımız. Bu aslında bizim bildiğimiz Winforms'daki Timer'dan pek farklı değil. Esasen tek farkı istemci tarafında farklı bir Thread içerisinde çalışmış gibi davranışması.

Dim Timer As Windows.Threading.DispatcherTimer

İlk olarak yukarıdaki şekilde Timer değişkenimizi global olarak tanımladık. Global tanımlamamızın nedeni hem **MouseLeftButtonDown** hem de **MouseLeftButtonUp** durumlarında bu Timer'a başvuracak olmamız. Aslında yapacağımız esas işlemi Timer'ı **MouseLeftButtonDown** durumunda yani kullanıcı düğmeye basında başlatmak ve **MouseLeftButtonUp** durumunda ise yani kullanıcı düğmeyi bıraktığında ise durdurmak.

```
If Scroll.ScrollableHeight > 0 Then
    Timer = New Windows.Threading.DispatcherTimer
    Timer.Interval = New TimeSpan(0, 0, 0, 0, 20)
    AddHandler Timer.Tick, AddressOf TimerTick
    Timer.Start()
End If
```

Yukarıdaki kodumuzu düğmemizin **MouseLeftButtonDown** durumuna yazıyoruz. Kullanıcı düğmeye tıkladığı anda global değişkenimize yeni bir **DispatcherTimer** nesnesi aktararak **Interval** değerini 20 milisaniye olarak düzenliyoruz. Böylece her 20 milisaniyede bir bir sonraki adımda **DispatcherTimer** nesnesine bağladığımız **Tick** event-handları çalıştırılıyor olacak. Tüm ayarlarımızı tamamladıktan sonra DispatchTimer'in **Start** metodu ile işlemi başlatıyoruz.

```
Sub TimerTick(ByVal sender As Object, ByVal e As EventArgs)
    Scroll.ScrollToVerticalOffset(Scroll.VerticalOffset + 1)
End Sub
```

Timer'ın her **Tick** durumunda daha önce kullandığımız kodu kullanarak kaydırma işlemi yapıyoruz. Bu sefer Tick durumları 20 milisaniyede bir olacağı için sadece 1 piksellik bir kayma yaratacağımız.

```
Private Sub Asagi_MouseLeftButtonUp(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Asagi.MouseLeftButtonUp
    Timer.Stop()
End Sub
```

Son olarak düğmenin **MouseLeftButtonUp** durumunda ise Timer'ımızı durdurarak kayma işlemini sonlandırıyoruz. Uygulamamızı hem **Asağı** hem de **Yukarı** düğmeleri için tamamladığımızda kodumuz aşağıdaki şekilde sonuçlanıyor.

Partial Public Class Page

Inherits UserControl

Public Sub New()

 InitializeComponent()

End Sub

Dim Timer As Windows.Threading.DispatcherTimer

Private Sub Asagi_MouseLeftButtonDown(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Asagi.MouseLeftButtonDown, Yukari.MouseLeftButtonDown

If Scroll.ScrollableHeight > 0 Then

 Timer = New Windows.Threading.DispatcherTimer

 Timer.Interval = New TimeSpan(0, 0, 0, 0, 20)

 AddHandler Timer.Tick, AddressOf TimerTick

 Timer.Start()

End If

End Sub

Sub TimerTick(ByVal sender As Object, ByVal e As EventArgs)

 If Yukari.Focused Then

 Scroll.ScrollToVerticalOffset(Scroll.VerticalOffset - 1)

 Else

 Scroll.ScrollToVerticalOffset(Scroll.VerticalOffset + 1)

 End If

End Sub

Private Sub Asagi_MouseLeftButtonUp(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Asagi.MouseLeftButtonUp, Yukari.MouseLeftButtonUp

 Timer.Stop()

End Sub

End Class

Yukarıdaki kod içerisinde MouseLeftButtonDown ve MouseLeftButtonUp event-handlerlarının sadece birer kere bulunduğu dikkatinizi çekecektir. Söz konusu event-handlerları her iki düğmeye de bağlamış durumdayız. Aslında her iki düğmenin de sadece uygun zamanlarda Timer nesnesini başlatması ve sonlandırma yeterli. Önemli olan Timer'in Tick durumunda içeriğini yukarı veya aşağıya kaydırılacağına karar verebiliyor olmak. Bunun için de ben örneğimde **Yukarı** düğmesinin **IsFocused** özelliğinden faydalandım. Eğer bir düğmeye basılmış ise doğal olarak söz konusu düğme Focus almış demektir. Böylece o an için hangi düğmeye basılmakta olduğunu yakalayıp one göre işlem yapılabilir.

Hepinize kolay gelsin.

Daron YÖNDEM

[Silverlight 2.0 içerisinde Silverlight Toolkit ve TreeView kullanımı](#)

Silverlight 2.0 ile beraber gelen kontrol sayısı 1.0'a kıyasla ciddi miktarda arttı. Fakat hala eksikler var! İşte bu eksikleri Silverlight sürümleri arasında doldurabilmek adına yeni bir proje [CodePlex](#) üzerinde yayında. Proje aslında uzun süredir geliştiriliyor. Fakat daha yeni yeni stabil kontroller sunmaya başladı. Aslında eski AJAX Control Toolkit'e benzetebileceğimiz bir yapıda ilerleyen [Silverlight Toolkit](#) unutmamak gereklidir. Microsoft tarafından geliştirilmiyor, tamamen açık kaynak kodları ile gönüllü programcılar tarafından ilettilen bu projede bazı kontroller stabil olarak işaretlenmişken bazıları ise hala "Preview" konumundalar.

İşte bu kütüphane içerisinde stabil olarak işaretlenmiş olan kontrollerden belki de en acil ihtiyaç duyacağımız **TreeView** kontrolünü bu yazımızda inceleyeceğiz.

Nasıl yüklenir?

Silverlight Toolkit içerisinde kontrollerden herhangi birini kullanabilmek için ilk olarak tabii ki söz konusu kütüphaneyi CodePlex üzerinden indirmeliyiz. Hemen aşağıdaki adresten son paketi indirebilirsiniz;

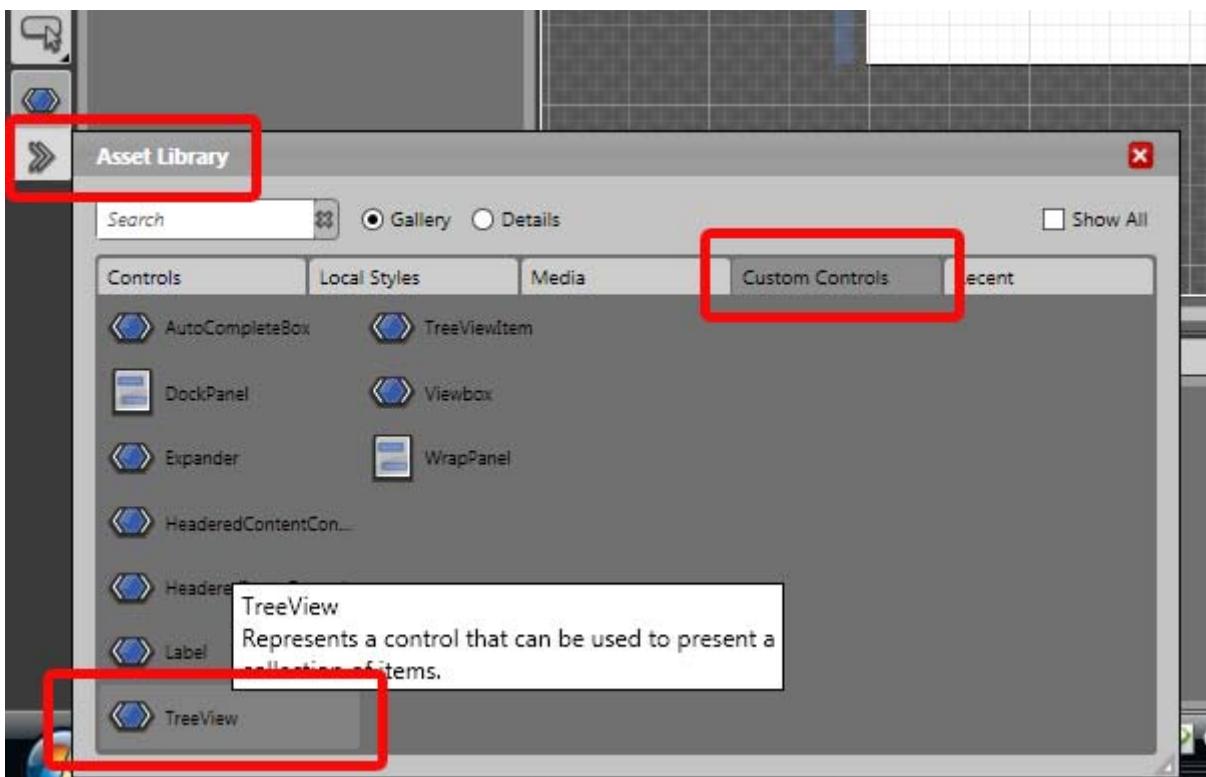
<http://www.codeplex.com/Silverlight/Release/ProjectReleases.aspx?ReleaseId=18804>

Paketi bilgisayarınıza indirdiğinizde **Binaries** klasörü içerisinde bulunan **Microsoft.Windows.Controls.dll** dosyasını yeni yarattığınız bir Silverlight projesine Visual Studio içerisinde referans olarak eklemeniz gerekiyor. Paket içerisinde bulunan diğer DLL'lere ve dosyalara ileriki makalelerimizde değineceğiz.

Unutmayın, referans ekleme işlemini doğrudan Silverlight projenize yapmanız gereklidir. Silverlight projenizle aynı Solution içerisinde bulunan ASP.NET proje ile herhangi bir ilişkimiz yok.

Bu işlemi tamamladıktan sonra artık **Expression Blend** tarafına geçerek kontrollerimizi kullanmaya başlayabiliriz. Eğer Expression Blend ile değil de Visual Studio tarafından XAML kodlarını yazmak isterseniz tek tek XAML içerisinde NameSpace tanımlarını yapmanız gerekecektir.

Expression Blend ile projemizi açtıktan sonra "Asset Library"e gidip "Custom Controls" tabına geçtiğimizde TreeView kontrolü karşımıza çıkıyor.



TreeView kontrolü Expression Blend içerisinde karşımıza çıkıyor!

TreeView kontrolünü Blend içerisinde projenizde herhangi bir XAML dosyasına eklediğinizde XAML koduna dönüp bakarsanız aşağıdaki şekilde gerekli NameSpace tanımlarının da yapılmış olduğunu göreceksiniz. Eğer Blend kullanmasaydık bu işlemleri Visual Studio içerisinde elle yapmamız gerekecekti.

[XAML]

```
<UserControl x:Class="SilverlightApplication7.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
    namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <Grid x:Name="LayoutRoot" Background="White">
        <controls:TreeView Margin="74,43,133,137"/>
    </Grid>
</UserControl>
```

Nasıl kullanılır?

Bir TreeView aslında birden çok TreeViewItem içerir. İsterseniz bu TreeViewItem'ları doğrudan Expression Blend içerisinde sürükle & bırak tekniği ile TreeView içerisinde yerleştireceğiniz gibi isterseniz doğrudan kod ile databind işlemleri de yapabilirsiniz.

[XAML]

```
<controls:TreeView Margin="74,43,133,137">
<controls:TreeViewItem Header="Deneme1"/>
<controls:TreeViewItem Header="Denem 2"/>
<controls:TreeViewItem Header="Deneme 3">
  <controls:TreeViewItem Height="100" Width="100" Header="İçinde!"/>
</controls:TreeViewItem>
</controls:TreeView>
```

Yukarıdaki kod içerisinde birden çok TreeViewItem içeren bir TreeView görünsünüz. TreeView altında toplam üç adet TreeViewItem varken son TreeViewItem içerisinde bir tane daha TreeViewItem var. Böylece iç içe açılarak devam eden sonsuz döngüde bir ağaç yapısı oluşturmak mümkün.

TreeViewItem'ların Header özelliğine gözükecek metin değerini girebileceğiniz gibiaslında farklı Silverlight kontrollerini de Header içerisinde koyma şansınız var.

[XAML]

```
<controls:TreeView Margin="74,43,133,137">
<controls:TreeViewItem Header="Deneme1"/>
<controls:TreeViewItem Header="Denem 2"/>
<controls:TreeViewItem Header="Deneme 3">
  <controls:TreeViewItem Height="100"
    Width="100">
    <controls:TreeViewItem.Header>
      <Image Height="50"
        Width="50"
        Source="Tree.jpg" />
    </controls:TreeViewItem.Header>
  </controls:TreeViewItem>
  </controls:TreeViewItem>
</controls:TreeView>
```

Yukarıdaki kod içerisinde de görebileceğiniz gibi TreeViewItem'lardan birinin Header'ında bir Image nesnesi, yani fotoğraf var. Siz uygulamalarınızda farklı fantaziler yaparak isterseniz MediaElement aracılığı ile video bile koyabilir veya belki de TreeView içerisinde bir DataGridView veya Calendar bile kabilirsiniz.

Gelelim tüm bu işlemlerin kod ile nasıl yapıldığına. Varsayılmış elimizde TreeView'de göstermek istediğimiz bir veri var. İlk olarak bu veriyi uygun şekilde düzenlememiz gereklidir. Bizim örneğimizde **Fiyati** ve **Adı** gibi özelliklere sahip bir sınıf kullanalım. Her bir TreeViewItem bu sınıf üzerinden oluşturulacak.

[VB]

Class Urun

```
Private PAdi As String
Public Property Adi() As String
  Get
```

```

        Return PAdi
    End Get
    Set(ByName value As String)
        PAdi = value
    End Set
End Property

Private PFiyati As Integer
Public Property Fiyati() As Integer
    Get
        Return PFiyati
    End Get
    Set(ByName value As Integer)
        PFiyati = value
    End Set
End Property

Private PList As List(Of Urun)
Public Property Liste() As List(Of Urun)
    Get
        Return PList
    End Get
    Set(ByName value As List(Of Urun))
        PList = value
    End Set
End Property

Public Sub New()
    Me.PList = New List(Of Urun)
End Sub
End Class

```

[C#]

```

public class Urun
{
    public string Adi { get; set; }
    public int Fiyati { get; set; }
    public List<Urun> Liste { get; set; }

    public Urun()
    {
        this.Liste = new List<Urun>();
    }
}

```

Gördüğünüz gibi sınıfımızın **Adı** ve **Fiyatı** özellikleri haricinde bir de **List** tipinden **Liste** adında Property'si var. Bunun nedeni aslında çok basit. Her bir TreeViewItem'in kendi içinde birden çok TreeViewItem olabilir demişti. Bizim her bir ürünümüzün için de bu şekilde

birden çok ürün saklanabiliyor olacak. Verimizi bu şekilde üretip TreeView'e DataBind ettiğimizde TreeView geri kalanı halledecek.

Gelin şimdi de bizim için deneme amaçlı olarak geçici veri yığını yaratacak bir kod yazalım.

[VB]

```
Function Veriyarat() As List(Of Urun)
    Dim Liste As New List(Of Urun)
    For x As Integer = 0 To 10
        Dim BirUrun = New Urun With {.Adi = "Ürün" & x, .Fiyati = Rnd() * 1000}
        For y As Integer = 0 To 5
            BirUrun.Liste.Add(New Urun With {.Adi = "Ürün" & y, .Fiyati = Rnd() * 1000})
        Next
        Liste.Add(BirUrun)
    Next
    Return Liste
End Function
```

[C#]

```
public List<Urun> Veriyarat()
{
    Random RastGele = new Random();
    List<Urun> Liste = new List<Urun>();
    for (int x = 0; x <= 10; x++) {
        Urun BirUrun = new Urun { Adi = "Ürün" + x.ToString(), Fiyati =
RastGele.Next(0,1000) };
        for (int y = 0; y <= 5; y++) {
            BirUrun.Liste.Add(new Urun { Adi = "Ürün" + y.ToString(), Fiyati =
RastGele.Next(0, 1000) });
        }
        Liste.Add(BirUrun);
    }
    return Liste;
}
```

Kodumuz içerisinde toplam 10 adet içerisinde 5'er ürün bulunan ürün yaratılıyor. Şimdi tüm bu listeyi alarak doğrudan TreeView'in ItemsSource özelliğine aktaracağız. Siz örneklerinizde farklı sınıflar ve farklı veri kaynakları kullanabilirsiniz. Özellikle LINQ2SQL veya Data Services kullandığınızı düşünürsek zaten elinize hazır nesneler gelecektir.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Agac.ItemsSource = Veriyarat()
End Sub
```

[C#]

```

public Page()
{
    InitializeComponent();
    this.Loaded += new RoutedEventHandler(Page_Loaded);
}

void Page_Loaded(object sender, RoutedEventArgs e)
{
    Agac.ItemsSource = Veriyarat();
}

```

Veri bağlantımızı yaptık. Fakat bu TreeViewItem'lar nasıl yaratılacak? **Fiyati** ve **Adı** adındaki özelliklerin içindeki değerler nerede gösterilecek? İşte bu noktada XAML tarafına geçerek bir ItemTemplate düzenlememiz gerekiyor. Böylece **TreeView** kendisine bağlanan veriye göre **TreeViewItem**'lar yaratırken nasıl bir görselliğten yola çıkacağını ve bu görsellik içerisinde gelen veriden hangi değerlerin nerelelere yerleştirileceğini anlayabileceğim.

[XAML]

```

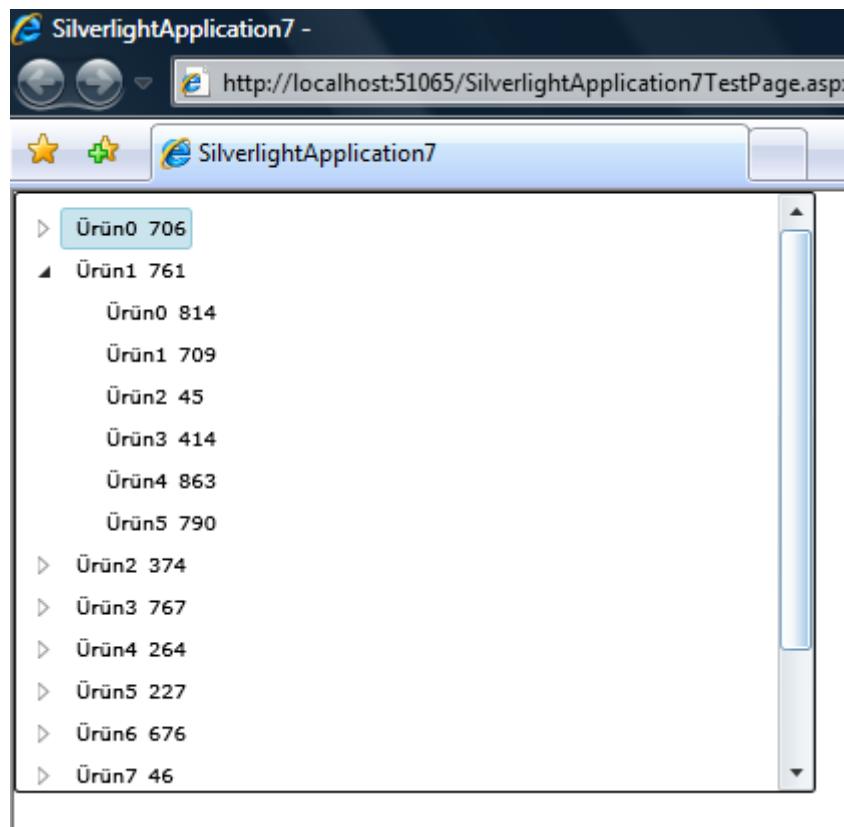
<controls:TreeView Margin="74,43,133,137"
    x:Name="Agac">
    <controls:TreeView.ItemTemplate>
        <controls:HierarchicalDataTemplate ItemsSource="{Binding Liste}">
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="{Binding Adı}"
                    Margin="5,0,0,0"
                    HorizontalAlignment="Left"
                    FontSize="10" />
                <TextBlock Text="{Binding Fiyati}"
                    Margin="5,0,0,0"
                    HorizontalAlignment="Left"
                    FontSize="10" />
            </StackPanel>
        </controls:HierarchicalDataTemplate>
    </controls:TreeView.ItemTemplate>
</controls:TreeView>

```

Yukarıdaki XAML kodu ile aslında uygulamamızı sonlandırmış olduk. Peki neler yaptık? İlk olarak **TreeView**'in **ItemTemplate**'i içerisinde yine Toolkit içerisindeki **HierarchicalDataTemplate** nesnesini yerleştirdik. Bu nesne bizim kod tarafında yarattığımız nested yapıyı algılayarak iç içe **TreeViewItem**'ların yaratılmasını sağlayacak fakat **DataBind** yaptığımız sınıfların hangi **Property**'sinin başka nested **TreeViewItem**'ların verilerini sakladığını anlayabilmesi için söz konusu **Property**'nin adını vermemiz lazım. Hatırlarsınız bizde **Urun** sınıfının **Liste** adında bir **property**'si vardı ve tüm alt öğeleri o saklıyordu. Biz de burada **HierarchicalDataTemplate** 'e **ItemsSource** olarak **Liste**'yi **Bind** ediyoruz. Geri kalanı kendisi halledecektir.

HierarchicalDataTemplate içerisinde doğrudan her bir **TreeViewItem**'nın Header bilgisini girer gibi istediğiniz Silverlight kontrollerini kullanabiliyorsunuz. Burada yapacağınız tasarım her **TreeViewItem** için kullanılacaktır. Tabi bunu yaparken Binding'lerimizi de unutmuyoruz

ve yerine göre **Fiyatı** ve **Adı** özelliklerini istediğimiz kontrollerin istediğimiz özelliklerine Bind ediyoruz. Bizim örneğimizde bir **StackPanel** içerisinde **Fiyatı** ve **Adı** özelliklerini gösterecek iki farklı **TextBlock** bulunuyor.



DataBind TreeView örneğimiz bitti.

Tüm verimizi bağladığımıza göre yeri geldiğinde seçili öğeyi nasıl yakalarız? Her bir TreeView'in kendi **SelectedIndexChanged** event'i var. Bu event çalıştığında söz konusu TreeView'in **SelectedItem** özelliğinden seçili öğeyi alabilirsiniz. En önemlisi **SelectedItem** size aslında **DataBind** esnasında bağlanan verinin içerisinde bulunan nesneyi döndürüyor. Yani bizim örneğimizde **SelectedItem**'ı aldığında elimize **Urun** tipinden bir nesne gelecek. Gönül daha ne ister? :)

Hepinize kolay gelsin.

Daron YÖNDEM

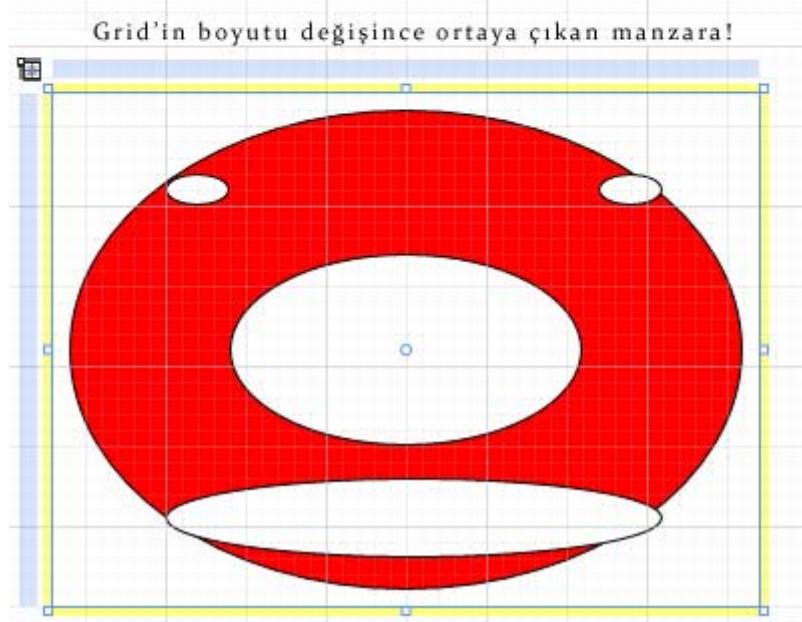
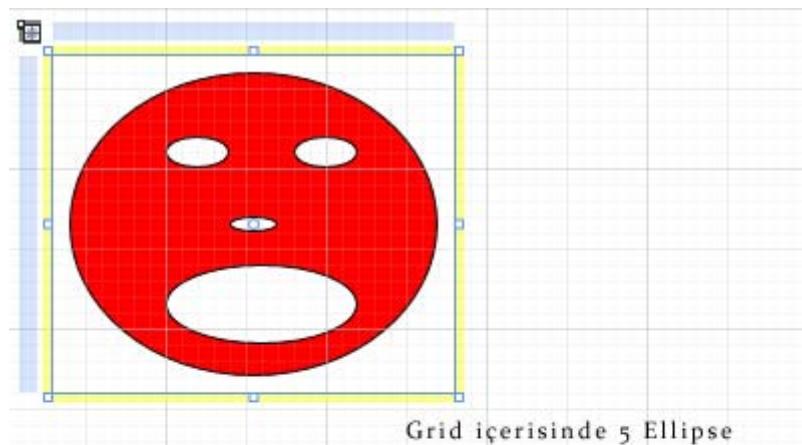
Silverlight 2.0 içerisinde Toolkit'den ViewBox kullanımı.

WPF'den Silverlight dünyasına geçince eksliğini hissettiğimiz kontrollerin Silverlight Toolkit projesi ile sağlanmaya çalışıldığına dair daha önceki yazılarımda ufak ipuçları vermiştim. Bu sefer de yine Toolkit içerisinde ViewBox kontrolünü inceleyeceğiz. **ViewBox** hali hazırda WPF içerisinde bulunan bir Layout kontrolü. Silverlight tarafında ise herhangi bir muadilinin olmaması bazı durumlarda ciddi sıkıntı yaratıyor.

Peki nedir ViewBox?

İster WPF ister Silverlight tarafında olun sahnenin planını Layout kontrolleri dediğimiz kontroller ile düzenlemek durumundasınız. Bu kontrollerin her birinin birbirinden farklı özellikleri var. Tüm bu özellikleri göz önünde aldığımızda ViewBox'ın eşsiz olduğu nokta içerisindeki tüm nesneleri vektörel olarak görsel anlamda tekrar boyutlandırabiliyor olması. Birkaç görsel örnek ile konuyu netlestirelim.

*Not: Silverlight Toolkit'i kullanabilmeniz için [CodePlex](#) üzerindeki adresten kütüphaneyi indirerek içerisindeki **Microsoft.Windows.Controls.dll** dosyasını projenize referans olarak eklemelisiniz.*



Grid içerisinde 5 farklı Ellipse'in değişen durumları.

Yukarıdaki gibi bir Grid içerisinde bulunan nesneler Grid'in kenarlarından olan uzaklarına ve farklı hizalama bilgilerine göre konumlandırılırlar. Bu nedenle Grid'in boyutu değiştiğinde nesnelerin kenarlara olan uzaklıklarını sabit tutabilmek adına nesneler garip şekillerde boyutlandırılır. Oysa biz bu Grid'in boyutlandırırken içindeki görselliği doğrudan aynı şekilde büyütmesini istiyorduk hem de vektörel olarak. Maalesef Silverlight 2.0 ile beraber hali hazırda gelen ve bu işlevselligi sağlayacak hiçbir Layout kontrolü yok! Tabi bu durum "*Böyle bir Layout kontrolü yazılamaz*" anlamına gelmiyor :) Nitekim yazmışlar ve Silverlight Toolkit içerisinde de biz **ViewBox** kontrolünü alıp kullanabileceğiz.

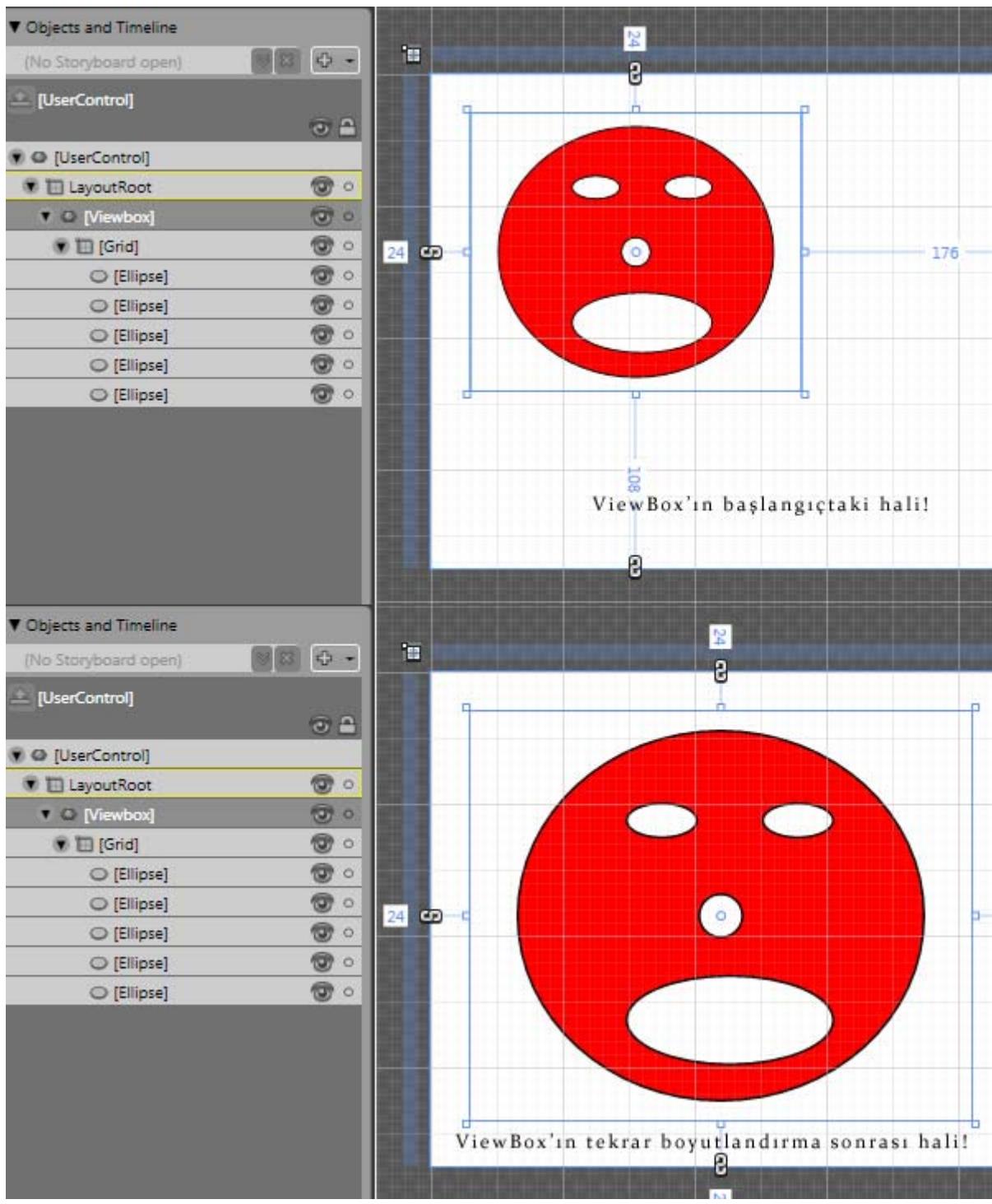
ViewBox kontrolümüzü kullanalım.

Silverlight Toolkit'i projenize referans olarak aldıktan sonra Expression Blend içerisinde doğrudan "Asset Library""de "Custom Controls" tabında "ViewBox" kontrolünü bulabilirsiniz. Bir önceki bölümdeki görsel örneğimizde kullandığımız Grid'i gelin bir ViewBox içerisinde yerleştirelim.

[XAML]

```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <Grid x:Name="LayoutRoot" Background="White">
        <controls:Viewbox Margin="24,24,176,108">
            <Grid Height="Auto" Width="Auto">
                <Ellipse Margin="8,8,8,8" Fill="#FFFF0000" Stroke="#FF000000"/>
                <Ellipse Height="16" HorizontalAlignment="Left" Margin="56,40,0,0"
VerticalAlignment="Top" Width="32" Fill="#FFFFFF00" Stroke="#FF000000"/>
                <Ellipse Height="16" HorizontalAlignment="Right" Margin="0,40,48,0"
VerticalAlignment="Top" Width="32" Fill="#FFFFFF00" Stroke="#FF000000"/>
                <Ellipse Margin="88,80,88,80" Fill="#FFFFFF00" Stroke="#FF000000" Width="20"
Height="20"/>
                <Ellipse Height="40" Margin="56,0,48,24" VerticalAlignment="Bottom"
Fill="#FFFFFF00" Stroke="#FF000000"/>
            </Grid>
        </controls:Viewbox>
    </Grid>
</UserControl>
```

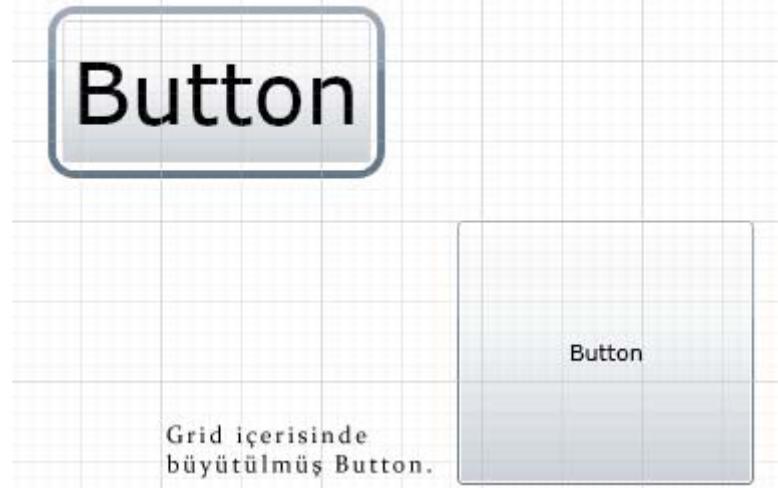
Yukarıdaki kod içerisinde de görebildiğiniz üzere elimizdeki Grid'i doğrudan alıp bir ViewBox içerisinde yerleştirdik. Böylece artık ViewBox'ı tekrar boyutlandırdığımızda içerisinde tüm görseller vektörel olarak bir bütün şeklinde kabul edilecek ve o şekilde tekrar boyutlandırılacak.



ViewBox güzelliği.

Gördüğünüz gibi görselde hiçbir değişiklik yok. Elimizdeki çizim vektörel olduğu için büyündüğünde de herhangi bir görsel bozulma olmuyor. Tabi **ViewBox**'ı her durumda kullanmak da doğru olmayacaktır. Örneğin Silverlight ekranınızda bir **Button** varsa (veya herhangi bir kontrol) ve bu kontrolün dinamik olarak boyutlandırılmasını istiyorsanız **ViewBox** yerine **Grid**'i tercih etmelisiniz. Çünkü **Grid** **Button**'un **Height** ve **Width** gibi özelliklerinden **Button**'a boyut verirken **ViewBox** doğrudan **Button**'un görselliği üzerinden vektörel olarak büyütürcektir.

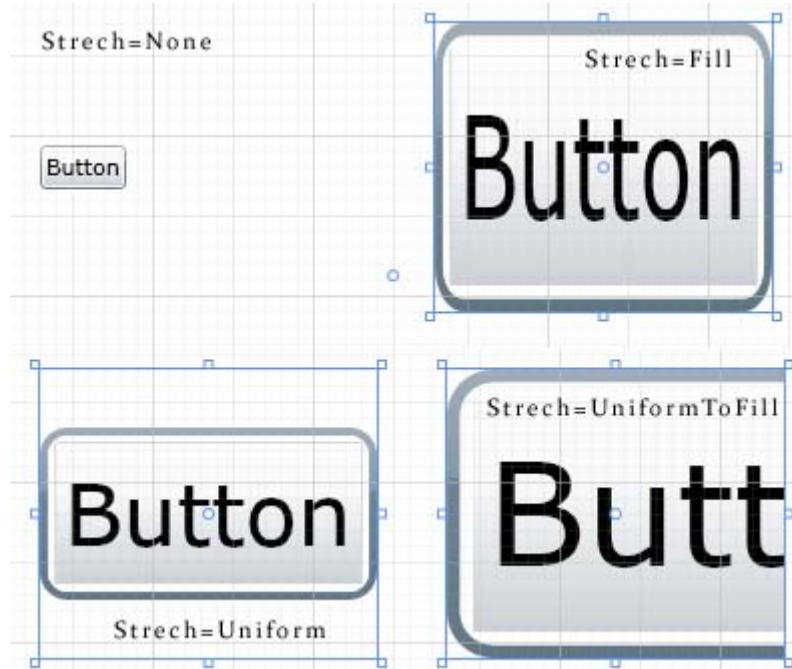
ViewBox içerisinde büyütülmüş Button :)



ViewBox ve Grid farkı.

ViewBox özellikleri...

Bir ViewBox kontrolü içerisinde görsellerin nasıl boyutlandırılacağı ile ilgili verebileceğimiz kararlar var. Bunlardan ilki Strech özelliği. Gelin Strech özelliğine verebileceğimiz değerler arasındaki farklara bir örnek ile bakalım.



ViewBox'in Strech özellikleri arasındaki farklar.

Farklar sanırım görselde açık bir şekilde belli oluyor. **None** değerini verdigimiz Strech özelliği ViewBox içerisindeki görsellerin boyutlandırılmamasını sağlıyor. **Uniform** değeri en/boy oranını koruyarak ViewBox'ın içerisinde tam sığacak şekilde görseli büyütürken **UniformToFill** ise yine en/boy oranını korusa da bu sefer ya eni ya da boyu her şekilde ViewBox'ın içine en büyük değeri ile yerleştiriyor. Strech özelliğine doğrudan **Fill** değerini

verirseniz bu sefer görselin en/boy oranı korunmadan tamamen ViewBox'in içerisine yayılıyor.

Ayrıca isterseniz ViewBox'in **VerticalAlignment** ve **HorizontalAlignment** özellikleri ile de **UniForm** modunda ViewBox içerisindeki görselin nasıl hizalanacağına karar verebilirsiniz.

StretchDirection özelliği.

Şu ana kadar **ViewBox** içerisinde görsellerin sürekli büyüdüğünü gördük fakat isterseniz ViewBox'ın boyutunu ufaltarak aynı şekilde elinizdeki görseli küçültebilirsiniz de. Bazı durumlarda ise ViewBox içerisindeki görselin sadece gerekiğinde büyütülmesini veya sadece küçültülmesini isteyebilirsiniz. Bu gibi durumlarda gerekli sınırlamaları yapmak için doğrudan **StretchDirection** özelliğini kullanabilirsiniz.

Eğer **StretchDirection** değeri **UpOnly** olursa ViewBox içerisindeki görsel gerekiğinde sadece büyütülecektir. ViewBox kendi içindeki görselden daha fazla ufaltılırsa içindeki görseli kesinlikle ufaltmayacaktır. Aynı şekilde **DownOnly** özelliği de ViewBox'ın kendi içerisindeki görseli gerekiğinde sadece ufaltmasını sağlayacaktır. **StretchDirection** özelliğinin varsayılan değeri **Both** olarak geldiği için normal şartlarda hem ufaltma hem de büyütme yapar.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Toolkit'ten Label kontrolünün kullanımı.

Silverlight ilk çıktığı günlerde en çok şaşırduğumuz noktalar biri "Label" adında bir kontrolün bulunmamasıydı. İşlevsellik olarak aynı çözümü sunan **TextBlock** kontrolünü çok kısa bir sürede keşfetmiş olsak da neden isminin değiştiğini pek anlayamamıştık. Bugünlerde [Silverlight Toolkit](#) paketi ile beraber özel bir **Label** kontrolü geldi.

Nedir **TextBlock** ile **Label**'ın farkı?

Aslında kaba tanımı ile **TextBlock** kontrolü Label'ın yapı taşıdır. Bir **Label** kontrolü ControlTemplating desteklerken **TextBlock** desteklemez. Zaten **Label** Template'leri oluştururken biz de TextBlock'lardan faydalanağız. Özette Label'lar gelişmiş TextBlock kontrolleridir de diyebiliriz. Sözü daha çok uzatmadan gelin neler yapabildiğimize göz atalım.

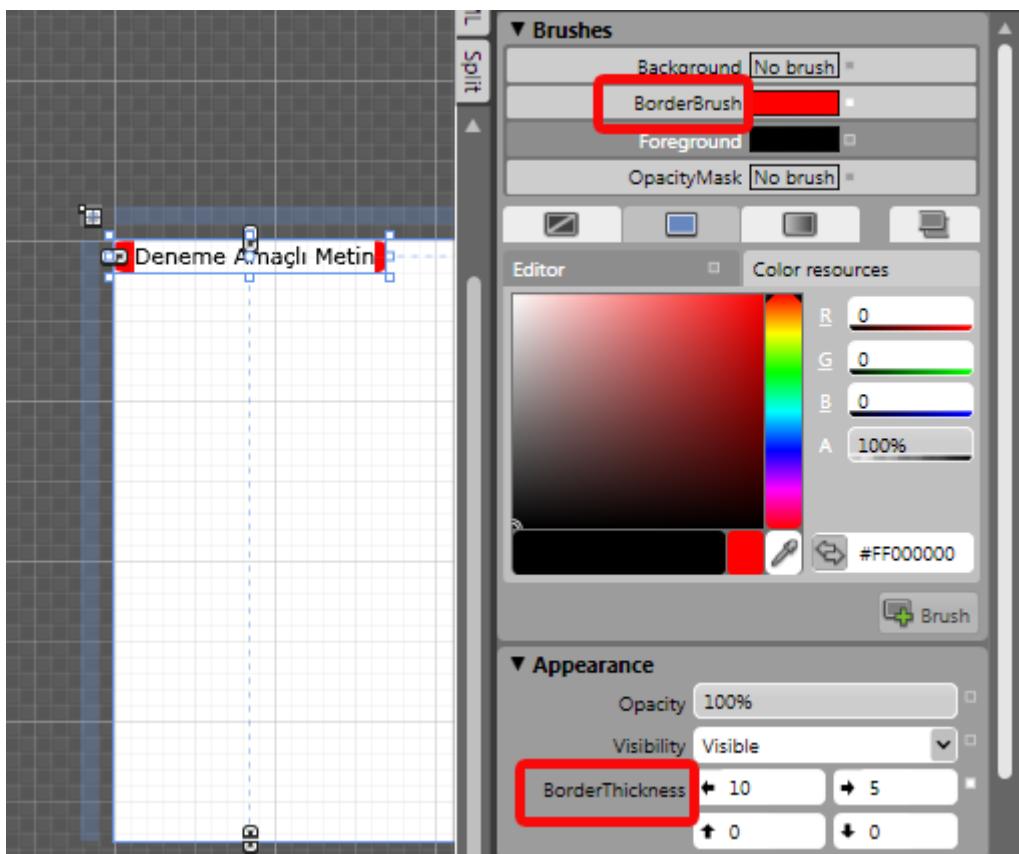
Not: Silverlight Toolkit'i kullanabilmeniz için [CodePlex](#) üzerindeki adresden kütüphaneyi indirerek içerisindeki Microsoft.Windows.Controls.dll dosyasını projenize referans olarak eklemelisiniz.

Bir Label kontrolünü Silverlight Toolkit'in referans alınmış olduğu herhangi bir projede rahatlıkla kullanabilirsiniz. Expression Blend içerisinde "Asset Library" kısmında "Custom Controls" sekmesinde **Label** kontrolünü bulabilirsiniz. TextBlock'larda da olduğu gibi herhangi bir **Label** içerisinde yazı yazmak için **Content** özelliğinden faydalanabilirsiniz.

[XAML]

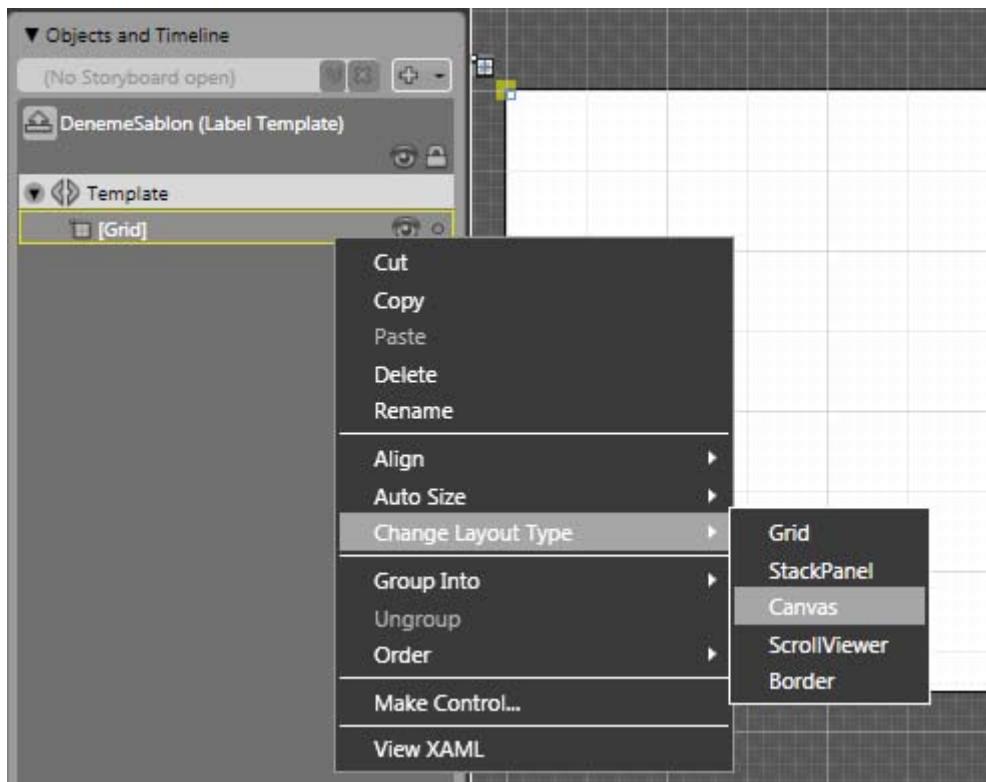
```
<UserControl x:Class="SilverlightApplication2.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
    namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <Grid x:Name="LayoutRoot" Background="White">
        <controls:Label HorizontalAlignment="Left" VerticalAlignment="Top"
        Content="Deneme Amaçlı Metin"/>
    </Grid>
</UserControl>
```

Basit bir şekilde **Label** kontrolünün kullanımına dair yukarıdaki XAML kodunu inceleyebilirsiniz. Bir TextBlock ile Label kontrolünü birbirinden ayıran özelliklerden ilki Label kontrolünün **BorderBrush** alabiliyor olması. Hatta sadece bu kadarla kalmayıp bir Label'ın hangi kenarlarında **Border** bulunacağına da karar verebiliyorsunuz.



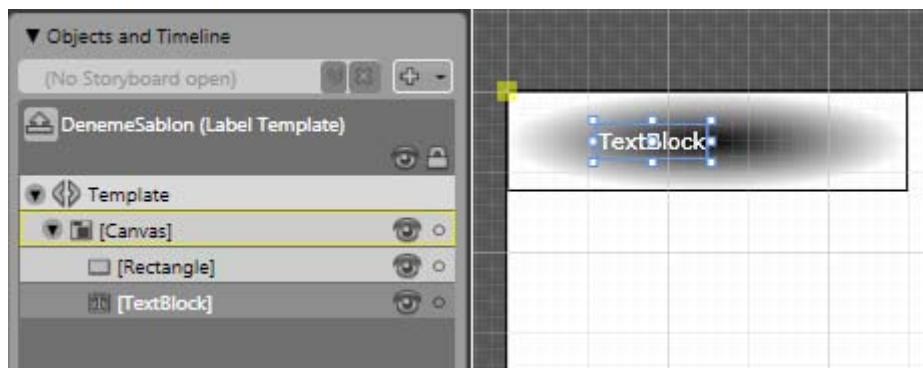
Label kontrolünün BorderBrush ayarları.

Aynı şekilde isterseniz bir Label için **Background** da belirleyebilirsiniz. Fakat makalemizin başından beridir bahsettiğimiz Label'in en önemli özelliği aslında ControlTemplating'e olanak tanımı. Gelin şimdi Expression Blend içerisinde Label kontrolümüze sağ tıklayarak gelen menüden "Edit Control Parts / Create Empty" komutunu verelim. Böylece hali hazırda sahnede olan Label'in görselliği yok varsayılarak bizim Label kontrolünün yapısını tekrar tasarılayabilmemiz sağlanacak. Bunun için ilk aşamada karşınıza gelen pencerede bu şablon'a bir de isim vermeniz gereklidir. Unutmayın hazırlanan şablonlar sonrasında birden çok Label'a linklenerek merkezi bir yerden kullanılabilir.



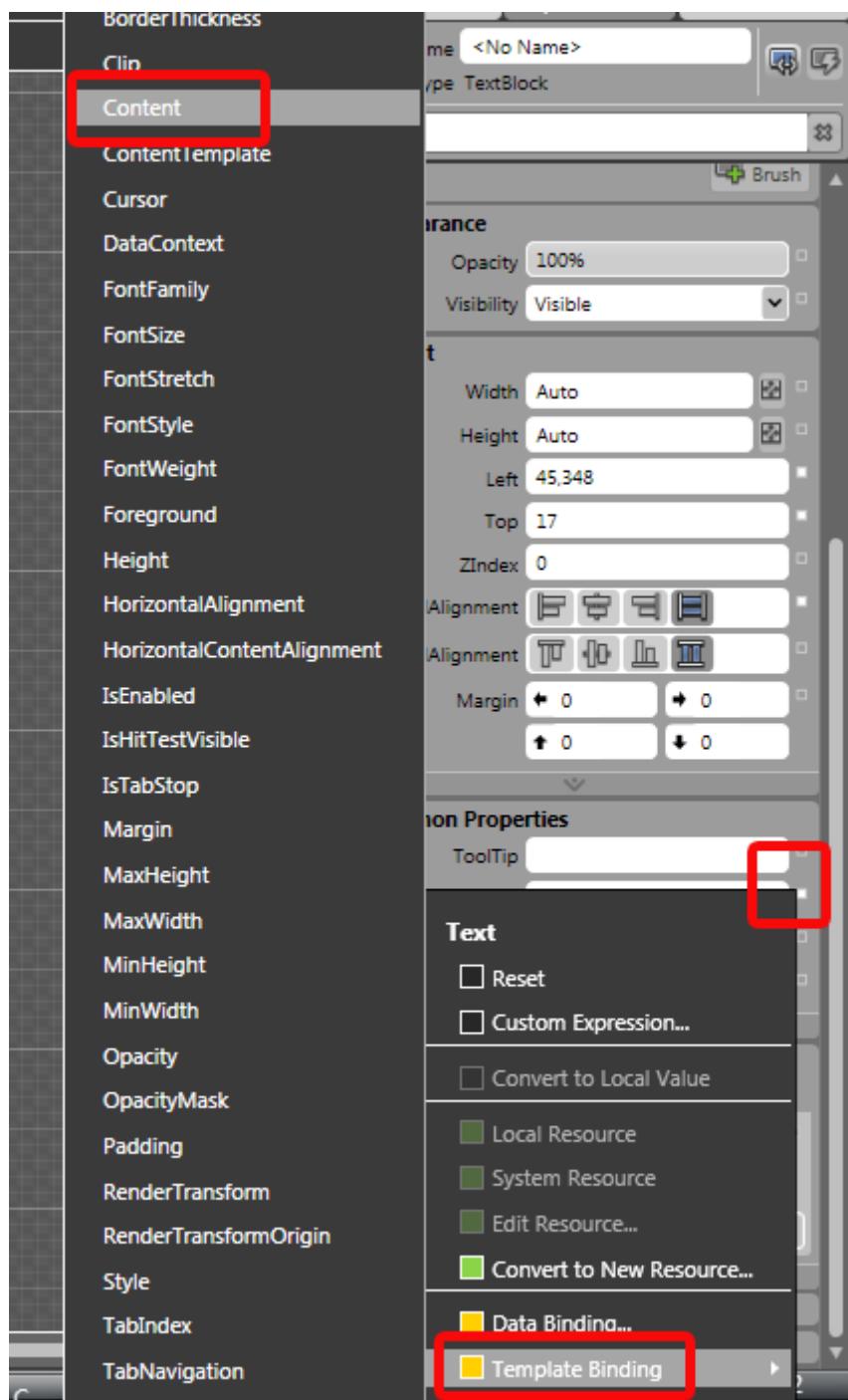
Label kontrolü için yeni bir ControlTemplate yaratıyoruz.

ControlTemplate'imizi yarattıktan sonra Blend bizi otomatik olarak **Template** tasarımına götürecektir. Bu sahnede otomatik olarak gelen Grid nesnesini bir **Canvas'a** çevireceğiz. Bu seçimi tamamen örneğin kolay ilerlemesi için yapıyoruz. Siz kendi tasarımlarınızda farklı Layout kontrolleri tabii ki kullanabilirsiniz.



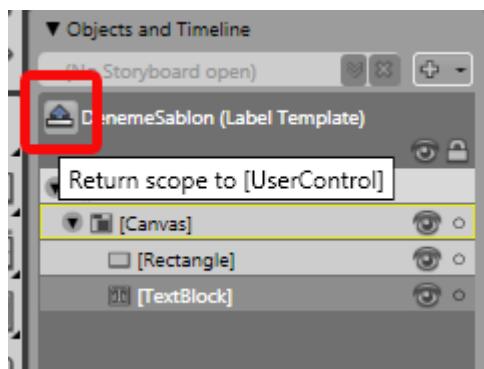
ControlTemplate tasarımımız bitmek üzere.

Tasarımımızda Canvas'in içerisinde bir **Rectangle** koyarak **Fill** özelliğine de **Radial** bir **GradientBrush** atadık. Rectangle'in önünde de bir TextBlock koyuyoruz. Label içerisinde metni gösterecek olan bu TextBlock kontrolü olacak. Label kontrolünün **Content** özelliğine verilen değerlerin otomatik olarak şablon içerisinde bu TextBlock'un **Content'ine** aktarılmasını sağlamalıyız. Bunu da ancak **TemplateBinding** ile yapabiliriz.



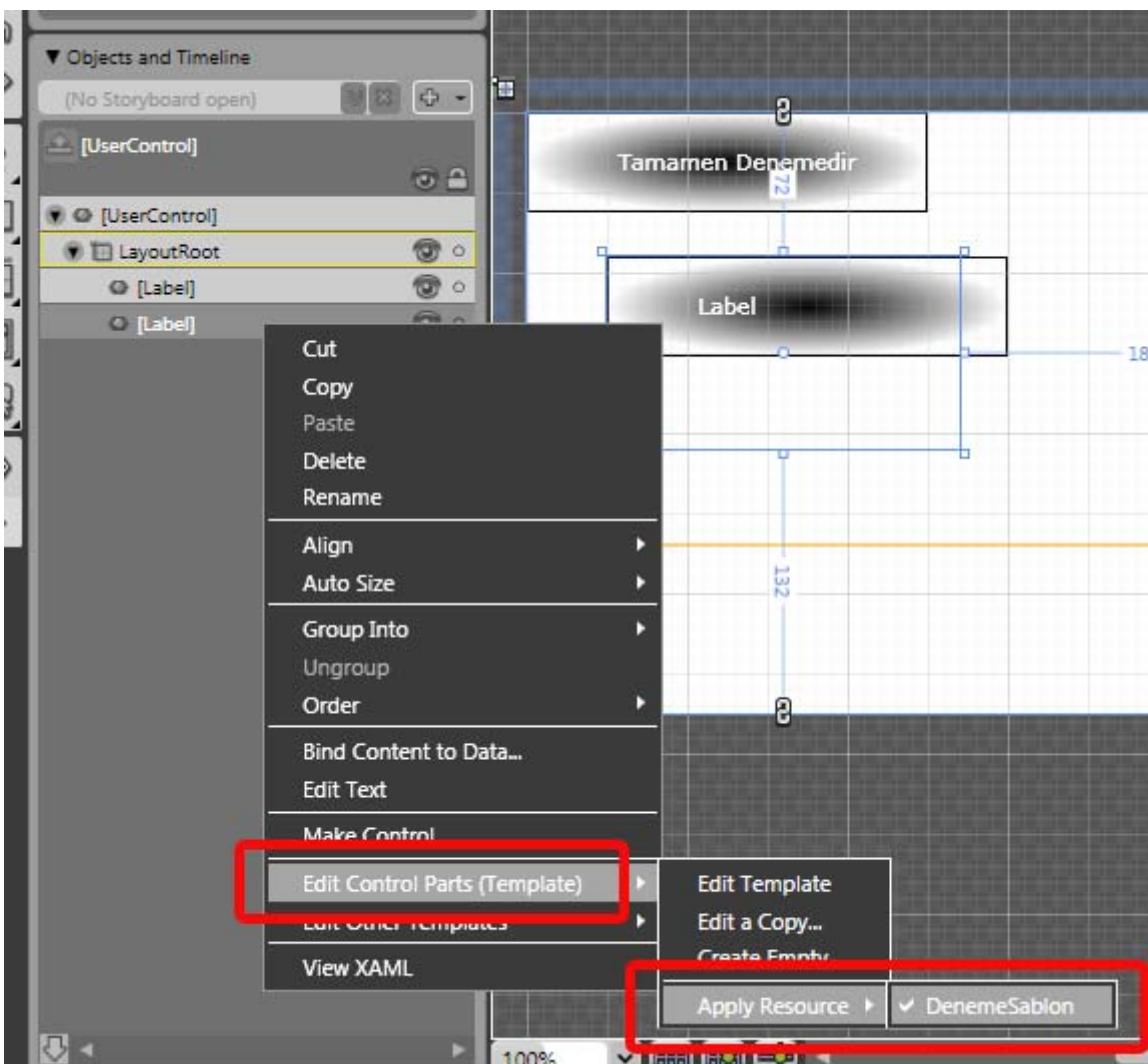
Blend arayüzünden TemplateBinding ayarlarını yapıyoruz.

Blend'in arayüzünde şablonumuzu tasarlarken **TextBlock** kontrolümüzü seçtikten sonra ekranın sağında kalan "Properties" sekmesinden söz konusu TextBlock'un **Content** özelliğini ayarladığımız yerin hemen yanındaki ufak kareye tıklıyoruz. Gelen menüden "**Template Binding**" komutunu verdikten sonra Blend bize ana kontrolün hangi özelliğinin şablonındaki bahsi geçen özelliğe bağlanacağını soruyor. Tabi biz de hemen **Content** özelliğini seçiyoruz. Böylece **Label** kontrolünün **Content** özelliğini şablonun içindeki **TextBlock'un Content** özelliğine bağlamış olduk.



Template tasarıımı modundan çıkışım.

Artık tasarımımızı tamamladığımızda göre şablon tasarım modundan çıkışıp ana uygulamaya geri dönebiliriz. Artık Label kontrolünün Content'ini değiştirdiğinizde ayarladığınız görsellik içindeki Textblock'a yerleştiğini görebilirsiniz. En güzel de yeni bir Label eklediğinizde aynı şablonu kullanmasını sağlayabilirsiniz.



Hazırladığımız şablonu istediğimiz Label kontrolünde kullanabiliriz.

Uygulamaniza yeni bir Label ekledikten sonra hazırladığınız şablonu bu Label üzerinde de kullanmak istiyorsanız doğrudan söz konusu Label'a sağ tuş ile tıklayarak gelen menüden "Edit Control Parts / Apply Resource" diyerek şablonunuzu adı ile seçebilirsiniz. Böylece

geri dönüp bu şablonda bir değişiklik yaptığınızda birden çok Label'da değişiklik yapmış olacaksınız. Merkezi yönetim :)

Yaptığımız tüm işlemlerin sonucunda aşağıdaki XAML kodu yaratılıyor.

[XAML]

```
<UserControl x:Class="SilverlightApplication2.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
    namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <UserControl.Resources>
        <ControlTemplate x:Key="DenemeSablon" TargetType="controls:Label">
            <Canvas>
                <Rectangle Height="50" Width="200" Stroke="#FF000000">
                    <Rectangle.Fill>
                        <RadialGradientBrush>
                            <GradientStop Color="#FF000000"/>
                            <GradientStop Color="#FFFFFF" Offset="1"/>
                        </RadialGradientBrush>
                    </Rectangle.Fill>
                </Rectangle>
                <TextBlock TextWrapping="Wrap" HorizontalAlignment="Stretch" Canvas.Top="17"
                    Canvas.Left="45.348" Foreground="#FFFFFF" Text="{TemplateBinding Content}"/>
            </Canvas>
        </ControlTemplate>
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White">
        <controls:Label HorizontalAlignment="Left" VerticalAlignment="Top"
            BorderBrush="#FFFF0000" BorderThickness="10,0,5,0" Template="{StaticResource
            DenemeSablon}" Content="Tamamen Denemedir"/>
    </Grid>
</UserControl>
```

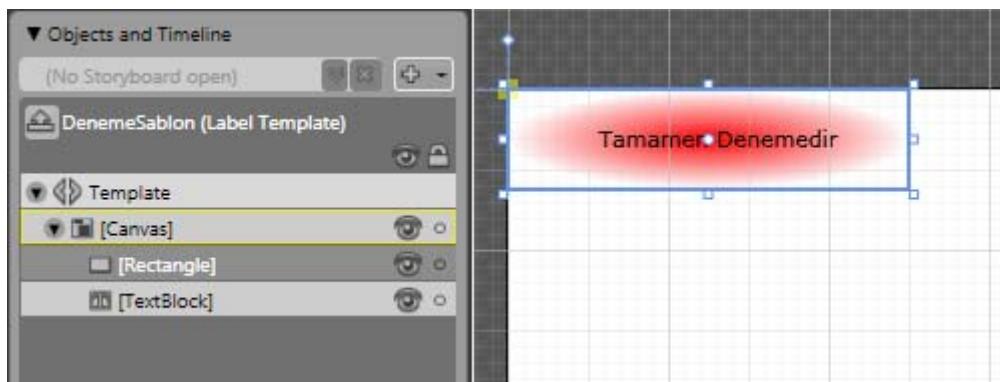
Yukarıdaki kodda özellikle önemli olan birkaç nokta var. İlk olarak nasıl olmuş da Label'ımız **UserControl.Resources** altındaki **DenemeSablon** adlı şablona bağlanmış? **TemplateBinding**'in kodu nasıl yazılmış? Tüm bunların cevaplarını kod içerisinde kalın yazılı bölgelerde bulabilirsiniz.

Label'a DataBinding nasıl yapılır?

Bir Label'e neden **DataBind** yapmak isteyelim? Sonuçta sadece **Text** göstermeyecek mi? Ve bu Text'in değişme şansı da yok ki **TwoWayBinding** vs gereksin? Tüm bu soruların cevabı aslında yukarıdaki **ControlTemplating** ile de alakalı. Bir **Label** düşünün ki kendisine Bind edilmiş nesneye göre şekil alabiliyor? Ne dersiniz?

Bir önceki örneğimizden devam edelim. Varsayıyalım ki Label'ımızda bir ürünün adını göstereceğiz fakat ürünün satış miktarına göre de Label'in arkasında Gradient'in (Aslında

Rectangle) şeffaflaşmasını veya görünür olmasını istiyoruz. Bunun için ilk olarak gelin kontrolümüzün tasarımını biraz değiştirelim.



DataBind yapacağımız Label kontrolünün şablonunu biraz değiştirdik.

Sadece renklerde biraz değişiklik yaptık. Böylece arkadaki Rectangle tamamen şeffaf olsa da yazı görünebilecek. Şimdi geçelim kod tarafına ve bu Label'a nasıl veri bağlarız ona bakalım.

İlk olarak Label'a bağlayacağım Urun'umuzu bir sınıf olarak programatik anlamda tanımlamamız gerekiyor.

[VB]

```

Public Class Urun

    Private PAdi As String
    Public Property Adi() As String
        Get
            Return PAdi
        End Get
        Set(ByVal value As String)
            PAdi = value
        End Set
    End Property

    Private PSatis As Double
    Public Property Satis() As Double
        Get
            Return PSatis
        End Get
        Set(ByVal value As Double)
            PSatis = value
        End Set
    End Property

End Class
  
```

[C#]

```
public class Urun
```

```
{
    public string Adi { get; set; }
    public double Satis { get; set; }
}
```

Gördüğünüz üzere **Urun** sınıfımızın iki **Property'sı** var. Bunlardan ilki ürünün ismini saklayacak olan **Adı**, diğer ise **Satis** miktarı. Özellikle **Satis** Property'sinin **Double** olarak tanımlandığına dikkat edelim. Herhangi bir **Convertor** kullanmadan bu Property'si **Bind** edeceğimiz için **Opacity** ile uyumlu şekilde 1 ile 0 arasında **Double** değerler taşımalı.

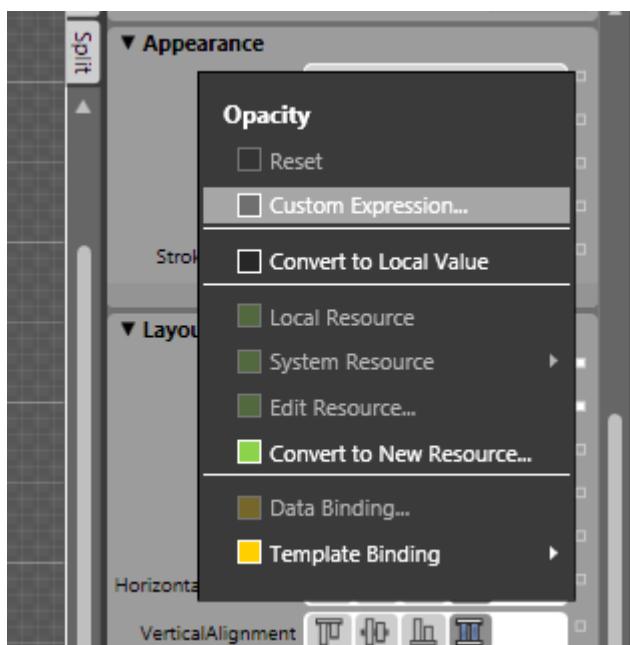
[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    Etiket.DataContext = New Urun() With {.Adi = "Bir ürün adı", .Satis = 0.5}
End Sub
```

[C#]

```
private void Page_Loaded(object sender, System.Windows.RoutedEventArgs e)
{
    Etiket.DataContext = new Urun { Adi = "Bir ürün adı", Satis = 0.5 };
```

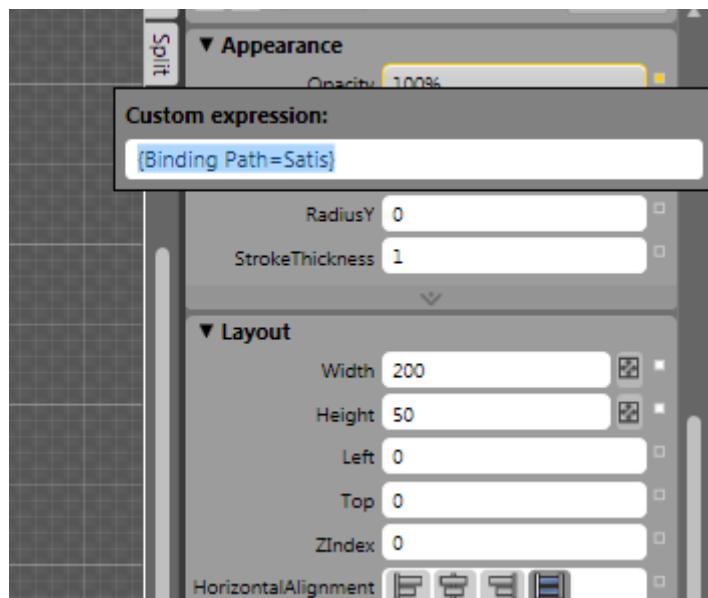
Kodumuzda basit bir şekilde yeni bir ürün yaratarak **Etiket** adındaki **Label** kontrolümüze **DataContext** özelliği üzerinden bağlıyoruz. Peki **Adı** ve **Satis** Property'leri **Label** içerisinde neleri etkileyeyecek? İşte bunun için ayrıca XAML tarafında ayarlar yapmamız gereklidir.



Rectangle'in Opacity'sinin DataBinding ayarlarını yapıyoruz.

Blend tarafında Label kontrolünün şablonuna tekrar geri dönüyoruz. Daha önce yarattığımız ControlTemplate'i açarak üzerinde değişiklikler yapmamız gerekiyor. İlk olarak şablon içerisinde Rectangle'i seçerek **Properties** tabından Opacity'sinin yanında ufak kareye

tıklıyoruz ve gelen menüden "**Custom Expression**" komutunu seçiyoruz. Böylece artık binding ile ilgili ayarı doğrudan el ile yazacağız.



DataBinding için Custom Expression yazıyoruz.

Yazdığımız Custom Expression ile artık **Label** kendi içindeki Rectangle'in **Opacity** değerinin kendisine verilen datadan geleceğini ve bağlanması gereken Property'nin adının da **Satis** olduğunu biliyoruz. Aynı işlemi **ControlTemplate** içerisindeki **TextBlock** için de yaparak bu sefer TextBlock'un **Content** özelliğini **Adı** Property'sine **{Binding Adı}** Custom Expression'i ile bağlamamız gereklidir.

Tüm bu işlemleri yaptıktan sonra uygulamayı çalıştırıldığınızda bağlılığınız verideki ürün adının TextBlock içeresine yerleştiğini ve gelen satış bilgisine göre de arkadaki Rectangle'in şeffaflığının belirlendiğini görebilirsiniz. Böylece gelen veriye göre görsellliğini değiştiren bir Label tasarlamış olduk.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde ToolTip Kontrolü ve Tooltip şablonları

Özellikle Windows uygulamalarında "erişilebilirlik" açısından çok önemli olan noktalardan biri de "İpucu" balonlarıdır. "Tooltip" olarak geçen bu sistem sayesinde herhangi bir kontrolün veya nesnenin üzerinde fare ile belirli bir süre durduğunuzda gerekli açıklama metni gösterilir. Böylece kullanıcı ufak yardım yönergeleri ile yönlendirilebilir.

Silverlight 2.0 Beta 2 için Tooltip işlevsellliğini otomatik olarak sağlayan bir **TooltipService** sınıfımız var. Gelin hemen bir örnekle konumuzu inceleyelim.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
    <Grid x:Name="LayoutRoot"
        Background="White">
        <Button Margin="35,41,170,124"
            Content="Button"
            x:Name="Dugme">
            <ToolTipService.ToolTip>
                <ToolTip Content="Bu bir düğme"></ToolTip>
            </ToolTipService.ToolTip>
        </Button>
    </Grid>
</UserControl>
```

Yukarıdaki kod içerisinde bir düğmeye **ToolTipService** ekleyerek service de bir **ToolTip** kontrolü aktarıyoruz. **ToolTip** kontrolümüzün **Content** özelliğine bir metin aktararak ip ucu olarak bu metnin gösterilmesini sağlıyoruz. Bu şekilde istediğiniz Silverlight kontrolüne **ToolTipService** ekleyebilirsiniz.



Basit bir Tooltip.

Eğer **Tooltip**'in görünümünü değiştirmek isterseniz **Background**, **FontSize**, **FontWeight**, **Foreground** gibi birçok özelliğe sahipsiniz. İsterseniz fon rengi gibi özelliklere farklı **Brush** yapıları da aktarabilirsiniz.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```

Width="400"
Height="300">
<Grid x:Name="LayoutRoot"
      Background="White">
<Button Margin="35,41,170,124"
       Content="Button"
       x:Name="Dugme">
<ToolTipService.ToolTip>
<ToolTip Content="Bu bir düğme">
<ToolTip.Background>
<LinearGradientBrush EndPoint="0.5,1"
                      StartPoint="0.5,0">
<GradientStop Color="#FF000000" />
<GradientStop Color="#FFFFFF"
               Offset="1" />
</LinearGradientBrush>
</ToolTip.Background>
<ToolTip.BorderBrush>
<LinearGradientBrush EndPoint="0.519999980926514,-0.0670000016689301"
                      StartPoint="0.519999980926514,0.899999976158142">
<GradientStop Color="#FF000000" />
<GradientStop Color="#FFFFFF"
               Offset="1" />
</LinearGradientBrush>
</ToolTip.BorderBrush>
</ToolTip>
</ToolTipService.ToolTip>
</Button>
</Grid>
</UserControl>

```

Gördüğü üzere rahatlıkla Tooltip'in görselliği değiştirilebiliyor. Bir Tooltip'in fonuna VideoBrush yerleştirmeye sizin hayallerinize bırakıyorum. Tooltip'e ait **VerticalOffset** ve **HorizontalOffset** özellikleri ile Tooltip'in fareden ne kadar uzakta gözükeceğini de ayarlayabilirsiniz.

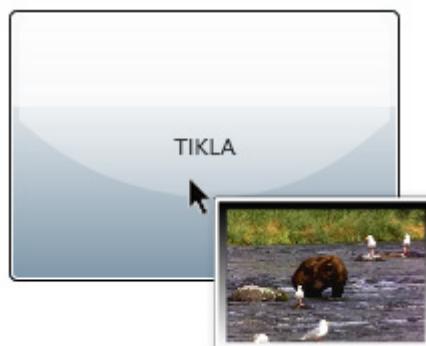


Özelleştirilmiş Tooltip

Biraz daha ileri giderek ToolTip'in Content özelliğine istersek farklı Silverlight kontrolleri de yerleştirebiliriz. Content içerisinde sadece bir Silverlight kontrolü yerleştirebilirsiniz fakat bu kontrolün içerisinde başka kontroller bulunabilir, yani Canvas veya Grid gibi Container

kontrollerini kullanarak farklı görsel şemalar yaratabilirsiniz. Aşağıdaki kod içerisinde biraz olayı abartarak ToolTip içerisinde bir video yerleştirdik.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
<Grid x:Name="LayoutRoot"
    Background="White">
<Button Margin="35,41,170,124"
    x:Name="Dugme"
    Content="TIKLA">
<ToolTipService.ToolTip>
<ToolTip>
<ToolTip.Content>
<MediaElement Height="66"
    VerticalAlignment="Bottom"
    Source="http://localhost:49167/SilverlightApplication6Web/Bear.wmv"
    Stretch="Fill" />
</ToolTip.Content>
</ToolTip>
</ToolTipService.ToolTip>
</Button>
</Grid>
</UserControl>
```



Tooltip içerisinde stream video!

Özel Tooltip!

Tooltip kontrolünün içerisinde gösterilecek Content'i değiştirmenin yanı sıra tamamen Tooltip'in görsel durumunu da değiştirebiliriz. Örneğin yuvarlak bir Tooltip yapılabilir. Bunun için ilk olarak Tooltip kontrolünün varsayılan Template yapısını MSDN üzerinden almamız gerekiyor. Adres şu şekilde; [http://msdn.microsoft.com/en-us/library/cc296244\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc296244(VS.95).aspx)

Elimizdeki **Template** içerisindeki ControlTemplate tagları arasındaki hersey bizim Tooltip'in görüntü şablonunu oluşturuyor. Buradaki Binding'lere dokunmamaya çalışıyoruz. Özellikle

ContentPresenter'ı silmememiz gerekiyor çünkü ToolTip'in Content'ına verilen içerik Template içerisinde buraya yerleştiriliyor olacak.

Gelin farklı bir şey yapalım ve kendi içine konan metni Scrollbar ile gösterebilen bir Tooltip oluşturalım. Bunun için hemen ControlTemplate'i düzenlememiz gerekiyor.

```
<UserControl x:Class="SilverlightApplication6.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
<UserControl.Resources>
    <Style x:Key="ToolTipStili"
        TargetType="ToolTip">
        <Setter Property="Foreground"
            Value="#FF313131" />
        <Setter Property="FontSize"
            Value="11" />
        <Setter Property="Background">
            <Setter.Value>
                <LinearGradientBrush x:Name="ToolTipBackground"
                    StartPoint="0,0"
                    EndPoint="0,1">
                    <LinearGradientBrush.GradientStops>
                        <GradientStop Color="#FFF9FAFA"
                            Offset="0" />
                        <GradientStop Color="#FFEDF1F4"
                            Offset="0.37259" />
                        <GradientStop Color="#FFE2E8EF"
                            Offset="0.372881" />
                        <GradientStop Color="#FFC3C9CD"
                            Offset="1" />
                    </LinearGradientBrush.GradientStops>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="ToolTip">
                    <ScrollViewer Height="30"
                        Width="100"
                        Background="WhiteSmoke">
                        <ScrollViewer.Content>
                            <ContentPresenter Content="{TemplateBinding Content}"
                                ContentTemplate="{TemplateBinding ContentTemplate}"
                                Cursor="{TemplateBinding Cursor}"
                                FontFamily="{TemplateBinding FontFamily}"
                                FontSize="{TemplateBinding FontSize}"
                                FontStretch="{TemplateBinding FontStretch}"
                                FontStyle="{TemplateBinding FontStyle}">
                        </ContentPresenter>
                    </ScrollViewer>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</UserControl.Resources>
<UserControl>
```

```

        FontWeight="{TemplateBinding FontWeight}"
        Foreground="{TemplateBinding Foreground}"
        HorizontalContentAlignment="{TemplateBinding
HorizontalContentAlignment}"
        Padding="{TemplateBinding Padding}"
        TextAlign="{TemplateBinding TextAlign}"
        TextDecorations="{TemplateBinding TextDecorations}"
        TextWrapping="{TemplateBinding TextWrapping}"
        VerticalContentAlignment="{TemplateBinding
VerticalContentAlignment}" />
    </ScrollViewer.Content>
</ScrollViewer>
</ControlTemplate>
<Setter.Value>
</Setter>
</Style>
</UserControl.Resources>
<Grid x:Name="LayoutRoot"
      Background="White">
    <Button Margin="35,41,170,124"
           x:Name="Dugme"
           Content="TIKLA">
        <ToolTipService.ToolTip>
            <ToolTip Style="{StaticResource ToolTipStili}"
                   Content="Herhangi bir açıklama metni burada olabilir.">
                </ToolTip>
            </ToolTipService.ToolTip>
        </Button>
    </Grid>
</UserControl>

```

Yukarıdaki kodumuz içerisinde ControlTemplate'in doğrudan içine bir **ScrollViewer** yerleştirdik ve Template içerisinde ContentPresenter'i da **ScrollViewer'ın** Content değerine aktardık. Böylece bu ToolTip kontrolüne aktarılan tüm Content bir ScrollViewer içerisinde gösterilecek.



Custom Tooltip

Siz de bu şekilde farklı ToolTip kontrolleri tasarlayarak uygulamalarınıza kullanabilirsiniz.

Silverlight 2.0 içerisinde VisualStateManager kullanımı.

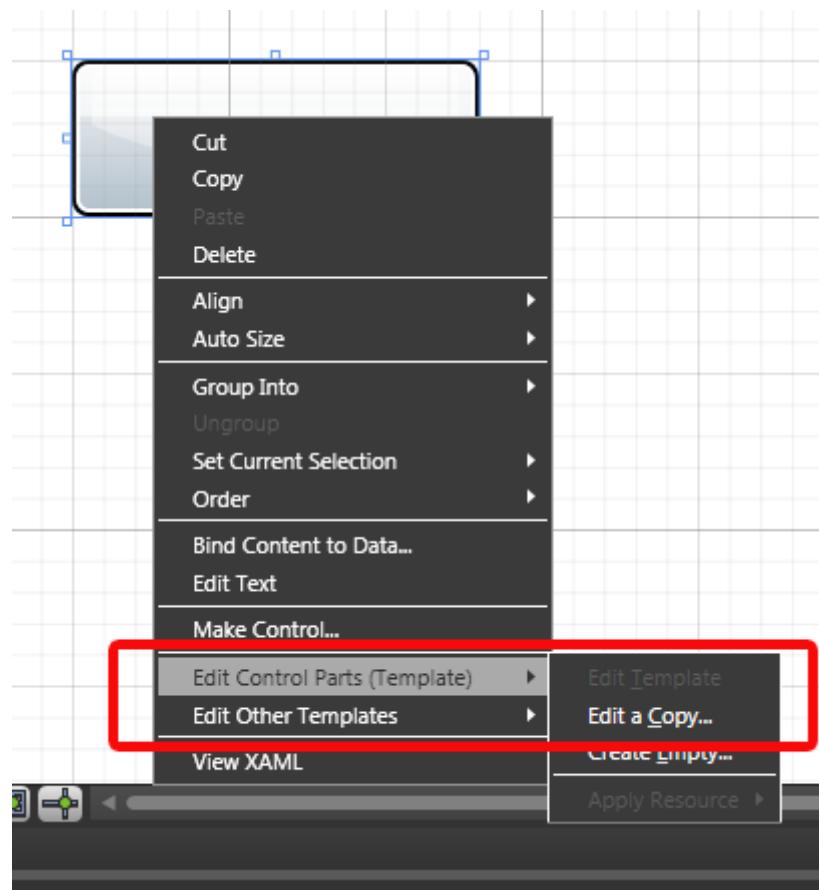
Bugüne kadar Silverlight içerisinde herhangi bir düğme veya farklı bir kontrolün değişik durumlarında ayrı şekiller alması için maalesef birbirinden bağımsız animasyonlar hazırlayarak tek tek gerekli eventler ile koddan eşleştirmemiz gerekiyordu. Örneğin bir düğmenin üzerine gelindiğinde kırmızı olması ve bunu ufak bir animasyonla yapması için ilk önce animasyonumuzu hazırlıyor sonra da düğmenin MouseOver event'ına event-handlar kodumuzu bağlayarak animasyonu elle çalıştırıyorduk. Bu şekilde çok sayıda proje yapıldı.

Oysa artık Silverlight 2.0 Beta 2 ile beraber karşımıza yepyeni bir sistem geliyor : **VisualStateManager!**

Artık bir nesnenin farklı durumlarında nasıl gözükeceğini önceden tanımlanmış durumlar için **Expression Blend 2.5** arayüzünden ayarlayabiliyorsunuz. Silverlight bununla kalmayıp bu farklı durumlar arasında geçiş efektlerini kendisi otomatik olarak yaratarak otomatik olarak oynatabiliyor. Gelin biraz işin içine girip nasıl yapıldığına göz atalım.

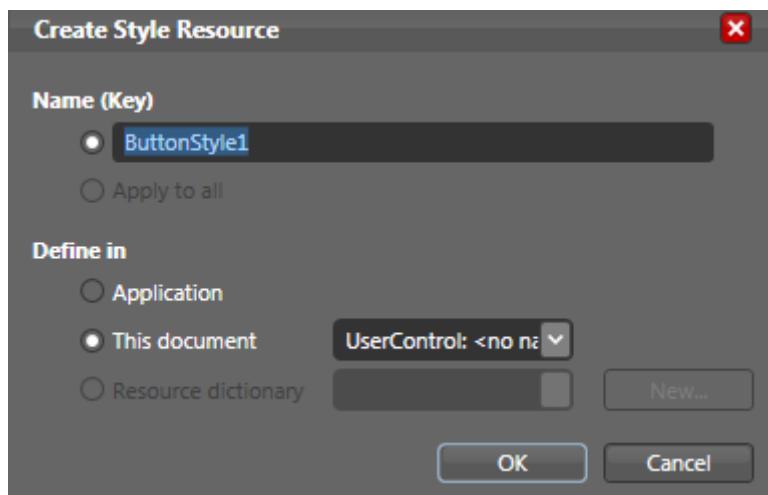
Animasyonlu bir Button...

Expression Blend 2.5 içerisinde sahneye bir Button yerleştiriyoruz. Hedefimiz bu Button'un farklı durumlarda fon rengini değiştirmek. Örneğin fare ile üzerine geldiğimizde düğme kırmızı olsun. Peki bunu Expression Blend 2.5 içerisinde nasıl yapacağız?



Düğmemizi Template'ini düzenliyoruz.

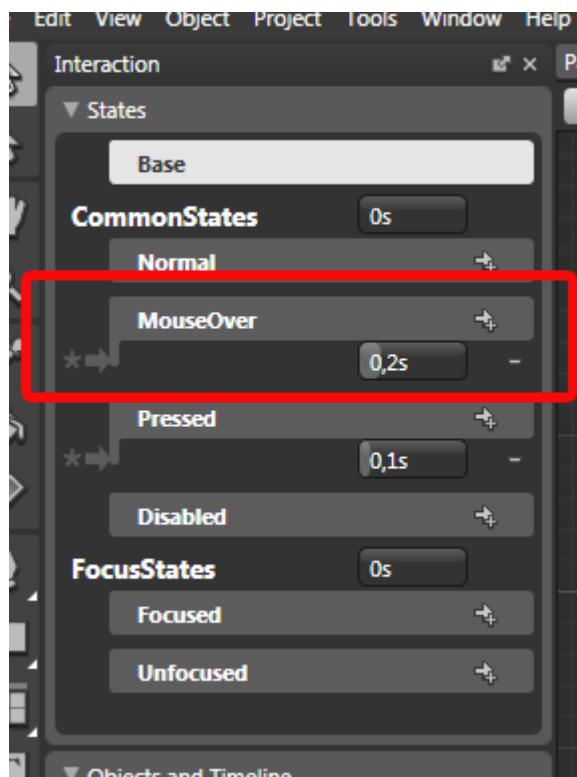
İlk olarak sahnedeki düğmemizin "Template"'ini düzenlememiz lazım. Düğmemizin zaten hali hazırda bir tasarımlı ve animasyonları var. Bunları projenizi çalıştırıldığınızda görebilirsiniz. Düğmenin üzerine geldiğinizde rengi değişecektir. Biz tamamen sıfırdan bir düğme tasarlamayağımız için "**Edit Control Parts**" menüsünden "**Edit a Copy**" seçeneğini kullanarak var olan tasarımın bir kopyasını alarak yola devam edeceğiz. Eğer sıfırdan bir düğme tasarlamak isterseniz bu ekranda "Create Empty" demeniz gerekecektir.



Düğme Template'imize bir isim verelim.

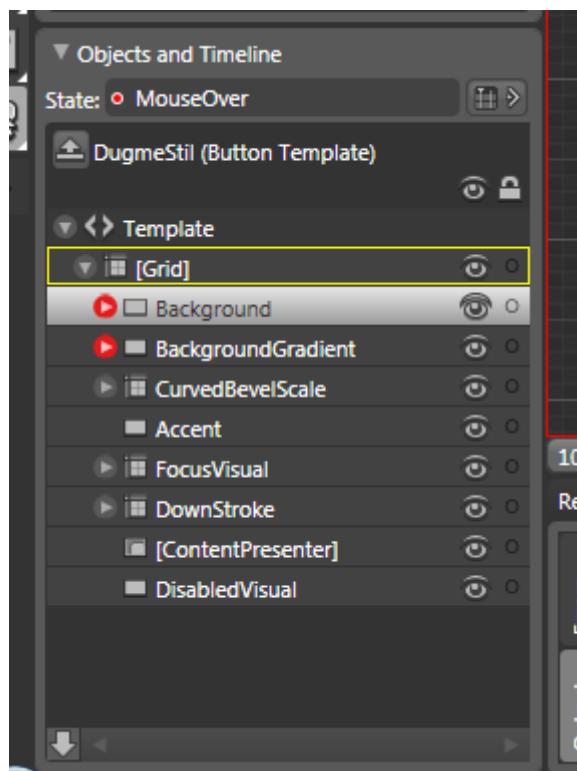
Düğmemizin "Template"'ini yaratırken Expression Blend bize minik bir soru soruyor. Şablonumuz bir "Resource" olarak yaratılacak ve projemizde istediğimiz bir isimle istediğimiz bir yere konabilecek. Bu ekranda eğer "**Application**" seçeneğini seçersen hazırlayacağımız şablonla ilgili tüm özellikleri tanımlayan XAML kodu projenin içerisinde App.XAML dosyası içerisinde saklanacaktır. Böylece şablonumuz aynı projedeki tüm Page'lerde kullanılabilecek. Eğer varsayılan ayar olan "**This document**" seçeneğini tercih ederseniz şablonla ilgili XAML kodları içerisinde çalıştığınız Page.XAML dosyasına kaydedilecektir. Bu durumda bu şablonu diğer Silverlight sayfalarınızda kullanmanız daha zor olacaktır. Son bir seçenek ise harici bir Resource.xaml dosyasında tüm kaynakları (şablonları) saklamak olabilir. "**Resource dictionary**" seçeneği üzerinden yeni bir XAML dosyası yaratarak saklayacağınız şablonlarınızı böylece birden fazla projede kullanabilirsiniz.

Biz şimdilik "**This document**" diyerek şablonumuza isim olarak da "**DugmeStil**" değerini verelim.



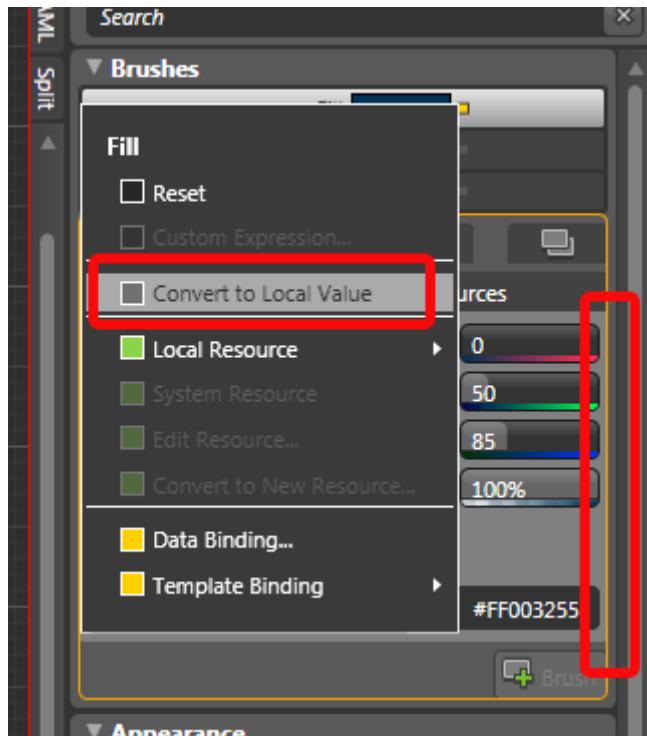
Düğmemizin farklı durumlarında tasarım değişiklikleri yapacağız.

Yukarıdaki ekran görüntüsünde de inceleyebileceğiz üzere düğmemizin şablonuna girdiğimiz anda düğmemizle ilgili farklı durumların listelendiği bir "Interaction" penceresi Expression Blend içerisinde ekleniyor. Buradan herhangi bir durumu fare ile seçtikten sonra düğme üzerinde yapmak istediğiniz görsel değişiklikleri gerçekleştirebilirsiniz.



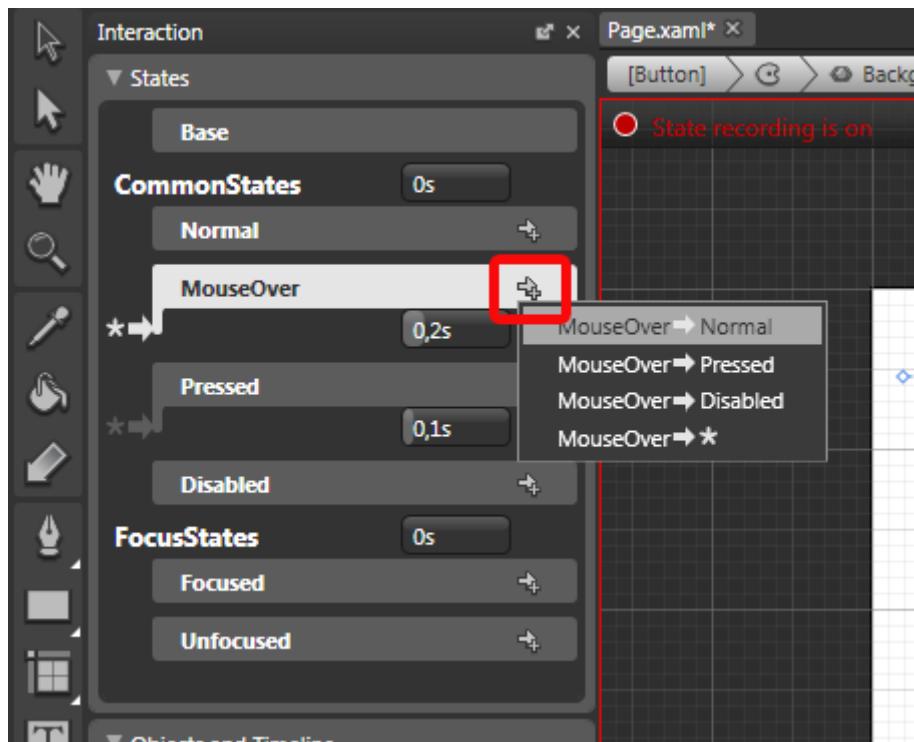
Düğmemizin fon gölge rengini değiştiriyoruz.

Biz örneğimizde düğmenin üzerindeki mavi gölgenin rengini değiştirmeye çalışalım. Standart bir düğmenin fare ile üzerine geldiğinde alt tarafa doğru ek mavi bir gölge geliyor. Expression Blend içerisinde baktığımızda bu gölge "**Background**" adı verilen bir Rectangle ile sağlanmış. Söz konusu Rectangle'in Fill renki aşağıdaki gibi DataBound gözükmektedir.



DataBound görsel değeri lokal bir değere aktarıyoruz.

Yukarıdaki görselde Rectangle'in Fill değerinin tanımlandığı bölgenin turuncu bir çizgi ile çevrelendiği görülebilirsiniz. Bu durum söz konusu özelliğin bir sistem değişkenine bağlı olduğu anlamına geliyor. Biz bu değeri değiştirmek istediğimiz için Fill özelliğinde değerin bir kopyasını almak amacıyla "Fill"in hemen yanındaki ufak kareye tıklayıp gelen menüden "**Convert to Local Value**" diyoruz. Artık söz konusu değer bizim şablon içerisinde taşındı, böylece üzerinde istediğimiz değişikliği yapıp gölgeyi kırmızı hale getirebiliriz. Renk paletinden yeni bir renk seçmeniz yeterli.

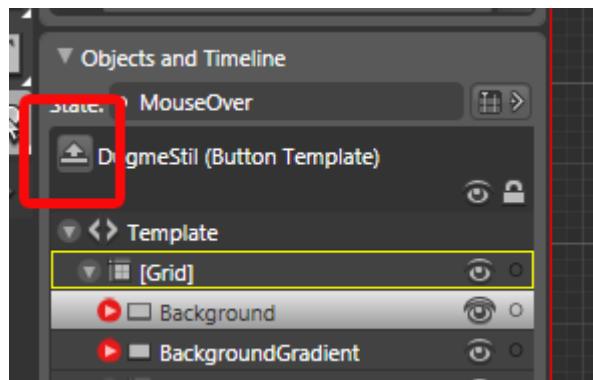


Transition efektlerini ve süreleri ayarlayalım.

Normal şartlarda bir düğmenin üzerine geldiğinde yeni görsel duruma geçiş esnasında bir animasyon çalıştırılıyor, fakat fareyi düğmenin üzerinden çektiğinizde normal haline "anında" geçiyor yani herhangi bir animasyon çalıştırılmıyor. MouseOver durumundan Normal'e geçişte de bir animasyon çalıştırılmasını istiyorsanız hemen Blend arayüzünde "MouseOver" durumunun yanındaki oka tıklayarak gelen menüden "MouseOver > Normal" seçeneğini seçerek bu geçiş esnasında da bir animasyon yaratılmasını istediğiniz belirtebilirsiniz. Böylece kırmızı hale geçişte ve eski hale dönüştür birer animasyon oynatılarak geçişler sağlanacaktır. Bu animasyonların sürelerini de hemen yanındaki 0.2s değerlerini düzenleyerek değiştirebilirsiniz.

Dikkat dikkat!

"MouseOver" veya "MouseLeave" gibi durumlarda görsel değişiklikler yaratırken aklınızda olması gereken ufak bir ipucu var. Animasyon uygulayacağınız nesnelerin her durumda sahnede yer alması gerekiyor. Yani sadece **MouseOver** durumunda gözükecek olan bir Rectangle'in **Normal** durumdan **MouseOver** duruma bir animasyonda kullanılması mümkün olmaz. Görünmezden görünüre doğru bir animasyon tasarlayacak olsanız da hedef nesnenin hem başlangıç hem de son durumlarında sahnede bulunması gerekiyor. Tek yapacağınız başlangıç noktasında söz konusu nesneyi görünmez ayarlamaktır.



Şablon düzenleme modundan geri çıkalım.

Arka planda neler oluyor?

Maalesef Expression Blend'in yaratmış olduğu tüm XAML kodunu burada yapıştırmayacağız, yaklaşık 180 satırlık bir kod söz konusu. Onun yerine özellikle bizim yaptığıımız işlemleri ilgili kısmını buraya alarak bir inceleyelim.

```

<vsm:VisualStateGroup.Transitions>
  <vsm:VisualTransition Duration="00:00:00.5000000"
    To="MouseOver"/>
  <vsm:VisualTransition Duration="0:0:0.1"
    To="Pressed"/>
  <vsm:VisualTransition Duration="00:00:00.5000000"
    From="MouseOver"
    To="Normal"/>
</vsm:VisualStateGroup.Transitions>
<vsm:VisualState x:Name="Normal"/>
<vsm:VisualState x:Name="MouseOver">
  <Storyboard>
    <ColorAnimationUsingKeyFrames Duration="0"
      Storyboard.TargetName="BackgroundGradient"
      Storyboard.TargetProperty="(Shape.Fill).(GradientBrush.GradientStops)[3].(GradientStop.Color)">
      <SplineColorKeyFrame KeyTime="0"
        Value="{StaticResource MouseOverLinearBevelDarkEndColor}" />
    </ColorAnimationUsingKeyFrames>
    <ColorAnimationUsingKeyFrames BeginTime="00:00:00"
      Duration="00:00:00.0010000"
      Storyboard.TargetName="Background"
      Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)">
      <SplineColorKeyFrame KeyTime="00:00:00"
        Value="#FFFF0000" />
    </ColorAnimationUsingKeyFrames>
  </Storyboard>
</vsm:VisualState>
```

Kod içerisinde de görebileceğiniz üzere **MouseOver** durumu içerisinde toplam iki adet animasyon var. Bunlardan biri **Background'u** diğeride **BackgroundGradient'i** değiştiriyor.

Adı **Background** olan animasyon bizim biraz önce Blend arayüzünden hazırladığımız animasyonun ta kendisi. Peki bu animasyon ne zaman nasıl çalıştırılacak? İşte tüm bunlara **VisualStateGroup.Transitions** tagları arasındaki bilgiler karar veriyor. Kaç saniyede hangi durumdan hangi duruma göre animasyonlar oynatılacağı doğrudan bu taglar arasında belirlenmiş durumda. Oynatılacak animasyonlar da zaten her durumun ismiyle ilişkilendirilmiş şekilde kod içerisinde duruyor.

Koddan geçişleri çalıştmak istersek?

Bazı durumlarda hazırlanan bu animasyonları kod ile de çalıştmak isteyebilirsiniz. Örneğin kullanıcının tıklaması gereken bir düğme vardır, fakat tıklamıyordur :) bu durumda sanki **MouseOver** olmuş gibi bir düğmeyi parlatıp söndürmek hoş olabilir.

[VB]

```
VisualStateManager.GoToState(Dugme, "MouseOver", true)
```

[C#]

```
VisualStateManager.GoToState(Dugme, "MouseOver", true);
```

Yukarıdaki kod ile **Dugme** adındaki düğmemizin **MouseOver** durumuna geçmesini ve arada gerekli animasyonun da oynatılmasını sağlamış oluyoruz. Eğer son Boolean parametreyi **False** olarak verirseniz geçiş animasyonu oynatılmaksızın söz konusu duruma geçilecektir.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 içerisinde Download penceresi açtırmak.

Silverlight 2.0 Beta 2 içerisinde istemci makinedeki herhangi bir dosyaya ulaşmakla ilgili örneklerimizde **Open File Dialog** nesnesini detayları ile incelemiştik. Tam zitti bir senaryoda da kullanıcıya sistemine kaydedebileceği bir dosya vermek isteyebiliriz. Bu gibi bir çözüm için maalesef Save File Dialog gibi bir kontrole sahip değiliz. O nedenle biraz farklı bir taktik uygulayarak kullanıcıya vermek istediğimiz dosyayı ilk olarak sunucuya göndererek kaydedeceğiz sonrasında da sunucudan dosyayı tekrar istemcinin alarak kaydedebilmesini sağlayacağız. Bu senaryo ilk bakışta çok saçma gibi gözüke de aslında dosya yaratma işlemini sunucu tarafında yaparsanız herhangi bir performans sorunu olmayacağındır. Gönül isterdi ki dosyayı tamamen istemicde Silverlight ile yaratarak kullanıcıya verebilelim. Fakat maalesef şimdilik en azından Beta 2 içerisinde böyle bir şansımız yok.

Peki herşeyi yaptık, dosyamızı **Silverlight 2.0 Beta 2** ile yarattık, sunucuya bir web servisi aracılığı ile gönderdik ve kaydettik. Bundan sonrasında bu dosyayı Silverlight içerisinde istemicideki kullanıcıya nasıl vereceğiz. Kullanacağımız taktik aslında epeyce basit. Tüm tarayıcılarda kullanıcıyı dosya downloadu için dosyanın bulunduğu adrese yönlendirirseniz hemen "Download" penceresi açılacaktır. Kullanıcının içerisinde bulunduğu web sayfası hala görüntülenmeye devam edecektir. Yani aslında siz kullanıcıyı başka bir adrese yönlendiriyor olsanız da söz konusu adreste bir sayfa olmadığı için görüntüde bir değişiklik olmayacağındır ve hedef adresste dosyanın download işlemi başlayacaktır. Biz de Silverlight tarafından bu taktikten faydalananız ve sayfanın adresini kullanıcının indirmesini istediği adres ile değiştirerek tarayıcının "Save File Dialog" açmasını sağlayacağız.

[VB]

Dim Adres As Uri

```
Uri.TryCreate(System.Windows.Browser.HtmlPage.Document.DocumentUri, New
Uri("indir.zip", UriKind.Relative), Adres)
```

[C#]

Uri Adres;

```
Uri.TryCreate(System.Windows.Browser.HtmlPage.Document.DocumentUri, new
Uri("tekliyazi.zip", UriKind.Relative), out Adres);
```

İlk olarak kullanıcıyı yönlendirmek istediğimiz adresi oluşturmamız gereklidir. Bunun için **Uri** sınıfındaki **TryCreate** metodunu kullanacağız. Bu metod bizden bir base **Uri** ve **Relative Uri** alarak ikisini birleştiriyor. **Base Uri** için içerisinde bulunduğu sayfanın adresini **System.Windows.Browser.HtmlPage.Document.DocumentUri** ile veriyoruz, relative **Uri** için yeni bir **Uri** değişkeni yaratıp download edilecek olan dosyanın adını veriyoruz. Son olarak da almak sonucun aktarılacağı yeni **Uri** değişkenimizi referans olarak aktarıyoruz.

[VB]

```
CType(System.Windows.Browser.HtmlPage.Document.GetProperty("location"),
System.Windows.Browser.ScriptObject).SetProperty("href", Adres.ToString)
```

[C#]

```
((System.Windows.Browser.ScriptObject)System.Windows.Browser.HtmlPage.Document.GetProperty("location")).SetProperty("href", Adres.ToString());
```

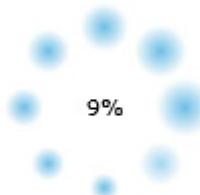
Simdi yapmamız gereken ise sayfanın adresini bir önceki adımda elde ettiğimiz adres ile değiştirmek. İlk olarak **System.Windows.Browser.HtmlPage.Document.GetProperty** ile sayfadaki dokümanın location özelliğini alıyoruz. Sonrasında da dokümanı bir **ScriptObject**'e cast ederek **SetProperty** ile **href** özelliğini değiştiriyoruz. Aslında bunu biraz JavaScript'teki **document.location.href** özelliğine benzetebilirsiniz. Artık kodumuzu çalıştırduğumızda Silverlight içerisinde herhangi bir düğmeye basıldığında otomatik olarak tarayıcının "Save File Dialog" penceresi açılacak ve kullanıcı istediği dosyayı Silverlight tarafından indirebilecek.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 için özel ön yükleme ekranları geliştirmek. (PreLoader)

Silverlight 2.0 uygulamaları hazırladığınızda sunucu tarafına yükleme işlemini yaptığınız gibi XAP dosyanızın büyülüğüne göre Silverlight Runtime tarafından otomatik olarak bir ön yükleme sistemi gösterilecektir. Kullanıcılar sitenizi ziyaret ettiğinde XAP dosyasının istemciye inme sürecini gösteren bu yükleme göstergelerini isterseniz rahatlıkla özelleştirebilir ve değiştirebilirsiniz. Tabi tüm bunları XAP dosyanız dışında daha XAP dosyası yüklenmeden bir şekilde yine Silverlight ile yapabiliyor olmamız gereklidir.



Silverlight ile beraber gelen standart ön yükleme animasyonu.

Aslında bu yükleme ekranını değiştirirken belki de eski Silverlight 1.0 günlerini biraz hatırlayacaksınız. XAP dosyası yüklenmeden önce bu şekilde bir yükleme animasyonu gösterebilmemiz için animasyonu oluşturacak ayrı bir XAML ve download durumunu kontrol edecek ayrı bir JavaScript koduna ihtiyacımız var.

Görsel kısmı halledelim...

İlk olarak ön yükleme işlemini gösterecek olan animasyonu ve görsel öğeleri düzenleyelim. Tüm bu görsel ögelerin tabii ki XAP dosyası dışında olması gereklidir. Bu durumda tek bir alternatif kalıyor, o da hazırlayacağımız tüm XAML kodunun harici bir dosya olarak sunucuda tutulması. Eğer Visual Studio ile bir Silverlight projesi yarattığınız büyük ihtimal ile yanında bir de ASP.NET siteniz olacaktır. İşte tam da o ASP.NET sitesine bir XAML dosyası eklemeliyiz. Bunu ister Visual Studio içerisinde yapın ister Expression Blend içerisinde, önemli olan projeleri karıştırmayarak XAML dosyasını kesinlikle web sitesinde tarafında bir dosya olarak yaratmanız gereklidir.

Maalesef bu noktadan sonra Expression Blend bize pek yardımcı olamayacak çünkü Web sitesindeki XAML dosyasının tam olarak ne tür bir projeye ait olduğunu algılayamayacaktır. O nedenle çoğu kodu elle yazmak zorunda kalacağız.

[XAML]

```
<StackPanel Orientation="Horizontal" xmlns="http://schemas.microsoft.com/client/2007"
           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
           x:Name="LayoutRoot" Background="White" Height="104">
    <Rectangle Height="33" x:Name="Progress" Fill="#FF00FF00"/>
    <TextBlock Height="18" Text="Yükleniyor..."/>
</StackPanel>
```

Yukarıdaki kodumuz bizim önyükleme ekranının tasarımını olacak. Tasarım oldukça basit; bir StackPanel içerisinde TextBlock ile Rectangle kullanarak işi çözüyoruz. Tabi siz örneklerinizde farklı tasarımlar kullanabilirsiniz. Bizim örneğimizde ekranda sürekli

"Yükleniyor..." yazacak ve yanında da 0 pikselden 100 piksele doğru genişleyecek olan bir Rectangle yer alacak. Böylece kullanıcıya standart görselden farklı bir şekilde XAP dosyasının yüklenmesine ait süreci göstermiş olacağız.

Bu kod içerisinde en önemli nokta Rectangle nesnesinin adının Progress olması. İleriki aşamalarda yazacağımız JavaScript kodları ile bu nesneyi bularak gerekli değişiklikleri yapacağız.

Ortamı hazırlayalım...

Silverlight uygulamamızı sayfaya yerleştirdiğimiz OBJECT tagları içerisinde bazı ek parametreler tanımlamamız gerekiyor. Böylece Silverlight Runtime XAP dosyasının indirirken nerede ve nasıl bir progress göstermesi gerektiğini bilebilecek. Gelin uygun bir OBJECT tagına göz atalım.

[XAML]

```
<object id="SL" data="data:application/x-silverlight-2," type="application/x-silverlight-2" width="100%" height="100%">
    <param name="splashscreensource" value="Scene1.xaml"/>
    <param name="onSourceDownloadProgressChanged"
    value="onSourceDownloadProgressChanged" />
    <param name="source" value="ClientBin/SilverlightApplication5.xap"/>
    <param name="onerror" value="onSilverlightError" />
    <param name="background" value="white" />
    <param name="minRuntimeVersion" value="2.0.31005.0" />
    <param name="autoUpgrade" value="true" />
    <a href="http://go.microsoft.com/fwlink/?LinkId=124807" style="text-decoration:none;">
        
    </a>
</object>
```

Gördüğünüz üzere kod içerisinde kalın olarak yazılı üç farklı nokta var. Bunlardan birincisi OBJECT taglarına vermiş olduğumuz ID olan SL ismi. OBJECT taglarını bu şekilde isimlendirmek zorundayız çünkü birazdan JavaScript ile bu OBJECT taglarına yanı Silverlight uygulamamıza ulaşmamız gerekecek. Son olarak ayarlamamız gereken iki tane de parametre bulunuyor. Bu parametrelerden ilki XAP dosyası yüklenmeden önce ve yüklenirken gösterilecek olan XAML dosyamızın tam yolu olacak. **splashscreensource** adındaki bu parametreye biz örneğimizde basit bir şekilde **Scene1.xaml** değerini verdik. Bu dosya içerisinde bir önceki adımda yazdığımız ve içerisinde **Progress** Rectangle'ının bulunduğu XAML yer alıyor. Son olarak **onSourceDownloadProgressChanged** event'ına da yine aynı isimde bir JavaScript event-listener ekleyerek XAP dosyasının download durumu ile ilgili değişiklikleri takip edebiliyoruz. Gelelim yazacağımız bu JavaScript event-listener koduna...

Biraz da JavaScript...

Bir önceki adımda hazırladığımız OBJECT tagının parametrelerinden birinde **onSourceDownloadProgressChanged** adında bir event-listener tanımlamıştık. Bu event-listener içerisinde ilk olarak sayfamızdaki OBJECT tagını ve içerisindeki Silverlight uygulamamızı bulmamız gerekiyor. Bu yapı Silverlight 1.0 günlerinden hatırladığımız bir yapı. Aşağıdaki kodumuzda SL adını verdigimiz OBJECT taglarını bulduktan sonra içeriğine bakarak **findName** ile **Progress** adındaki Rectangle'ımızı bulmuş olduk.

[JavaScript]

```
function onSourceDownloadProgressChanged(sender, eventArgs) {
    var Progress = document.getElementById("SL").content.findName("Progress");
    if (eventArgs.progress)
        Progress.Width = eventArgs.progress * 100;
    else
        Progress.Width = eventArgs.get_progress() * 100;
}
```

Son olarak söz konusu Rectangle'in genişliğini elimizdeki yükleme durumuna göre değiştireceğiz. Yükleme durumunu bize 1 üzerinden decimal olarak verecek olan şey tarayıcı tipine göre ya bir Property ya da bir metod olacağı için bir IF kontrolü ile onu de kontrol ederek gerekli işlemi yapıyoruz.

Artık Silverlight Runtime ile beraber gelen ön yükleme animasyonlarından kurtulabilir ve kendi özgür iradeniz ile :) kendi tasarımlarınızı kullanabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ile Analog Saat Uygulaması

Silverlight 2.0 Beta 2 kullanarak analog bir saat gösterimi yapacağımız bu yazı içerisinde StoryBoard kullanımını ve StoryBoard'ların belirli zamanlara özel olarak konumlandırılmasını inceleyeceğiz. Gelin ilk olarak işin görsel tarafını halledelim ve Expression Blend 2.5 ile bir saat görseli yaratalım.

Saatimizde toplam üç adet dikdörtgene ihtiyacımız var. Bu dikdörtgenler saatin akrep, yelkovan ve saniye göstergesini temsil edecek. Ayrıca saatin ana gövdesini oluşturacak bir de Ellipse kullanacağız.

```
<UserControl x:Class="SilverlightApplication16.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300"
    xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid x:Name="LayoutRoot" Background="White">
        <Ellipse HorizontalAlignment="Stretch" Margin="0,0,120,20"
            VerticalAlignment="Stretch" Fill="#FFFF0000" Stroke="#FF000000" x:Name="Govde"/>
        <Rectangle Height="80" HorizontalAlignment="Left" Margin="136,59,0,0"
            VerticalAlignment="Top" Width="8" Fill="#FFFFFF" Stroke="#FF000000"
            x:Name="Saat"/>
        <Rectangle Height="107" HorizontalAlignment="Left" Margin="136,32,0,0"
            VerticalAlignment="Top" Width="8" Fill="#FF00C6FF" Stroke="#FF000000"
            x:Name="Dakika"/>
        <Rectangle Height="131" HorizontalAlignment="Left" Margin="136,8,0,0"
            VerticalAlignment="Top" Width="8" Fill="#FFD600FF" Stroke="#FF000000"
            x:Name="Saniye"/>
    </Grid>
</UserControl>
```

Bu noktadan sonra animasyonlarımızı hazırlamaya başlayalım. Saniyeyi gösterecek olan saat kolunun toplam 60 saniye içerisinde 360 derece dönmesi gerekiyor. Buna uygun animasyon yaratmadan önce tüm dikdörtgenlerimizin merkez noktasını saatin ortasına gelecek şekilde düzenleyelim. Böylece dikdörtgenler saatin merkez noktasının etrafında dönecektir.

```
<UserControl x:Class="SilverlightApplication16.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300"
    xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid x:Name="LayoutRoot" Background="White">
        <Ellipse HorizontalAlignment="Stretch" Margin="0,0,120,20"
            VerticalAlignment="Stretch" Fill="#FFFF0000" Stroke="#FF000000" x:Name="Govde"/>
```

```

<Rectangle Height="80" HorizontalAlignment="Left" Margin="136,59,0,0"
VerticalAlignment="Top" Width="8" RenderTransformOrigin="0.5,1" Fill="#FFFFFF"
Stroke="#FF000000" x:Name="Saat"/>
<Rectangle Height="107" HorizontalAlignment="Left" Margin="136,32,0,0"
VerticalAlignment="Top" Width="8" RenderTransformOrigin="0.5,1" Fill="#FF00C6FF"
Stroke="#FF000000" x:Name="Dakika"/>
<Rectangle Height="131" HorizontalAlignment="Left" Margin="136,8,0,0"
VerticalAlignment="Top" Width="8" RenderTransformOrigin="0.5,1" Fill="#FFD600FF"
Stroke="#FF000000" x:Name="Saniye"/>
</Grid>
</UserControl>

```

Uygulamamızın XAML kodunun son hali yukarıdaki şekilde. Hemen saniye koluna ait animasyonu hazırlayalım. **SaniyeAnim** adını vererek yeni bir **Storyboard** yaratıyoruz ve söz konusu StoryBoard'un 60. saniyesinde saniye adındaki dikdörtgeni dönüş açısını 360 derece olarak tanımlıyoruz.

```

<Storyboard x:Name="SaniyeAnim" RepeatBehavior="Forever">
  <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Saniye"
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[2].(RotateTransform.Angle)">
    <SplineDoubleKeyFrame KeyTime="00:01:00" Value="360"/>
  </DoubleAnimationUsingKeyFrames>
</Storyboard>

```

Yukarıdaki animasyon içerisinde de gördüğünüz üzere ilk olarak animasyonun **RepeatBehavior** özelliğine **Forever** değerini vererek animasyonun sürekli tekrar edeceğini belirtiyoruz. Sonrasında **Saniye** dikdörtgenini hedef alan animasyon dikdörtgenin dönüş açısını bir dakika içinde 360 dereceye getiriyor.

Saniye kolu için yaptığımız animasyonlarla aynı mantıkta dakika ve saat kolları için de animasyonlar düzenleyeceğiz. Dakika kolu için düzenlediğimiz animasyon dakika kolunu 60 dakika içerisinde 360 derece döndürecek.

```

<Storyboard x:Name="DakikaAnim" RepeatBehavior="Forever">
  <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Dakika"
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[2].(RotateTransform.Angle)">
    <SplineDoubleKeyFrame KeyTime="01:00:00" Value="360"/>
  </DoubleAnimationUsingKeyFrames>
</Storyboard>

```

Son olarak saat kolu için düzenleyeceğimiz animasyonda da saat kolunu 12 saat içerisinde 360 derece döndürecekiz.

```

<Storyboard x:Name="SaatAnim" RepeatBehavior="Forever">
  <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
Storyboard.TargetName="Saat"

```

```

Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[2].(RotateTransform.Angle)">
    <SplineDoubleKeyFrame KeyTime="12:00:00" Value="360"/>
</DoubleAnimationUsingKeyFrames>
</Storyboard>

```

Tüm bu animasyonlar çalıştığında saatimiz doğru bir şekilde işliyor olacak. Fakat esas mesele Silverlight uygulaması ilk açıldığında tüm saat kollarını doğru başlangıç noktalarına getirmek. Bunu yapabilmek için StoryBoard'ların **Seek** metodundan faydalanağız. Seek metodu ile istediğimiz bir StoryBoard'u kendi içinde istediğimiz zaman aralığına getirebiliyoruz.

```

SaniyeAnim.Begin()
DakikaAnim.Begin()
SaatAnim.Begin()

```

```

SaniyeAnim.Seek(New TimeSpan(0, 0, Now.Second))
DakikaAnim.Seek(New TimeSpan(0, Now.Minute, Now.Second))
SaatAnim.Seek(New TimeSpan(Now.Hour, Now.Minute, Now.Second))

```

Yukarıdaki kodun başında tüm animasyonları hemen başlatıyoruz sonra da her animasyonu uygun başlangıç noktasına getirmek için **Seek** metodunu kullanıyoruz. Seek metoduna verdigimiz **TimeSpan** parametresini yaratırken her bir kola uygun değerleri aktarıyoruz. Saniye koluna sadece içerisinde bulduğumuz zamanın saniyesini, dakika koluna dakikayı ve saat koluna da saat bilgisini veriyoruz ve animasyonlarımız o konumdan gelerek oynamaya devam ediyor.

Saat uygulamamızı tamamladık, hepинize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ile Oyun Programlama'ya Giriş

Silverlight 2.0 ile beraber .NET programlama geldiğinde göre hemen aklımıza gelen konulardan biri de "Oyun Programlama" oluyor. Silverlight içerisinde var olan **Storyboard** yapısını kullanarak oyun programlamaya çalışığınız noktada ise "*Bu iş böyle olmaz*" diyerek bırakacağınız kesin. Maalesef StoryBoard yapısı çoğu işi kolaylaştırırsa da ve oyun programlama içerisinde belirli bir yeri olsa da işin tamamını halletmek için yeterli değil.

Oyun programlamada çoğu zaman kare kare çizim yapabilmemiz ve nesnelerin koordinatları ile tek tek ilgilenebilmemiz gereklidir. Daha net bir tanımla aslında ekranda çizimi bizim kod ile yaptırmamız gereklidir. Bu noktada **Silverlight 2.0 Beta 1** ile beraber gelen **DispatcherTimer** nesnesini kullanarak bu makalemizde ufak bir örnek yapacağız.

Hedefimiz uygulamamızda bir kare içeresine bir top yerleştirerek bu topun kare içerisinde duvarlara çarparak sürekli gezmesini sağlamak. "*Bu ne biçim oyun?*" diyebilirsiniz. Tabii ki makale sonunda elimizde hazır bir oyun olmayacak, amacımız oyun programlamada kullanılanının bir modelini yaratmak.

İlk olarak gelin uygulamamızın görsel kısmını tamamlayalım. Yarattığımız yeni Silverlight 2.0 uygulamasına bir Ellipse yerleştirerek en üst sol köşeye konumlandıralım. Ellipse'in adı **Top** olsun. İşlemleri tamamladığımızda elde edeceğimiz XAML aşağıdaki gibi olacak.

```
<UserControl x:Class="SilverlightApplication15.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="#FFF4E80C">
        <Ellipse Height="30" HorizontalAlignment="Left" Margin="0,0,0,0"
            VerticalAlignment="Top" Width="30" Fill="#FFFF0000" Stroke="#FF000000"
            x:Name="Top"/>
    </Grid>
</UserControl>
```

XAML kodumuz hazır olduğuna göre artık arkaplana geçip programlamaya başlayabiliriz. Uygulamamızda kullanılmak üzere toplam dört adet global değişken tanımlayacağız.

```
Dim Konum As New System.Windows.Media.TranslateTransform()
Dim Timer As New System.Windows.Threading.DispatcherTimer
Dim XDegisim = 5
Dim YDegisim = 5
```

Bu değişkenlerden ilki olan Konum bizim için sürekli olarak topun ekranın sol üst köşesinden ne kadar sağda ve aşağıda olacağını saklayacak. Konum değişkeninin tipi **TranslateTransform** şeklinde. Bu değişken tipi herhangi bir nesnenin **RenderTransform** grubuna aktardığımızda nesnenin X ve Y koordinatlarında yer değiştirmesini sağlıyor. Eğer Expression Blend 2.5 içerisinde bir nesne yaratır ve herhangi bir animasyon ile konumunu değiştirirseniz nesnenizin XAML kodunun aşağıdaki hale geldiğini görebilirsiniz.

```

<Rectangle Height="80" HorizontalAlignment="Left" Margin="136,93,0,0"
VerticalAlignment="Top" Width="155" Fill="#FFFFFF" Stroke="#FF000000"
x:Name="rectangle" RenderTransformOrigin="0.5,0.5">
<Rectangle.RenderTransform>
<TransformGroup>
<ScaleTransform/>
<SkewTransform/>
<RotateTransform/>
<TranslateTransform X="10" Y="10" />
</TransformGroup>
</Rectangle.RenderTransform>
</Rectangle>

```

Expression Blend'in bizim için yarattığı **Rectangle** koduna da baktığımızda aslında yapıyı anlayabiliyoruz. Rectangle içerisinde yerleştirilen bir **RenderTransform** yapısındaki **TransformGroup** içerisinde **TranslateTransform** nesnesi yer alıyor. Bu nesnenin X ve Y özellikleri de Rectangle'ın ne kadar yer değiştireceğine karar veriyor. Biz bu aynı yapıyı kendi **Top** nesnemiz için programatik olarak kullanacağız.

Şimdi tanımladığımız global değişkenlere geri dönelim.

```

Dim Konum As New System.Windows.Media.TranslateTransform()
Dim Timer As New System.Windows.Threading.DispatcherTimer
Dim XDegisim = 5
Dim YDegisim = 5

```

İkinci değişkenimiz bir **DispatcherTimer** tipinde bir Timer nesnesi. Bu nesne aslında Windows programlamadan bildiğimiz Timer'dan farklı değil. Bu Timer'in da bir **Tick** durumu ve **Interval'i** var. Biz örneğimizde Interval'i çok düşük vererek ekranda çizim yapmak için bu nesneyi kullanacağız.

Diğer iki global değişkenimizin aslında Top'umuzun bir defada ne kadar yer değiştireceğine karar veriyor. Bu değerleri büyüterek topun hızını artırabilir veya tam tersine azaltarak topunu hızını da yavaşlatılabilirsiniz.

```

Timer.Interval = New TimeSpan(0, 0, 0, 0, 15)
AddHandler Timer.Tick, AddressOf Timer_Tick
Timer.Start()

```

Silverlight uygulamamız sayfada ilk açıldığında hemen Timer nesnemizin Interval'ini 15 milisaniye olarak ayarlıyoruz. Her 15 milisaniyede bir ekranın Top'un yeni konumunu belirleyeceğiz böylece animasyon oluşturmuş olacağız. Doğal olarak Timer nesnemizin **Tick** durumunu da yakalamamız lazım. Onun için bir de event-handler tanımıyoruz. Event-Handler kodumuza birazdan bakacağınız. Son olarak Timer'in **Start** metodu ile sayacı başlatıyoruz.

Sıra geldi Timer'in her bir Tick durumunda topun yeni pozisyonunu ayarlamaya. Aslında kullanacağımız kod çok basit.

```

Konum.X += XDegisim
Konum.Y += YDegisim

```

Top.RenderTransform = Konum

Global değişkenimiz olan **Konum** değişkeninin X ve Y değerlerine değişim miktarlarını ekleyerek Top'un **RenderTransform'una** eşitleyerek Top'un yeni konumunu almasını sağlıyoruz. Fakat ortada bir sorun var; Top'un duvarlardan sekmesi lazım. Kabaca baktığımızda aslında yapmamız gereken dört kontrol var. Eğer top alt kenara çarptıysa artık XDegisim miktarı 5 değil de -5 olmalı. Böylece top tekrar yukarıya doğru gitmeye başlamalı. Aynı şekilde her kenar için bu kontrolün yapılması lazım. Eğer top sağ kenara çarptıysa artık YDegisim miktarı 5 değil de -5 olmalı. Tüm bu kontrolleri yaptığımızda aşağıdaki gibi bir kod elde ediyoruz.

```
If Konum.X = Me.LayoutRoot.ActualWidth - Top.Width Then
    XDegisim = -XDegisim
End If
If Konum.Y = Me.LayoutRoot.ActualHeight - Top.Height Then
    YDegisim = -YDegisim
End If
If Konum.X = 0 Then
    XDegisim = Math.Abs(XDegisim)
End If
If Konum.Y = 0 Then
    YDegisim = Math.Abs(YDegisim)
End If
Konum.X += XDegisim
Konum.Y += YDegisim
Top.RenderTransform = Konum
```

IF kontrollerimizi sırasıyla incelersek; ilkinde eğer topun konumu sahnemizdeki ana Canvas'ımız olan **LayoutRoot'un** genişliğinden **Top'un** genişliğini de çıkararak aldığımız koordinata ulaşmış ise hemen **XDegisim** miktarını - hale getiriyoruz. Böylece Top geri gitmeye başlayacak. Aynı şekilde ikinci IF kontrolünde de Y yönünde aynı kontrolü yapıyoruz. Tabi bir de işin diğer tarafı var, yani değişim miktarı eksi iken top ekranın üstüne veya soluna doğru gidecek. Bu durumda da üçüncü ve dördüncü IF kontrolümüz ile Top bu pozisyonlara gelmiş ise değişim miktarlarının mutlak değerini alarak pozitif hale çeviriyoruz.

Bu hali ile uygulamamızı çalıştırduğumızda topumuzun duvarlardan sekerek sonsuza dek gezecektir. Kodumuzun son hali aşağıdaki şekilde;

Partial Public Class Page
Inherits UserControl

```
Public Sub New()
    InitializeComponent()
End Sub

Dim Konum As New System.Windows.Media.TranslateTransform()
Dim Timer As New System.Windows.Threading.DispatcherTimer
Dim XDegisim = 5
Dim YDegisim = 5
```

```

Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Timer.Interval = New TimeSpan(0, 0, 0, 0, 15)
    AddHandler Timer.Tick, AddressOf Timer_Tick
    Timer.Start()
End Sub

Private Sub Timer_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
    If Konum.X = Me.LayoutRoot.ActualWidth - Top.Width Then
        XDegisim = -XDegisim
    End If
    If Konum.Y = Me.LayoutRoot.ActualHeight - Top.Height Then
        YDegisim = -YDegisim
    End If
    If Konum.X = 0 Then
        XDegisim = Math.Abs(XDegisim)
    End If
    If Konum.Y = 0 Then
        YDegisim = Math.Abs(YDegisim)
    End If
    Konum.X += XDegisim
    Konum.Y += YDegisim
    Top.RenderTransform = Konum
End Sub

Private Sub Page_MouseLeftButtonDown(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseButtonEventArgs) Handles Me.MouseLeftButtonDown
    If Timer.IsEnabled Then
        Timer.Stop()
    Else
        Timer.Start()
    End If
End Sub
End Class

```

Ek olarak bir de sayfanın genelinde **MouseLeftButtonDown** durumunu yakalayarak Silverlight uygulamasına tıklandığında Timer'in durdurulabilerek tekrar başlatılabilmesi için gerekli kodu da yazarsak örneğimiz artık tamamlanmış demektir. Artık farklı animasyon yapmak, farklı koordinat kontrolleri yapmak hatta matematiksel koordinat hesaplamalarını ortaokula dönüp biraz da trigonometri ekleyerek lise bilgisi ile de vektörel hesaplamalara çevirmek tamamen size kalmış.

Hepinize kolay gelsin.

Daron YÖNDEM

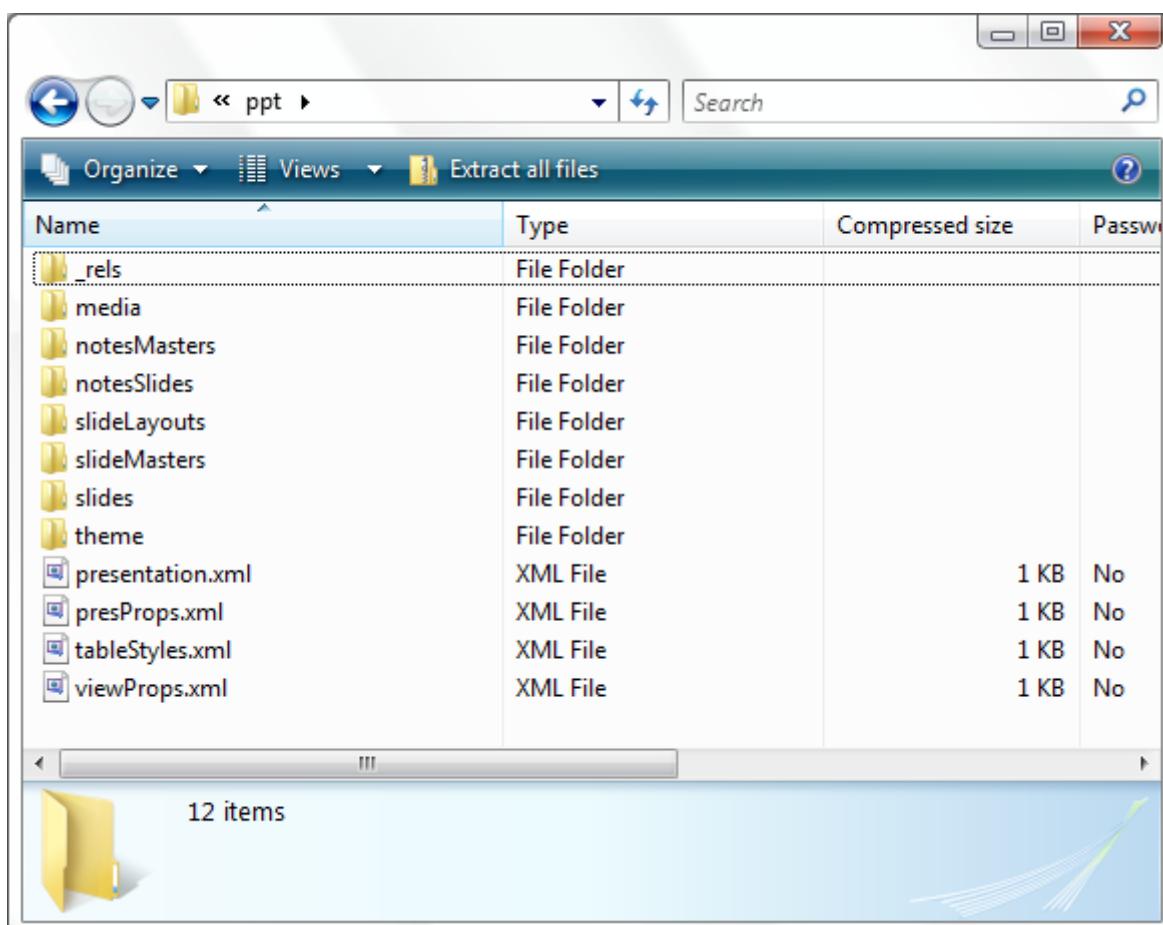
Silverlight 2.0 ile Powerpoint dosyalarının Thumbnail'lerini göstermek.

Her programcının hayatında en az birkaç defa Office dosya formatları ile ilgilendiği projeler olmuştur. Bu durum ister yeni bir Word dosyası yaratmak olsun, ister bir Excel dosyasından veri çekmek olsun **OpenXML** öncesi ciddi sıkıntılardan çektığımız bir gerçek. En basit bir sebep olarak binary formattılı dokümanlar ile uğraşmak zorunda kaldığımızı söyleyebiliriz. Office 2007 ile beraber tanıştığımız ve sonlarına X harfleri eklenen dosya uzantıları aslında OpenXML'in geldiğinin bir işaretiydi. İyi ki geldin diyerek yavaş yavaş konumuza geçiş yapalım.

Daha önceki yazılarımda Silverlight 2.0 ile istemci tarafındaki dosyalara [OpenFileDialog](#) ile ulaşabildiğimizi görmüştük. Yine aynı sistemi kullanarak bu sefer amacımız bir Powerpoint dosyasını açarak söz konusu dosyanın Thumbnail görüntüsünü Silverlight içerisinde göstermek. Aslında çok şanslıyız, neden mi? Çünkü OpenXML var.

OpenXML'in bize faydası ne?

OpenXML formatındaki her dosya aslında bir ZIP dosyasıdır. Eğer dosyanın uzantısını ZIP olarak değiştirirseniz aslında içerişine girerek tüm detayları görebilirsiniz.



OpenXML dosyalarının içinde her şey XML formatında, açık seçik karşımızda.

Yukarıdaki basit bir Powerpoint dosyasının içeriği görebilirsiniz. Gördüğünüz üzere aslında her şey ayrı ayrı XML dosyaları şeklinde kaydedilmiş. Yani bir Office 2007 dosyası yaratmak özünde XML'ler yaratarak ZIP'lemekten farklı değil. Hele bir de bu dosyaları açarak

programlarınızda göstermek istiyorsanız LINQ2XML'in gücünden de faydalananarak çok hızlı çözümler geliştirmeniz mümkün.

Konumuza dönersek; bize Powerpoint dosyasının Thumbnail'i lazımdı. Bunun için XML'ler içerisinde ilk slayt ile ilgili bilgileri alarak gerekli çizimleri hazırlayabiliriz fakat buna gerek yok. Çünkü her PowerPoint dosyası zaten işletim sisteminde gösterilecek olan Thumbnail'i kendi içinde barındırır. OpenXML formatı sağ olsun, dosyamızın içini biraz karıştırdığımızda görüyoruz ki ZIP dosyası içerisinde **docProps** adındaki bir klasörün içinde **thumbnail.jpeg** adında bir JPG dosyası bulunuyor. İşte bu dosya bizim Silverlight tarafında göstereceğimiz thumbnail görselini içeriyor.

ZIP'ten resmi nasıl alacağız?

Yine daha önceki bir yazımdayken [ZIP dosyalarını Silverlight tarafında açabilmeyi](#) yollarına değinmiştim. Hatta daha da ileri giderek Silverlight 2.0 XAP dosyaları da birer ZIP dosyası olduğu için onları açmıştık. Şimdi ise sıra başka bir ZIP dosyası açmaya geldi, Powerpoint dosyaları.

İlk olarak Powerpoint dosyasını açarak Thumbnail'i gösterecek olan uygulamamızın tasarımını basit bir şekilde yapalım. Sahnede bir düğme ve bir de Image nesnesi olsun. Düğmeye başında kullanıcıdan bilgisayarından bir Powerpoint dosyası seçmesini isteyelim ve sonrasında da seçili dosyanın Thumbnail'ini Image nesnesi içerisinde gösterelim.

```
<UserControl x:Class="SilverlightApplication58.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Height="32" HorizontalAlignment="Left" Margin="24,32,0,0"
    VerticalAlignment="Top" Width="104" Content="Button" x:Name="btnDosyaAc"/>
        <Image Margin="160,32,80,140" x:Name="imgOnIzleme"/>
    </Grid>
</UserControl>
```

Sıra geldi gerekli işlemleri sırası ile yapacak olan kodumuzu yazmaya.

[VB]

```
Dim DosyaAc As New OpenFileDialog
DosyaAc.Filter = "PowerPoint 2008 Dosyaları (*.pptx)|*.pptx"
DosyaAc.Multiselect = False
DosyaAc.ShowDialog()
```

[C#]

```
OpenFileDialog DosyaAc = new OpenFileDialog();
DosyaAc.Filter = "PowerPoint 2008 Dosyaları (*.pptx)|*.pptx";
DosyaAc.Multiselect = false;
DosyaAc.ShowDialog();
```

İlk olarak yukarıdaki şekilde Silverlight tarafından kullanıcıdan bilgisayarından bir Powerpoint dosyası seçmesini isteyelim. Sonrasında bize dosyanın Stream'i gelecek ve biz de içerisinde istedigimiz JPEG dosyasını almaya çalışacağız. Kodumuz içerisinde istemcideki dosyalara ulaşmak için bir **OpenFileDialog** kullanıyor, filtresini de sadece PPTX'leri gösterecek şekilde ayarlıyoruz.

[VB]

```
If DosyaAc.SelectedFile IsNot Nothing Then
    Dim Foto As New Imaging.BitmapImage()
    Dim Gelen = Application.GetResourceStream(New
Windows.Resources.StreamResourceInfo(DosyaAc.SelectedFile.OpenRead(), Nothing), New
Uri("docProps/thumbnail.jpeg", UriKind.Relative)).Stream
    Foto.SetSource(Gelen)
    imgOnIzleme.Source = Foto
End If
```

[C#]

```
if (DosyaAc.SelectedFile != null) {
    System.Windows.Media.Imaging.BitmapImage Foto = new
System.Windows.Media.Imaging.BitmapImage();
    System.IO.Stream Gelen = Application.GetResourceStream(new
System.Windows.Resources.StreamResourceInfo(DosyaAc.SelectedFile.OpenRead(), null),
new Uri("docProps/thumbnail.jpeg", UriKind.Relative)).Stream;
    Foto.SetSource(Gelen);
    imgOnIzleme.Source = Foto;
}
```

Bu noktada Silverlight ile beraber gelen Resource yapısını kullanacağımız ve sanki harici bir kaynak açmış gibi ZIP dosyasını açtıracagız. Sonuçta hedefin bir Powerpoint dosyası olup olmadığı önemli değil. Bizim istedigimiz **ZIP** dosyasındaki **docProps** klasöründe bir dosyayı almak. Kullandığımız taktik daha önceki Silverlight 2.0 XAP dosyalarından dosya almak için kullandığımız taktik ile birebir aynı. **StreamResource** olarak **OpenFileDialog**'dan gelen dosyanın içerisinde JPEG dosyasını alıyor ve **GetResourceStream** ile de dosyayı çekiyoruz. Çektigimiz dosyayı daha önce **Foto** adında tanımladığımız **BitmapImage** değişkenine **SetSource** ile atayarak sonunda da delimizdeki resmi sahnedeki **Image** nesnesine aktarıyoruz.

Böylece Powerpoint dosyasında Thumbnail'i bularak Silverlight 2.0 içerisinde çok kolay bir şekilde gösterebildik. Bu aynı taktiği ASP.NET ile de kullanarak gerektiğinde sunucu taraflı olarak Powerpoint dosyalarında **Thumbnail'ler** yaratmak **OpenXML** sayesinde çocuk oyunçaguina dönüşmüş durumda.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ile Twitter ve TwitXR Combo Widget

Bugün sizlerle yeni bir açık kaynak kodlu **Silverlight 2.0** uygulamamı paylaşacağım. Hatırlarsınız bundan aylar önce [Silverlight 1.0 ile bir Twitter Widget](#) hazırlamış ve sizlerle kaynak kodlarını paylaşmıştım. Şimdi de 2.0 ile yeni bir Widget hazırladım. Bu sefer Widget'in [TwitXR](#) desteği gibi ilginç özellikleri var

Bir Widget hikayesi...

Silverlight 2.0 ile geliştirme yapmanın 1.0'a kıyasla çok sayıda avantajı var fakat unutmayalım ki artık .NET tarafındayız ve JavaScript ile olduğu kadar low level kod yazmıyoruz. Ne demek istedigimi makale boyunca daha net anlayacağınızdan eminim. Peki size neler anlatacağım?

Uygulamanın içindeki kodların açıklamalarını zaten kodlar içerisindeki yorum satırlarında bulabilirsiniz. Yorum satırlarını İngilizce yazdım çünkü uygulamayı yurt dışı ile de paylaşacağım. Önemli olan ve benim özellikle değerim istedigim noktalar bir Widget hazırlarken karşılaştığım duruma özel sorunlarla ilgili. Gelin daha fazla konuşmak yerine sorumlara el atalım.

Twitter API saçmalığı!

Twitter'ın çok güzel bir XML API yapısı var. Sorunlardan ilki bu API için ister Flash ister Silverlight hiç fark etmez **ClientAccessPolicy** dosyası konmamış, yani Cross-Domain-Request yasak! Durum böyle olunca sunucu tarafında bir proxy oluşturarak veriyi kendi sunucunuza alıp kendi Widget'inize aktarmanız gerekiyor, ama bunu da yapamıyoruz çünkü söyle bir saçmalık mevcut; istemci başına bir saatte 70 request sınırı var. Eeee? Tüm requestleri benim Web Server'im yapacak sonuçta tüm ziyaretçiler için, 70 kesinlikle kabul edilebilir bir sınır değil. Büyük ihtimal ile Windows uygulamaları falan düşünülmüş.

Sonuç olarak Silverlight 1.0 Widget'da kullandığımız **Remote Script Injection** ile Cross-Domain-Request taktığını Silverlight 2.0 da da kullanmamız gerek. Dinamik olarak sayfaya bir **script** tagı ekleyip twitter sitesinden JavaScript alıp sitemizde çalıştırıyorduk. Parametre olarak da bizim istediğimiz veriler geliyordu. İşte sorular;

Silverlight 2.0 ile dinamik Script tagları sayfaya nasıl eklenir?

```
'Insert our dynamic Script Tag to get Cross Domain Data
Dim MyDoc As HtmlDocument = HtmlPage.Document
Dim ScriptTag = MyDoc.CreateElement("script")
ScriptTag.SetAttribute("type", "text/javascript")
ScriptTag.SetAttribute("src", "http://twitter.com/statuses/user_timeline/" &
InitParams("twitterid") & ".json?callback=TwitterIncData&count=" & InitParams("count"))
MyDoc.Body.AppendChild(ScriptTag)
```

Yukarıdaki kod ile rahatlıkla dinamik olarak bir **Script** tagı yaratıp özelliklerini de ayarlayıp içerisinde bulunduğu sayfanın **Body'sine** ekleyebiliyoruz. Böylece uzaktaki dosya çağrılmaktır. Twitter adresinin içindeki parametrelerin konumuzla şimdilik alakası yok. Ama bu adres üzerinden gönderdiğimiz **callback** parametresi Twitter'dan data gelince sayfamızda

çalıştırılacak olan JavaScript'in ta kendisi. Peki bu durumda data gelince çağrılan JavaScript'ten bizim Silverlight 2.0'ın nasıl haberi olacak?

JavaScript'ten Silverlight 2.0 fonksiyonları nasıl çağrırlar?

Buyurun makalemi okuyun : <http://daron.yondem.com/tr/PermaLink.aspx?guid=a1426eb0-7120-4a66-9d5c-de5027fd59ed>

Şimdi veri geldi elimize ama gelen veri JSON! Bunu nasıl olacak da anlaşılabilir bir hale çevireceğiz ve .NET nesneleri şeklinde kullanabileceğiz?

JSON verisi Silverlight 2.0'a nasıl alınır?

Buyurun bir makale daha: <http://daron.yondem.com/tr/PermaLink.aspx?guid=457fbb28-892e-4a37-b7d3-cb297d97020b>

Ama ben yukarıdaki makalede anlatılanı yapmadım, JSON'dan çekmek istediğim veri belli olduğu için ve twitter'in JSON formatı basit olduğu için doğrudan aşağıdaki kod ile gelen her JavaScript nesnesine ScriptObject muamelesi yaptım.

```
For x As Integer = 0 To CInt(InitParams("count")) - 1
    AllData.Add(New TwitPost(JSON.GetProperty(x).GetProperty("text").ToString,
    JSON.GetProperty(x).GetProperty("created_at").ToString,
    JSON.GetProperty(x).GetProperty("id").ToString))
Next
```

Yukarıda JSON değişkeninin içerisinde doğrudan JavaScript tarafından gelen JSON objesi bulunuyor. **GetProperty** metodu ile herhangi bir diziden istediğimiz öğeyi ve özelliklerini tek tek alabiliyorum.

Bir sonraki sorunumuz hangi Twitter hesabından veri çekeceğimiz Widget'i kullananların nasıl karar verebileceği. Bunun için Silverlight 2.0'ın sayfaya yerleştirildiği **Object** taglarının parametrelerini kullanacağız.

Parametreli Silverlight 2.0 dosyaları kullanımı nasıl olur?

Buyurun size bir makale : <http://daron.yondem.com/tr/PermaLink.aspx?guid=4834596e-b5ec-450f-8e3c-cfba929d958e>

Twitter'dan data alma işleminin Widget'in bulunduğu sayfa tamamen yüklenikten sonra başlasın istiyoruz. Bunun durum data yüklenikten sonra DOM'a ulaşıyor olmamızdan kaynaklıyor, eğer sayfa tam yüklenmemişse daha DOM listeleri sabitlenmemiş olabilir.

HTML / DOM eventlarını Silverlight 2.0 ile nasıl yakalarım?

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim MyDoc As HtmlDocument = HtmlPage.Document
    HtmlPage.Window.AttachEvent("onload", AddressOf DocLoaded)
End Sub
```

```
Private Sub DocLoaded(ByVal sender As Object, ByVal e As EventArgs)
```

```
End Sub
```

Yukarıdaki kod ile yaratmış olduğunuz bir event-handleri nasıl sayfanın eventlarının veya bulduğunuz herhangi bir HTML nesnesinin eventlarına ekleyebileceğinizi görebilirsiniz. Kullanım şekli epey basit.

Bir sonraki adımda bir TextBlock içerisinde farklı formatlarda yazı eklemek istiyoruz. Bunda zor ne var? diyebilirsiniz :) Ama bir TextBlock'un Text özelliğine baktığınızda String tipinde olduğunu görürorsunuz. Peki buna nasıl bir formatlama bilgisi aktarabilirim ki? Aktaramayız :) Aslında TextBlock'un bir de **Inlines** diye bir dizisi var. Bunun içerisinde satır içi Item'lar saklanıyor. Eğer **Inlines** ile ilgili bir ayarlama yapılmamışsa ve doğrudan **Text** özelliği set edilmiş ise arkaplanda otomatik olarak bir **Inline** Item yaratılarak bu diziye ekleniyor. Tüm bunlar programatik olarak da yapılabilir.

Bir TextBlock'un içindeki belirli bir metnin rengi programatik olarak nasıl değiştirilir?

```
Dim Span As New Documents.Run
Span.Text = AllText.Substring(0, AllText.IndexOf(FoundURLs.Item(0).Value))
Span.Foreground = CType(App.Current.Resources("PostTextForeGround"),
SolidColorBrush)
Current.Inlines.Item(0) = Span
Span = New Documents.Run
Span.Text = FoundURLs.Item(0).Value
Span.Foreground = CType(App.Current.Resources("PostTextLinkForeGround"),
SolidColorBrush)

Span.TextDecorations = TextDecorations.Underline
Current.Inlines.Add(Span)
Span = New Documents.Run
Span.Text = AllText.Substring(AllText.IndexOf(FoundURLs.Item(0).Value) +
FoundURLs.Item(0).Value.Length, AllText.Length -
(AllText.IndexOf(FoundURLs.Item(0).Value) + FoundURLs.Item(0).Value.Length))
Span.Foreground = CType(App.Current.Resources("PostTextForeGround"),
SolidColorBrush)
Current.Inlines.Add(Span)
```

Yukarıdaki kod içerisinde **Current** değişkeni aslında bir TextBlock. Bu **TextBlock** içerisinde URL'i alıp URL'den öncesini ayrı bir **Run** olarak, URL'i ayrı bir **Run** olarak, kalanı da ayrı bir Run olarak TextBlock'un **Inlines** dizisine ekliyoruz. Böylece her bir **Run** için görsel olarak farklı ayarlar yapabiliyoruz.

Yazımın en başında TwitXR desteği derken ne demek istediğimi biraz anlatıyorum. Twitter ile TwitXR beraber çalışan sitelerdir aslında. TwitXR üzerine yolladığınız bir resim ve yazı otomatik olarak Twitter'a da aktarılır. Bizim Widget Twitter'dan Post'lari alırken kontrol edecek eğer o Post TwitXR'dan gelmişse uygun fotoğrafı da bularak gösterecek. Tüm bu hikayedede benim istediğim nokta TwitXR'dan alınan resim istemciye yüklenirken yüklemenin

durumundan haberdar olmaktı. Yani resmi istedim ama tam olarak yüklendi mi yoksa indiriliyor mu?

Remote Image yüklerken Progress göstermek!

Başından beridir Remote-Request'e izin verilmiyor ve Policy dosyaları yok diyoruz! Bu durumda benim normal bir WebClient sınıfı ile resmi indirmem ve sonra gelen Stream'i Image nesnesine bağlayıp sahneye koyma şansım yok. Mecburen Image'ı doğrudan Image nesnesine bağlamak zorundayım, ancak bu şekilde remote resim alabiliyorum. Ama bunu yaparken de zaten **Imaging.BitmapImage** sınıfını kullanmak zorundayım ve bu sınıfın kendine özel bir DownloadProgress'i var :)

WithEvents PhotoDownload As New Imaging.BitmapImage

'The Photo for the PhotoFrame has been downloaded.

```
Private Sub PhotoDownload_DownloadProgress(ByVal sender As Object, ByVal e As System.Windows.Media.Imaging.DownloadProgressEventArgs) Handles PhotoDownload.DownloadProgress
    If e.Progress = 100 Then
        GetBig(MousePos.Y)
    End If
End Sub
```

'Play the anim when the mouse is on

```
Private Sub Clickable_MouseEnter(ByVal sender As System.Object, ByVal e As System.Windows.Input.MouseEventArgs)
    PhotoDownload.UriSource = New Uri(MyPhoto, UriKind.Absolute)
    Photo.Source = PhotoDownload
End Sub
```

Yukarıdaki kod içerisinde **PhotoDownload** adındaki **BitmapImage**'e fotoğrafın URL'ini veriyorum ve sonra da Photo adında **Image** UIElement nesneme **Source** olarak veriyorum. Böylece **Silverlight** resmi **BitmapImage** ile indirerek Image nesnesine bağlayıp göstermeye çalışıyor. Bu esnada **BitmapImage'in DownloadProgress'i** çalışıyor. İlginç bir şekilde bu Progress event'i içerisindeki Progress özelliği 1 ile 0 arasında olması gerekipen ya 0 ya da 100 döndürüyor. Normalde 0 = indirilemedi ve 1 = indirildi anlamına gelmeliydi. Sanırım bu bir BUG :) Her neyse bir işimizi şimdilik halletti. Unutmayın buradaki Progress WebClient'taki biri Double değil, Integer. Yani aslında bir **Progress** değil de **Status** değeri veriyor.

Şeffaf uygulama ve Overlay Sorunu

Aslında uygulama fonunu şeffaf yapmak çok kolay. Buyurun makalesi :

<http://daron.yondem.com/tr/PermaLink.aspx?guid=b334e195-feb7-4411-a77d-b6f07d482068>

Esas sorun ben şeffaf olan yerlerin sadece şeffaf gözükmemesini değil aynı anda o şeffaflığın arkasındaki HTML'in de kullanılabilir olmasını istiyorum. İşte bu olmuyor! Hedefim Widget içerisinde herhangi bir Post'un fare ile üzerine gelince yana doğru sayfanın üzerine taşıacak şekilde uygulamanın genişlemesi ve orada da postun fotosunun gözükmesi. Tabi tüm uygulama gelişmeyecek sadece fotoğrafın gözükeceği bir kısım açılacak. Şeffaf fon

kullanıldığında görsel olarak bir sorun yok gibi gözüksede bir bakıyorsunuz ki şeffaf olmasına rağmen uygulama alanınız fotoğrafın gösterileceği yerleri de kapsadığı yani sayfaya taşıdığı için o kısımlardaki HTML kontrolleri çalışmıyor. Bu konu maalesef Flash'ta daha iyi çözülmüş durumda, Flash'ta şeffaf olan yerler gerçekten şeffaf :(

Peki nasıl çözeriz, dinamik olarak Silverlight uygulamasının sayfadaki kapladığı alanı yine kodlarımız ile düzenlememiz gerek. Silverlight'in OBJECT taglarına bir ID değeri vererek bunun üzerinden OBJECT'in genişliğini değiştirebiliriz.

```
HtmlPage.Document.GetElementById("TwitterSLWidget").SetAttribute("width",
"360px")
```

Bu kod ile kullandığımız OBJECT tagları aşağıdaki gibi.

```
<object style="position:absolute; z-index: 2;" id="TwitterSLWidget"
data="data:application/x-silverlight-2,"
type="application/x-silverlight-2">
<param name="source" value="ClientBin/TwitterWidget.xaml" />
<param name="initParams"
value="twitterid=daronyondem,count=10,twitxrid=daronyondem" />
<param name="onerror" value="onSilverlightError" />
<param name="background" value="Transparent" />
<param name="pluginbackground" value="Transparent" />
<param name="windowless" value="true" />
<param name="background" value="white" />
<param name="minRuntimeVersion" value="2.0.31005.0" />
<param name="autoUpgrade" value="true" />
<a href="http://go.microsoft.com/fwlink/?LinkID=124807" style="text-decoration: none;">
    
</a>
</object>
```

Widget'i nasıl kullanırız?

Birazdan kaynak kodları ile beraber Widget'i sizinle paylaşacağım. Ama onun öncesinde hemen Widget'i nasıl kullanırız ona bakalım.

```
<div id="TwitterSLWidgetHost">
    <object style="position:absolute; z-index: 2;" id="TwitterSLWidget"
data="data:application/x-silverlight-2,"
type="application/x-silverlight-2">
    <param name="source" value="ClientBin/TwitterWidget.xaml" />
    <param name="initParams"
value="twitterid=daronyondem,count=10,twitxrid=daronyondem" />
    <param name="onerror" value="onSilverlightError" />
    <param name="background" value="Transparent" />
    <param name="pluginbackground" value="Transparent" />
```

```

<param name="windowless" value="true" />
<param name="background" value="white" />
<param name="minRuntimeVersion" value="2.0.31005.0" />
<param name="autoUpgrade" value="true" />
<a href="http://go.microsoft.com/fwlink/?LinkId=124807" style="text-decoration: none;">
    
</a>
</object></div>

```

Yukarıdaki HTML kodunu sayfanıza yerleştirmeniz gerekiyor. Özellikle koyu olan yerbere dikkat. **daronyondem** yerine kendi **Twitter** ve **TwitXT** hesaplarınızın adlarını yazmanız gerek. **count** kısmına da koç Post gözüksün istiyorsanız onu yazabilirsınız. OBJECT ve DIV taglarının ID'leri çok önemli. Bu ID'ler kodlarda kullanılıyor, eğer değiştirirseniz kodları da tekrar düzenleyip XAP dosyasını **Compile** etmeniz gerekecektir.

```

function TwitterIncData(object)
{
    document.getElementById("TwitterSLWidget").Content.Page.IncData(object);
}

```

Son olarak yukarıdaki kodu da sayfanızda uygun bir JavaScript dosyasına veya tagları arasında koymanız ve XAP dosyasını sunucuya kopyalamanız yeterli olacaktır.

Kaynak Kodlar

Tüm projenin kaynak kodlarını aşağıdaki adresten indirebilirsınız. Proje epey karışık oldu, özellikle Remote Script Injection kullandığımız için Cross-Browser uyumluluğu konusunda sıkıntılar var. Diğer yandan Overlay konusunda da farklı tarayıcılarda sorunlar var. O nedenle şimdilik ben ancak IE 7 desteği sunabiliyorum, uğraşacak pek zaman olmadı. Eğer siz uygulamayı diğer tarayıcıları da destekleyecek şekilde modifiye ederseniz beni haberdar etmeniz yeterli. Böylece mini bir açık kaynak projesi olmuş olur.

Makaledeki tüm kodlar VB :) çünkü uygulamadan kesip yapıştırdım. Uygulamayı da malum VB ile yazdım :) C#'cılara selamlar :)

Daron YÖNDEM

Silverlight 2.0 ItemsControl ve ObservableCollection ile DataBinding

ASP.NET kullanırken en sevdiğim kontrol Repeater kontrolüdür. Bana herseyi istediğim gibi esnek bir şekilde düzenleme şansı tanır. Aynı mantıkla **Silverlight 2.0 Beta 1** tarafına geçtiğimizde karşımızda **ItemsControl** çıkıyor. ItemsControl'e bağladığınız herhangi bir veri içerisindeki her Item'in nasıl gözükeceği aynı Repeater içerisinde olduğu gibi bir **ItemTemplate** aracılığı ile karar verebiliyorsunuz. Ayrıca tüm bu Item'ların nasıl bir kontrol içerisinde ekrana yerleştirileceğini de belirleme şansınız var. Özünde **ItemsControl** tek başına herhangi bir görsellik barındırmıyor, herseyi sizin tek tek ayarlamış olmanız şart.

Örneğimizde elimizde bulunan bir ürün listesini ItemsControl'e bağlayarak ürünlerin isimleri ile satış grafiklerini göstereceğiz. Bunu yaparken de Silverlight 2.0 ile beraber gelen DataBinding sistemini kullanacağız. İlk olarak gelin code-behind tarafından başlayalım ve ürün listemizi yaratacağımız ürün tipini tanımlayalım.

Public Class Urun

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

```
Private PSatis As Integer
Public Property Satis() As Integer
    Get
        Return PSatis
    End Get
    Set(ByVal value As Integer)
        PSatis = value
    End Set
End Property
```

```
Sub New()
```

```
End Sub
```

```
Sub New(ByVal adi As String, ByVal satis As Integer)
    Me.Adi = adi
    Me.Satis = satis
End Sub
```

```
End Class
```

Gördüğünüz gibi aslında klasik bir .NET sınıfından farklı değil. Her zamanki gibi **Urun** sınıfımızı / tipini yarattık ve bu sınıf üzerinde bir liste üreterek ItemsControl'e bağlayacağız.

Normal şartlarda olsa belki bir **Generic.List** kullanırdık oysa Silverlight 2.0 ile beraber bir **ObservableCollection** kullanacağız. Bunun tabii ki mantıklı bir nedeni var.

ObservableCollection yapıları Silverlight 2.0 tarafında DataBind işlemleri için kullanıldıklarında Public bir **ObservableCollection** listesi bir defa herhangi bir kontrole bağlandıktan sonra sürekli organik bir bağ içerisinde kalıyor. Böylece eldeki liste üzerinde herhangi bir değişiklik kod tarafında yapıldığında otomatik olarak sonuç görsel öğelere de yansıyor. Biz de bu nedenle örneğimizde **ObservableCollection** listesi kullanacağız, böylece kod tarafında listede bir değişiklik yaptığımızda sonuç doğrudan görsel olarak ItemsControl'e yansıyacak.

```
Public Liste As New System.Collections.ObjectModel.ObservableCollection(Of Urun)
```

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    For x As Integer = 0 To 9
        Liste.Add(New Urun("Urun Adı " & x, Rnd() * 150))
    Next
    UrunlerControl.ItemsSource = Liste
End Sub
```

Gördüğünüz gibi aslında **ObservableCollection'ların** normal **Generic.List**'lerden kullanım açısından pek bir farkı yok. Biz örneğimizde rastgele ürünler ve satış rakamları yaratarak ürettiğimiz listeyi, adını **UrunlerControl** olarak koyduğumuz **ItemsControl** nesnesine **ItemsSource** özelliği üzerinden bağlıyoruz.

Gelelim XAML kodumuza...

```
<ItemsControl Margin="27,8,44,8" x:Name="UrunlerControl">
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            Her bir yaratılan ögenin görsel tanımı buraya gelir.
        </DataTemplate>
    </ItemsControl.ItemTemplate>
    <ItemsControl.ItemsPanel>
        <ItemsPanelTemplate>
            Tüm yaratılan öğelerin içerisinde yerleştirileceği ortamı tanımlar.
        </ItemsPanelTemplate>
    </ItemsControl.ItemsPanel>
</ItemsControl>
```

Yukarıdaki ItemsControl içerisinde toplamda iki farklı **Template** bulunuyor. Bunlardan ilki **ItemTemplate** içerisinde **DataTemplate**. ItemsControl'e bağladığımız verideki her biri **Urun** için bir adet **DataTemplate** oluşturulacaktır. Bu sistemi ASP.NET'teki Repeater içerisinde ItemTemplate yapısına benzetebiliriz. Tüm bu yaratılan DataTemplate'lar ise **ItemsPanel** içerisinde **ItemsPanelTemplate'ta** tanımlanmış görsel yapının içerisinde oturtulacaktır. ItemsControl'un kendine ait bir görsel yapısı olmadığı için aslında ItemsPanelTemplate bu yapıyı oluşturuyor olacak.

Biz örneğimizde ana yapı olarak bir **StackPanel** kullanacağız ve **ItemsPanelTemplate** içerisinde içindeki nesneleri dikey hizalamaya ayarlanmış bir StackPanel bulunacak.

```
<ItemsControl.ItemsPanel>
  <ItemsPanelTemplate>
    <StackPanel Orientation="Vertical"></StackPanel>
  </ItemsPanelTemplate>
</ItemsControl.ItemsPanel>
```

Diğer yandan üretilen her biri **Urun'un** görselliği için de **DataTemplate** içerisinde bir **TextBlock** bir de **Rectangle** kullanacağız. **DataTemplate** içerisinde sadece bir element yerleştirebildiğimiz için bir ContainerElement olarak yine **StackPanel** kullanacağız. Bu sefer her bir **Urun** için oluşturulacak olan StackPanel'ler kendi içindeki nesneleri yatay olarak hizalayacaklar. Böylece StackPanel'ler içerisindeki TextBlock ve Rectangle'lar yan yana duracaklar.

```
<ItemsControl.ItemTemplate>
  <DataTemplate>
    <StackPanel Orientation="Horizontal">
      <TextBlock/>
      <Rectangle Height="30" Fill="#FFFF0000"/>
    </StackPanel>
  </DataTemplate>
</ItemsControl.ItemTemplate>
```

Amacımız TextBlock içerisinde ürünün adını gösterirken ürün satış rakamına göre Rectangle'in da genişliğini ayarlayarak genişliğine göre ürünlerin yanında bir satış grafiği göstermek. Bu durumda hemen Silverlight 2.0'a özel DataBind sistemini kullanarak TextBlock'un **Text** özelliğini **ItemsControl'a** aktarılan verilerin **Adı** özelliğine, Rectangle'in **Width** özelliğini de **Satis** değerine bağlamamız gerekiyor.

```
<StackPanel Orientation="Horizontal">
  <TextBlock Text="{Binding Adı}"/>
  <Rectangle Height="30" Width="{Binding Satis}" Fill="#FFFF0000"/>
</StackPanel>
```

İşimiz tamamlandı. Artık uygulamamızı çalıştıracak sonucu görebiliriz. Fakat onun öncesinde gelin sürekli ürünlerin listesini değiştiren bir de Düğme ekleyelim uygulamamıza ve elimizdeki **ObservableCollection** yapısındaki ürün listesini sürekli değiştirsin. Düğmemizin arkaplandaki koduna aşağıdaki satırları yazmamız yeterli olacaktır.

```
Private Sub Dugme_Click(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Dugme.Click
  Liste.Clear()
  For x As Integer = 0 To 9
    Liste.Add(New Urun("Urun Adı " & x, Rnd() * 150))
  Next
End Sub
```

Yukarıdaki kod içerisinde **ObservableCollection** listemizi temizliyor ve tekrar ürünler ekliyoruz. Herhangi bir şekilde tekrar DataBind işlemi yapmasak da söz konusu işlemlerin sonucu ItemsControl'un görsel arayüzüne yansıyor.

Uygulamamızın tam XAML kodunu aşağıda inceleyebilirsiniz.

```
<UserControl x:Class="SilverlightApplication8.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
<Grid x:Name="LayoutRoot" Background="White">
    <ItemsControl Margin="27,8,44,8" x:Name="UrunlerControl">
        <ItemsControl.ItemTemplate>
            <DataTemplate>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="{Binding Adi}" />
                    <Rectangle Height="30" Width="{Binding Satis}" Fill="#FFFF0000"/>
                </StackPanel>
            </DataTemplate>
        </ItemsControl.ItemTemplate>
        <ItemsControl.ItemsPanel>
            <ItemsPanelTemplate>
                <StackPanel Orientation="Vertical"></StackPanel>
            </ItemsPanelTemplate>
        </ItemsControl.ItemsPanel>
    </ItemsControl>
    <Button Height="30" HorizontalAlignment="Right" Margin="0,0,17,8"
        VerticalAlignment="Bottom" Width="70" Content="Yenile" x:Name="Dugme"/>
</Grid>
</UserControl>
```

Code-Behind kısmındaki VB kodumuz da bu şekilde;

```
Partial Public Class Page
    Inherits UserControl
```

```
    Public Sub New()
        InitializeComponent()
    End Sub
```

```
    Public Class Urun
```

```
        Private PAdi As String
        Public Property Adi() As String
            Get
                Return PAdi
            End Get
            Set(ByVal value As String)
                PAdi = value
            End Set
        End Property
```

```
        Private PSatis As Integer
```

```

Public Property Satis() As Integer
    Get
        Return PSatis
    End Get
    Set(ByVal value As Integer)
        PSatis = value
    End Set
End Property

Sub New()
End Sub

Sub New(ByVal adi As String, ByVal satis As Integer)
    Me.Adi = adi
    Me.Satis = satis
End Sub

End Class

Public Liste As New System.Collections.ObjectModel.ObservableCollection(Of Urun)

Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    For x As Integer = 0 To 9
        Liste.Add(New Urun("Urun Adı " & x, Rnd() * 150))
    Next
    UrunlerControl.ItemsSource = Liste
End Sub

Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme.Click
    Liste.Clear()
    For x As Integer = 0 To 9
        Liste.Add(New Urun("Urun Adı " & x, Rnd() * 150))
    Next
End Sub
End Class

```

Silverlight 2.0 Beta 1 ile beraber gelen bu DataBind özellikleri aslında WPF tarafından tanıdık olsa da Cross-Platform olarak istemci tarafında bu işlemleri yapabiliyor olmak gerçekten heyecan verici.

Hepinize kolay gelsin.

Daron YÖNDEM

[Silverlight 2.0 kontrolleri yaratmanın yolu.](#)

Silverlight 2.0 içerisinde kullandığımız standart kontrollerin yanı sıra istersek kendi kontrollerimizi de hazırlama şansımız var. Daha önceleri ister ASP.NET ister Windows uygulamalarından alışık olduğumuz bu uygulama ile hazırladığınız size özel kontrolleri farklı projelerde rahatlıkla kullanma şansına sahip olabilirsiniz. Bu yazımızda deneme amaçlı olarak Silverlight 2.0 ile beraber gelen Button kontrolünü baz alıp geliştirmek yeni bir kontrol oluşturacağız. Yaratacağımız yeni **TimeOutButton** kontrolü kullanıcından bir **TimeOut** değeri alacak. Milisaniye cinsinden verilen bu süre sonunda Button kullanılamaz hale gelecek. Bu esnada Button'un üzerinde sürekli kaç saniye kaldıgı da yazılacak. Kullanıcıların belirli bir sürede bir işlemi tamamlamasını istiyorsanız bu gibi bir Button işinizi görebilir. Gelin hemen işe koyulalım.

Çalışma ortamımızı hazırlayalım.

Kontrolümüzü geliştirirken sürekli test etmek isteyeceğiz. O nedenle ilk olarak sahaklı bir çalışma ortamı hazırlamamız lazım. Yeni bir Silverlight projesi yarattığınızda Visual Studio içerisinde Solution içinde bir ASP.NET ve bir de Silverlight projeniz bulunur. Solution dosyasına bir proje daha ekleyeceğiz. Solution Explorer içerisinde solution dosyasına sağ tuş ile tıkladıktan sonra "Add / New Project" diyerek "**Silverlight Class Library**" seçeneğini işaretlemeniz yeterli olacaktır. Böylece yeni kontrolümüzü oluşturmak için kodlarımızı yazacağımız ortamı geliştirdik.

Solution'ı Compile ettiğimizde Silverlight projemize otomatik olarak bu kontrol projemizin de eklenmesi gerekiyor. O nedenle hemen Silverlight projesine sağ tuş ile tıklayarak gelen menüden "Add Reference" komutunu verin ve karşınıza gelen pencereden de "Projects" tabına geçerek Class Library projenizi seçin. Böylece artık F5'e bastığınızda her şey otomatik olarak yapılacaktır ve kontrolünüzü kullanılır şekilde inceleyebileceksiniz. Tabi testlere geçmeden önce biraz kod yazmamız gerekiyor ;)

Kontrolümüzü hazırlayalım.

Class Library içerisinde kod dosyamızda düğmemizin adını taşıyan bir Class bulunuyor. Bu Class'ı biz örneğimizde standart Silverlight Button kontrolünden inherit edeceğiz. Böylece yeni ve gelişmiş bir Button yaratırken her şeye sıfırdan başlamayacağız, Silverlight içerisinde hazır Button kontrolünü alarak üzerine yeni işlevsellikler ekleyeceğiz.

[VB]

```
Public Class TimeOutButton
    Inherits Button
```

```
End Class
```

[C#]

```
namespace TimeOutBtn
{
    public class TimeOutButton : Button
    {
    }
```

}

Bir sonraki adımda hemen kontrolümüze yeni bir Property tanımlayalım. Property'mız sayesinde kullanıcıların düğmenin ne kadar süre aktif kalacağını tanımlayabilecekler.

[VB]

```
Private PTimeOut As Integer
Public Property TimeOut() As Integer
    Get
        Return PTimeOut
    End Get
    Set(ByVal value As Integer)
        PTimeOut = value
    End Set
End Property
```

[C#]

```
public int TimeOut { get; set; }
```

Düğmemiz ilk olarak sayfada gözüktüğünde hemen işlemelere başlamamız gerekiyor. İlk olarak düğmenin için sayaç bilgisi yazacak yeni bir kontrol eklememiz şart. Düğmenin **Content** özelliği duruma göre doğrudan bir String olabilir veya Content içerisinde farklı Silverlight nesneleri bulunabilir. Tüm bunları göz önünde bulundurmak zorundayız.

[VB]

```
Dim Container As New StackPanel
Container.Orientation = Orientation.Vertical
Container.HorizontalAlignment = Windows.HorizontalAlignment.Center
Container.VerticalAlignment = Windows.VerticalAlignment.Center

If Me.Content.GetType() Is System.Type.GetType("System.String") Then
    Dim TextBl As New TextBlock
    TextBl.Text = Me.Content
    Container.Children.Add(TextBl)
Else
    Container.Children.Add(Me.Content)
End If
Container.Children.Add(MyBox)
Me.Content = Container
```

[C#]

```
StackPanel Container = new StackPanel();
Container.Orientation = Orientation.Vertical;
Container.HorizontalAlignment = System.Windows.HorizontalAlignment.Center;
Container.VerticalAlignment = System.Windows.VerticalAlignment.Center;
```

```

if (object.ReferenceEquals(this.Content.GetType(),
System.Type.GetType("System.String")))
{
    TextBlock TextBl = new TextBlock();
    TextBl.Text = this.Content.ToString();
    Container.Children.Add(TextBl);
}
else {
    Container.Children.Add(this.Content as UIElement);
}
Container.Children.Add(MyBox);
this.Content = Container;

```

Kodumuz içerisinde ilk olarak bir StackPanel yaratıyoruz. StackPanel'in özelliklerini ayarladıkten sonra hemen düğmemize atanmış Content'i kontrol ediyoruz. Amacımız Content içerisinde her ne varsa hepsini StackPanel içine eklemek. Sonrasında bizim kalan saniye miktarını göstereceğimiz TextBlock'u da yine StackPanel içerisinde ekleyerek StackPanel'i de düğmenin Content'i yapacağız. Böylece bu TimeOutButton kontrolünü kullananlar düğmenin Content'ine ne koyarlarsa kalan saniye miktarı sürekli bu Content'in içerisinde gösterilecek.

Eğer düğmeye atanmış Content String tipindeyse bu String'i alıp yeni bir TextBlock yaratarak içerisinde yerleştiriyoruz. TextBlock'umuzu da StackPanel içine koyuyoruz. Eğer Content String değilse demek ki kullanıcı düğmenin Content'ine bir Silverlight nesnesi koymuş. Bu durumda söz konusu nesneyi alıp doğrudan StackPanel içine koyabiliriz. Son olarak kodumuzda gözükmeyen fakat uygulamamızda global bir değişken olarak yarattığımız MyBox adındaki TextBlock'u da StackPanel'e ekliyoruz ve StackPanel'i de düğmenin Content'i yapıyoruz. MyBox kontrolünü global yaratmamızın nedeni ileriki adımlarda MyBox'in içeriğine sürekli kalan saniye miktarını yazdıracak olmamız.

[VB]

```

Dim MyBox As New TextBlock
WithEvents Timer As New System.Windows.Threading.DispatcherTimer

```

[C#]

```

TextBlock MyBox = new TextBlock();
System.Windows.Threading.DispatcherTimer Timer = new
System.Windows.Threading.DispatcherTimer();

```

Global değişkenimiz arasında bir de **DispatcherTimer** bulunuyor. Bu Timer ile kalan saniye miktarını sürekli hesaplayıp kontrol ederek düğmenin içine yazdıracağız.

[VB]

```

Private Sub Timer_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Timer.Tick
If PTimeOut <= 0 Then
    Timer.Stop()
    MyBox.Visibility = Windows.Visibility.Collapsed

```

```

Me.IsEnabled = False
Else
    PTimeOut -= 1000
    MyBox.Text = (PTimeOut / 1000).ToString() + " saniye kaldı."
End If
End Sub

```

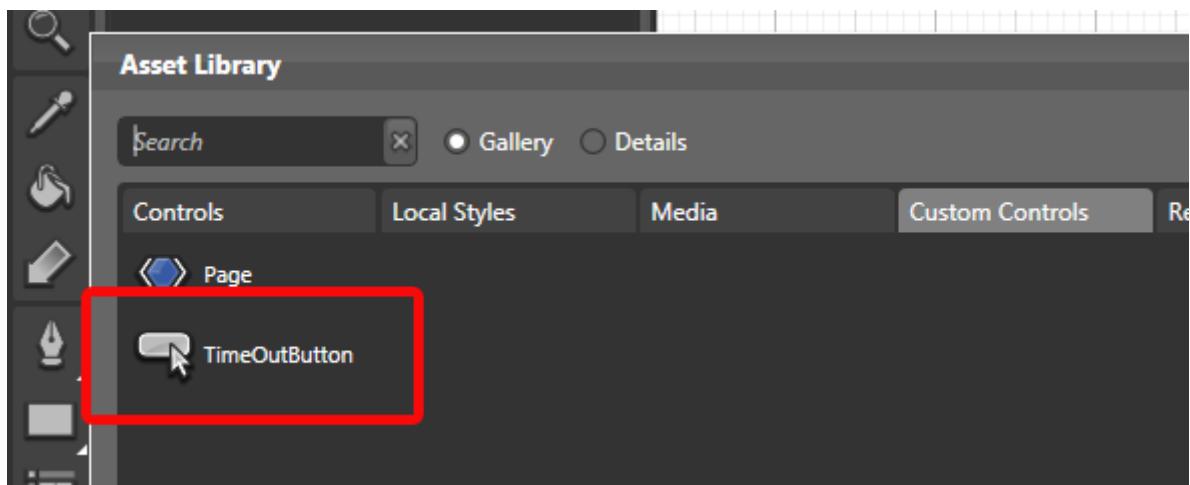
[C#]

```

private void Timer_Tick(object sender, System.EventArgs e)
{
    if (TimeOut <= 0)
    {
        Timer.Stop();
        MyBox.Visibility = System.Windows.Visibility.Collapsed;
        this.IsEnabled = false;
    }
    else {
        TimeOut -= 1000;
        MyBox.Text = (TimeOut / 1000).ToString() + " saniye kaldı.";
    }
}

```

Kodumuzda kalan süreyi **PTimeOut** adındaki private değişkenimizde saklayacağız. Eğer kalan süre sıfırın altında ise hemen Timer'ımızı durduruyor ve kalan süreyi gösteren TextBlock'umuz olan MyBox'i sahneden kaldırıp düğmeyi de pasif hale getiriyoruz. Kalan süre sıfırdan büyük olduğu sürece **MyBox** TextBlock içerisinde gerekli uyarıyı yazdırıp kalan süreyi Timer'in Intervalı kadar azaltıyoruz.



Expression Blend içerisinde hazırladığımız yeni kontrol

Artık kontrolümüz hazır. Projemizi Blend ile açtığımızda Asset Library içerisinde Custom Controls tabında kontrolümüzü görebiliyoruz. Sahneye bir TimeOutButton ekleyip TimeOut özelliğini de ayarladıkten sonra kontrolümüzü kullanabiliriz.

```

<UserControl x:Class="SilverlightApplication47.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="400" Height="300" xmlns:TimeOutBtn="clr-
namespace:TimeOutBtn;assembly=TimeOutBtn">
<Grid x:Name="LayoutRoot" Background="White">
<TimeOutBtn:TimeOutButton TimeOut="5000" Height="72"
HorizontalAlignment="Left" Margin="56,48,0,0" VerticalAlignment="Top" Width="128"
Content="TimeOutButton"/>
</Grid>
</UserControl>

```

Hepinize kolay gelsin. Uygulamanın tam kodunu aşağıda inceleyebilirsiniz.

[VB]

```

Public Class TimeOutButton
Inherits Button

Private PTimeOut As Integer
Public Property TimeOut() As Integer
Get
    Return PTimeOut
End Get
Set(ByVal value As Integer)
    PTimeOut = value
End Set
End Property

Dim MyBox As New TextBlock
WithEvents Timer As New System.Windows.Threading.DispatcherTimer

Private Sub TimeOutButton_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
Dim Container As New StackPanel
Container.Orientation = Orientation.Vertical
Container.HorizontalAlignment = Windows.HorizontalAlignment.Center
Container.VerticalAlignment = Windows.VerticalAlignment.Center

If Me.Content.GetType() Is System.Type.GetType("System.String") Then
    Dim TextBl As New TextBlock
    TextBl.Text = Me.Content
    Container.Children.Add(TextBl)
Else
    Container.Children.Add(Me.Content)
End If
Container.Children.Add(MyBox)
Me.Content = Container
Timer.Interval = New TimeSpan(0, 0, 1)
Timer.Start()
End Sub

```

```

Private Sub Timer_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Timer.Tick
    If PTimeOut <= 0 Then
        Timer.Stop()
        MyBox.Visibility = Windows.Visibility.Collapsed
        Me.IsEnabled = False
    Else
        PTimeOut -= 1000
        MyBox.Text = (PTimeOut / 1000).ToString + " saniye kaldi."
    End If
End Sub
End Class

```

[C#]

```

using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace TimeOutBtn
{
    public class TimeOutButton : Button
    {
        public int TimeOut { get; set; }

        TextBlock MyBox = new TextBlock();
        System.Windows.Threading.DispatcherTimer Timer = new
        System.Windows.Threading.DispatcherTimer();

        public TimeOutButton()
        {
            this.Loaded += new RoutedEventHandler(TimeOutButton_Loaded);
            Timer.Tick += new EventHandler(Timer_Tick);
        }

        private void TimeOutButton_Loaded(object sender, System.Windows.RoutedEventArgs
e)
        {
            StackPanel Container = new StackPanel();
            Container.Orientation = Orientation.Vertical;
            Container.HorizontalAlignment = System.Windows.HorizontalAlignment.Center;
            Container.VerticalAlignment = System.Windows.VerticalAlignment.Center;
        }
    }
}

```

```
if (object.ReferenceEquals(this.Content.GetType(),  
System.Type.GetType("System.String"))) {  
    TextBlock TextBl = new TextBlock();  
    TextBl.Text = this.Content.ToString();  
    Container.Children.Add(TextBl);  
}  
else {  
    Container.Children.Add(this.Content as UIElement);  
}  
Container.Children.Add(MyBox);  
this.Content = Container;  
Timer.Interval = new TimeSpan(0, 0, 1);  
Timer.Start();  
}  
private void Timer_Tick(object sender, System.EventArgs e)  
{  
    if (TimeOut <= 0)  
    {  
        Timer.Stop();  
        MyBox.Visibility = System.Windows.Visibility.Collapsed;  
        this.IsEnabled = false;  
    }  
    else {  
        TimeOut -= 1000;  
        MyBox.Text = (TimeOut / 1000).ToString() + " saniye kaldi.";  
    }  
}  
}
```

Daron YÖNDEM

Silverlight 2.0 Plug-In Algılama ve OBJECT Tagı

Silverlight 2.0 Beta 2 ile beraber 1.0 sürümüne göre uygulamaların dağıtım sistemi değişti ve karşımıza **XAP** dosyaları çıktı. Bu dosyaları standart HTML veya herhangi bir sunucu taraflı programlama altyapısında kullanmanın yolu da aslında eskiden Flash tarafından alışık olduğumuz Object taglarına dönüştü. Bu durumun tabi ki güzel yanları var, Silverlight 1.0'da olduğu gibi harici JavaScript dosyalarına (Silverlight.js) ve DIV elementlerine vs ihtiyacımız olmuyor. Şimdi gelin beraber Silverlight 2.0 uygulamalarını sayfalarımıza yerleştirirken kullanacağımız **OBJECT** tagları ile gelen yenilikleri inceleyelim.

Yeni bir Silverlight 2.0 projesi

Visual Studio 2008 içerisinde yeni bir Silverlight 2.0 projesi yarattığımızda bizim için test amaçlı olarak örnek bir de HTML dosyası hazırlanıyor. Bu dosya içerisinde bir Silverlight uygulamasının gösterimi için konulmuş kodları yavaş inceleyelim.

```
<div id="silverlightControlHost">
    <object data="data:application/x-silverlight," type="application/x-silverlight-2-b2"
width="100%" height="100%">
        <param name="source" value="ClientBin/SilverlightApplication56.xap"/>
        <param name="onerror" value="onSilverlightError" />
        <param name="background" value="white" />

        <a href="http://go.microsoft.com/fwlink/?LinkID=115261" style="text-decoration: none;">
            
        </a>
    </object>
    <iframe style='visibility:hidden;height:0;width:0;border:0px'></iframe>
</div>
```

En dışta silverlightControlHost adında bir DIV elementi bulunuyor. Aslında böyle elementin bulunmasına hiç gerek yok, sadece CSS ile Silverlight uygulamasının boyutu ayarlanabilisin diye yerleştirilmiş bu DIV elementinin başka herhangi bir işlevselligi olmadığı için rahatlıkla koddan kaldırılabilir.

Esas **OBJECT** tagımıza geçtiğimizde karşımıza dört farklı özellik çıkıyor. Bunlardan **Data** özelliği **OBJECT** taglarının farklı tarayıcılarda farklı işlevselliklerle karşılaşmasından dolayı yerleştirilmiş. Data özelliği olmayan **OBJECT** tagları bazı tarayıcılarda sorun çıkartabiliyor. **TYPE** özelliği tarayıcıya Silverlight Plug-In'in bu **OBJECT** tagından sorumlu olduğu bilgisini verirken **width** ve **height** ise uygulamanın sayfadaki boyutunu belirtiyor.

```
<param name="source" value="ClientBin/SilverlightApplication56.xap"/>
<param name="onerror" value="onSilverlightError" />
<param name="background" value="white" />
```

OBJECT tagları arasında bazı parametreler var. Bu parametreler uygulamanın özelliklerine göre değişecektir, hatta burada tanımlanmamış olan bazı farklı parametreler de kullanmak mümkün. Mevcut parametreler arasında **Source** parametresi istemciye yüklenecek olan **XAP**

dosyasının adresini taşıırken **onerror** parametresi ise hata durumunda istemci tarafından çalıştırılacak olan **JavaScript** metodunun adını saklıyor.

```
<a href="http://go.microsoft.com/fwlink/?LinkId=115261" style="text-decoration:none;">
    
</a>
```

Buradaki kod ise aslında istemcide Silverlight Runtime yüklü olmadığından gösterilecek olan içeriği tanımlıyor. Gördüğünüz gibi basit bir şekilde "Get Microsoft Silverlight" logosu gösterilerek microsoft.com'da bir web sayfasına yönlendirilmiş. Artık SL 1.0'da olduğu gibi Silverlight Plug-In'in hangi sürümünün veya hangi platforma özel yüklemesinin yapılacağına JavaScript ile istemci tarafında karar vermiyoruz. Herhangi bir şekilde yükleme gereğinde doğrudan Microsoft sitesine yönlendirme yaparak tüm işi Microsoft'a bırakıyoruz. Bu biz yazılım geliştiriciler için çok rahat bir sistem.

Eğer isterseniz yukarıdaki HTML kodunu değiştirerek **Silverlight RunTime** yüklü olmadığından ekranda gösterilecek içeriği farklı bir şekilde tanımlayabilirsiniz. Önemli olan tek nokta herhangi bir şekilde bir link ile kullanıcıları doğru adrese yönlendirerek Microsoft sitesinden RunTime paketini indirmelerine olanak tanımak.

```
<div id="silverlightControlHost">
    <object data="data:application/x-silverlight," type="application/x-silverlight-2-b2"
width="100%" height="100%">
        <param name="source" value="ClientBin/SilverlightApplication56.xap"/>
        <param name="onerror" value="onSilverlightError" />
        <param name="background" value="white" />

        <a href="http://go.microsoft.com/fwlink/?LinkId=115261" style="text-decoration:none;">
            
        </a>
    </object>
    <iframe style='visibility:hidden;height:0;width:0;border:0px'></iframe>
</div>
```

Örneğin yukarıdaki kod içerisindeki HTML sayesinde eğer istemci tarafında Silverlight Runtime yüklü değil ise silverlightyok.jpg adında bir resim gösteriliyor ve tıklandığında RunTime yüklenmek üzere Microsoft web sitesine yönlendiriliyor.

```
<iframe style='visibility:hidden;height:0;width:0;border:0px'></iframe>
```

Bu da nesi? Bu tamamen Safari tarafında bir önbellekleme sorununu gidermek için yerleştirilmiş normalde kullanılmayan bir IFRAME. Safari tarayıcısında bir sayfada eğer bir IFRAME varsa sayfa hiçbir şekilde önbelleğe alınmıyor.

Bunların haricinde Visual Studio'da yaratılan sayfa içinde hata yakalama için **onSilverlightError** adında bir JavaScript fonksiyonu ve **errorLocation** adında bir DIV

elementi bulunuyor. Bahsi geçen bu JavaScript fonksiyonu herhangi bir hata durumunda hata mesajını alarak **errorLocation** içerisinde yazdırırmakla sorumlu.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 PopUp Kontrolü

Silverlight 2.0 Beta 1 ile gelen yeni Silverlight kontrollerinden biri olan PopUp kontrolü ile beraber Silverlight uygulamaları içerisinde dahili PopUp sistemleri oluşturabiliyoruz. Özellikle farklı Silverlight kullanıcı kontrollerinin bulunduğu projelerde farklı arayüzler arası geçişleri sağlamak için PopUp kontrolleri büyük kolaylık sağlıyor.

Hemen bir PopUp kontrolü örneği yapmak için yeni bir Silverlight 2.0 Beta 1 projesi yaratarak uygulamamızın ana XAML'ı olan Page.XAML içerisinde bir düğme yerleştiriyoruz. Söz konusu düğmeye basıldığında PopUp kontrolü açılacak.

[Page.xaml]

```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="800" Height="600">
    <Grid x:Name="LayoutRoot" Background="#FFFF0000">
        <Button Height="36" HorizontalAlignment="Left" Margin="26,40,0,0"
VerticalAlignment="Top" Width="65" Content="POPUP&#xd;=&#xa;AÇ"
TextAlignment="Center" x:Name="Dugme"/>
    </Grid>
</UserControl>
```

Uygulamamızın ana sayfasını yukarıdaki şekilde tasarladıkten sonra sıra geldi PopUp kontrolü içerisinde kullanacağımız özel kullanıcı kontrolünü (User Control) hazırlamaya. Söz konusu User Control içerisinde de PopUp'ı kapatacak olan bir düğme ve bir de TextBlock yer alacak.

[UserControl1.xaml]

```
<UserControl
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expsion/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    x:Class="SilverlightApplication1.UserControl1"
    d:DesignWidth="640" d:DesignHeight="480">

    <StackPanel x:Name="LayoutRoot">
        <Canvas Height="323" Width="478" Background="#FF2DFF00">
            <TextBlock TextWrapping="Wrap" FontSize="72" FontWeight="Bold" Text="PopUp
Test" Height="232" Width="478" Canvas.Left="31" Canvas.Top="29"/>
            <Button Height="112" Content="POPUP Kapat" x:Name="Dugme" Width="171"
Canvas.Left="132" Canvas.Top="159"/>
        </Canvas>
    </StackPanel>
</UserControl>
```

Üstteki şekli ile UserControl'ümüzü de tamamladıktan sonra artık XAML dosyalarımızı ve tasarımlarını bitirdiğimize göre Visual Studio ile kod yazma kısmına geçebiliriz. İlk olarak ana XAML dosyamızdaki Dugme'mizin arkasına PopUp kontrolünü yaratarak ekrana getirecek olan kodu yazalım.

```
Dim PopupKontrol As New System.Windows.Controls.Primitives.Popup
PopupKontrol.Child = New UserControl1
PopupKontrol.IsOpen = True
```

Kodumuzun ilk satırında **System.Windows.Controls.Primitives** NameSpace'i altında yer alan PopUp sınıfını kullanarak yeni bir PopUp kontrolü yaratıyoruz. Bir sonraki satırda ise UserControl olarak hazırladığımız ve adı **UserControl1** olan kontrolümüzden bir kopya yaratarak PopUp kontrolünün **Child** özelliğine eşitliyoruz. İşlemleri tamamladıktan sonra artık PopUp kontrolümüzü ekrana getirebileceğimize göre **IsOpen** özelliğine **True** değerini vererek ilerleyebiliriz.

PopUp kontrolümüz ekranda gösterildiğine göre sıra geldi PopUp kontrolündeki UserControl1 içerisindeki düğmeye basıldığında söz konusu PopUp'ı ekrandan kaldırmaya.

```
Me.Visibility = Windows.Visibility.Collapsed
```

Yukarıdaki kod ile basit bir şekilde **UserControl** içerisindeki Düğme'nin **Click** durumunda UserControl'ümüzü görünmez hali getirerek sahneden kaldırıyoruz. Silverlight uygulamamızında code-behind dosyalarının son hali aşağıdaki şekilde;

[Page.xaml.vb]

```
Partial Public Class Page
    Inherits UserControl

    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub Dugme_Click(ByVal sender As Object, ByVal e As
        System.Windows.RoutedEventArgs) Handles Dugme.Click
        Dim PopupKontrol As New System.Windows.Controls.Primitives.Popup
        PopupKontrol.Child = New UserControl1
        PopupKontrol.IsOpen = True
    End Sub
End Class
```

[Usercontrol1.xaml.vb]

```
Imports System
Imports System.Windows
Imports System.Windows.Controls
Imports System.Windows.Media
Imports System.Windows.Media.Animation
Imports System.Windows.Shapes
```

```
Partial Public Class UserControll1
```

```
    Inherits UserControl
```

```
    Public Sub New()
```

```
        ' Required to initialize variables
```

```
        InitializeComponent()
```

```
    End Sub
```

```
    Private Sub Dugme_Click(ByVal sender As Object, ByVal e As  
System.Windows.RoutedEventArgs) Handles Dugme.Click
```

```
        Me.Visibility = Windows.Visibility.Collapsed
```

```
    End Sub
```

```
End Class
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 RC0 içerisinde ComboBox kullanımı.

Silverlight 2.0 RC0 ile gelen yeni kontrollerden biri de ComboBox kontrolü. Bu yazımızda Combobox'ın kullanımına, görsel düzenlemelerin nasıl yapıldığında göz atacağız. İlk olarak yeni bir Silverlight projesi yaratalım ve Expression Blend 2 içerisinde Asslet Library'de bir **Combobox** bularak sahneye yerleştirelim.

```
<UserControl x:Class="SilverlightApplication5.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
    <Grid x:Name="LayoutRoot"
        Background="White">
        <ComboBox HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Width="157"
            Margin="38,43,0,0"/>
    </Grid>
</UserControl>
```

Yukarıdaki XAML kodu sahnede boş bir Combobox yaratacaktı. **Combobox** içerisinde XAML kodu ile yeni öğeler eklemek istersek Combobox'in **Items** dizisine **ComboBoxItem'lar** eklememiz gerekecek.

```
<ComboBox HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Width="157"
    Margin="38,43,0,0">
    <ComboBox.Items>
        <ComboBoxItem Content="İlk Seçenek"></ComboBoxItem>
        <ComboBoxItem Content="İkinci Seçenek"></ComboBoxItem>
        <ComboBoxItem Content="Son Seçenek"></ComboBoxItem>
    </ComboBox.Items>
</ComboBox>
```

Her ComboBoxItem'in ayrıca bir de **IsSelected** özelliği var. Böylece uygulama ilk çalıştırıldığında ve ComboBox sahneye ilk geldiğinde hangi Item'in seçili olacağına karar verebilirsiniz.

```
<ComboBox HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Width="157"
    Margin="38,43,0,0">
    <ComboBox.Items>
        <ComboBoxItem IsSelected="True"
            Content="İlk Seçenek"></ComboBoxItem>
        <ComboBoxItem Content="İkinci Seçenek"></ComboBoxItem>
        <ComboBoxItem Content="Son Seçenek"></ComboBoxItem>
    </ComboBox.Items>
```

</ComboBox>

Tabi çoğu zaman bizler uygulamalarımızda bu şekilde seçenekleri XAML kodu içerisinde gömmeyeceğiz. Genelde bir veri kaynağımız olacak ve veri kaynağındaki listelerin ComboBox içerisinde gösterilmesini tercih edeceğiz. Bu durumda gelin şimdilik de ComboBox'a nasıl veri bağlayabileceğimizi inceleyelim.

İlk olarak **Urun** adında sınıfımızı tanımlayalım ve bu sınıf üzerinden örneğimizde kullanacağımız geçici veriyi üretelim.

[VB]

Class Urun

```
Private PAdi As String
Public Property Adi() As String
    Get
        Return PAdi
    End Get
    Set(ByVal value As String)
        PAdi = value
    End Set
End Property
```

```
Private PFiyat As Integer
Public Property Fiyat() As Integer
    Get
        Return PFiyat
    End Get
    Set(ByVal value As Integer)
        PFiyat = value
    End Set
End Property
```

End Class

[C#]

```
public class Urun
{
    private string PAdi;
    public string Adi
    {
        get { return PAdi; }
        set { PAdi = value; }
    }
}
```

```
private int PFiyat;
```

```
public int Fiyat
{
    get { return PFiyat; }
    set { PFiyat = value; }
}
```

Uygulamamız ilk çalıştırıldığında yukarıda tanımladığımız **Urun** sınıfından nesneler yaratarak bir **List** değişkenine ekleyeceğiz. Sonra da bu listeyi **ComboboxUrunler** adını verdigimiz Combobox'ımıza bağlayacağız.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Urunler As New List(Of Urun)
    Urunler.Add(New Urun With {.Adi = "Ürün Adı1", .Fiyat = 1000})
    Urunler.Add(New Urun With {.Adi = "Ürün Adı2", .Fiyat = 2000})
    Urunler.Add(New Urun With {.Adi = "Ürün Adı3", .Fiyat = 3000})
    Urunler.Add(New Urun With {.Adi = "Ürün Adı4", .Fiyat = 4000})

    comboboxUrunler.DisplayMember = "Adı"
    comboboxUrunler.ItemsSource = Urunler
End Sub
```

[C#]

```
void Page_Loaded(object sender, RoutedEventArgs e)
{
    List<Urun> Urunler = new List<Urun>();
    Urunler.Add(new Urun { Adi = "Ürün Adı1", Fiyat = 1000 });
    Urunler.Add(new Urun { Adi = "Ürün Adı2", Fiyat = 2000 });
    Urunler.Add(new Urun { Adi = "Ürün Adı3", Fiyat = 3000 });
    Urunler.Add(new Urun { Adi = "Ürün Adı4", Fiyat = 4000 });

    comboboxUrunler.DisplayMember = "Adı";
    comboboxUrunler.ItemsSource = Urunler;
}
```

Combobox'ın **ItemsSource** özelliğine aktardığımız liste otomatik olarak Combobox'ın içerisine tüm öğelerin yerleştirilmesini sağlayacaktır fakat bizim yarattığımız **Urun** sınıfının hangi özelliğinin Combobox içerisinde gösterileceğini belirlememiz gerekiyor. Bunun için ComboBox'in **DisplayMemberPath** özelliğine istediğimiz **Urun** sınıfının bir özelliğinin adını atıyoruz. Böylece **ComboBox** kendisine atanmış dizideki her ögenin söz konusu özelliğindeki veriyi kullanıcıya gösterecektir.

Eğer veri bağlantısı sonrası ComboBox içerisinde seçili olacak ögeyi belirlemek isterseniz kullanabileceğiniz iki yöntem var.

[VB]

```
comboboxUrunler.SelectedItem = (From gelenler In Urunler Where gelenler.Adi = "Ürün Adı2").SingleOrDefault
comboboxUrunler.SelectedIndex = Urunler.IndexOf((From gelenler In Urunler Where gelenler.Adi = "Ürün Adı2").SingleOrDefault)
```

[C#]

```
comboboxUrunler.SelectedItem = (from gelenler in Urunler where gelenler.Adi == "Ürün Adı2" select gelenler).SingleOrDefault();
comboboxUrunler.SelectedIndex = Urunler.IndexOf((from gelenler in Urunler where gelenler.Adi == "Ürün Adı2" select gelenler).SingleOrDefault());
```

Bunlardan ilki doğrudan ComboBox'in **SelectedItem** özelliğini tanımlayarak seçili öğeyi belirlemek. Yukarıdaki kod içerisinde elimizde diziden istediğimiz Item'ı bir LINQ sorgusu ile bularak söz konusu Item'in **SelectedItem** olması gerektiğini belirlemiş oluyoruz. Bir diğer seçenek ise doğrudan seçili olacak Item'in **Index** numarasını **SelectedIndex** değerine aktarmak.

ComboBox içerisinde kullanıcı bir Item seçtiğinde ise doğrudan ComboBox'in SelectionChanged event'i çalıştırılacaktır. Böylece doğrudan SelectedItem özelliği üzerinden seçili öğeyi alabilir ve bu ögleyle ilgili bilgilere ulaşabiliyoruz. Bizim örneğimizde seçili öğenin tipinin de Urun olduğunu bildiğimiz için doğrudan Urun sınıfına Cast ederek seçili öğeye ait tüm bilgilere ulaşabiliyoruz.

[VB]

```
Private Sub comboboxUrunler_SelectionChanged(ByVal sender As Object, ByVal e As System.Windows.Controls.SelectionChangedEventArgs) Handles comboboxUrunler.SelectionChanged
    MessageBox.Show(CType(comboboxUrunler.SelectedItem, Urun).Fiyat)
End Sub
```

[C#]

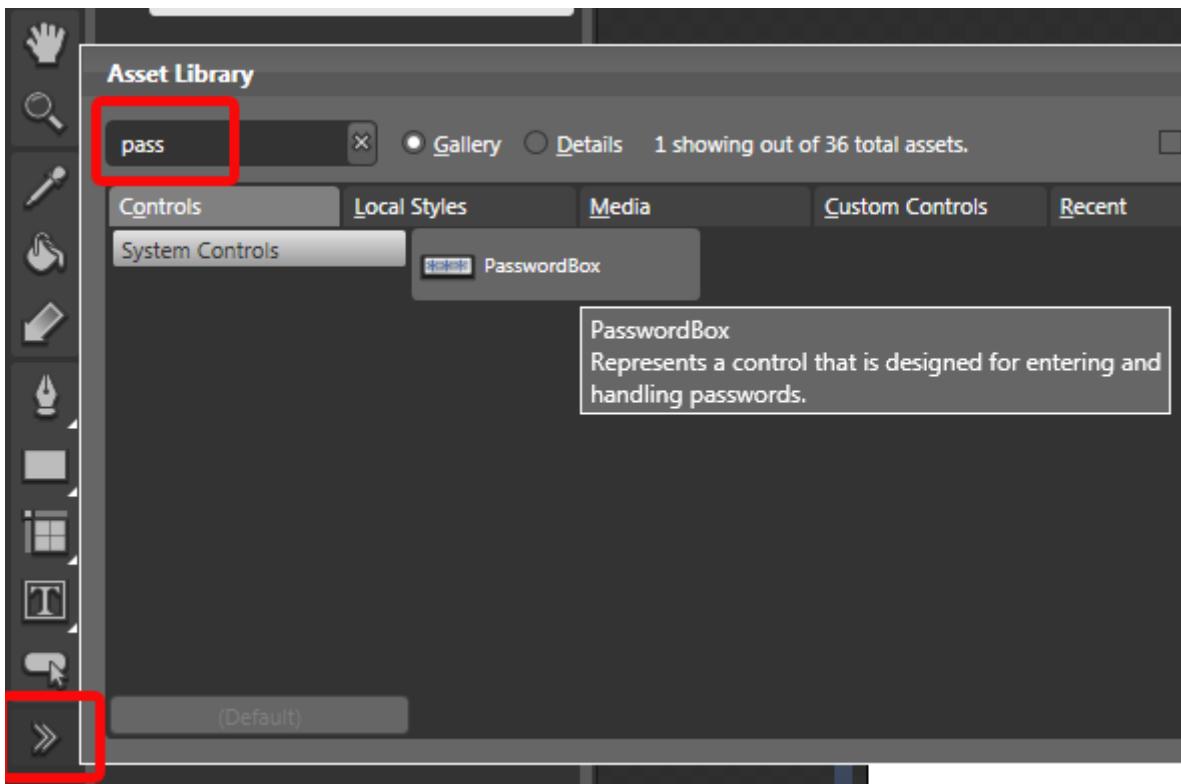
```
void comboboxUrunler_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    MessageBox.Show(((Urun)comboboxUrunler.SelectedItem).Fiyat.ToString());
}
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 RC0 içerisinde PasswordBox kullanımı.

Silverlight 2.0 RC0 öncesinde Silverlight içerisinde özellikle kullanıcı girişlerinde şifrelerin yazılacağı bir TextBox oluşturmak epey zahmetli bir i̇sti. Bunun için normal bir TextBox'ı sahneye yerleştirdi̇r后 sonra da bu TextBox'a tüm karakterleri * olan bir font bağılıyordu. RC0 ile beraber artık bu işleme özgü bir **PasswordBox** kontrolümüz var. Bu yazımızda PasswordBox'in kullanımına hızlıca dėgineceğiz.

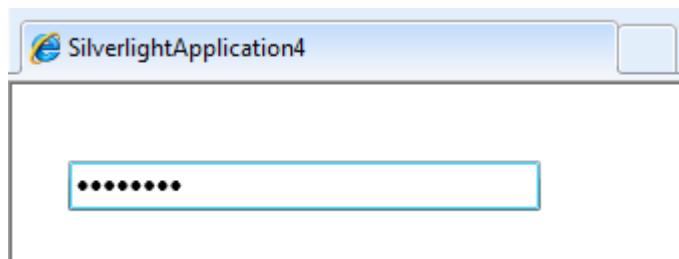


Expression Blend 2 içerisinde PasswordBox kontrolü.

Yeni bir Silverlight projesi yarattıktan sonra Expression Blend 2 içerisinde Asset Library bölümünde PasswordBox'ı bulabilirsiniz. Tasarım arayüzünden yukarıdaki şekilde bir PasswordBox alarak sahneye yerleştirdi̇ginizde XAML dosyasında aşağıdaki gibi bir kod ile karşılaşacaksınız.

```
<UserControl x:Class="SilverlightApplication4.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <PasswordBox Height="25" Margin="28,38,136,0" VerticalAlignment="Top"/>
    </Grid>
</UserControl>
```

Kodumuzda **PasswordBox** bir **Grid** içerisinde bulunduğu için **Margin**'i ayarlanmış durumda. Bu şekilde hazırlanan basit bir uygulamayı çalıştırıldığınızda **PasswordBox** içerisinde girdiğiniz her karakter bir nokta ile gösterilecektir.



PasswordBox'in varsayılan görünümü.

PasswordBox'in noktalarla doldurduğu karakterlerin yerine farklı bir karakter seçmek için PasswordBox'in **PasswordChar** özelliğinden faydalansınız. Bu özelliğe aktardığınız karakter PasswordBox içeresine yazılan her karakterin gösteriminde kullanılacaktır.

```
<UserControl x:Class="SilverlightApplication4.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
<Grid x:Name="LayoutRoot"
    Background="White">
<PasswordBox PasswordChar="*"
    Height="25"
    Margin="28,38,136,0"
    VerticalAlignment="Top"/>
</Grid>
</UserControl>
```

Programatik olarak da yukarıda bahsettiğimiz tüm özelliklere VB veya C# kodunuz ile ulaşabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Uygulamaları Parametre Gönderimi

Silverlight 2.0 uygulamalarını web sayfalarımıza **OBJECT** tagları ile koyacağımızı biliyoruz. Artık Silverlight 1.0'daki gibi JavaScript ile uğraşmak durumunda kalmayacağız. Durum böyle olunca tabi ki bu uygulamalara dışarıdan belirli durumlarda parametreler de göndermek gerekecek.

Örneğin bir Video Player hazırladınız ve aynı sayfada birden çok Video göstermek için kullanacaksınız fakat bu videolar da sunucu tarafındaki veriye bağlı olacak. Yani özetle Video Player Silverlight uygulamamız bir ASP.NET Repeater içerisindeyse video dosyasının adını nasıl Silverlight uygulamamıza aktarırız?

Dışarıdan Parametre Gönderimi

İlk olarak sayfamız içerisinde Object tagları arasında bir yererde parametrelerimizi belirtmemiz lazım. Bunun için aşağıdaki gibi bir yapı kullanabiliriz.

```
<object type="application/x-silverlight"
       width="100%" height="100%">
  <param name="source"
        value="ClientBin/deneme.xaml">
  <param name="initParams"
        value="metin=osman" />
</object>
```

İsterseniz parametre sayısını artırmak için yukarıdaki param tagının **value** özelliğine birden çok parametre ve değer çifti verebilirsiniz. Tek yapmanız gereken *metin=osman,deger=xx* şeklinde çiftleri birbirinden birer virgül ile ayırmak. Eğer ASP.NET ile beraber gelecek Silverlight sunucu kontrolünü kullanarak uygulamanızı sayfanıza ekliyorsanız bu durumda aşağıdaki gibi bir yapı kullanabilirsiniz.

```
<asp:Silverlight ID="Xaml1" runat="server"
  Source("~/ClientBin/deneme.xaml"
  InitParameters="metin=osman"
  Version="2.0"
  Width="100%"
  Height="100%" />
```

Peki uygulama içerisinde nasıl kullanacağız?

Bir önceki bölümde verdigimiz parametrelere Silverlight uygulamaları içerisinde **Application** nesnesinin **Startup** durumunda erişebiliyoruz. Söz konusu durumu uygulamanızın **App.xaml** dosyası içerisinde kodlayabiliyorsunuz.

```
Private Sub Application_Startup(ByVal o As Object, ByVal e As StartupEventArgs) Handles
Me.Startup
    e.InitParams("metin")
End Sub
```

Yukarıdaki şekli ile Application nesnesinin Startup durumuna parametre olarak gelen **StartupEventArgs** üzerinden **InitParams** dizisinde parametrelerimizi bulabiliyoruz. Fakat aslında bizim esas istediğimiz bu parametrelere doğrudan uygulamamızın ana XAML dosyalarında ulaşabiliyor olmak. Bunun için biraz daha uğraşmamız gerekecek. İlk olarak gelin içerisinde bir **TextBlock** olan XAML kodumuza bakalım. **Metin** parametresi ile Silverlight uygulamasına aktarılan metni bu TextBlock içerisinde göstereceğiz.

```
<UserControl x:Class="SilverlightApplication10.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Margin="29,26,41,48" Text="TextBlock" TextWrapping="Wrap"
x:Name="Metin"/>
    </Grid>
</UserControl>
```

Gördüğü üzere ortada çok karışık bir durum yok. Sadece bir TextBlock var. Peki nasıl olacak da parametrelerimizi sayfamıza aktaracağız. Aslında sayfa dediğimiz XAML dosyaları birer Class.

Partial Public Class Page
Inherits UserControl

.....
End Class

Yukarıdaki kod bizim herhangi bir XAML kodumuzun arkasında .NET kodunu gösteriyor. Page adında bir sınıf tanımlanmış ve bu sınıf aslında aşağıdaki şekilde XAML kodumuza da bağlanmış durumda.

```
<UserControl x:Class="SilverlightApplication10.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
```

Yani her XAML dosyası aslında birer sınıf olarak tanımlanıyor. Peki başlangıçta hangi XAML dosyasının açılacağı nasıl ayarlanıyor? Gelin App.XAML içerisindeki orijinal StartUp eventinin koduna bir göz atalım.

```
Private Sub Application_Startup(ByVal o As Object, ByVal e As StartupEventArgs) Handles
Me.Startup
    Me.RootVisual = New Page()
End Sub
```

İşte tam bu noktada uygulama açıldığında bizim Page sınıfından bir adet yaratılarak uygulamanın ana görseli haline getirilmiş. Yani bir XAML dosyasını yüklemek için aslında söz konusu XAML koduna bağlı .NET sınıfı kullanılmış. Bu durumda biz Page sınıfımızı bir Property eklesek ve bu Property'ye Application Startup'daki parametreleri aktarsak Page sınıfı içerisinde de tüm parametrelere ulaşmaz mıyz?

Kesinlikle ulaşırız. Hatta üzerine bir de yeni alternatif bir Constructer yazdık mı aslında işimiz daha da kolaylaşır. Gelin tek tek bunları yapalım.

```
Private PInitParams As System.Collections.Generic.IDictionary(Of String, String)
Public Property InitParams() As System.Collections.Generic.IDictionary(Of String, String)
    Get
        Return PInitParams
    End Get
    Set(ByVal value As System.Collections.Generic.IDictionary(Of String, String))
        PInitParams = value
    End Set
End Property
```

Yukarıdaki gördüğünüz Property'yi **Page** sınıfı içerisinde kullanacağımız. Bu Property aslında Application Startup'taki tüm **InitParams'lari** taşıyabilecek. Zaten söz konusu **InitParams'in** tipine de baktığımızda **System.Collections.Generic.IDictionary(Of String, String)** ile karşılaşıyoruz. Sıra geldi bir de yeni Constructor yazmaya.

```
Public Sub New(ByVal IncInitParams As System.Collections.Generic.IDictionary(Of String, String))
    Me.InitParams = IncInitParams
    InitializeComponent()
End Sub
```

Yukarıdaki kodu da ekledikten sonra artık istersek yeni bir **Page** sınıfı yaratırken atanacak olan Parametre listesini de verebiliriz. **Page** sınıfımızın tam kodu aşağıdaki şekilde sonlanıyor.

```
Partial Public Class Page
    Inherits UserControl
```

```
Private PInitParams As System.Collections.Generic.IDictionary(Of String, String)
Public Property InitParams() As System.Collections.Generic.IDictionary(Of String, String)
    Get
        Return PInitParams
    End Get
    Set(ByVal value As System.Collections.Generic.IDictionary(Of String, String))
        PInitParams = value
    End Set
End Property
```

```
Public Sub New()
    InitializeComponent()
End Sub
```

```
Public Sub New(ByVal IncInitParams As System.Collections.Generic.IDictionary(Of String, String))
    Me.InitParams = IncInitParams
    InitializeComponent()
```

End Sub

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Metin.Text = InitParams("metin")
End Sub
End Class
```

Page.Loaded durumunda da **Page** sınıfımızın kendi Property'si olan **InitParams** üzerinden **Metin** parametresini alarak TextBlock içerisinde yazdırıyoruz. Peki App.xaml'ın arkasına ne yazdık?

```
Partial Public Class App
    Inherits Application
```

```
    Public Sub New()
        InitializeComponent()
    End Sub
```

```
    Private Sub Application_Startup(ByVal o As Object, ByVal e As StartupEventArgs)
Handles Me.Startup
    Me.RootVisual = New Page(e.InitParams)
    End Sub
```

End Class

Gördüğünüz gibi Page sınıfını yaratırken doğrudan uygulamaya gelen tüm parametrelerin listesini de sınıfımıza aktarıyoruz. Böylece artık Page sınıfında da söz konusu tüm parametrelere ulaşabilecek.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Uygulamalarında farklı XAML dosyaları kullanmak.

Bir Silverlight uygulamasında birden çok XAML dosyası kullanarak dosyalar arasında geçiş yapmak isteyebilirsiniz. Fakat bu noktada maalesef sizi ufak bir sorun bekliyor. Herhangi bir Silverlight uygulamasının ana görsel elementini maalesef ki değiştirme şansınız yok. Ana görsel element olarak bahsettiğimiz şey aslında Silverlight 2.0 uygulamasının XAML koduna ait "Root Element" oluyor. Söz konusu root element sadece uygulama ilk çalıştırıldığında belirlenebiliyor. Peki bu durumda nasıl bir çözüm geliştirebiliriz?

Root Elementi bir Container olarak kullanınsak?

Uygulamamızın ana elementini bir Grid yapsak ve içerisinde XAML dosyalarımızı birer UserControl olarak yerleştirsek, böylece istediğimiz zaman Grid içerisindeki elementlerini değiştirerek farklı XAML dosyaları yükletemez miyiz? Güzel bir fikre benziyor. Deneyip görelim.

İlk olarak içerisinde basit bir şekilde iki adet XAML dosyası içeren yeni bir Silverlight 2.0 projesi yaratıyoruz. Bu dosyalardan birinde sadece bir TextBlock varken diğerinde bir TextBlock ve bir de Button olarak. İlk dosyadaki Button'a basıldığında ikinci dosyanın yüklenmesini sağlayacağız. Gelin önce dosyalarımızın XAML kodlarına hızlıca bir göz atalım.

[Page.xaml]

```
<UserControl
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SilverlightApplication4.Page"
    Width="640" Height="480">

    <Grid x:Name="LayoutRoot" Background="White" >
        <TextBlock x:Name="etiket" Margin="146,120,270,225" TextWrapping="Wrap"
        FontSize="72">
            <Run FontFamily="Portable User Interface" FontSize="14.66666984558106"
        FontStretch="Normal" FontStyle="Normal" FontWeight="Normal"
        Foreground="#FF000000" Text="1"/>
        </TextBlock>
        <Button Height="30" HorizontalAlignment="Left" Margin="169,0,0,78"
        VerticalAlignment="Bottom" Width="105" Content="Button" x:Name="Dugme"/>
    </Grid>
</UserControl>
```

[Page2.xaml]

```
<UserControl
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
```

```

x:Class="SilverlightApplication4.Page2"
d:DesignWidth="640" d:DesignHeight="480">

<Grid x:Name="LayoutRoot" Background="White" >
  <TextBlock Margin="154,113,206,172" TextWrapping="Wrap">
    <Run FontFamily="Portable User Interface" FontSize="14.666666984558106"
FontStretch="Normal" FontStyle="Normal" FontWeight="Normal"
Foreground="#FF000000" Text="2"/>
  </TextBlock>
</Grid>
</UserControl>

```

Herhangi bir Silverlight 2.0 projesinde ilk açılacak olan XAML dosyasının sınıfı **App.xaml** ile beraber çalışan code-behind dosyasına belirlenir. Bu dosyayı WPF uygulamalarında app.xaml'a veya ASP.NET uygulamalarındaki Global.asax'a benzetebilirsiniz. App.xaml uygulama ile ilgili global işlemlerin yapıldığı yerdir. İlk olarak app.xaml dosyasının code-behind kısmına geçerek uygulamanın **OnStartup** durumuna özel bir kod yazacağımız.

```

Private Sub OnStartup(ByVal o As Object, ByVal e As EventArgs) Handles Me.Startup
  Dim BirGrid As New System.Windows.Controls.Grid
  BirGrid.Children.Add(New Page)
  Me.RootVisual = BirGrid
End Sub

```

Yukarıdaki kod içerisinde ilk olarak bir **Silverlight 2.0 Grid** kontrolü yaratıyoruz. Unutmayın ki WPF'de de olduğu gibi **Grid** kontrolleri birer **DataGrid** değil. **Grid** kontrolleri HTML'deki karşılığı ile **table** diyebileceğimiz **container** kontrollerinden sadece biri. Yarattığımız **Grid** kontrolü içerisinde **Page.xaml** dosyamızı yüklemek için Page.xaml'a ait code-behind'daki sınıfından bir kopya yaratarak Grid'imizin children listesine ekliyoruz. Son olarak da uygulamamızın ana / root elementini Grid kontrolü olarak tanımlıyoruz. Böylece bundan sonra ana elementi değiştirmeden doğrudan Grid içerisindeki kontrolleri değiştirerek aslında tüm uygulama arayüzüne de değiştirmiş olacağız.

Gelelim Page.xaml dosyamızın code-behind kısmına ve bakalım Page.xaml içerisinde bulunan düğmemize basıldığında nasıl olacak da uygulama arayüzünden page.xaml'ı kaldırarak page2.xaml'ı yükleyeceğiz.

```

Private Sub Dugme_MouseLeftButtonDown(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Dugme.MouseLeftButtonDown
  Dim Root As Grid = CType(Me.Parent, Grid)
  Root.Children.Clear()
  Root.Children.Add(New Page2)
End Sub

```

Yapmamız gereken aslında şu an içerisinde olduğumuz Page.xaml dosyasının parent elementini bularak içerisindekileri silmek. Bunun için **Me.Parent** ile Gridimizi yakalıyoruz ve sonrasında **children.clear** ile sayfadaki herşeyi siliyoruz. Son olarak da Page2.xaml dosyamından bir kopya yaratarak aynı Grid'in içerisinde ekliyoruz. Böylece Silverlight 2.0 uygulamamızın içerisinde artık başka bir XAML dosyası yüklemiş olduk.

Silverlight 2.0 ve Adaptive Streaming

Silverlight dünyasında 1.0 sürümü ile başlayan video uygulamalarındaki yüksek performans gibi avantajların yenileri 2.0 sürümünde de tabii ki devam ediyor. Bu yenilikler arasında en ilginçlerinden biri **Adaptive Streaming**. Bugün internet ortamında video yayını dediğimizde en büyük sorunlarımızdan biri farklı bant genişliklerine hitap edebilecek içeriği oluşturmak. Maalesef ülkemizde 1 MBit bağlantıyı baz alarak ilerlemek zorunda kalsak da aslında 4 MBit'e kadar ADSL hatlarının kullananların sayısı hiç de az değil. Peki tüm bu kullanıcılar en uygun kalitede video yayını nasıl yapabiliriz?

Farklı bant genişliklerine farklı kalitelerde video yayımı!

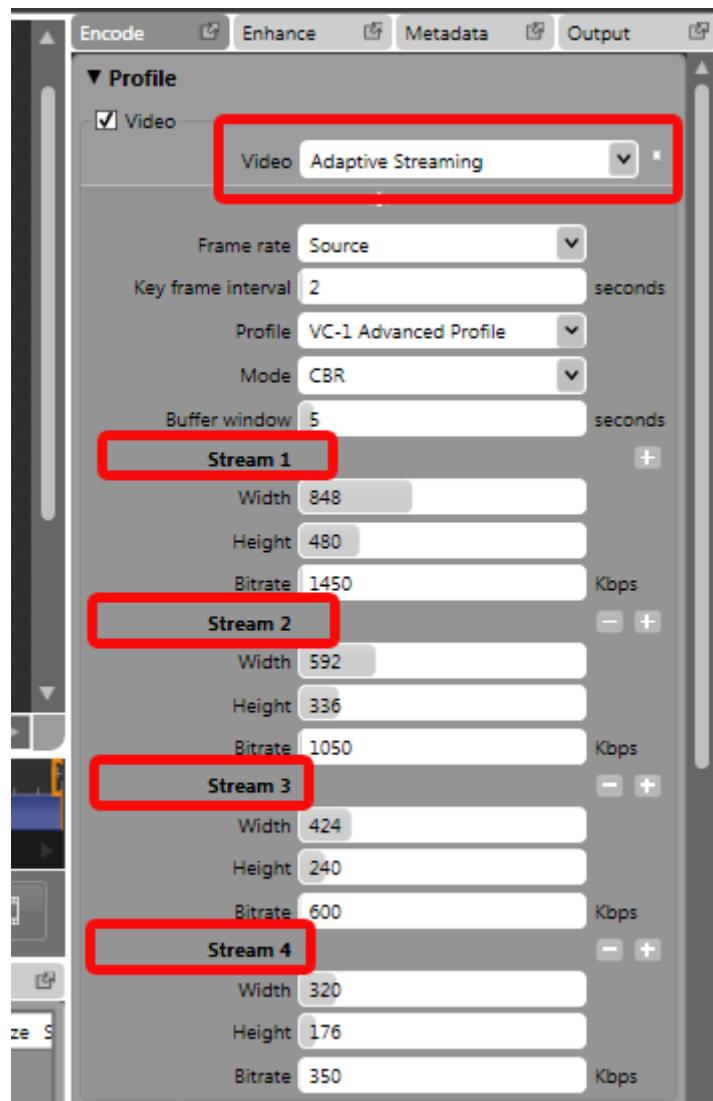
Bu hiç de yabancı olduğumuz bir çözüm değil. Elimizdeki video dosyasını farklı bitrate'lerde encode ederiz ve kullanıcılarımıza sitemize girdiklerinde farklı seçenekler sunarız. Her kullanıcı kendi bağlantısına göre istediği kaliteyi seçer. Peki ya videoyu seyrederken bant genişliğinde değişiklikler olursa? Varsayılmış ki kullanıcımız paylaşımı internet bulunan bir ortamda ve kişisel olarak elde ettiği bant genişliği değişimliyor. Bu durumda ne yapacağız? Tabii ki "Yükleniyor." mesajları göstereceğiz, yapacak pek bir şey yok! Aslında var!

Adaptive Streaming! Bırakın yayın akışı bant genişliğine uysun!

Silverlight 2.0 ile beraber kullanabildiğimiz Adaptive Streaming teknikleri ile artık videonuzun kalitesi ile kullanıcının bant genişliği arasında ilişki ile ilgilenmeniz gerekmıyor. Adaptive Streaming otomatik olarak kullanıcının bant genişliğini algılayarak uygun kalitedeki video dosyasının sunucudan çekiyor. Hatta bunu sadece video ilk oynatılırken değil video oynatıldığı sürece yapıyor! Böylece videoyu başta kaliteli bir şekilde izleyen bir kullanıcının bant genişliği düştüğünde video duracağına ve "yükleniyor" mesajları gösterileceğine videonun daha düşük kaliteli sürümlerine otomatik geçiş yapılıyor. Tabii ki bu geçişlerin hepsi otomatik olduğu üzere video duraksıyor ve kullanıcılarımız hiçbir şey hissetmiyor. Peki tüm bunlar için biz ne mi yapıyoruz? Hmm bir kaç düğmeye tıklamak.

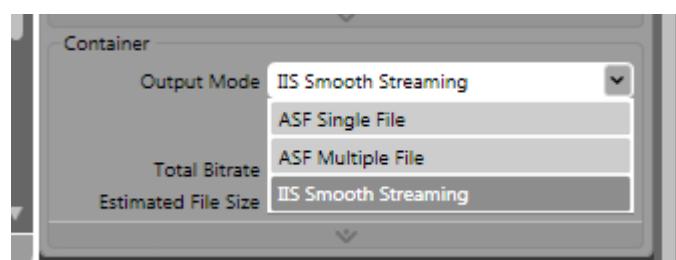
Expression Encoder 2 SP 1 ile gelenler!

Adaptive Streaming için video içeriği hazırlamanın birkaç yolu var. Aslında ilki oturup gerekli dosyaları tek tek encode ederek gerekli manifest XML dosyalarını da elle hazırlamak. Fakat bu zahmete girmek yerine doğrudan Expression Encoder 2'yi de kullanabilirsiniz. SP1 ile beraber Expression Encoder'a gerekli Adaptive Streaming şablonları da eklendi.



Adaptive Streaming için seçenekler.

Video profili olarak "Adaptive Streaming'i seçtiğinizde varsayılan ayarları ile dört farklı kalitede videonun encode edilmesi sağlanacaktır. İsterseniz ek kalite sekmeleri tanımlayabilir veya var olanları değiştirebilirsiniz. Tüm bu ayarları tamamladıktan sonra sıra geliyor Adaptive Streaming için hangi altyapıyı kullanacağınızı.



Adaptive Streaming için ne kullanalım?

Adaptive Streaming ile yayın yapmanın birkaç yolu var ama bunların öncesinden şu ufak detaylardan bahsedelim. Adaptive Streaming sadece HTTP üzerinden çalışıyor. Zaten Silverlight'in MMS üzerinden de HTTP protokolü ile çalıştığını biliyoruz. O nedenle pek bir değişiklik yok fakat ek olarak burada yarattığımız tüm video dosyalarının doğrudan HTTP

üzerinden yayınlanması gerektiğini hatırlatmak isterim. Yani dosyalar IIS gibi bir web sunucusunu üzerinden sunulmalı.

Yukarıdaki ekran görüntüsünde de görebileceğiniz üzere Expression Encoder bizden bir seçim daha yapmamızı istiyor. "Output Mode" olarak kastedilen aslında Adaptive Streaming için dosyaların ve Manifest'lerin nasıl hazırlanması gereği ile ilgili. Eğer "IIS Smooth Streaming"'i işaretlerseniz manifest harici bir XML olarak tutuluyor. "ASF" seçeneği ise duruma göre tek bir dosyada tüm farklı streamleri ve manifesti veya her stream için bir dosya ve dahili manifestleri oluşturuyor.

Ama tüm bunları kullanabilene kadar yaklaşık bir altı ay kadar beklemeniz gerekecek :) Bunun nedeni "Smooth Streaming" için IIS tarafından yüklü olması gereken HTTP Handler'ların daha Microsoft tarafından yayınlanmamış olması. "Smooth Streaming" handlerlarının 2009'un ilk çeyreğinde yayınlanması bekleniyor. Tabi MS'ten önce bu konuda başka bir sunucu altyapısı için kendi HTTP Handler'ını yazan olmaz ise :) sonuçta kodlar açık.

Peki ne yapacak Smooth Streaming?

IIS tarafından çalışacak olan Smooth Streaming altyapısını şu an <http://www.smoothhd.com/> adresinden test edebilirsiniz. Tabi gerçekten HD kalitesine ulaşabilmek için bağlantınızın kuvvetli olması şart aksi halde sistem düşük kaliteye otomatik olarak geçecektir.

Smooth Streaming sunucu tarafında video dosyalarını tamamen istek üzerine parçalara bölgerek istemciye gönderiyor. Bu parçaları ayrı ayrı 2, 3 saniyelik video dosyaları olarak düşünebilirsiniz. Bu parçalama sistemi sayesinde internetin doğasında yer alan proxy ve cache mekanizmalarından otomatik olarak video içerikleri de faydalananmiş oluyor.

İstemci tarafındaki tüm işlemler ise [MediaStreamSource](#) sınıfı ile hallediliyor. Bu sınıf ile gelen videodan kaç karenin eksik olduğu ve saniyede kaç karenin mevcut internet hattı üzerinden alınabildiği gerçek zamanlı olarak kontrol edilerek kaynak değişimi yapılabiliyor. Tüm bu işlemleri yapacak olan kodları ayrı bir AdaptiveStream sınıfı olarak Expression Encoder 2 SP1 ile beraber gelen Silverlight 2 video oynatıcılarına dahil edilmiş durumda. İsterseniz haricen bu sınıfları alıp kendi yarattığınız uygulamalara da ekleyebilirsiniz.

Şimdilik bu kadar, IIS 7 için "Smooth Streaming" yayınlığında işin sunucu tarafına da ayrı bir yazıda değineceğiz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ve ADO.NET Data Services

Silverlight 2.0 tarafından veritabanı erişimi için mecburen web servisleri kullanmak zorundayız. Durum böyle olunca tek tek veritabanındaki işlemler için ayrı web servisleri yazmak bir noktadan sonra işkenceye dönüştürüyor. Bugünlerde özellikle LINQ ve Entity Framework ile beraber data layer'larımızda ciddi kolaylıklarından faydalananabiliyoruz fakat web servisleri tarafından geldiğinde ise LINQ vs ile gelen nesneleri geri döndüren web servislerini tek tek yazmak yine can sıkıcı bir hal alıyor.

Aslında tüm bu sorunları çözebilecek bir altyapı .NET Framework içerisinde artık mevcut. **ASP.NET Data Services** adını verdigimiz altyapı ile beraber bir veritabanına erişimi doğrudan REST üzerinden yapabiliyorsunuz.

Peki nasıl?

İlk önce gelin ASP.NET Data Services sonuç olarak nasıl bir hizmet yaratıyor ona bakalım. Bugün herhangi bir veritabanına farklı where sorguları ile select'ler göndermek istesek bu sorgulardaki where cümleciklerini parametreli hale getirmemiz ve bu parametreleri alarak uygun datayı döndüren web servisleri yazmamız gerekiyor. Ancak bu şekilde Silverlight ile veritabanına erişebiliyoruz. Farklı senaryolardan eğer sorgularınızın filtreleme şekilleri değişirse bu sefer tekrar gidip uygun web servisini yazmak zorunda kalıyorsunuz. Başka bir seçenek olarak where cümleciklerini parametre alan bir servis yazılabılır fakat bu pek güvenli bir manzara olmaz.

Tüm bu problemleri çözmek için ASP.NET Data Services ile sorgularınızı yazabileceğiniz özel bir syntax geliyor ve artık URL üzerinden sorgu gönderebiliyoruz.

http://localhost:4351/WebDataService1.svc/Uruns(2)

Örneğin yukarıdaki gibi bir adrese gittiğimizde veritabanındaki Uruns tablosunda ID'si 2 olan ürünün bilgilerini XML olarak almış oluyoruz.

[XML]

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<entry xml:base="http://localhost:4351/WebDataService1.svc/" 
      xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" 
      xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" 
      xmlns="http://www.w3.org/2005/Atom">
  <id>http://localhost:4351/WebDataService1.svc/Uruns(2)</id>
  <title type="text"></title>
  <updated>2009-01-11T22:18:06Z</updated>
  <author>
    <name />
  </author>
  <link rel="edit" title="Urun" href="Uruns(2)" />
  <link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Kategori" 
        type="application/atom+xml;type=entry" title="Kategori" href="Uruns(2)/Kategori" />
  <category term="SilverlightApplication17.Web.Urun" 
            scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
```

```

<content type="application/xml">
<m:properties>
<d:ID m:type="Edm.Int32">2</d:ID>
<d:Adi>Ürün2</d:Adi>
<d:KategoriID m:type="Edm.Int32">1</d:KategoriID>
</m:properties>
</content>
</entry>

```

Tahmin ettiğiniz üzere aslında bizim normal şartlarda yazdığımız web servisleri de kısmen bu işi yapıyor. Yani bize XML ile istediğimiz veriyi gönderiyoruz. Şimdi bir düşünelim, bu şekilde esnek olarak URL üzerinden farklı sorgular gönderdiğimde bana istediğim veriyi XML ile sanki web servisinden sonuç dönüyormuş gibi döndüren bir sistem aslında benim Silverlight tarafından veritabanını sorgulamam için çok daha esnek bir yapı olmaz mı? Her farklı sorgu için ayrı ayrı web servisleri yazmaktan kurtulmaz mıymış? Evet :) amaç da zaten bu.

Tabi bu arada bir web servisinin çalışma şeklinde ve sağladığı XML'lerin syntax'ı ile buradaki biraz farklı. Konumuz dışında olsa da aradaki bu farkın bilincinde olmakta fayda var.

Yapalım şu işi...

Gelin hızlı bir örnek ile ASP.NET Data Services yapısının kullanımını ve Silverlight tarafından yansımalarını giriş seviyesine inceleyerek ilerleyelim. İlk olarak yeni bir Silverlight projesi yaratıyor ve yanına da güzel bir ASP.NET sitesi alıyoruz. ASP.NET sitenize sağ tıklayarak "Add New Item" diyerek gelen pencereden uygun dosyayı seçip yapabilirsiniz.

[VB]

```

Imports System.Data.Services
Imports System.Linq
Imports System.ServiceModel.Web

Public Class WebDataService1
    Inherits DataService(Of DataClasses1DataContext)

    Public Shared Sub InitializeService(ByVal config As IDataServiceConfiguration)
        config.SetEntitySetAccessRule("(*", EntitySetRights.AllRead)
    End Sub

End Class

```

ADO.NET Data Service'i projenize eklediğinizde hemen karşınıza yukarıdaki kodlar çıkacaktır. Koyu ile yazılı kısımları bizim elle eklememiz gerekiyor. **DataClasses1DataContext** olarak adı geçen şey aslında projemizdeki bir LINQ2SQL Classes dosyası. ADO.NET'in Data Servisleri üzerinden hangi Entity'leri yayinallyayacağını belirlememiz gerek. Bu nedenle aslında bir data servisi yaratmadan önce ya LINQ2SQL DBML dosyanızı hazırlamanız ya da Entity Framework tarafından Entity'lerinizi hazırlamanız

gerekıyor. Yazımızın konusu dışında olduğu için işin bu kısmına şimdilik degeinmeyeceğim. Önemli olan yarattığınız bu veri kaynağının yukarıdaki şekilde Data Servisi'nize aktarmanız.

Bir sonraki adımda ise veri kaynağındaki hangi Entity'lere ne şekilde erişim hakları vereceğiniz. Yani bu data servislerini kullanarak insanlar sadece SELECT mi yapabilecek, yoksa Update veya Delete işlemi de yapabilecekler mi ona karar vermemiz gerekiyor. Bu noktada güvenlik açısından epey dikkatli olmak gerek. Ben şimdilik **AllRead** diyerek sadece SELECT için veri kaynağındaki tüm tabloları * işaretini ile açtım.

Silverlight tarafından macelerar.

Servisimiz hazır olduğuna göre artık sıra geldi Silverlight tarafında bu servisi kullanmaya. Başlangıç için normal bir web servisi kullanmaktan pek farklı olmadığını söyleyebilirim. Silverlight projemize sağ tıklayarak "Add Service Reference" diyoruz ve ADO.NET Data Service'imizin SVC dosyasının adresini veriyoruz. Böylece gerekli istemci taraflı proxy yaratılmış oluyor.

Her zamanki gibi veri kaynağımızı kullanmadan önce servis üzerinden bir bağlantı kopyası almamız gerekecektir. Normal şartlarda Silverlight ile SoapClient sınıflarından kopya alırken bu sefer doğrudan **DataContext** alacağız.

[VB]

```
Dim Veri As New ServiceReference1.DataClasses1DataContext(New
Uri("http://localhost:4351/WebDataService1.svc/"))
```

DataContext'imizi alırken servisimizin tam yolunu da parametre olarak veriyoruz. Böylece artık bu servis üzerindeki tüm veriye ulaşabiliriz. Sıra geldi sorgularımızı yazmaya. Silverlight tarafında web servislerinin kullanımında da olduğu üzere tüm veri trafiğinin asenkron ilerleyeceğini hatırlarsak aslında sorgularımızı gönderip sonrasında ayrı bir event-listener ile sonucu alacağımızı da tahmin etmek zor değil. ADO.NET Data Services sorgularının URL üzerinden farklı bir syntax ile gittiğini görmüştük fakat elimizde bir DataContext olduğuna göre geri gelen IQueryable nesneleri LINQ ile sorgulayabiliyor olmamız gereklidir. Söz konusu LINQ sorguları otomatik olarak Data Services tarafına uygun şekilde çevrilerek gönderilecektir. Sözü daha fazla uzatmadan kodumuzu inceleyelim.

[VB]

```
Dim Sorgu As System.Data.Services.Client.DataServiceQuery(Of
ServiceReference1.Urun) = From gelenler In Veri.Uruns Where gelenler.ID = 2 Select
gelenler
Sorgu.BeginExecute(New AsyncCallback(AddressOf Geldi), Sorgu)
```

[C#]

```
System.Data.Services.Client.DataServiceQuery<ServiceReference1.Urun> Sorgu =
(System.Data.Services.Client.DataServiceQuery<ServiceReference1.Urun>)
from gelenler in Veri.Uruns where gelenler.ID == 2 select gelenler;
Sorgu.BeginExecute(new AsyncCallback(Geldi), Sorgu);
```

Yukarıdaki kodlar bir sorgunun sunucu tarafına gönderilmesini sağlayacak olan kodlar. Sorgumuzu ilk satırda standart LINQ sorgusu olarak yazıyoruz fakat unutmayın ki burada bazı sınırlamalar var. ADO.NET Data Services sorgularında tüm keyword'leri kullanmak mümkün olmuyor. O nedenle eğer buradaki LINQ sorgularında Take vs gibi bazı keyword'leri kullanırsanız Visual Studio hata verecektir.

Yazdığımız sorguyu bir **DataServiceQuery** değişkenine eşitliyoruz ve **Query** nesnemizi yaratırken de geriye ne tür bir nesne döneceğini yine servis üzerinden gelen nesne tanımı ile belirtiyoruz. Artık sorgumuz hazır olduğuna göre **BeginExecute** ile çalıştırabiliriz. Fakat bu noktada da iki parametreye ihtiyacımız var; birincisi bu sorgu tamamlandığında hangi event çalıştırılacak? Yani bir Callback lazım bize. Asenkron bir **Callback** yaratarak ilerliyoruz. İkinci parametre ise sorgunun kendisi. Böylece CallBack çalıştığında buradaki sorgunun state'i de geri dönecek.

[VB]

```
Sub Geldi(ByVal ar As IAsyncResult)
    Dim Sorgu As System.Data.Services.Client.DataServiceQuery(Of
ServiceReference1.Urun) = ar.AsyncState
    Dim result = Sorgu.EndExecute(ar)
    MessageBox.Show(result.SingleOrDefault().Adı)
End Sub
```

[C#]

```
void Geldi(IAsyncResult ar)
{
    System.Data.Services.Client.DataServiceQuery<ServiceReference1.Urun> Sorgu =
(System.Data.Services.Client.DataServiceQuery<ServiceReference1.Urun>)ar.AsyncState;
    var result = Sorgu.EndExecute(ar);
    MessageBox.Show(result.SingleOrDefault().Adı);
}
```

Geldi adındaki asenkron callback'imiz çalıştığında hemen kendisine parametre olarak gelen **Result'ın** için **AsyncResult** üzerinden **Sorgu** değişkenimizi alıyoruz. Artık sorgu tamamlandığında göre çalışma işlemini de sonlandırip sonucu almak gereklidir. **EndExecute** metoduna tekrar Callback'e gelen parametreyi verip sonucun bir değişkene aktarılmasını sağlıyoruz ve aldığımız değişken üzerinden istediğimiz verİYE ulaşabiliyoruz.

İste bu kadar...

Data Services yapısı ile Silverlight tarafından kodlamanın çok kolaylaştığını söylemek pek doğru olmaz. Elimizde veriyi sağlayan hazır bir web servisi olsaydı çok daha rahat bir kodlama ortamına sahip olabilirdik fakat Data Services bize web servislerine dokunmadan tek bir altyapıya bağlanarak istediğimiz sorguları çalışma zamanında oluşturma şansı tanıyor. Tabi bu sorgular sadece SELECT sorguları olmak zorunda değil, yeri geldiğinde Update, Delete ve Insert de yapabiliyoruz. Bu makalemizde giriş seviyesinde kalacağımız için şimdilik diğer işlemlere pek dokunmayacağız.

Silverlight 2.0 ve JavaScript kardeşliği

Silverlight 2.0 ile beraber istemci tarafında VB.NET veya C# kullanabileceğimizi artık biliyoruz. Durum böyle olunca söz konusu .NET dilleri ile Silverlight uygulamasının dışına çıkararak HTML sayfasındaki klasik HTML elementlerine ulaşmak da büyük kolaylıklar sağlayabilir. Böylece aslında kısmen şu an için istemci tarafındaki JavaScript'in hakimiyetini kırmak da mümkün. Tabi bunun tam tersi senaryolarda da JavaScript tarafından yola çıkarak Silverlight içerisinde .NET metodlarına ulaşmak isteyebiliriz.

HTML elementlerine ulaşalım

İlk olarak Silverlight tarafında yazdığımız kod ile sayfadaki HTML elementlerine nasıl ulaşabileceğimize bir göz atalım. Bunun için ilk olarak **Silverlight 2.0 Beta 1** uygulaması içerisinde dışarı çıkararak **Browser** içerisinde **HTMLPage**'in **Document** nesnesini yakalamamız gerekiyor.

```
Private Sub Dugme_MouseLeftButtonDown(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs) Handles Dugme.MouseLeftButtonDown
    Dim MevcutBelge As System.Windows.Browser.HtmlDocument =
        System.Windows.Browser.HtmlPage.Document
    MevcutBelge.GetElementById("icerik").SetAttribute("innerHTML", "DENEME METNİ")
End Sub
```

MevcutBelge değişkenimize sayfadaki mevcut **HTMLDocument** nesnesini aktardıktan sonra artık klasik JavaScript metodu gibi **GetElementById** metodunu kullanabiliyoruz. Silverlight 2.0 animasyonunun bulunduğu sayfadaki "icerik" adındaki HTML elementini yakaladıktan sonra basit bir şekilde **innerHTML** özelliğine farklı bir metin aktarıyoruz. Bu şekilde VB veya C# kodu ile sayfalardaki HTML elementlerine ulaşarak farklı işlemler rahatlık istemci tarafında yapılabilir.

Peki ya bir JavaScript kodu çalıştırmak istersek?

İstemcideki .NET kodunuz ile sayfalarınızda hali hazırda yer alana JavaScript kodlarını birbiri ile konuşturabiliyor olmak tabii ki çok önemli. HTML sayfası içerisinde tanımlanmış olan herhangi bir JavaScript metodunu doğrudan çalıştırmak için aşağıdaki kodu kullanabilirsiniz.

```
System.Windows.Browser.HtmlPage.Window.CreateInstance("Uyarı")
```

Yukarıdaki kod içerisinde "Uyarı" adındaki JavaScript fonksiyonu çalıştırılıyor. Eğer bu fonksiyon aşağıdaki gibi tanımlanarak bir parametre alsaydı daha farklı bir şekilde çalıştırılmamız gerekecekti.

```
function Uyarı(mesaj)
{
    alert(mesaj);
}
```

Yukarıdaki JavaScript fonksiyonunu Silverlight tarafından VB veya C# ile çalıştırırken göndereceğimiz parametreyi de ek olarak belirtmemiz gerekecek.

```
System.Windows.Browser.HtmlPage.Window.CreateInstance("Uyarı", New String("23"))
```

CreateInstance metoduna verdigimiz ilk parametre çalıştırırmak istediğimiz JavaScript metodunun adı şeklinde düzenlenirken verdigimiz diğer parametre ise aslında JavaScript fonksiyonumuza aktarmak istediğimiz diğer olası tüm parametrelerin bir dizisi.

JavaScript tarafından .NET'e yolculuk...

.NET tarafında hazırladığımız ve Silverlight ile istemci tarafında çalışan bir metodu JavaScript ile kullanabilmemiz için ilk aşamada yapmamız gereken bazı ayarlar var. Bunlardan ilki uygulamamız ilk yüklendiğinde elimizdeki Silverlight sayfasını HTML sayfasına bir **"ScriptableObject"** olarak tanımlamamız gerektiği. Bunu yapmak için uygulamamızın **App.xaml** dosyasının arkasındaki kodlardan faydalananacağız.

```
Private Sub OnStartup(ByVal o As Object, ByVal e As EventArgs) Handles Me.Startup
    Dim Sayfam As New Page()
    System.Windows.Browser.HtmlPage.RegisterScriptableObject("Page", Sayfam)
    Me.RootVisual = Sayfam
End Sub
```

Gördüğünüz gibi ilk satırda Silverlight uygulamamızda gösterilecek sayfayı yaratıyoruz ve bir sonraki satırda hemen söz konusu sayfayı mevcut HTML sayfamıza bir **ScriptableObject** olarak kaydediyoruz. Sayfayı Silverlight uygulamamızın ana görsel ögesi haline getirerek kullanıcıya gösterilmesini de sağmalayı unutmayalım.

Bu ayarı tamamladıktan sonra sıra geldi JavaScript tarafı ile paylaşacağımız fonksiyonumuzu yazmaya.

```
<System.Windows.Browser.ScriptableMember()>_
Public Function Kare(ByVal X As Integer) As Integer
    Return X ^ 2
End Function
```

Yukarıdaki örnekte basit fonksiyon kendisine parametre olarak verilen sayının karesini alarak geri döndürüyor. Bu fonksiyonun JavaScript tarafı ile paylaşılabilmesi için kesinlikle yukarıdaki şekilde **ScriptableMember** olarak işaretlenmiş olması gerekiyor. Son olarak sıra geldi bu metodu kullanabilecek olan JavaScript kodunu yazmaya fakat onun öncesinde dikkat etmemiz gereken bir nokta. Sayfamızdaki herhangi bir Silverlight uygulaması içerisindeki bir metodu dışarıdan kullanabilmek için ilk olarak söz konusu Silverlight uygulamasını bulmamız gerekiyor. Yani Silverlight uygulamamızın bir isminin olması şart.

```
<object id="SL" data="data:application/x-silverlight," type="application/x-silverlight-2-b1" width="100%" height="100%">
    <param name="source" value="SilverlightApplication8.xap"/>
    <param name="onerror" value="onSilverlightError" />
    <param name="background" value="white" />

    <a href="http://go.microsoft.com/fwlink/?LinkID=108182" style="text-decoration: none;">
        
```

```
</a>
</object>
```

Yukarıdaki şekliyle sayfaya yerleştirilen bir Silverlight uygulamasına isim vermek için uygulamaya ait object taglarına bir ID bilgisi atamak yeterli olacaktır. Artık ID bilgisi üzerinden Silverlight uygulamamız ulaşarak aşağıdaki gibi JavaScript içerisinde .NET metodumuzu çalıştırabiliriz.

SL.Content.Page.Kare(2)

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 ve JSON Serialize / DeSerialize İşlemleri

Günümüzde çoğu AJAX uygulamasında veri transferi için JSON formatı kullanılıyor. ASP.NET programcılığı tarafında biz farkında olmasak da PageMethod'lar, Web Servisleri veya WCF Servisleri sunucudan JavaScript istemciye asenkron (AJAX) veri taşırken JSON ile çalışıyor. ASP.NET dışındaki dünyaya da baktığımızda tabi ki bu kural geçerliliğini koruyor, örneğin bugün twitter.com kendi uygulamalarından dışarıya asenkron veri aktarırken JSON formatını kullanıyor. Peki Silverlight 2.0 ile salt AJAX mantığından kurtularak artık Web Servislerimizi veya WCF servislerimizi doğrudan asenkron olarak kullanabildiğimize göre karşımıza eski JSON kaynakları gelirse ne yapacağız?

.NET nesnelerinden JSON oluşturmak.

Hikayenin tersinden başlayalım ve ilk olarak istemci tarafında JSON verisi nasıl yaratırız onu inceleyelim. Zaten genelde harici bir web servisinden JSON verisi alacaksanız büyük ihtimal ile elinizdeki hazır bir JSON verisini de web servisine göndermek durumunda kalacaksınız. Bu gibi bir durumda rahatlıkla elimizdeki .NET nesnelerini JSON formatına çevirebiliyor olmalıyız.

Silverlight 2.0 Beta 1 ile beraber gelen sınıflardan **System.Runtime.Serialization.JsonDataContractJsonSerializer** sınıfını kullanacağız. Bu sınıfı normal şartlarda kullanmak isterseniz herhangi bir Silverlight uygulamasında Intellisense ile bulma şansınız olmayacağından emin olabilirsiniz. Söz konusu sınıf harici olarak **System.ServiceModel.Web.dll** dosyası içerisinde kendisini projemize referans olarak almamızı bekliyor. Gerekli referansı projeye ekledikten sonra rahatlıkla JSON işlemlerini tamamen istemci tarafında yürütebiliyoruz.

Uygulamamızda bir öğrencinin adını ve soyadını taşıyacak bir **Ogrenci** sınıfı kullanacağız. JSON ile çeviri işlemleri yaparken elimizdeki çevireceğimiz nesnenin tipinin belirli şekilde tanımlanmış olması gerekiyor. O nedenle hemen aşağıdaki kod ile tipimizi tanımlayalım.

Public Class Ogrenci

```

Private Padi As String
Public Property Adi() As String
  Get
    Return Padi
  End Get
  Set(ByVal value As String)
    Padi = value
  End Set
End Property

```

```

Private PSoyadi As String
Public Property Soyadi() As String
  Get
    Return PSoyadi
  End Get
  Set(ByVal value As String)

```

```

    PSoyadi = value
    End Set
End Property

Sub New()
End Sub

Sub New(ByVal adi As String, ByVal soyadi As String)
    Me.Adi = adi
    Me.Soyadi = soyadi
End Sub
End Class

```

Bu basit işlemi tamamladıktan sonra uygulamamıza üç adet metin kutusu ve iki de düğme yerleştirelim. Bu metin kutularından ikisi öğrencinin adının ve soyadının gözükeceği yer, diğeri ise yarattığımız JSON verisinin yazdırılacağı konum olacak. Düğmelerimizi de işlemleri yapmak için kullanacağız.

```

<UserControl x:Class="JSON.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Height="31" HorizontalAlignment="Left" Margin="22,19,0,0"
            VerticalAlignment="Top" Width="151" Text="Adı" x:Name="Adi1"/>
        <TextBox Height="26" HorizontalAlignment="Left" Margin="22,54,0,0"
            VerticalAlignment="Top" Width="151" Text="Soyadı" x:Name="Soyadi1"/>
        <TextBox Margin="22,142,23,19" Text="TextBox" x:Name="Sonuc"/>
        <Button Height="29" HorizontalAlignment="Right" Margin="0,99,97,0"
            x:Name="Dugme2" VerticalAlignment="Top" Width="102" Content="JSON'dan Al"/>
        <Button Height="29" HorizontalAlignment="Left" Margin="45,99,0,0"
            x:Name="Dugme1" VerticalAlignment="Top" Width="102" Content="JSON Yarat"/>
    </Grid>
</UserControl>

```

İlk olarak **Dugme1** nesnesinin arkasına gerekli kodları yazarak metin kutuları içerisinde öğrencinin adını ve soyadını alıp bir öğrenci nesnesi yaratalım. Sonrasında da bu nesneyi JSON verisine çevirerek **Sonuc** adındaki metin kutusuna yazdırıralım.

```

Dim Cevirici As New
System.Runtime.Serialization.JsonDataContractJsonSerializer(GetType(Ogrenci))
Dim Veri As New IO.MemoryStream
Cevirici.writeObject(Veri, New Ogrenci(Adi1.Text, Soyadi1.Text))
Sonuc.Text = Text.Encoding.UTF8.GetString(Veri.ToArray, 0, Veri.Length)

```

Cevirici adını verdigim nesne bir **DataContractJsonSerializer** nesnesi. Bu nesnenin **WriteObject** metodunu kullanarak elimizdeki uygun bir .NET nesnesini JSON formatına çevirebiliyoruz. **WriteObject** metodu toplamda iki parametre alıyor; bunlardan ilki çeviri işlemi esnasında oluşan JSON verisinin yazdırılacağı **Stream** nesnesi, diğeri ise çevrilecek

olan nesnenin ta kendisi. Ben bu örnekte bir **MemoryStream** kullandım. Son olarak eldeki Stream'i de bir metne çevirerek **Sonuc** adındaki metin kutusu içerisinde yazdırıyoruz.

Böylece uygulamamızda dinamik olarak JSON yaratma sorununu çözmüş olduk. Tamamen istemci tarafında rahatlıkla yarattığımız .NET nesnelerini JSON formatına çevirebiliyoruz. Şimdi de tam tersi bir senaryoya göz atalım.

JSON verisinden .NET nesneleri yaratmak

Bir önceki bölümde kullandığımız örneği aynen kullanmaya devam edelim. Bu sefer de tam tersi bir işlem yaparak **Sonuc** adındaki metin kutusu içerisinde yazılan JSON verisini okuyarak içerisinde öğrencinin adını ve soyadını alıp diğer metin kutularının içerisinde yerleştirelim.

Dim Cevirici As New

System.Runtime.Serialization.Json.DataContractJsonSerializer(GetType(Ogrenci))

Dim Veri As New IO.MemoryStream(System.Text.Encoding.UTF8.GetBytes(Sonuc.Text))

Dim GenelOgrenci = CType(Cevirici.ReadObject(Veri), Ogrenci)

Adi1.Text = GenelOgrenci.Adi

Soyadi1.Text = GenelOgrenci.Soyadi

Her zamanki gibi ilk olarak **Cevirici** nesnemizi **Ogrenci** tipinden yaratıyoruz. Bu sefer **DataContractJsonSerializer** sınıfının **ReadObject** metodunu kullanacağız. **ReadObject** metodu okuyacağı veriyi bir **Stream** olarak istediği için **Sonuc** adındaki metin kutusu içerisindeki metinden ilk önce bir **Byte** dizisi sonra da bu diziden bir **MemoryStream** yaratıyoruz. Aldığımız **MemoryStream**'ı **Cevirici** nesnemizin **ReadObject** metoduna verdığımızde söz konusu metod bize bir **Object** döndürüyor. Doğal olarak JSON verisi içerisindeki nesnenin hangi .NET nesnesine denk geldiğini bilme şansı yok. O nedenle biz elle casting yaparak aldığımız **Object** tipindeki değişkeni **Ogrenci** tipine değiştiriyor ve gerekli verileri alarak diğer metin kutularının içerisinde yerleştiriyoruz.

Her iki uygulamayı da bir örnek projede yaptığımızda ilk önce metin kutularına veri girerek JSON verisini yaratıyor sonrasında da JSON verisini **Sonuc** metin kutusunda elle değiştirip tekrar diğer metin kutularına güncel değerlerin aktarılabilmesi için **DeSerialize** işleminin yapılmasını sağlayabiliyoruz.

Uygulamamızın tam kodu aşağıdaki şekilde sonlanıyor.

Partial Public Class Page

Inherits UserControl

Public Sub New()

 InitializeComponent()

End Sub

Public Class Ogrenci

 Private Padi As String

 Public Property Adi() As String

 Get

 Return Padi

```

End Get
Set(ByName value As String)
    Padi = value
End Set
End Property

```

```

Private PSoyadi As String
Public Property Soyadi() As String
    Get
        Return PSoyadi
    End Get
    Set(ByName value As String)
        PSoyadi = value
    End Set
End Property

```

```
Sub New()
```

```
End Sub
```

```
Sub New(ByVal adi As String, ByVal soyadi As String)
```

```
    Me.Adi = adi
    Me.Soyadi = soyadi
End Sub
```

```
End Class
```

```

Private Sub Dugme1_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme1.Click
    Dim Cevirici As New
System.Runtime.Serialization.JsonDataContractJsonSerializer(GetType(Ogrenci))
    Dim Veri As New IO.MemoryStream
    Cevirici.writeObject(Veri, New Ogrenci(Adi1.Text, Soyadi1.Text))
    Sonuc.Text = Text.Encoding.UTF8.GetString(Veri.ToArray, 0, Veri.Length)
End Sub

```

```

Private Sub Dugme2_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Dugme2.Click
    Dim Cevirici As New
System.Runtime.Serialization.JsonDataContractJsonSerializer(GetType(Ogrenci))
    Dim Veri As New
IO.MemoryStream(System.Text.Encoding.UTF8.GetBytes(Sonuc.Text))
    Dim GenelOgrenci = CType(Cevirici.readObject(Veri), Ogrenci)
    Adi1.Text = GenelOgrenci.Adi
    Soyadi1.Text = GenelOgrenci.Soyadi
End Sub
End Class

```

Hepinize kolay gelsin.

Silverlight 2.0 ve Socket Programlama Mucizesi

Socket programlama Silverlight çıktılarından beri biz yazılım geliştiricilerin en büyük hayali ve bu hayal gerçek oluyor. Silverlight 2.0 Beta 1 ile beraber Socket Programlama karşımızda. Yani artık istemci ile sunucu arasında TCP/IP ile haberleşmek mümkün. Tabi belirli kurallar var; bu kurallardan ilki sunucudaki uygulamanın istemci uygulamanın yüklediği web sitesi ile aynı konumda olması. Yani sunucu uygulamanızın web siteniz ahmet.com ise ahmet.com'un reverse DNS Look-Up ile bakıldığından çıkan IP adresine sahip sunucuda bulunması gerekiyor. Bu durumun Silverlight'in Beta 1 sonrası sürümlerinde policyfile gibi sistemlerde daha esnek hale getirileceği söylentiler arasında fakat baktığımızda şu anki hali ile bile süper bir potansiyel söz konusu.

Peki nedir bunun avantajı?

Diyorum ya, hayalimizdi diye, peken neden? Web sitelerinde güncel bilgi göstermek her zamanki en büyük derttir. Bunu yapabilmek için çok eskilere döndüğümüzde bazı meta tagları ile belirli aralıkla sayfanın refresh olmasını sağladığımız günler bile olurdu. IFRAME vs nin gelmesi ile en azından bunu sayfada kısmi bölgümlerde uygulayabilir hale geldik. Sonrasında AJAX geldi ve çok daha sinsi bir şekilde kullanıcı farkında olmadan belirli aralıklarla sunucudan yeni veri talebinde bulunarak sayfa değişmeden yeni içeriği gösterebildik. Oysa hep bizi rahatsız eden bir nokta vardı, o da şu; sürekli istemciden sunucuya bağlanarak bir veri değişikliğinin olup olmadığını kontrol etmek durumunda kalıyordu. Sunucuya "Yeni birşey var mı?" diye dakikada bir soruyor ve çoğunda da hırsız ile geri dönüyor. Keşke sunucu bize bir "Alo" diyebilse ve değişiklik olduğunda istemciyi haberدار edebilse? Teknik olarak bu güvenlik sebepleri nedeniyle zaten mümkün değil çünkü bir istemci bilgisayara dışarıdan içeriye bağlantı kuramazsınız (kuramamanız gereklidir). Peki nasıl oluyor da Socket Programming bunu yapıyor? Aslında aşçımıyor, yine istemci sunucuya bağlanıyor fakat söz konusu bağlantı TCP bazında olduğu herhangi bir trafiğe neden olmadan sürekli açık tutulabiliyor. Durum böyle olunca sunucu kendisine bağlı istemciye istediğiinde söz konusu bağlantı üzerinden rahatlıkla ulaşabiliyor.

Sunucu tarafından işe başlayalım.

İlk olarak sunucudaki programımızı hazırlayalım. Söz konusu program kendisine gelen tüm istekleri karşılayarak gerektiğinde istemcilere veri gönderecek. Bizim programımız içerisinde bir TextBox bulunacak ve kutu içerisinde metin yazıldıkça kendisine bağlı tüm istemcilere bu metin sürekli gönderilecek.

```
Dim Baglilar As New System.Collections.Generic.List(Of System.IO.StreamWriter)
Dim yeniTR As System.Threading.Thread
Dim TCPBaglantilari As New System.Threading.ManualResetEvent(True)
Dim Dinleyici As New System.Net.Sockets.TcpListener(System.Net.IPAddress.Any, 4530)
```

İlk olarak global değişkenlerimizi tanımlıyoruz. Bunlardan ilki olan **Baglilar** değişkeni sunucuya bağlı olan istemcilere veri gönderecek olan **StreamWriter** nesnelerini bir listesini taşıyacak. Böylece istediğimizde bu liste içerisinde gezerek tüm bağlı olan istemcilere veri gönderebileceğiz. **yeniTR** adındaki değişkenimizi yeni bir Thread yaratmak ve her yerden kendisine ulaşabilmek için kullanacağız. **TCPBaglantilari** değişkenimiz var olan Threat'in blocklanması ve tüm event-larının sıfırlanması için kullanılacak. **Dinleyici** adındaki değişkenimizi ise tüm istemcileri dinleyecek olan ve gelen bağlantıları algılayacak olan

TCPListener nesnemizin ta kendisi. Gördüğünüz gibi bu nesneyi tanımlarken iki parametre aktarmışız. Bunlardan ilki herhangi bir IP adresi üzerinden bu uygulamaya bağlanabileceğin anlamına gelirken diğer ise sadece 4530 portu üzerinden bağlantı yapılabileceği anlamına geliyor. **Silverlight 2.0 Beta 1** şu anda **4502-4532** aralığındaki portları kullanabiliyor.

```
Private Sub btn_Basla_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Basla.Click
    yeniTR = New System.Threading.Thread(AddressOf Bekle)
    yeniTR.Start()
End Sub
```

Uygulamamızdaki düğmeye basıldığında dinleme işlemini başlatmak üzere yeni bir Thread yaratıyoruz. Söz konusu Thread **Bekle** Sub'ına bağlı. Yarattığımız thread'i hemen başlatıp yolumuza devam edelim.

```
Sub Bekle()
    Dinleyici.Start()
    While True
        TCPBaglantilari.Reset()
        Dinleyici.BeginAcceptTcpClient(New System.AsyncCallback(AddressOf
BaglantiGeliyor), Nothing)
        TCPBaglantilari.WaitOne()
    End While
End Sub
```

Yeni Thread içerisinde hemen Dinleyici nesnemizi başlatıyoruz ve kısır bir döngüye giriyoruz. Sürekli olarak elimizdeki Threadi sıfırlayarak Dinleyici'nin **BeginAcceptTcpClient** metodu ile istemciden bir bağlantı geleceğini belirterek **WaitOne** metodu ile de bekliyoruz. Eğer burada bir bağlantı gelir ve başarılı veya başarısız şekilde sonuçlanırsa bu döngü başa gelerek tekrar yeni bir bağlantı bekleyecek. **BeginAcceptTcpClient** içerisinde parametre olarak verdığımız **BaglantiGeliyor** event-handları herhangi bir bağlantı geldiğinde çalıştırılacak.

```
Private Sub BaglantiGeliyor(ByVal ar As System.IAsyncResult)
    TCPBaglantilari.Set()
    Dim Musteri As System.Net.Sockets.TcpClient = Dinleyici.EndAcceptTcpClient(ar)
    If Musteri.Connected Then
        Dim yazici As New System.IO.StreamWriter(Musteri.GetStream)
        yazici.AutoFlush = True
        Baglilar.Add(yazici)
        yazici.WriteLine("Bağlandınız.")
    End If
End Sub
```

Baglanti geldiği anda bekleyen Threat'leri devam ettirmek adına **TCPBaglantilari.Set()** metodunu çağrıyoruz. Unutmayalım ki programımız aynı anda sadece tek istemcinin bağlantısını authenticate edebilir, yani diğerleri bir önceki istemci bağlanıp bağlantısını oluşturana kadar bekleyecektir. Bu noktada artık istemci bağlantı kurma işlemini tamamladığı için diğerlerine yol veriyoruz. Musteri adında bir **TCPClient** yarattıktan sonra **Dinleyici'nin EndAcceptTcpClient** metodu ile args parametresi üzerinden gelen Request'i alıyoruz. Eğer

Musteri bağlı ise, yani **Connected** ise artık sıra geldi ona veri göndermeye. Musteri'nin yani **TCPClient'in Stream'ini** alarak bundan bir **StreamWriter** oluşturuyoruz. Bu Stream üzerinden artık istemciye istediğimiz veriyi gönderebiliriz.

Unutmayın ki uygulamamızda bir **TextBox** vardı ve içerisinde birşey yazıldığında tüm bağlı kullanıcılaraya gönderecektik. Bunun için sonra kullanabilmek adına elimizdeki canlı **Stream'leri** saklamak sorundayız. Global olarak tanımladığımız **Baglilar** adında **Generic.List**'e elimizdeki **Stream'i** aktarıyoruz. Bu arada kullanıcıya "Bağlandınız" diye de bir metin gönderiyoruz.

```
Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged
    For Each x As System.IO.StreamWriter In Baglilar
        x.WriteLine(TextBox1.Text)
    Next
End Sub
```

Artık **TextBox** içerisinde değişiklik olunca bunu istemcilere göndermek çok kolay. Basit bir şekilde **Generic.List** içerisinde gezin ve her Stream'e elinizdeki veriyi gönderin.

İstemci tarafında neler olacak?

Silverlight tarafında çok basit görsellikte bir uygulamamız olacak. Sadece bir **TextBlock!** Uygulama tarayıcı içerisinde ilk açıldığında sunucuya bağlanacak ve gelen veriyi sürekli olarak söz konusu **TextBlock** içerisinde gösterecek.

```
<UserControl x:Class="SocketsClient.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Margin="42,46,37,125" Text="TextBlock" TextWrapping="Wrap"
x:Name="Metin"/>
    </Grid>
</UserControl>
```

XAML kodumuzu yukarıdaki şekilde düzenledikten sonra hemen code-behind dosyasına gecerek bağlantı kodlarını yazmaya başlayalım.

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Hat As New System.Net.Sockets.Socket(Net.Sockets.AddressFamily.InterNetwork,
    Net.Sockets.SocketType.Stream, Net.Sockets.ProtocolType.Tcp)
    Dim Args As New System.Net.Sockets.SocketAsyncEventArgs

    Args.UserToken = Hat
    Args.RemoteEndPoint = New System.Net.DnsEndPoint("localhost", 4530)
    AddHandler Args.Completed, AddressOf Baglandi
    Hat.ConnectAsync(Args)
End Sub
```

Silverlight uygulaması ilk yüklendiğinde hemen bir Socket bağlantısı yaratmak için System.Net.Sockets.Socket üzerinden ilerliyoruz. **Hat** adındaki değişkenimizi yaratırken verdığımız parametrelerden ilki olan **InterNetwork** bizim bağlantı için IPv4 kullanacağımızı ve ikinci parametre de TCP kullanacağımızı belirtiyor. Asenkron bir çalışma yapısı için bir de **SocketAsyncEventArgs** nesnesi yaratarak söz konusu nesneyi **Hat** adındaki Socket'imize bağlıyoruz. **Args'in RemoteEndPoint** özelliği istemcinin bağlanacağı sunucunun adresini ve port bilgisini içeriyor. Bağlantı oluşturulduğunda elimizdeki **SocketAsyncEventArgs** nesnesi olan **Args'in Completed** event-handları çalışacağı için ona da dinamik bir event-handler olarak **Baglandi** metodunu atıyoruz. Son olarak **ConnectAsync** diyerek **Socket** değişkenimizin eldeki **SocketAsyncEventArgs** ile sunucuya bağlanması sağlıyoruz.

```
Private Sub Baglandi(ByVal sender As Object, ByVal e As
System.Net.Sockets.SocketAsyncEventArgs)
    Dim Gelen(1024) As Byte
    e.SetBuffer(Gelen, 0, Gelen.Length)
    RemoveHandler e.Completed, AddressOf Baglandi
    AddHandler e.Completed, AddressOf Geldi
    Dim Baglanti As System.Net.Sockets.Socket = CType(e.UserToken,
System.Net.Sockets.Socket)
    Baglanti.ReceiveAsync(e)
End Sub
```

Artık istemci sunucuya bağlandığında göre sıra geldi karşı taraftan yeri geldiğinde veriyi almaya. Hatta ilk bağlantı esnasında hatırlarsanız bizim sunucumuzun "Bağlandınız" diye bir metin gönderiyordu. Gelen veriyi alabilmek için ve sürekli gelen veriyi dinlemek için eldeki Socket'i alarak sürekli dinleme durumunda olmamız şart. Kodumuzdaki event-handler içerisinde e parametresi aslında bizim bir önceki adımda tanımladığımız **Args** adındaki **SocketAsyncEventArgs**'nın ta kendisi. SetBuffer ile veriyi önbelleklemek için kullanacağımız ayarları da bir **Byte** değişkeni üzerinden aktardıktan sonra ilginç bir şekilde elimizdeki event-handlerları değiştiyoruz. Bundan sonra **Args'in Compeleted** durumu yeni bir bağlantı olduğunu değil yeni veri geldiğini bildireceği için farklı bir event-handleri bağlamamız gerekiyor. **Baglandi** adındaki metodumuzla **Args'in** ilişkisi keserek **Geldi** adında farklı bir metoda bağlıyoruz. Son olarak **Page.Load**'a atadığımız ve **e.UserToken** üzerinden alabileceğimiz ana **Socket** değişkenimizi de yakalayarak **ReceiveAsync** metodu ile veri alımını başlatıyoruz.

Geldi ve **Baglandi** metodları aslında Silverlight içerisinde ayrı bir Thread içerisinde çalışıyor. Bu nedenle tüm bu işlemler yapılrken kullanıcının uygulama ile olan interaktivitesi kesinlikle kesilmiyor. Tabi ayrı bir Thread gibi davranışını olmasının dezavantajı ise birazdan karşımıza çıkacak. Kısır döngü içerisinde sürekli **ReceiveAsync** ile sunucuyu dinlerken istemci tarafında görsel arayüzde değişiklik yapamayacağız. Bu da bizim sunucudan veri alabilmemizi fakat ekranda göstermememize neden olacak. Tabi demokrasilerde çare tükenmez...

```
Delegate Sub MyDelegate(ByVal myArg2 As String)
```

```
Sub GelGel(ByVal x As String)
```

```
    Metin.Text = x
```

```
End Sub
```

İlk olarak bir **Delegate** tanımlayacağımız, söz konusu delegemiz sadece bir parametre alacak. Ayrıca bir de Sub yaratıyoruz. Aynı şekilde Sub'da bir metin parametresi alıyor ve bizim uygulamamızda adı **Metin** olan **TextBlock** içerisinde yerleştiriyoruz. İşte bu yapı ile biraz önce bahsettiğimiz sorundan kurtuluyor olacağız.

```

Private Sub Geldi(ByVal sender As Object, ByVal e As
System.Net.Sockets.SocketAsyncEventArgs)
    Dim Gelen As String = System.Text.Encoding.UTF8.GetString(e.Buffer, e.Offset,
e.BytesTransferred)
    Me.Dispatcher.BeginInvoke(New MyDelegate(AddressOf GelGel), New String()
{Gelen})
    Dim Baglanti As System.Net.Sockets.Socket = CType(e.UserToken,
System.Net.Sockets.Socket)
    Baglanti.ReceiveAsync(e)
End Sub
```

Aslında sunucudan gelen veriyi almak çok kolay. Kodumuz içerisindeki ilk satır bu işi hallediyoruz. Esas mesele veriyi aldiktan sonra sahnedeki göstergemiz. Dispatcher nesnesi belki de Silverlight içerisinde en ilginç yapılardan biri; **Dispatcher** ile mevcut Thread'i yakalayarak **BeginInvoke** ile başka bir metod çalıştırıyoruz. Çalıştıracağımız metod sunucudan gelen veriyi parametre olarak vereceğiz ve söz konusu metod (GelGel) bu veriyi alarak sahnedeki **Metin** adındaki **TextBlock** içerisinde yerlestirecek. **BeginInvoke** ile ilgili işimizi de tamamladıktan sonra artık tekrar sunucunun dinlenmeye başlanması için elimizdeki Socket'i yakalayarak **ReceiveAsync** metodunu çalıştırıyoruz. Bu sistem böyle sonsuza tek dönecek ve sunucudan gelen veri sürekli olarak tüm istemcilerde anında gösterilecek.

Son olarak hem istemci hem de sunucu uygulamanın tam kodunu sizlerle paylaşmak istiyorum.

[İstemci: Silverlight uygulaması]

```
Partial Public Class Page
    Inherits UserControl
```

```

Public Sub New()
    InitializeComponent()
End Sub
```

```

Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Dim Hat As New System.Net.Sockets.Socket(Net.Sockets.AddressFamily.InterNetwork,
Net.Sockets.SocketType.Stream, Net.Sockets.ProtocolType.Tcp)
    Dim Args As New System.Net.Sockets.SocketAsyncEventArgs

    Args.UserToken = Hat
    Args.RemoteEndPoint = New System.Net.DnsEndPoint("localhost", 4530)
    AddHandler Args.Completed, AddressOf Baglandi
    Hat.ConnectAsync(Args)
End Sub
```

```

Private Sub Baglandi(ByVal sender As Object, ByVal e As
System.Net.Sockets.SocketAsyncEventArgs)
    Dim Gelen(1024) As Byte
    e.SetBuffer(Gelen, 0, Gelen.Length)
    RemoveHandler e.Completed, AddressOf Baglandi
    AddHandler e.Completed, AddressOf Geldi
    Dim Baglanti As System.Net.Sockets.Socket = CType(e.UserToken,
System.Net.Sockets.Socket)
    Baglanti.ReceiveAsync(e)
End Sub

Private Sub Geldi(ByVal sender As Object, ByVal e As
System.Net.Sockets.SocketAsyncEventArgs)
    Dim Gelen As String = System.Text.Encoding.UTF8.GetString(e.Buffer, e.Offset,
e.BytesTransferred)
    Me.Dispatcher.BeginInvoke(New MyDelegate(AddressOf GelGel), New String()
{Gelen})
    Dim Baglanti As System.Net.Sockets.Socket = CType(e.UserToken,
System.Net.Sockets.Socket)
    Baglanti.ReceiveAsync(e)
End Sub

Delegate Sub MyDelegate(ByVal myArg2 As String)

Sub GelGel(ByVal x As String)
    Metin.Text = x
End Sub

End Class

```

[Sunucu: Winforms Uygulaması]

```

Public Class Form1

Dim Baglilar As New System.Collections.Generic.List(Of System.IO.StreamWriter)
Dim yeniTR As System.Threading.Thread
Dim TCPBaglantilari As New System.Threading.ManualResetEvent(True)
Dim Dinleyici As New System.Net.Sockets.TcpListener(System.Net.IPAddress.Any, 4530)

Private Sub btn_Basla_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Basla.Click
    'İzin Verilen Port Aralığı 4502-4532
    yeniTR = New System.Threading.Thread(AddressOf Bekle)
    yeniTR.Start()
End Sub

Sub Bekle()
    Dinleyici.Start()
    While True
        TCPBaglantilari.Reset()

```

```

Dinleyici.BeginAcceptTcpClient(New System.AsyncCallback(AddressOf
BaglantiGeliyor), Nothing)
    TCPBaglantilari.WaitOne()
End While
End Sub

Private Sub BaglantiGeliyor(ByVal ar As System.IAsyncResult)
    TCPBaglantilari.Set()
    Dim Musteri As System.Net.Sockets.TcpClient = Dinleyici.EndAcceptTcpClient(ar)
    If Musteri.Connected Then
        Dim yazici As New System.IO.StreamWriter(Musteri.GetStream)
        yazici.AutoFlush = True
        Baglilar.Add(yazici)
        yazici.WriteLine("Bağlandınız.")
    End If
End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox1.TextChanged
    For Each x As System.IO.StreamWriter In Baglilar
        x.WriteLine(TextBox1.Text)
    Next
End Sub
End Class

```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 XAP Paketleri ve Kaynak Dosyalar

Silverlight 2.0 uygulamalarında tüm uygulamaya ait VB veya C# kodu ile beraber XAML dosyalarının da birer DLL olarak düzenlendiğini ve sonrasında XAP adında, özünde ZIP dosyaları şeklinde paketlenerek istemciye gönderildiğini biliyoruz. Durum böyle olunca bir Silverlight uygulaması ile beraber sunucudan istemciye farklı kaynaklar göndermeyle ilgili değişik yollar söz konusu oluyor.

Bunlardan en basiti tabi ki sunucuda yer alan hali hazırladıktan sonra dosyayı asenkron bir istek ile istemciye taşımak. Fakat bu durumda eğer istenecek olan veri çok ufaksa aslında ilk Silverlight uygulamasının yüklenmesinde kullanılan veri transferinde bu ufak dosyalar da ana XAP dosyası ile beraber gönderilebilirdi. Böylece hem toplamda sunucuya gönderilen istek sayısı azalırırdı hem de XAP dosyalarının yapısı gereği sıkıştırma özelliğinden faydalananmış olurdu.

Build Action : Resource

Visual Studio içerisinde Silverlight projelerinde herhangi bir dosyayı seçtikten sonra "Properties" paneline göz attığımızda "Build Action" adında bir ayar görebilirsiniz. Bu ayar ile söz konusu dosyanın ne şekilde sunucudan istemciye gönderileceğini ayarlamış oluyoruz.

Varsayılan ayarları ile projenize bir resim dosyası eklediğinizde Build Action ayarı **Resource** olarak düzenlenmiş olacaktır. Bu dosyalar doğrudan Silverlight uygulaması için yaratılacak DLL dosyası içerisinde Resource olarak yerleştirilecektir. DLL dosyasının yüklenme süresini uzatmamak adına olabildiğince ufak ve önemli dosyaları bu şekilde projelere eklemekte faydalıdır.

<Image Source="Foto.jpg"/>

Bu şekilde projelere eklenmiş dosyaları XAML içerisinde doğrudan yukarıdaki gibi kullanabilirsiniz.

Build Action : Content

Eğer dosyanızın orijinal DLL'i şıyrılmamasını istemiyorsanız fakat yine de aynı XAP dosyası içerisinde istemciye gitmesini istiyorsanız. Kullanmanız gereken seçenek **"Content"** seçeneği. Bu şekilde işaretlenmiş dosyalar XAP dosyası içerisinde konarak istemciye gönderilir. Eğer istemci tarafında Plug-In hedef dosyayı XAP dosyası içinde bulamazsa bu sefer XAP dosyası ile aynı klasörde sunucu üzerinden dosyayı almaya çalışır.

<Image Source="/Foto.jpg"/>

Normalinden farklı olarak bu sefer tüm verilen adreslerin başında bir / yerleştirilmesi ve relative konum verilmesi gerekiyor. Her ihtimale karşı yine de çok büyük dosyaları da bu şekilde kullanmamakta fayda var. Çünkü unutmayın XAP dosyası istemciye tamamen gitmeden uygulamanız çalışmamayacaktır.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 XAP Paketleri

Silverlight 2.0 (Beta 1) yazılarının yavaş yavaş geleceğinden bahsetmiştim. İşte ilki ile karşınızdayım. Bu yazıda **Silverlight 2.0 Beta 1** ile beraber gelen yeni dağıtım şeklärinden bahsedeceğiz. Silverlight 1.0 uygulamalarında sitenize herhangi bir Silverlight animasyonu yerleştirebilmeniz için epey emek harcayarak birden çok JavaScript dosyasını sayfanızı linklemeniz sonrasında uygun HTML elementlerini ayarlamanız hatta bir de yanında XAML dosyanızı koyma gerekiyordu. Bu durum Silverlight 2.0 ile beraber değişiyor ve karşımıza **XAP** uzantılı dosyalar geliyor. XAP'ın herhangi bir açılımı yok (en azından şimdilik).

XAP dosyaları aslında özünde birer ZIP dosyası. Bu dosyalar içerisinde Silverlight projenize ait DLL'ler bulunuyor. Silverlight 2.0 ile beraber VB veya C# kullanarak kodlama yapabildiğimiz için bu kodlardan DLL dosyaları yaratılıyor ve söz konusu DLL'ler XAP dosyası içerisinde istemciye gönderilerek istemci tarafında Silverlight Run-Time ile çalıştırılıyor. Expression Blend ile hazırladığınız **XAML** dosyaları da aynı DLL dosyaları içerisinde birer **Resource** olarak ekleniyor.

Şu an **ASP.NET 3.5 Extensions** paketi içerisinde yer alan ve ilerde ASP.NET'e eklenmesi düşünülen kontrollerden biri olan Silverlight kontrolü XAP dosyalarını alarak doğrudan ASP.NET sayfalarına yerleştirilebilecek. Örnek bir Silverlight kontrolünün kodunu aşağıda inceleyebilirsiniz.

```
<asp:Silverlight runat="server"
    PluginBackground="White"
    Source="animasyon.xap"
    Version="2.0">
    <PluginNotInstalledTemplate>
        Plug-In Yüklü Değil
    </PluginNotInstalledTemplate>
</asp:Silverlight>
```

Gördüğünüz gibi artık bir Silverlight 2.0 animasyonunu herhangi bir ASP.NET sitesine yerleştirmek çocuk oyunçagına dönüşmüştür. Tabii "artık" derken bu kullandıklarımızın hiçbirinin daha yayında olan yazılımlar olmadığını da akılda tutmakta fayda var. Fakat geleceğin böyle olduğunu görmek hoş.

Bir XAP dosyasında neler var?

İlk olarak tabi ki bizim yazdığımız uygulamanın kodlarını ve XAML kaynaklarını içeren DLL dosyamız var. Bu DLL dosyasını De-Compile ettiğimizde içerisinde **Resource** olarak XAML dosyalarımızın var olduğunu görebiliyoruz.

Name	Value
page.xaml	ef bb bf 3c 55 73 65 72 ... (403 bytes)
app.xaml	ef bb bf 3c 41 70 70 6c ... (181 bytes)

Silverlight 2.0 DLL dosyası içerisindeki XAML dosyalarımız.

Haricen Silverlight 2.0 projemizde kullandığımız kontrollere ait DLL dosyaları da yine XAP içerisinde bulunuyor. Silverlight 2.0 içerisinde standart kontroller bile ayrı DLL dosyaları olarak geliyor. Bunların zamanla Plug-In'e dahil edilip edilmeyeceği belli değil fakat Silverlight 2.0 mimarisinde harici DLL dosyalarından farklı kontroller kullanılabileceği için bu yapı her zaman var olacaktır.

Extracted files	
Name	Type
AppManifest	XAML File
SilverlightApplication3.dll	Application Extension
System.Windows.Controls.dll	Application Extension
System.Windows.Controls.Extended.dll	Application Extension

Silverlight 2.0 Beta 1 XAP dosyası içeriği.

XAP dosyası içerisinde bir de XAML dosyası bulunuyor. Aşağıda içeriğini inceleyebileceğiniz örnek bir **AppManifest.xaml** dosyasına baktığımızda XAP paketi içerisinde tüm DLL'lerin sınıf isimleri ile ilişkilendirildiklerini görüyoruz.

```
<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  EntryPointAssembly="SilverlightApplication3"
  EntryPointType="SilverlightApplication3.App" RuntimeVersion="2.0.30226.2">
  <Deployment.Parts>
    <AssemblyPart x:Name="SilverlightApplication3" Source="SilverlightApplication3.dll" />
    <AssemblyPart x:Name="System.Windows.Controls"
      Source="System.Windows.Controls.dll" />
    <AssemblyPart x:Name="System.Windows.Controls.Extended"
      Source="System.Windows.Controls.Extended.dll" />
  </Deployment.Parts>
</Deployment>
```

Deployment tagı içerisindeki **EntryPointAssembly** uygulama ilk çalıştırıldığında hangi DLL'in başlatılacağını belirliyor.

Bu haliyle yeni Silverlight 2.0 paketlerine baktığımızda olayın epey derlenip toparlandığını görüyoruz. Özellikle Silverlight 1.0 ile ilgili sıkça akla gelen "kod güvenliği" sorunları ile ilgili kısmı bir ilerleme var diyebiliriz. En azından artık XAP (ZIP) dosyasını açmanız DLL'i

çıkarmanız ve De-Compile etmeniz gerekiyor. Tabi ki tüm bunlar mümkün :) fakat biraz daha zağmetli hale geldi diyebiliriz. Ayrıca artık Silverlight uygulamalarının dağıtımını ve paylaşımı çok daha kolay. Özellikle bu tip animasyonları içerik yönetim sistemlerinde kullanmak isteyenler için SL 2.0 büyük kolaylık olacaktır.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0'da farenin hareket etmediğini anlamak.

Bu yazımızda **Silverlight 2.0 RC0** ile Silverlight uygulaması içerisinde minik bir ekran koruyucusu yapacağız. Aslında amacımız ekran korumak değil tabii ki, istediğimiz şey kullanıcı Silverlight uygulamasını kullanmadığında farklı bir içerik göstermek. Bu belki bir reklam, belki farklı bir "Ekrana Geri Dön Kullanıcı" sesli mesajı veya çok daha farklı bir uyarı sistemi olabilir. Özellikle veritabanı üzerinden veri alarak bu veriyi düzenleyen bir Silverlight uygulamasını düşünürsek belki de kullanıcı uzun süre ekran başında değilse verileri sunucuya göndererek kaydetmek için doğru zamanı yakalamış demektir. Bu sistemin kullanılabileceği diğer örnekleri sizin hayal gücünze bırakıyorum.

Animasyonlarımızı hazırlayalım

Örneğimizde görsel olarak herhangi bir ek öğe yer almayacak. Kullanıcı fareyi hareket ettirmezse bir süre sonra uygulamanın ana Grid'inin rengini siyaha çevireceğiz. Kullanıcı fare ile herhangi bir hareket yaptığı anda ise tekrar söz konusu Grid'i beyaza çevireceğiz. Siz örneklerinizde çok daha farklı işlemler yapabilirsiniz. Şimdi gelin bu iki animasyonla beraber oluşan XAML kodumuza göz atalım.

```
<UserControl x:Class="SilverlightApplication2.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
<UserControl.Resources>
    <Storyboard x:Name="Gitti">
        <ColorAnimationUsingKeyFrames BeginTime="00:00:00"
            Storyboard.TargetName="LayoutRoot"

Storyboard.TargetProperty="(Panel.Background).(SolidColorBrush.Color)">
        <SplineColorKeyFrame KeyTime="00:00:01"
            Value="#FF000000"/>
    </ColorAnimationUsingKeyFrames>
</Storyboard>
    <Storyboard x:Name="Geldi">
        <ColorAnimationUsingKeyFrames BeginTime="00:00:00"
            Storyboard.TargetName="LayoutRoot"

Storyboard.TargetProperty="(Panel.Background).(SolidColorBrush.Color)">
        <SplineColorKeyFrame KeyTime="00:00:01"
            Value="#FFFFFFF"/>
    </ColorAnimationUsingKeyFrames>
</Storyboard>
</UserControl.Resources>
<Grid x:Name="LayoutRoot"
    Background="White">
</Grid>
</UserControl>
```

Gördüğünüz gibi kodumuzda yer alan iki animasyon da **LayoutRoot** adındaki Grid'in rengini bir saniyede değiştirmeyi hedefliyor. Animasyonlardan **Geldi** adındaki Storyboard **Grid'in** rengini beyaza alırken, **Gitti** adındaki ise Siyah'a götürüyor. Kullanıcı fareyi hareket ettirmediginde yani ekran başından gittiğinde **Gitti**, geri geldiğinde ise **Geldi** animasyonunu oynatacağız.

DispatcherTimer yetiş imdadımıza!

Farenin kullanıcı tarafından hareket ettirilip ettirilmediği sürekli kontrol etmek yerine aslında bizim ana bir Timer'a ihtiyacımız var. Varsayılm ki beş saniye boyunca herhangi bir hareket olmamışsa ekranı karartacağız. Bu durumda farenin oynatıldığı son harekette bu beş saniyelik sayacı başlatmamız gereklidir. Biz hangi fare hareketinin son hareket olduğunu anlayamayacağımız için aslında farenin her hareketinde sayacımızı başlatmalıyız fakat sonrasında eğer fare tekrar hareket ettirilirse sayacı durdurup baştan başlatmamız gereklidir.

İlk olarak Timer olarak kullanacağımız DispatcherTimer'i global olarak tanımlayalım.

[VB]

```
WithEvents Kontrol As New System.Windows.Threading.DispatcherTimer
```

[C#]

```
System.Windows.Threading.DispatcherTimer Kontrol = new  
System.Windows.Threading.DispatcherTimer();
```

Uygulama ilk olarak istemciye yüklenliğinde hemen elimizdeki Timer'ın Interval yani tekrar aralığını düzenleyerek Timer'i başlatmamız gerekiyor.

[VB]

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As  
System.Windows.RoutedEventArgs) Handles Me.Loaded  
    Kontrol.Interval = New TimeSpan(0, 0, 5)  
    Kontrol.Start()  
End Sub
```

[C#]

```
void Page_Loaded(object sender, RoutedEventArgs e)  
{  
    Kontrol.Interval = new TimeSpan(0, 0, 5);  
    Kontrol.Start();  
}
```

Eğer fare hareket etmezse yukarıdaki Timer sonuna kadar çalışacak ve doğal olarak Timer'in Tick event'i çalıştırılacaktır. Fakat bu süreçte eğer fare oynatılırsa bizim bu Timer'i durdurup baştan başlatmamız gereklidir ki sayacımızı da bir anlamda sıfırlamış olalımy. Uygulamamızın MouseMove durumunu yakalamamız yeterli olacaktır.

[VB]

```
Private Sub Page_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Me.MouseMove
    Kontrol.Stop()
    Kontrol.Start()
End Sub
```

[C#]

```
void Page_MouseMove(object sender, MouseEventArgs e)
{
    Kontrol.Stop();
    Kontrol.Start();
}
```

Sıra geldi Timer'in Tick eventine gerekli kodu yazmaya. Aslında bu noktada yazacağımız tek kod bizim Gitti animasyonunu çalıştıracak olan kod olacak. Böylece uygulama kullanıcının fareyi beş saniyedir oynatmadığını algılamış ve gerekli işlemi yapmış olacak.

[VB]

```
Private Sub Kontrol_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Kontrol.Tick
    Gitti.Begin()
    Gitmis = True
End Sub
```

[C#]

```
void Kontrol_Tick(object sender, EventArgs e)
{
    Gitti.Begin();
    Gitmis = true;
}
```

Kod içerisinde ilginizi çeken nokta eminim ki Gitmis değişkenidir. Gitmis değişkeni uygulamamızda global olarak tanımlayacağımız bir Boolean değişkeni. Bu değişkeni ScreenSaver yapımızın çalışıp çalışmadığını kontrol etmek için kullanacağız. Böylece MouseMove durumunda daha önce eğer ekran karartılmış ise ekranı aydınlatabileceğiz. Gelin MouseMove durumundaki yeni kodumuzu da inceleyelim.

[VB]

```
Private Sub Page_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Me.MouseMove
    Kontrol.Stop()
    Kontrol.Start()
    If Gitmis Then
        Gitmis = False
    End If
End Sub
```

```
Geldi.Begin()
End If
End Sub
```

[C#]

```
void Page_MouseMove(object sender, MouseEventArgs e)
{
    Kontrol.Stop();
    Kontrol.Start();
    if(Gitmis)
    {
        Gitmis = false;
        Geldi.Begin();
    }
}
```

Gördüğünüz gibi fare her oynatıldığında sayacımızı sıfırlarken daha önce ekran karartılmış mı kontrol ediyoruz. Eğer karartılmış ise hemen aydınlatma işlemini başlatıyoruz ve tabi ki **Gitmis** değişkenimizi de **False** olarak ayarlıyoruz.

Uygulamamızın tam kodu aşağıdaki şekilde sonlanıyor;

[VB]

```
Partial Public Class Page
    Inherits UserControl
```

```
Public Sub New()
    InitializeComponent()
End Sub
```

```
WithEvents Kontrol As New System.Windows.Threading.DispatcherTimer
Dim Gitmis As Boolean = False
```

```
Private Sub Page_Loaded(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
    Kontrol.Interval = New TimeSpan(0, 0, 5)
    Kontrol.Start()
End Sub
```

```
Private Sub Page_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Me.MouseMove
    Kontrol.Stop()
    Kontrol.Start()
    If Gitmis Then
        Gitmis = False
        Geldi.Begin()
    End If
End Sub
```

```

Private Sub Kontrol_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Kontrol.Tick
    Gitti.Begin()
    Gitmis = True
End Sub
End Class

```

[C#]

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace SilverlightApplication3
{
    public partial class Page : UserControl
    {
        System.Windows.Threading.DispatcherTimer Kontrol = new
        System.Windows.Threading.DispatcherTimer();
        bool Gitmis;

        public Page()
        {
            InitializeComponent();
            this.Loaded += new RoutedEventHandler(Page_Loaded);
            this.MouseMove += new MouseEventHandler(Page_MouseMove);
            this.Kontrol.Tick += new EventHandler(Kontrol_Tick);
        }

        void Kontrol_Tick(object sender, EventArgs e)
        {
            Gitti.Begin();
            Gitmis = true;
        }

        void Page_MouseMove(object sender, MouseEventArgs e)
        {
            Kontrol.Stop();
            Kontrol.Start();
            if (Gitmis)
            {

```

```
    Gitmis = false;
    Geldi.Begin();
}

void Page_Loaded(object sender, RoutedEventArgs e)
{
    Kontrol.Interval = new TimeSpan(0, 0, 5);
    Kontrol.Start();
}

}
```

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight içerisinde ClipBoard kullanımı

Silverlight uygulamaları içerisinde "Clipboard"a ulaşmak istediğinizde maalesef hazır bir altyapı ile en azından şimdilik **Silverlight 2.0 Beta 2** içerisinde karşılaşmıyoruz. Aynı şekilde Silverlight 1.0 içerisinde de bu sorun için bir çözüm yok. Fakat özellikle Silverlight 1.0 tarafından zaten JavaScript'in ana programlama yapısı olduğunu düşünürsek "*Acaba tarayıcı içerisinde JavaScript ile bir çözüm oluşturabilir miyiz?*" sorusu akla geliyor. Bu sorunun cevabı en azından Internet Explorer için "Evet". FireFox varsayılan ayarları ile bu gibi işlemlere JavaScript tarafından olanak tanımıyor.

Silverlight 1.0 ile Clipboard kullanımı

Yeni bir Silverlight 1.0 projesi yaratarak içerisinde bir **TextBlock** ve **Rectangle** yerleştirelim. Yapacağım işlem TextBlock içerisinde yazılı metni Rectangle'a basıldığından ClipBoard'a taşımak olacak.

```
<Canvas
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="640" Height="480"
    Background="White"
    x:Name="Page">
    <TextBlock Width="274.242" Height="43.939" Canvas.Left="31.818"
    Canvas.Top="27.273" Text="Kopyalanacak Metin" TextWrapping="Wrap"
    x:Name="Etiket"/>
    <Rectangle MouseLeftButtonDown="Kopyala" Width="122.727" Height="43.939"
    Fill="#FFFF0000" Stroke="#FF000000" Canvas.Left="31.818" Canvas.Top="92.425"
    RadiusY="16.167" RadiusX="16.167" x:Name="Dugme"/>
</Canvas>
```

Yukarıdaki kod uygulamamızın görsel arayüzüne oluşturuyor. "**Dugme**" adındaki Rectangle nesnemizin **MouseLeftButtonDown** durumunda çalıştırılacak olan kodu birazdan yazacağız.

```
function Kopyala(sender)
{
    window.clipboardData.setData("text", sender.findName("Etiket").Text);
}
```

Kodumuz içerisinde kullandığımız clipboardData sınıfı ile ilgili detaylara [MSDN](#) üzerinden ulaşabilirsiniz. setData metodu toplamda iki parametre alıyor; bunlardan ilki ClipBoard'a kopyalanacak olan verinin tipi, ikincisi ise kopyalanacak olan içeriğin ta kendisi. Aynı şekilde isterseniz ClipBoard'dan veri almak için **getData** metodunu da kullanabilirsiniz.

```
function Getir(sender)
{
    sender.findName("Etiket").Text = window.clipboardData.getData("text");
}
```

getData metodu sadece ClipBoard'dan alacağı verinin tipini parametre olarak alarak geriye doğrudan elde ettiği veriyi döndürüyor.

Peki ya Silverlight 2.0 tarafından neler yapacağız?

Aşında çok farklı bir işlem yapmayacağız. Silverlight 2.0 Beta 2 tarafından da şimdilik JavaScript'in nimetlerinden faydalananmak zorundayız. O nedenle istemci tarafındaki VB veya C# kodumuz ile sayfanın JavaScript tarafındaki özelliklerine ulaşıp yine JavaScript tarafındaki metodlarını çalıştıracağız.

İlk olarak Silverlight 2.0 Beta 2 uygulamamızın arayüzüni aşağıdaki şekilde düzenleyelim.

```
<UserControl x:Class="SilverlightApplication14.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox Height="40" HorizontalAlignment="Left" Margin="16,8,0,0"
            VerticalAlignment="Top" Width="120" Text="Herhangi bir Metin" TextWrapping="Wrap"
            x:Name="txtMetinKutusu"/>
        <Button Height="24" HorizontalAlignment="Left" Margin="16,72,0,0"
            VerticalAlignment="Top" Width="88" Content="Kopyala" x:Name="btnKopyala"/>
        <Button Height="24" HorizontalAlignment="Left" Margin="16,112,0,0"
            VerticalAlignment="Top" Width="88" Content="Yapıştır" x:Name="btnYapistir"/>
    </Grid>
</UserControl>
```

Bir TextBox ve iki Button'dan oluşan uygulamamızın ilk olarak kopyalama işlemini yapacak olan kodunu yazalım.

[VB]

```
Private Sub btnKopyala_Click(ByVal sender As Object, ByVal e As
System.Windows.RoutedEventArgs) Handles btnKopyala.Click
    Dim Clipboard As Browser.ScriptObject =
Browser.HtmlPage.Window.GetProperty("clipboardData")
    Clipboard.Invoke("setData", "text", txtMetinKutusu.Text)
End Sub
```

[C#]

```
void btnKopyala_Click(object sender, RoutedEventArgs e)
{
    System.Windows.Browser.ScriptObject Clipboard =
(System.Windows.Browser.ScriptObject)System.Windows.Browser.HtmlPage.Window.GetProperty("clipboardData");
    Clipboard.Invoke("setData", "text", txtMetinKutusu.Text.ToString());
}
```

Kodumuz içerisinde ilk olarak tarayıcının **clipboardData** sınıfını yakalamamız gerekiyor. Bunun için içerisinde olduğumuz tarayıcının (Browser) yine mevcut HTML sayfasının (HtmlPage) ait olduğu pencereyi (Window) yakalayıp onun üzerinden **GetProperty** ile **clipboardData**'yı alarak **ScriptObject** tipinde yarattığımız Clipboard değişkenimize

aktarıyoruz. Sonrasında doğrudan değişkenimiz üzerinden **Invoke** diyerek aynı **Reflection** yapar gibi **setData** metodunu çalıştırıyoruz. Normal şartlarda JavaScript tarafına vereceğimiz parametreleri de yine **Invoke** metoduna aktarıyoruz. Böylece kopyalama işlemini tamamlamış olduk. Aynı şekilde ClipBoard'dan veri almayı da hemen **getData** ile yapabiliriz.

[VB]

```
Private Sub btnYapistir_Click(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs) Handles btnYapistir.Click
    Dim Clipboard As Browser.ScriptObject =
    Browser.HtmlPage.Window.GetProperty("clipboardData")
    txtMetinKutusu.Text = Clipboard.Invoke("getData", "text")
End Sub
```

[C#]

```
void btnYapistir_Click(object sender, RoutedEventArgs e)
{
    System.Windows.Browser.ScriptObject Clipboard =
    (System.Windows.Browser.ScriptObject)System.Windows.Browser.HtmlPage.Window.GetProperty("clipboardData");
    txtMetinKutusu.Text = Clipboard.Invoke("getData", "text").ToString();
}
```

Clipboard'dan veri alırken de aynı şekilde **clipboardData** nesnemizi yakaladıktan sonra bu sefer **getData** metodunu çalıştırıyoruz ve geriye dönen değeri de örneğimizde **TextBox** içerisine yazdırıyoruz.

C# kullananların haricen aşağıdaki şekilde uygulama başlangıcında Event-Handlar bağlantılarını yapmaları gerekecektir. VB kodlarındaki Handles keyword'ü ile bu işlem satır içinde yapılabildiği için ek olarak yazmak gerekmeyor.

[C#]

```
public Page()
{
    InitializeComponent();
    btnKopyala.Click += new RoutedEventHandler(btnKopyala_Click);
    btnYapistir.Click += new RoutedEventHandler(btnYapistir_Click);
}
```

Böylece hem Silverlight 1.0 hem 2.0 içerisinde Clipboard'dan veri alarak veri aktarımı yapabildik. Tabi tüm bu işlemlerin sadece Internet Explorer içerisinde olması üzücü. Diğer tarayıcılar için de geçerli olacak şekilde umarız ilerde Silverlight Runtime içerisinde standart işlevsellikler eklenir.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight içerisinde sayfa adresine ulaşmak

Bugünlerde bana sıkça gelen sorulardan biri Silverlight tarafından uygulamanın çalıştığı adresin nasıl alınacağı ile ilgili oluyor. Aslında basit bir şekilde Silverlight'tan DOM'a çıkışınız bunun için yeterli olacaktır. Peki bunu nasıl yaparız?

Silverlight içerisinde DOM'a çıkmak için yolculüğünuzun başlayacağını namespace System.Windows.Browser namespace'si olacaktır. Bu NameSpace içerisinde ister sayfadaki JavaScript metodlarına ulaşır ister sayfadaki HTML elementlerine ulaşabilirsiniz.

[C#]

```
MessageBox.Show(System.Windows.Browser.HtmlPage.Document.DocumentUri.AbsoluteUri.ToString());
```

Yukarıda gördüğünüz kod Silverlight uygulamanızın çalıştığı sayfanın tam yolunu verecektir. **DocumentUri** üzerinden giderek bu adrese ait farklı bilgileri de alabilirsiniz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight Runtime ve SDK DLL'leri ve açıklamaları

Silverlight 2.0 ile beraber artık tüm yazdığımız kodların DLL olarak XAP dosyaları içerisinde istemciye gönderildiği bir ortamda harici DLL kütüphaneleri de kullanabiliyor olmak büyük avantajlar getiriyor. Tabi ki bu avantajlardan ürünün kendisi de faydalıyor ve **Silverlight 2.0 Beta 2 Runtime ve SDK** ile beraber gelen çoğu harici kontrol normal şartlarda Expression Blend 2.5 içerisinde gözükmese de aslında DLL'ler şeklinde bilgisayarlarımıza bulunuyor. Bu yazımızda Silverlight 2.0 Runtime'in ve SDK'nın derinliklerine inip bahsettiğimiz DLL'leri inceleyerek neler yapabileceğimize göz atacağız.

Silverlight Runtime yüklemesinde bilgisayarınızda *C:\Program Files\Microsoft Silverlight\2.0.30523.6* konumuna yerleştirilecek olan DLL'leri söz konusu klasörü açarak bir inceleyelim.

Name	Date modified	Type	Size
coreclr.dll	04.06.2008 13:53	Application Extens...	2.972 KB
dbgshim.dll	04.06.2008 15:04	Application Extens...	156 KB
Microsoft.VisualBasic.dll	04.06.2008 13:53	Application Extens...	230 KB
mscordaccore.dll	04.06.2008 15:04	Application Extens...	759 KB
mscordbi.dll	04.06.2008 15:04	Application Extens...	809 KB
mscorlib.dll	04.06.2008 13:53	Application Extens...	1.429 KB
mscorrc.debug.dll	04.06.2008 15:04	Application Extens...	323 KB
mscorrc.dll	04.06.2008 13:53	Application Extens...	14 KB
npctrl.dll	04.06.2008 13:53	Application Extens...	747 KB
npctrlui.dll	04.06.2008 13:53	Application Extens...	733 KB
Silverlight.ConfigurationUI.dll	04.06.2008 13:53	Application Extens...	536 KB
sos.dll	04.06.2008 15:04	Application Extens...	461 KB
System.Core.dll	04.06.2008 13:53	Application Extens...	286 KB
system.dll	04.06.2008 13:53	Application Extens...	237 KB
System.Net.dll	04.06.2008 13:53	Application Extens...	165 KB
System.Runtime.Serialization.dll	04.06.2008 13:53	Application Extens...	278 KB
System.ServiceModel.dll	04.06.2008 13:53	Application Extens...	310 KB
System.ServiceModel.Web.dll	04.06.2008 13:53	Application Extens...	74 KB
System.Windows.Browser.dll	04.06.2008 13:53	Application Extens...	134 KB
System.Windows.dll	04.06.2008 13:53	Application Extens...	710 KB
System.Xml.dll	04.06.2008 13:53	Application Extens...	313 KB

Silverlight 2 Beta 2 Runtime DLL'leri.

Yukarıdaki DLL'lere baktığımızda özellikle eski VB sınıflarını desteklemek adına **Microsoft.VisualBasic.dll**'in Silverlight RunTime ile beraber de dağıtıldığını görüyoruz. Buradaki DLL'lerden bazlarına özellikle eğilmek gerekirse dikkat çekenler arasında **System.XML.dll** yer alıyor. Windows'takine kıyasla bu DLL içerisinde artık XSL veya XPath sınıfları bulunmuyor, tabi ki bunun çok mantıklı bir nedeni var; özellikle XPATH yerine artık elimizde XLINQ olduğu için bu sınıfları muhafaza etmek saçma olurdu. **System.Core.dll** ve **System.Xml.Linq.dll** işte tam bu noktada yardımımıza koşuyor ve LINQ'in neredeyse tüm özelliklerini Silverlight tarafına taşıyor. Son olarak **System.Windows.Browser.dll** ise bir Silverlight uygulamasının içerisinde bulunduğu tarayıcı ile ilgili işlemler yapabilmesi için gerekli sınıfları içeriyor.

Şimdi de gelin SDK ile beraber gelen DLL'lere göz atalım; bunun için bilgisayarınızda **C:\Program Files\Microsoft SDKs\Silverlight\v2.0\Libraries\Client** klasörüne ufak bir yolculuk yapmanız yeterli olacaktır.

Name	Size	Date modified	Type
IronPython.dll	817 KB	04.06.2008 15:09	Application Extens...
IronPython.Modules.dll	218 KB	04.06.2008 15:09	Application Extens...
IronRuby.dll	417 KB	04.06.2008 15:09	Application Extens...
IronRuby.Libraries.dll	302 KB	04.06.2008 15:09	Application Extens...
Microsoft.JScript.Compiler.dll	82 KB	04.06.2008 15:09	Application Extens...
Microsoft.JScript.Runtime.dll	310 KB	04.06.2008 15:09	Application Extens...
Microsoft.Scripting.Core.dll	646 KB	04.06.2008 15:09	Application Extens...
Microsoft.Scripting.dll	86 KB	04.06.2008 15:09	Application Extens...
Microsoft.Scripting.Silverlight.dll	46 KB	04.06.2008 15:09	Application Extens...
System.Data.Services.Client.dll	314 KB	04.06.2008 15:09	Application Extens...
System.Json.dll	50 KB	04.06.2008 15:09	Application Extens...
System.ServiceModel.PollingDuplex.dll	82 KB	04.06.2008 15:09	Application Extens...
System.ServiceModel.Syndication.dll	114 KB	04.06.2008 15:09	Application Extens...
System.Windows.Controls.Data.Design.dll	17 KB	04.06.2008 15:09	Application Extens...
System.Windows.Controls.Data.dll	290 KB	04.06.2008 15:09	Application Extens...
System.Windows.Controls.Extended.Design.dll	20 KB	04.06.2008 15:09	Application Extens...
System.Windows.Controls.Extended.dll	206 KB	04.06.2008 15:09	Application Extens...
System.Windows.Design.dll	23 KB	04.06.2008 15:09	Application Extens...
System.Xml.Linq.dll	122 KB	04.06.2008 15:09	Application Extens...
System.Xml.Serialization.dll	318 KB	04.06.2008 15:09	Application Extens...
System.Xml.Utils.dll	110 KB	04.06.2008 15:09	Application Extens...

Silverlight 2 SDK paketindeki DLL Kütüphaneleri

SDK içerisinde DLL'leri RunTime içerisindekilerden daha heyecan verici. İlk olarak **IronPython** ve **IronRuby** programlama altyapısını sunan DLL'lerini burada bulabiliyoruz. Tüm bu DLL'ler arasında bize DataGrid gibi farklı kontroller sunan DLL'ler ise **System.Windows.Controls** sınıfındaki dosyalar. **System.Windows.Controls.dll** içerisinde standart Silverlight kontrolleri, **System.Windows.Controls.Extended.dll** içerisinde Calendar, DatePicker, Slider, WatermarkedTextBox gibi kontroller, **System.Windows.Controls.Data.dll** içerisinde ise daha WPF'de bile olmayan DataGrid bulunuyor.

Tüm bu DLL kütüphaneleri Silverlight uygulamalarınızın çalışması için hayatı önem taşırken özellikle ek kontrolleri kullanabilmek adına gerekli DLL'leri projenize referans olarak da eklemeniz gerekecektir.

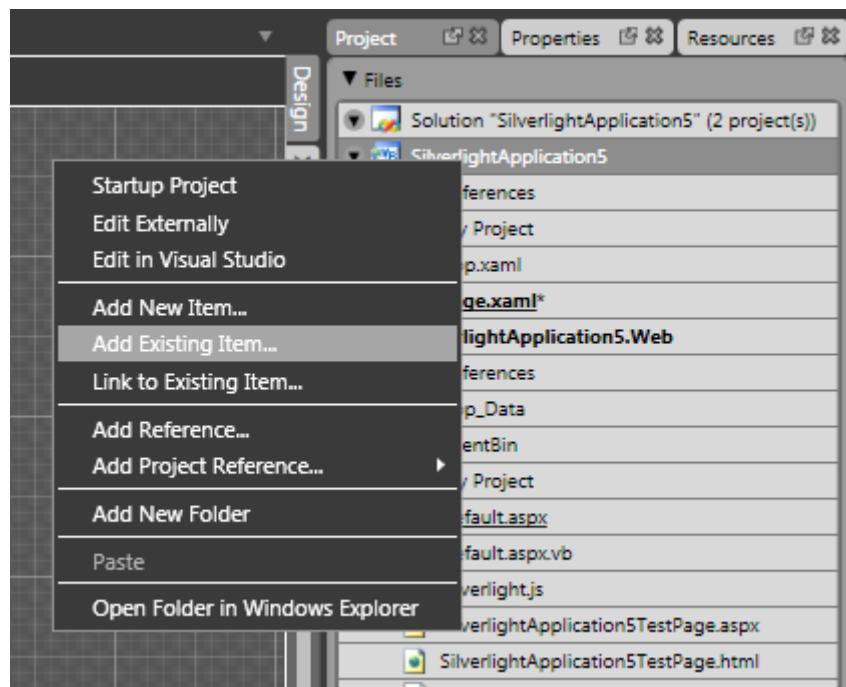
Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight Toolkit içerisinde gelen hazır renk şablonları (Thema) inceliyoruz.

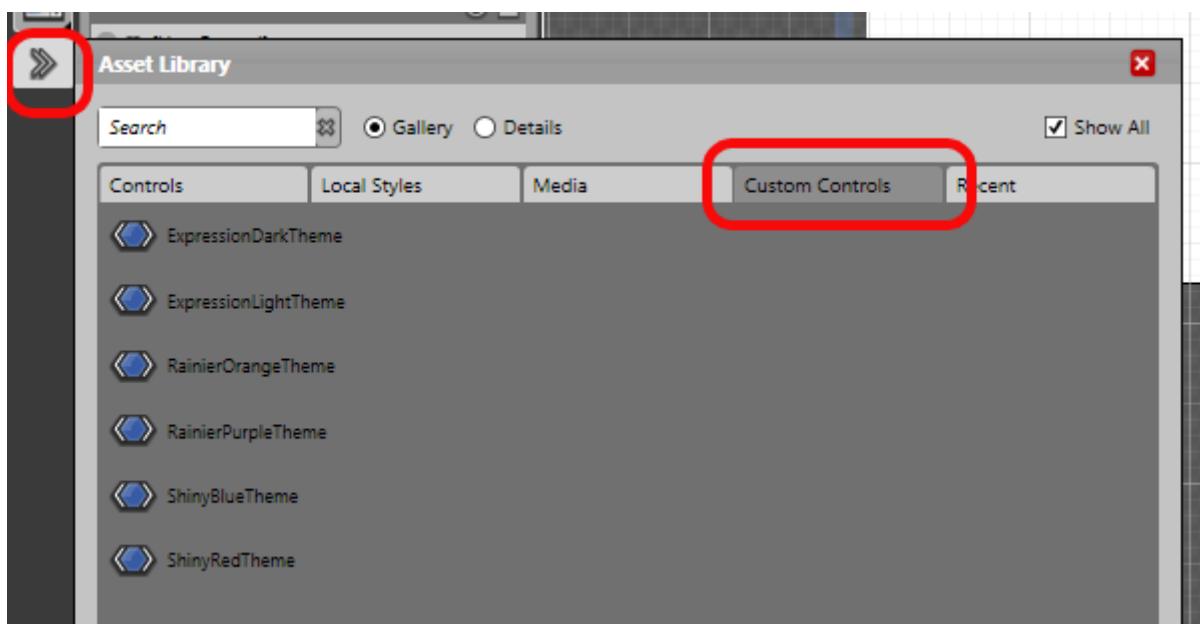
Silverlight Toolkit konusunda yazılarımı devam edeceğimden bahsetmiştim. Bu yazımızda Silverlight Toolkit ile beraber gelen theming (renk şablonları) yapısına göz atacağız. Hali hazırda Toolkit ile beraber gelen şablonların nasıl kullanılabildiğini inceleyeceğiz.

Yarattığımız yeni bir Silverlight 2.0 projesini Blend 2 ile açtıktan sonra ilk aşamada hemen Silverlight Toolkit içerisindeki gerekli DLL'leri referans almamız gerekiyor. Toolkit paketini bilgisayarınızda açtığınızda içinde Themes adında bir klasör olduğunu göreceksiniz. Bu klasör içerisinde her bir DLL farklı bir renk şablonunu temsil ediyor. İsterseniz hepsini referans olarak ekleyebilir veya sadece kullanacaklarınızı projenize dahil edebilirsiniz. Örneğimizde ben hepsini referans olarak alacağım ki aradaki farkları görelim.



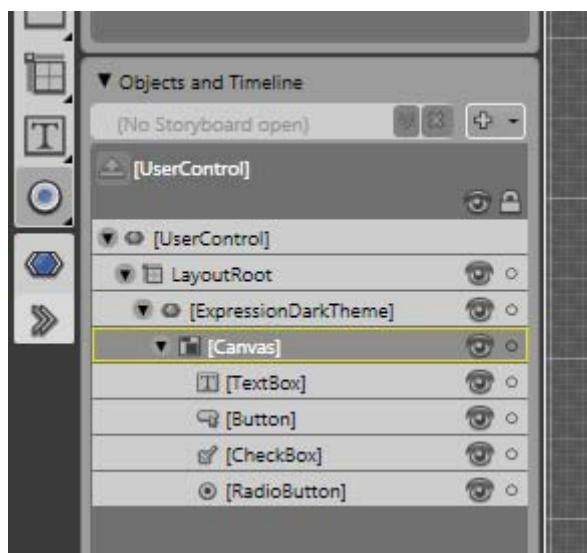
Blend 2 içerisinde projeye referans eklerken.

Themes klasöründeki tüm DLL'leri referans olarak aldıktan sonra bir de Toolkit paketinin ana klasöründeki **Microsoft.Windows.Controls.Theming.dll** dosyasını referans olarak almalısınız.



Tüm hazır Theme'ler kontroller şeklinde Asset Library'de.

Tüm DLL'leri doğru şekilde projenize referans olarak eklediğinizde Blend 2 içerisinde Asset Library'nin Custom Controls bölümünde her bir Thema için ayrı bir kontrol göreceksiniz. Bu kontrollerin kullanımı biraz garip :) Herhangi bir thema'dan etkilenmesini istediğiniz tüm kontrolleri bu yukarıdaki thema kontrolleri içerisinde koymamız gerekiyor. Yani aslında bir anlamda kendileri birer LayoutControl gibi davranışlıyorlar.

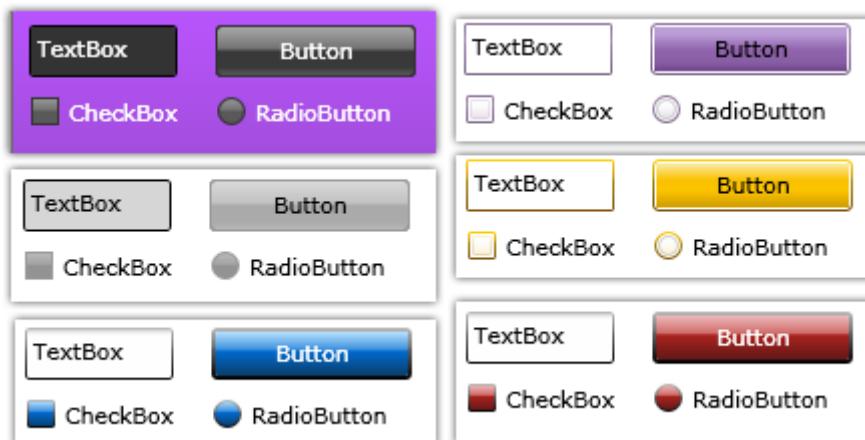


Blend içerisinde Theme kontrollerinin yapısı.

Yukarıdaki ekran görüntüsünde gördüğünüz üzere **ExpressionDarkTheme**'den etkilenmesini istediğimiz tüm kontrolleri bir Canvas içerisinde kendisine teslim etmişiz. Theme kontrolleri aslında içlerine sadece bir kontrol alabiliyorlar o nedenle Canvas gibi ayrı bir LayoutControl daha kullanmamız gerekiyor.

Bu yapı sayesinde bir uygulamada birden çok şablonu farklı uygulama bölgelerinde kullanabilirsiniz. Örneğin bir Grid'in iki kolonu içerisinde farklı Theme kontrolleri koyarak bu

kontrollerin içerisinde söz konusu şablonlardan faydalananabilir ve bu şekilde farklı kombinasyonlar ile de çalışabilirsiniz.



Toolkit içerisinde hazır renk şablonları.

Yukarıdaki şablonlar doğrudan Toolkit içerisinde hazır olarak gelen şablonların örnekleri. Tasarımcıların ne kadar hoşuna gider bilemiyorum :) ama yazılımcıların çok işine yarayacağından eminim. İsteyenler Toolkit paketi içerisinde Themes/XAML klasöründe bu tasarımların XAML kodlarını da bulabilirler.

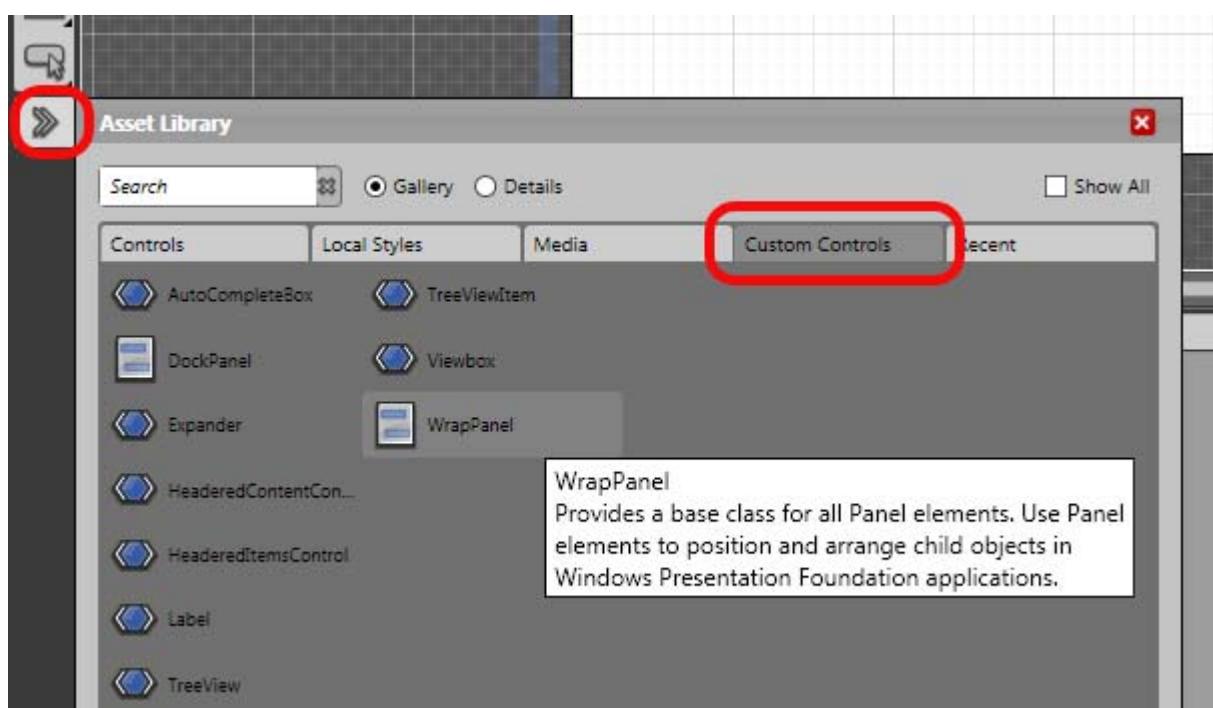
Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight Toolkit'ten WrapPanel'in kullanımı.

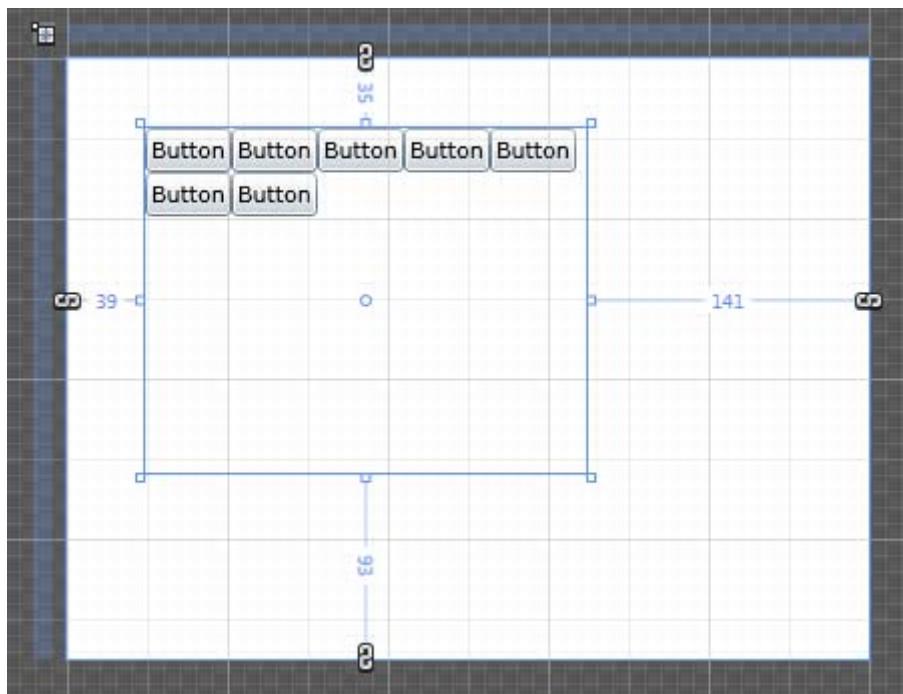
Layout kontrolleri XAML ile arayüzler oluştururken "olmazsa olmazlar" listemizin en başında geliyor. Silverlight 1.0'daki Canvas kontrolünden sonra 2.0'da birçok kontrol yardımımıza yetişse de maalesef WPF'deki zenginliğe ulaşamamıştı. Bu eksikliği Silverlight Toolkit karşılıyor ve paket içerisindeki WrapPanel aynı WPF içerisindeki işlevsellikleri Silverlight için de sağlıyor.

Silverlight Toolkit DLL'lerinden **Microsoft.Windows.Controls.DLL** dosyasını Silverlight 2 projenize referans olarak ekledikten sonra WrapPanel kontrolünü Blend 2 içerisinde Asset Library'de "Custom Controls" tabında bulabilirsiniz.



Expression Blend 2 içerisinde Silverlight Toolkit'ten WrapPanel.

Bu noktadan sonra Blend içerisinde doğrudan bir WrapPanel'i sahneye yerleştirebilirsiniz. Peki nedir WrapPanel? WrapPanel içerisindeki kontrolleri bir sepete atılmış gibi sürekli toplayarak kendi içine sığdırırmaya çalışan bir kontroldür. Gelin örnekler ile tam olarak nasıl çalıştığını göz atalım.



WrapPanel içerisinde birden çok nesne.

Yukarıdaki görselde sahnede bir WrapPanel yer alıyor. Söz konusu WrapPanel içinde toplam 6 tane Button var. Bu Button'ların konumları ile ilgili hiçbir ayar yapmadık. Düğmeleri WrapPanel içerisinde attıkça WrapPanel kendisi ilk başta düğmeleri yan yana koymaya başladı sonra yer kalmayınca alt satırda attı ve alt satırda düğmeler dizmeye başladı. İşte WrapPanel'in mantığı da budur. İçerisinde bulunan nesneleri sırası ile toparlayarak konumlandırır.

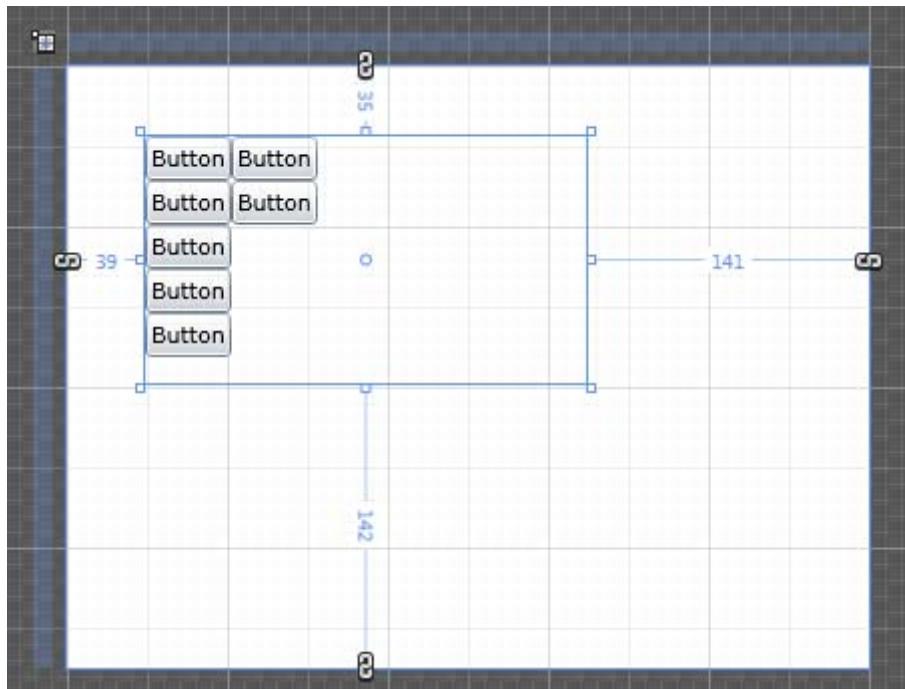
[XAML]

```
<UserControl x:Class="SilverlightApplication23.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:controls="clr-
    namespace:Microsoft.Windows.Controls;assembly=Microsoft.Windows.Controls">
    <Grid x:Name="LayoutRoot" Background="White">

        <controls:WrapPanel Margin="39,35,141,93">
            <Button Content="Button"/>
            <Button Content="Button"/>
            <Button Content="Button"/>
            <Button Content="Button"/>
            <Button Content="Button"/>
            <Button Content="Button"/>
            <Button Content="Button"/>
        </controls:WrapPanel>

    </Grid>
</UserControl>
```

Örneğimizin XAML kodunu da yukarıda bulabilirsiniz. Gördüğünüz gibi WrapPanel içerisindeki düğmelerin hiçbirinin konumlandırma bilgisi yok. Düğmelerin içerisinde yazılacak yazılar dışında hiçbir özelliklerini set edilmiş değil.



WrapPanel'in Orientation özelliği Vertical yapılinca...

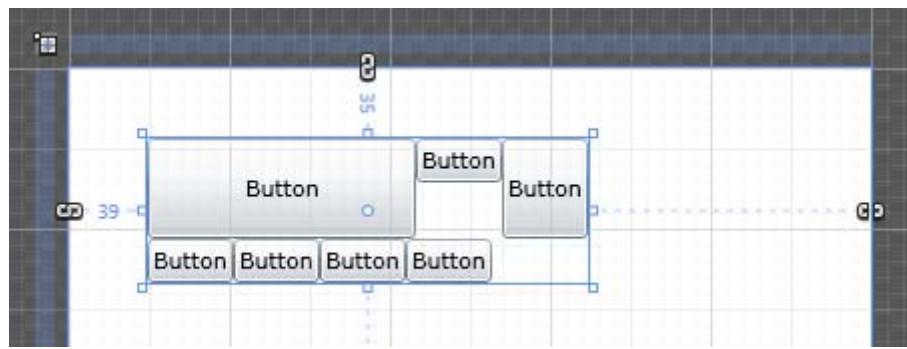
İsterseniz bir WrapPanel'in içerisindeki nesnelerin ekranın üstüne doğru değil de soluna doğru da toparlanmalarını sağlayabilirsiniz. Bunun için WrapPanel'in **Orientation** özelliğini **Vertical** olarak ayarlamamanız yeterli olacaktır. Böylece sıralama işlemi dikey olarak yapılacak ve dikey boy doldurulduğunda yeni bir kolon yaratılarak hizalama devam edecektir.

[XAML]

```
<controls:WrapPanel Margin="39,35,141,142" Orientation="Vertical">
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
</controls:WrapPanel>
```

WrapPanel içerisinde nesnelerin hizalama ayarlarına dikkat!

WrapPanel içerisindeki nesnelerin her biri kendi satır ve sütunları içerisinde hizalandırılabilirler. Stretch modunda olan nesnelerin oluşan bir kolon veya sütun içerisinde tüm nesnelerin en geniş veya en yüksek olan nesneye uyum sağlayacaktır. Bu nedenle eğer bir satır veya sütun içerisinde nesnelerin orijinal büyülüklerinde kalmalarını istiyorsanız hizalamalarını kesinlikle tek tek ayarlamalısınız.



Farklı hizalamalara sahip nesneler!

Yukarıdaki görselde ikinci düğmenin dikey hizalaması değiştirilerek kendi özgün boyutunda gösterilmesi sağlanmıştır. Söz konusu düğmenin bulunduğu satırın yüksekliğini arttıran isen birinci düğmedir.

[XAML]

```
<controls:WrapPanel Margin="39,35,141,142" Orientation="Horizontal"
HorizontalAlignment="Left" VerticalAlignment="Top">
    <Button Content="Button" Width="134" Height="50" HorizontalAlignment="Left"
VerticalAlignment="Top"/>
    <Button Content="Button" HorizontalAlignment="Left"
VerticalAlignment="Top"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
    <Button Content="Button"/>
</controls:WrapPanel>
```

Nerelerde kullanılabilir?

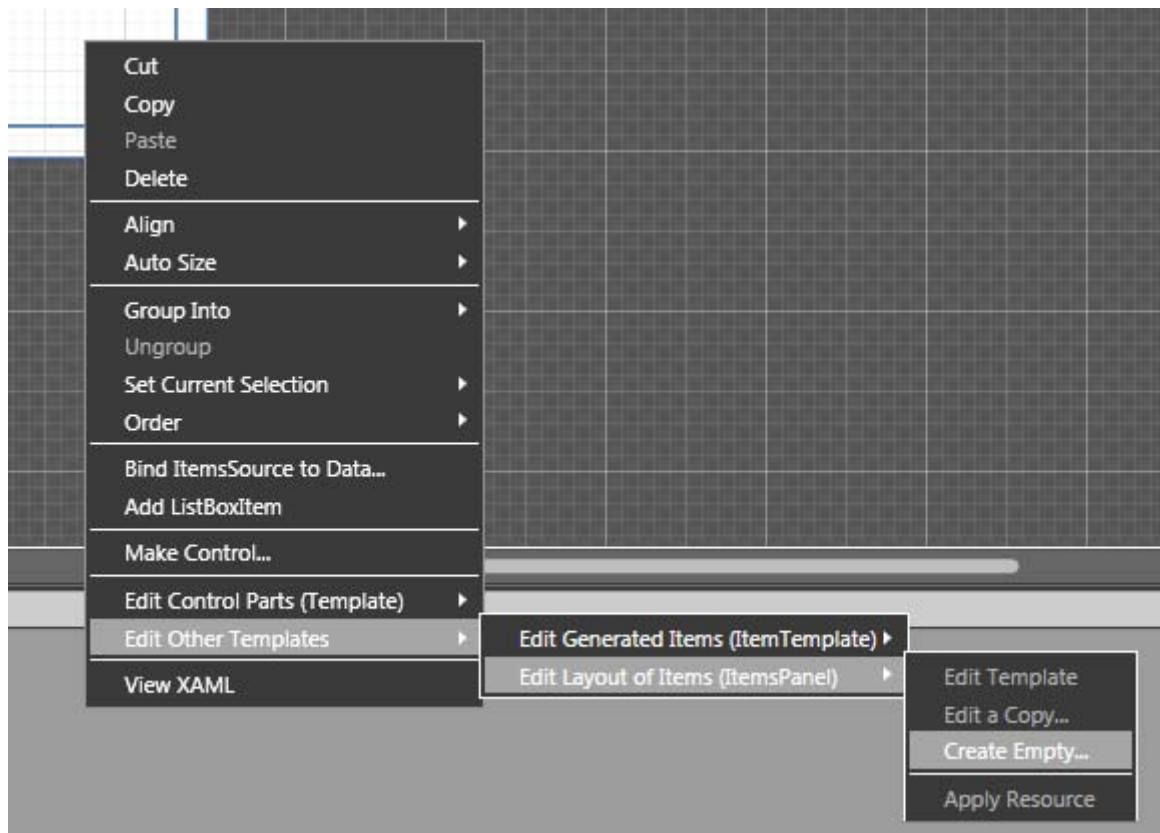
WrapPanel'in belki de en sık kullanıldığı basit yerlerden biri harici kontrollerdeki diğer Layout kontrollerinin yerini alarak farklı görsellikler sağlama noktasıdır. Örneğin normal şartlarda içerisindeki öğelerin dikey veya yatay olarak tek bir satır veya sütunda gösterilebileceği bir ListBox kontrolünün Layout kontrolünü değiştirerek ListBox içerisinde tüm nesnelerin ekrana sıgacak şekilde bir WrapPanel içerisinde toplanmasını sağlayabilirsiniz.



Standart bir ListBox.

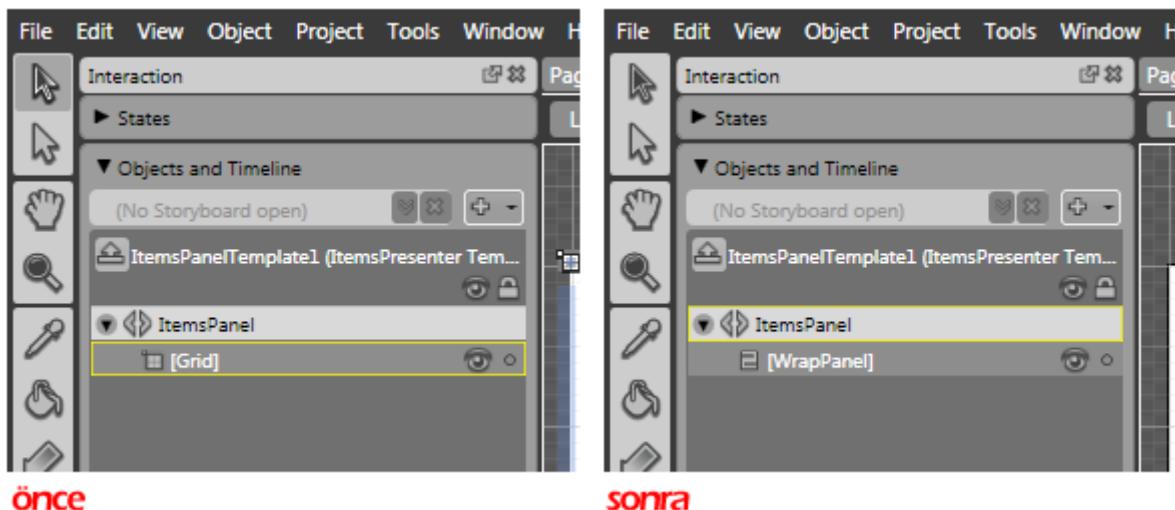
Yukarıda gördüğünüz standart ListBox içerisinde öğeler dikey olarak sıralanmış durumda. Oysa bu öğelerin uzunlukları kısa ve belki de "Deneme 1" ile "Deneme 2" yan yana konabilirdi. Böylece çok daha fazla öğe aynı anda ekranda gösterilirken kullanıcının çok daha hızlı şekilde aradığı şeye ulaşması sağlanabilirdi. Bu detayların haricinde görsel olarak da daha güzelbir sonuç alabiliriz.

Bu durumda bizim yapmamız gereken yukarıda hazırladığımız standart ListBox'in Layout kontrolünü değiştirmek. Bunun için Expression Blend içerisinde kontrolü seçerek sağ tuş tıklayıp aşağıdaki görselde görebileceğiniz menüye doğru hızlı bir yolculuk yapıyoruz.



ListBox'in Layout kontrolünü değiştireceğiz.

ListBox'ın içerisinde normal şartlarda bir Grid Layout kontrolü bulunuyor. Bu kontrolü silerek yerine bir WrapPanel koymamız gerekecek.



Grid yerine WrapPanel karşınızda.

WrapPanel'i koyduktan sonra WrapPanel'in genişliğini de ListBox'inizin genişliğine eşitlemeye unutmayın. Bunu dinamik arayüzlerde LayoutUpdated eventlarında yapabilir veya XAML tarafında bir Binding ile çözebilirsiniz.



ListBox içerisinde WrapPanel.

Yukarıdaki görselde WrapPanel ile ListBox'in iş ortaklığının sonucunu görebilirsiniz. ListBox içerisinde nesneler mümkün oldukça yan yana alınarak yukarıya doğru toplanmışlar. Bizim örneğimizde nesne sayısı artık az geldiği için ListBox'in scroll barları da gözükmüyor.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight uygulamaları ve IIS MIME Type ayarı.

Silverlight uygulamalarında kullandığımız XAML dosyalarının bazı durumlarda IIS tarafından istemciye gönderilmemişinden daha önceki yazılarımda bahsetmiştim. Bazen bu durum özellikle sunucu admini (hosting sağlayıcı) tarafından ek ücret talebi yapabilmek adına kasıtlı olarak yapılabildiği gibi bazen de gerçekten ortada bir sorun olabiliyor. Peki nasıl düzelteceğiz?

MIME Type tanımı nasıl yapılır?

Silverlight XAML dosyalarının IIS tarafından istemciye sunulabilmesi için aşağıdaki şekilde gerekli MIME type'ların tanımlanması gerekektir.

Dosya uzantısı: .xaml

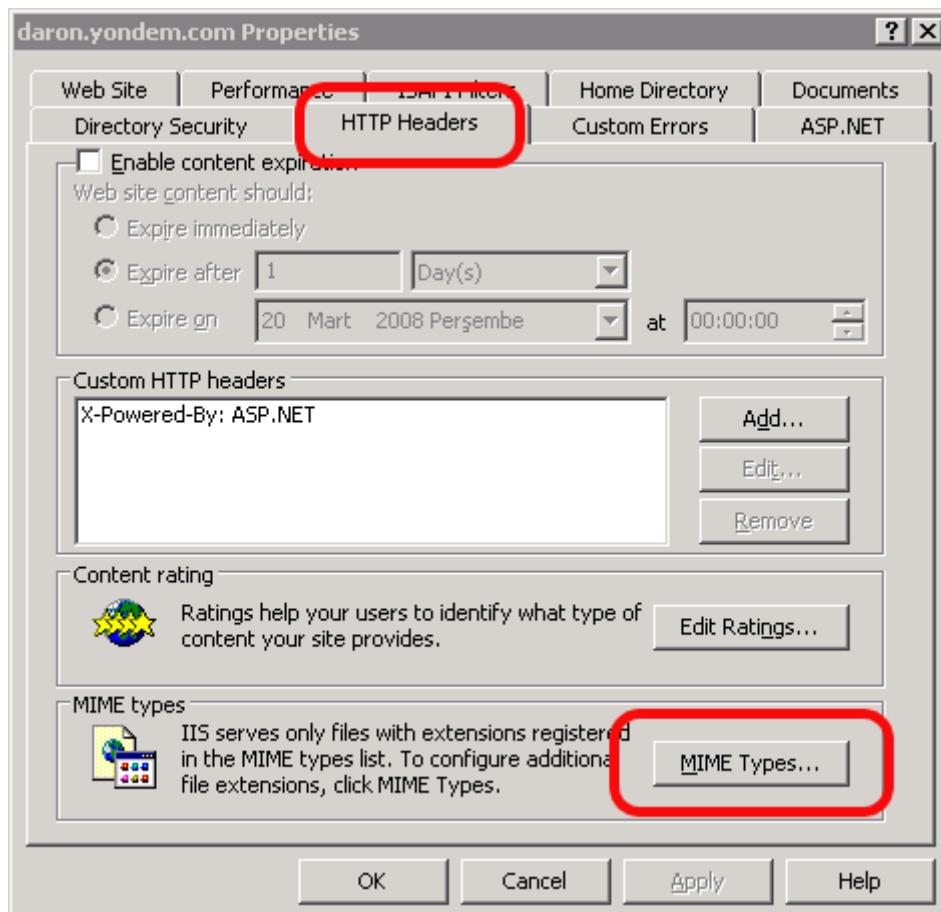
MIME type: application/xaml+xml

Dosya uzantısı: .xap

MIME type: application/x-silverlight-app

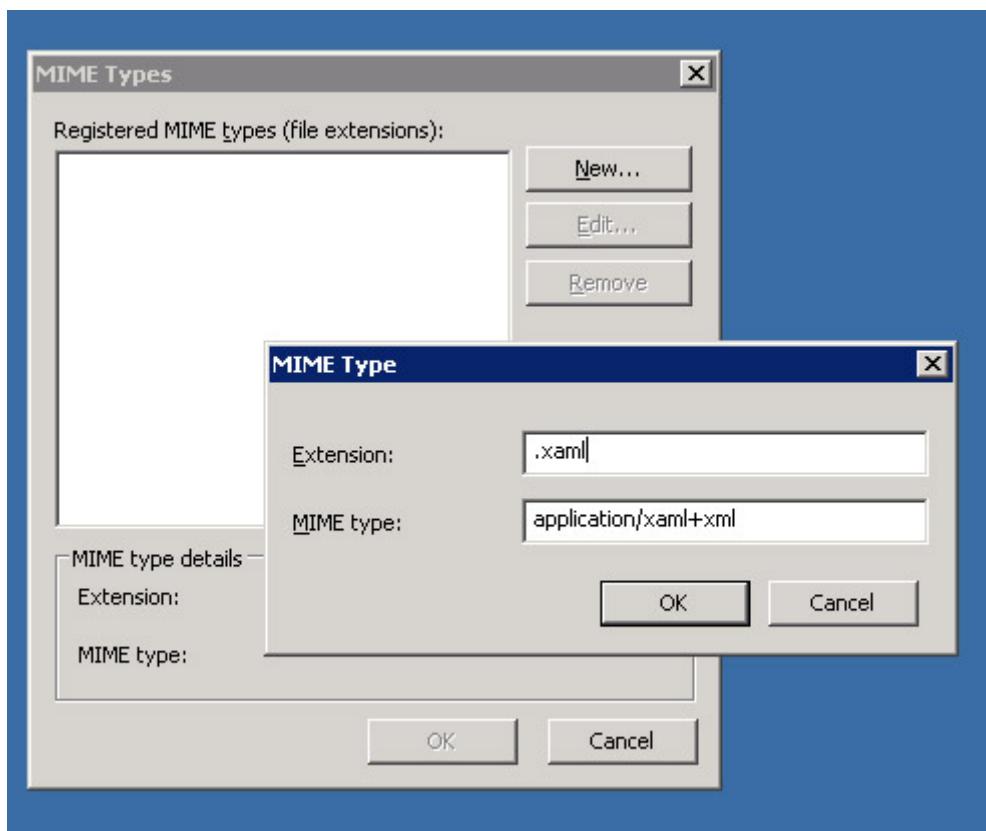
XAML'ı biliyorduk da XAP da nesi? :) XAP şu an Silverlight 2.0 Beta 1 ile beraber kullanılan bir dosya uzantısı. Eğer olur ya bir fantezi çerçevesinde Silverlight 2.0 Beta 1 ile sanal makinenizde geliştirdiğiniz bir uygulamayı sunucuya koymak isterseniz şu an için XAP dosyaları için de gerekli MIME Type'ları tanımlamanız gerekecektir. İleride Silverlight 2.0'in yayına çıkacak orijinal sürümlerinde bu uzantının kullanılmaya devam edeceği tabii ki garanti değil. Bahsettiğimiz teknoloji daha beta aşamasında.

Gelelim bu tanımlama işini nasıl yapacağımıza. Sunucuda **IIS Manager**'ı açtıktan sonra **MIME Type** ayarı yapmak istediğiniz siteyi seçerek sağ tuş tıklayarak gelen menüden "**Properties**" komutunu veriyoruz. Karşımıza gelen pencereden "HTTP Header" sekmesine geçerek en alttaki "MIME Types" düğmesine tıklıyoruz.



IIS Manager içerisinde MIME Types seçeneğini bulduk.

"MIME Types" bölümüne girdikten sonra hemen "New" düğmesine tıklayarak yeni bir "MIME Type" eklemek üzere bir önceki paragraftaki ayarları buraya aynen yazıyoruz ve gördüğümüz tüm "OK" düğmelerine basarak IIS Manager arayüzüne geri dönüyoruz.



Yeni bir MIME Type ekliyoruz.

Artık gerekli uzantılar tanımlandığı için herhangi bir sorun yaşamayacağız.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight 2.0 Beta 2 ile gelen DataGrid yenilikleri

Silverlight 2.0 Beta 1 ile beraber gelen DataGrid kontrolünün ana yapılarını daha önce incelemiştik. Bu yazımızda da **Silverlight 2.0 Beta 2** ile beraber gelen DataGrid yeniliklerine göz atacağız.

Sıralama İşlemleri

Maalesef Silverlight 2.0 Beta 1 ile beraber gelen DataGrid içerisinde herhangi bir otomatik sıralama yapısı gelmiyordu. Oysa çoğu ASP.NET'teki GridView'den de alışık olduğumuz üzere bazı durumlarda kolonların en üst başlık kısımlarını tıklanabilir hale getirerek kolon içerisindeki veriye göre satırları sıralatabiliyor olmak çok önemli bir işlevsellik. Artık Silverlight 2.0 Beta 2 ile beraber gelen DataGrid içerisinde bu özelliğin sağlayıcı hazır bir yapı var.

Peki neler yapmamız gerekiyor? Aslında yapmanız gereken neredeyse hiçbir şey yok. **IList** sınıfından türetilmiş herhangi bir listeyi DataGrid'e bağladığınızda sıralama (sorting) işlemleri otomatik olarak yapılabilecektir. Zaten baktığımızda DataGrid'lerimizi bu yapıyı destekleyen **List** veya **ObservableCollection** listeleri bağlıyoruz.

Normal şartlarda listeleme için kullanılan veri tıklanan kolona bağlı olan ve **DisplayMemberBinding**'de gözüken veri kaynağı oluyor. Oysa bazı durumlarda bu senaryo da düzgün çalışmamayacaktır. Örneğin normal kolonlar yerine bir **TemplateColumn** kullanıyorsanız **DisplayMember** olmadığı için sıralama işlemi de yapılamayacaktır. Bu gibi durumlarda kolonun görsel kısmından bağımsız olarak **SortMemberPath** özelliğini ayarlayarak kolonda ne gözükürse gözüksün arka planda sıralama işleminin farklı bir kolona veya veriye göre yapılmasını sağlayabilirisiniz.

Görsel Değişiklikler

Özellikle listeleme işlemini yaptığınız kolonların başlıklarında oklar meydana gelecektir. Bu okların tasarımdan tutun, meydana gelme animasyonlarına kadar her şeyi tek tek değiştirebilirisiniz. Bunun için MSDN'den aşağıdaki adresi incelemeniz yeterli olacaktır.

[http://msdn.microsoft.com/tr-tr/library/cc278066\(en-us,vs.95\).aspx](http://msdn.microsoft.com/tr-tr/library/cc278066(en-us,vs.95).aspx)

Bu adreste hali hazırda DataGrid içerisinde kullanılan tüm kaynakların, stillerin ve animasyonların kodları bulunuyor. Bunları alıp özelleştirip rahatlıkla uygulamanıza ekleyebilirisiniz. Örneğin aşağıda sadece kolonların başlıklarına ait kısımların özelleştirildiği bir XAML kodunu inceleyebilirisiniz. Satır arası yorumlarına özellikle dikkat etmenizde fayda var.

```
<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
    x:Class="SilverlightApplication5.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400"
    Height="300">
<UserControl.Resources>
```

```

<Style x:Key="KolonBasi" TargetType="local:DataGridColumnHeader"
    xmlns:local="clr-namespace:System.Windows.Controls;
    assembly=System.Windows.Controls.Data"
    xmlns:controls="clr-namespace:System.Windows.Controls;
    assembly=System.Windows">

    <Setter Property="SeparatorBrush">
        <Setter.Value>
            <SolidColorBrush Color="#FFA4A4A4" />
        </Setter.Value>
    </Setter>
    <Setter Property="SeparatorVisibility"
        Value="Visible" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="local:DataGridColumnHeader">
                <!-- Gridi boyayan Gradient. -->
                <Grid Name="RootElement">
                    <Grid.Background>
                        <LinearGradientBrush StartPoint="0.276463,-0.00385181"
                            EndPoint="0.276463,0.71">
                            <GradientStop Color="#FFF9FAFA"
                                Offset="0" />
                            <GradientStop Name="StopColor2"
                                Color="#FFEDF1F4"
                                Offset="0.37259" />
                            <GradientStop Name="StopColor3"
                                Color="#FFE2E8EF"
                                Offset="0.372881" />
                            <GradientStop Name="StopColor4"
                                Color="#FFC3C9CD"
                                Offset="1" />
                        </LinearGradientBrush>
                    </Grid.Background>
                </Grid>
                <Grid.Resources>
                    <!-- Normal duruma getirme animasyonu. -->
                    <Storyboard x:Key="Normal State">
                        <ColorAnimation Storyboard.TargetName="StopColor2"
                            Storyboard.TargetProperty="(Color)"
                            Duration="00:00:0.3"
                            To="#FFEDF1F4" />
                        <ColorAnimation Storyboard.TargetName="StopColor3"
                            Storyboard.TargetProperty="(Color)"
                            Duration="00:00:0.3"
                            To="#FFE2E8EF" />
                        <ColorAnimation Storyboard.TargetName="StopColor4"
                            Storyboard.TargetProperty="(Color)"
                            Duration="00:00:0.3"
                            To="#FFC3C9CD" />
                    </Storyboard>
                </Grid.Resources>
            </ControlTemplate>
        </Setter.Value>
    </Setter>

```

```

</Storyboard>
<!-- Fare üzerine gelince çalışan animasyon. -->
<Storyboard x:Key="MouseOver State">
    <ColorAnimation Storyboard.TargetName="StopColor2"
        Storyboard.TargetProperty="(Color)"
        Duration="00:00:0.3"
        To="#FFE6EFF7" />
    <ColorAnimation Storyboard.TargetName="StopColor3"
        Storyboard.TargetProperty="(Color)"
        Duration="00:00:0.3"
        To="#FFD3E4F5" />
    <ColorAnimation Storyboard.TargetName="StopColor4"
        Storyboard.TargetProperty="(Color)"
        Duration="00:00:0.3"
        To="#FF87A5BA" />
</Storyboard>
<!-- Sıralama bozulduğunda gösterilen animasyon. -->
<Storyboard x:Key="Unsorted State">
    <DoubleAnimation Storyboard.TargetName="SortIconElement"
        Storyboard.TargetProperty="Opacity"
        Duration="00:00:0.3"
        To="0.0" />
    <DoubleAnimation Storyboard.TargetName="SortIconTransform"
        Storyboard.TargetProperty="ScaleY"
        BeginTime="00:00:0.3"
        Duration="00:00:0.0"
        To="1" />
</Storyboard>
<!-- Sıralama anındaki animasyon -->
<Storyboard x:Key="SortedAscending State">
    <DoubleAnimation Storyboard.TargetName="SortIconElement"
        Storyboard.TargetProperty="Opacity"
        Duration="00:00:0.3"
        To="1.0" />
    <DoubleAnimation Storyboard.TargetName="SortIconTransform"
        Storyboard.TargetProperty="ScaleY"
        Duration="00:00:0.3"
        To="-1" />
</Storyboard>
<!-- Sıralama anındaki animasyon -->
<Storyboard x:Key="SortedDescending State">
    <DoubleAnimation Storyboard.TargetName="SortIconElement"
        Storyboard.TargetProperty="Opacity"
        Duration="00:00:0.3"
        To="1.0" />
    <DoubleAnimation Storyboard.TargetName="SortIconTransform"
        Storyboard.TargetProperty="ScaleY"
        Duration="00:00:0.3"
        To="1" />
</Storyboard>

```

```

</Grid.Resources>

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

<Rectangle Stretch="Fill"
    StrokeThickness="2"
    Stroke="#FFFFFF"
    Grid.ColumnSpan="2"
    Grid.RowSpan="2" />
<Rectangle Grid.Row="2"
    Grid.ColumnSpan="3"
    Height="1"
    HorizontalAlignment="Stretch"
    Fill="#FFA4A4A4" />
<Rectangle Grid.RowSpan="2"
    Grid.Column="2"
    Width="1"
    VerticalAlignment="Stretch"
    Fill="{TemplateBinding SeparatorBrush}"
    Visibility="{TemplateBinding SeparatorVisibility}" />
<!-- Kolonun için kolon adı buraya yazılıyor Bind edilmiş -->
<controls:ContentPresenter Margin="3,0,3,0"
    Content="{TemplateBinding Content}"
    VerticalAlignment="Center" />
<!-- Sıralam oku görseli burada -->
<Path Name="SortIconElement"
    Margin="3,0,3,0"
    Opacity="0"
    Grid.Column="1"
    Stretch="Uniform"
    Width="8"
    Data="F1 M -5.215,0.0L 5.215,0.0L 0,6.099L -5.215,0.0 Z ">
<Path.Fill>
    <SolidColorBrush Color="#FF313131" />
</Path.Fill>
<Path.RenderTransform>
    <ScaleTransform Name="SortIconTransform"
        CenterX="4"
        CenterY="2.5"
        ScaleX="1"
        ScaleY="1" />

```

```

        </Path.RenderTransform>
    </Path>
</Grid>
</ControlTemplate>
<Setter.Value>
</Setter>
</Style>

</UserControl.Resources>
<Grid x:Name="LayoutRoot"
      Background="White">
    <!-- Stilimizi gride bağladık. -->
    <my:DataGrid x:Name="Veri"
                  ColumnHeaderStyle="{StaticResource KolonBasi}"
                  AutoGenerateColumns="False"
                  AlternatingRowBackground="#FFFFFF00"
                  HorizontalGridlinesBrush="#FFD4FF00"
                  RowBackground="#FFE3E3E3">
        <my:DataGrid.Columns>
            <my:DataGridTextColumn Header="Adı"
                                  DisplayMemberBinding="{Binding Adı}" />
        </my:DataGrid.Columns>
    </my:DataGrid>
</Grid>
</UserControl>

```

Otomatik boyutlandırma

Yeni bir Silverlight uygulaması yarattığınızda sahnede yer alan LayoutRoot aslında bir Grid nesnesidir. Silverlight 2.0 Beta 1'den farklı olarak Beta 2 ile beraber gelen DataGrid'in kendi içerisindeki veriye göre kendini otomatik boyutlandırma şansı var. Varsayılan ayarları ile bir DataGrid yarattığınızda bu özellik açık geliyor. Tabi DataGrid'in istediği kadar büyüyebilmesi için uygun kontroller içeresine yerleştirilerek herhangi bir şekilde boyutlarının sınırlanılmamış olması gerekiyor. Programatik olarak bir nesnenin yükseklik veya genişliğini otomatik olarak ayarlamak istiyorsanız `Double.NaN` 'a eşitlemeniz yeterli olacaktır.

DataGrid'in genişliğini otomatik olarak ayarlamadan sıra artık kolonlar da içlerindeki veriye göre otomatik olarak boyutlandırılabilirler. Bu boyutlandırma için ise 4 farklı teknik kullanılabiliyor. Birincisi "**Auto**" değerini vererek işlemi tamamen otomatiğe bırakmak, ikincisi **SizeToHeader** diyerek kolonun kendi başlığı kadar genişleyebilmesini sağlamak, üçüncüsü **SizeToCells** diyerek kolonun başlığından bağımsız olarak içeriğindeki veri kadar genişlemesini sağlamak ve dördüncüsü de doğrudan sayısal bir değer vererek kolonun genişliğini ayarlamak. Aşağıda örnek bir kullanımı görebilirsiniz.

```

<my:DataGrid x:Name="Veri"
              ColumnHeaderStyle="{StaticResource KolonBasi}"
              AutoGenerateColumns="False"
              AlternatingRowBackground="#FFFFFF00"
              HorizontalGridlinesBrush="#FFD4FF00"

```

```

RowBackground="#FFE3E3E3">
<my:DataGrid.Columns>
  <my:DataGridTextBoxColumn Width="SizeToHeader"
    Header="Adı"
    DisplayMemberBinding="{Binding Adı}" />
</my:DataGrid.Columns>
</my:DataGrid>

```

Aynı işlemi kod ile yaptığınızda ise doğrudan DataGridLength sınıfı üzerinden gerekli seçeneklere ulaşabiliyorsunuz. Sayısal bir değer atayarak kolonun genişliğini sabitlemek istediğinizde ise maalesef sizi ufak bir sorun bekliyor. Kolon genişliğini **DataGridLength** olarak atamanız lazım, **integer** veya **double** kesinlikle kabul edilmiyor. Bu durumda yardımımıza **DataGridLengthConverter** adında bir sınıf yetişiyor.

[VB]

```
kolon.Width = (New DataGridLengthConverter).ConvertFrom(50)
```

[C#]

```
kolon.Width = (DataGridLength)(new DataGridLengthConverter()).ConvertFrom(100);
```

DataGridLengthConverter sınıfındaki ConvertFrom metodu ile elimizdeki herhangi bir değişkenden DataGridLength yaratarak kolonlarımızın genişliğini kod tarafından da düzenleyebiliyoruz.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverlight ve WPF'de Design Mode ve Init durumunda kodlar sorunsalı.

WPF veya Silverlight projelerinde **Init** durumu ile **PageLoad** veya **WindowLoad** event'ları arasındaki fark bazen ilk bakışta yokmuş gibi varsayılarak kodların doğrudan **Init** kısmına yazıldığını çok gördüm. Bazen bu durum sorun çıkarmasa da aslında tam olarak **Init** durumu bitmediği için bazı kaynaklara veya kontrollere ulaşmama hatta bu ulaşıp / ulaşmama durumunun da belli olmaması :) gibi garip hatalar ile karşılaşabilirsiniz. O nedenle benim genel tavsiyem sürekli **Loaded** event'larının kullanılması ve **Init'in** sadece ek event-listener tanımlamalarının yapılacağı bir konum olarak saklanması.

Bu çerçevede bir diğer sorun ise **Init** durumuna yazdığınız kodların aslında hem Blend hem de Visual Studio tarafından Design modundayken çalıştırılıyor olması. Eğer bu bilgiye sahip değilseniz maalesef ki **Init** durumunda yaptığınız ağır bir işlemin bir anda Visual Studio ve Blend'in arayüzüne de binmesi ve Page.XAML gibi bir dosyayı açtığınız anda yüksek işlemci kullanımları ile karşılaşmanız olası. En basit çözüm bu kodları Init'den çıkarmak ve Loaded'a yerleştirmek olabilir fakat ya Init'i kesinlikle kullanmanız gerekiyorsa?

[C#]

```
if (System.ComponentModel.DesignerProperties.GetIsInDesignMode(this) == false)
{
}
```

İşte yukarıdaki gibi bir kontrol ile söz konusu Init kodunun içinde uygulamanın DesignMode'da olup olmadığını kontrol edebilirsiniz. Böylece eğer sayfa Visual Studio veya Blend ile Design modunda açılmış ise bu IF içerisinde kodlar çalıştırılmayacak. Oysa programı F5 ile compile edip çalıştırıldığınızda ise herhangi bir sorun ile karşılaşamaycaksınız.

Hepinize kolay gelsin.

Daron YÖNDEM

Silverligh 2.0'da uygulama fonunu şeffaf kullanmak

Özellikle dikdörtgen köşelere sahip olmayan Silverlight uygulamalarında sayfanın fonunun Silverlight'in fonunda da gözükmesini isteyebilirsiniz. Aslında basit bir şekilde **Silverlight 2.0 Beta 2** uygulamasının fonunu şeffaf yapabilsek sorunumuz çözülmüş olacaktır. Bunun için yapmamız gereken ufak bir kaç ayar var.

Eğer bir ASP.NET sayfasında Silverlight sunucu kontrolünü kullanıyorsanız aşağıdaki şekilde **PluginBackground** özelliğini **Transparent** ve **Windowless** özelliğini de **True** olarak ayarlamınız yeterli olacaktır. ASP.NET Silverlight sunucu kontrolü gerekli HTML içeriği sizin için üretecektir.

```
<asp:Silverlight PluginBackground="Transparent" Windowless="true"
ID="Xaml1" runat="server" Source="~/ClientBin/SilverlightApplication29.xap"
MinimumVersion="2.0.30523" Width="100%" Height="100%" />
```

Eğer Silverlight uygulamanızı ASP.NET dışı bir sayfada kullanacaksanız bu sefer söz konusu parametreleri OBJECT tagları arasında belirtmeniz gerekiyor.

```
<object data="data:application/x-silverlight," type="application/x-silverlight-2-b2"
width="100%" height="100%">
    <param name="source" value="ClientBin/SilverlightApplication29.xap"/>
    <param name="onerror" value="onSilverlightError" />
    <param name="background" value="Transparent" />
    <param name="pluginbackground" value="Transparent" />
    <param name="windowless" value="true" />
    <a href="http://go.microsoft.com/fwlink/?LinkID=115261" style="text-decoration:
none;">
        
    </a>
</object>
```

Tabi tüm bunları yaparken Silverlight uygulaması içerisinde Root görselinizin fonunun da şeffaf bırakıldığını kontrol etmekte fayda var.

Hepinize kolay gelsin.

Daron YÖNDEM

SL 2.0'da ZIP içerisinde asenkron kaynak kullanımı.

Silverlight 1.0 içerisinde sunucu tarafından ZIP dosyaları alarak bunların içerisindeki dosyaları istemci tarafında çıkartarak kullanabiliyor olmak büyük avantaj sağlıyordu. Özellikle böyle bir işlevsellik sağlamak için de neredeyse hiçbir ek kod yazmamız şarşırıcı bir kolaylıktı. Bu şaşkınlığımızı almak adına **Silverlight 2.0 Beta 1** içerisinde işler biraz daha zorlaştırılmış durumda. Gelin bir örnek ile ilerleyelim. Sayfamızda iki adet resim göstermek istiyoruz. Bu resimler bir ZIP dosyası içerisinde sunucuda yer alacak.

```
<UserControl
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SilverlightApplication6.Page"
    Width="640" Height="480">

    <Grid x:Name="LayoutRoot" Background="White" >
        <Image HorizontalAlignment="Left" Margin="42,64,0,235" Width="221"
x:Name="Resim1" />
        <Image Margin="303,64,118,235" x:Name="Resim2"/>
        <Button Height="53" HorizontalAlignment="Stretch" Margin="200,0,301,138"
VerticalAlignment="Bottom" Content="Button" x:Name="Dugme"/>
    </Grid>
</UserControl>
```

Yukarıdaki örnek Silverlight 2.0 Beta 1 projemize ait **Page.xaml** dosyasının kodunu inceleyebilirsiniz. Hedefimiz düğmeye tıklandığında sunucudaki **fotolar.zip** dosyasını istemciye indirerek içerisindeki iki farklı resmi ekrandaki iki boş **Image** tagının içerisine yerleştirmek. İlk olark sunucudan asenkron bir istek ile veri alabilmek için bir **WebClient** nesnesi yaratmamız gerekiyor.

```
Private Sub Dugme_MouseLeftButtonDown(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs) Handles Dugme.MouseLeftButtonDown
    Dim Talep As New System.Net.WebClient
    AddHandler Talep.OpenReadCompleted, AddressOf VeriGeldi
    Talep.OpenReadAsync(New Uri("fotolar.zip", UriKind.Relative))
End Sub
```

Yukarıdaki kodumuz Silverlight uygulamamızda düğmeye basıldığında çalışacak olan kodun ta kendisi. İlk olarak **Talep** adında bir **WebClient** nesnesi yaratıyoruz. Sonrasında bu **WebClient**'in **OpenReadCompleted** durumunda çalışacak olan **VeriGeldi** fonksiyonuna ait atamayı tamamlıyoruz. Böylece sunucudan istediğimiz veri istemciye ulaştığında birazdan yazacağımız **VeriGeldi** fonksiyonu / event handler'ı çalışacak.

Son satırda **OpenReadAsync** metodu ile sunucudan **fotolar.zip** dosyasını istiyoruz. Söz konusu isteği gerçekleştirirken **OpenReadAsync** metoduna bir **Uri** vermemiz gerekiyor bunun için **Relative** bir Uri yaratıyoruz. Relative yaratmamızın nedeni **fotolar.zip** dosyasının Silverlight uygulamasının çalıştığı adres ile aynı konumda bulunacak olması.

```
Dim GelenVeri As System.IO.Stream = e.Result
```

```
Dim FotoStream As System.IO.Stream = Application.GetResourceStream(New
System.Windows.Resources.StreamResourceInfo(GelenVeri, Nothing), New Uri("creek.jpg",
UriKind.Relative)).Stream
```

Artık yavaş yavaş **VeriGeldi** event-handlerimizda çalışacak kodları yazmaya başlayalım. İlk satırda söz konusu event-handlerimize parametre olarak gelecek olan **System.Net.OpenReadCompletedEventArgs** tipinde **e** değişkeni üzerinden **e.Result** diyerek sunucudan gelen Stream'i alıyoruz. Sonraki satırda işler biraz karışık. Elimizdeki Stream'i, yani **GelenVeri** değişkenini bir **StreamResourceInfo**'ya dönüştürmemiz gerekiyor. Bunu da **Application.GetResourceStream** metoduna birinci parametre olarak veriyoruz. Böylece **GetResourceStream** metodu artık nereden Resource çıkartacağını biliyor. ZIP dosyamızın gelen stream'ini dönüştürerek **GetResourceStream**'e verdikten sonra tanımlamamız gereken ikinci parametre ise ZIP dosyası içerisindeki hangi dosyayı yani **Resource'u** almak istediğimiz. Bunun için de bir **Relative Uri** yaratarak **GetResourceStream'a** metoduna ikinci parametre olarak veriyoruz. Artık **GetResourceStream** gerekli veriyi kendisine verdiği ZIP Stream içerisinde alarak bize yine bir Stream olarak verecektir. **FotoStream** değişkenimizi de bu şekilde yakaladıktan sonra artık işlemlere devam edebiliriz.

```
Dim Foto As New System.Windows.Media.Imaging.BitmapImage
Foto.SetSource(FotoStream)
Resim1.Source = Foto
Resim1.Measure(New System.Windows.Size)
```

FotoStream'imizden bir **Foto** yaratabilmek için **Foto** adında bir **BitmapImage** nesnesi yaratıyoruz. Söz konusu nesnenin **SetSource** metodunu kullanarak doğrudan elimizdeki **FotoStream'i** kendisine aktarıyoruz. Artık elimizde bir **Foto** bulunduğu göre bunu XAML kodumuzdaki **Resim1'in Source** özelliğine aktarabiliriz. Son olarak **Resim1'e** ait **Measure** metodunu da çağrıarak görsel anlamda **Resim1'in** ekranда kaplayacağı yer olarak kendisini toparlamasını sağlamamız gerekiyor. Aksi halde resim yüklense de daha önce **Resim1'in** içi boş olduğu ve boyutu ekranда ufak olduğu için gözükmeyecektir. Artık tek yapmamız gereken aynı işlemleri ikinci resim için de yaparak **GelenVeri'den** ikinci dosyamızı da almak ve **Resim2** içerisinde yerleştirmek.

```
Private Sub VeriGeldi(ByVal sender As Object, ByVal e As
System.Net.OpenReadCompletedEventArgs)
    Dim GelenVeri As System.IO.Stream = e.Result
    Dim FotoStream As System.IO.Stream = Application.GetResourceStream(New
System.Windows.Resources.StreamResourceInfo(GelenVeri, Nothing), New Uri("creek.jpg",
UriKind.Relative)).Stream
    Dim Foto As New System.Windows.Media.Imaging.BitmapImage
    Foto.SetSource(FotoStream)
    Resim1.Source = Foto
    Resim1.Measure(New System.Windows.Size)
    Foto = New System.Windows.Media.Imaging.BitmapImage
    Foto.SetSource(Application.GetResourceStream(New
System.Windows.Resources.StreamResourceInfo(GelenVeri, Nothing), New Uri("autumn
leaves.jpg", UriKind.Relative)).Stream)
    Resim2.Source = Foto
    Resim2.Measure(New System.Windows.Size)
End Sub
```

Peki bu ZIP dosyası içerisinde sadece resimler mi bulunabilir? Tabi ki hayır. Aynı ZIP dosyası içerisinde bir de **deneme.txt** adında metin dosyası bulunduğu varsayıyalım ve söz konusu dosya içerisinde ufak bir metni alıp düğmemizin üzerine yazdırıralım.

```
Dim MetinStream As System.IO.Stream = Application.GetResourceStream(New
System.Windows.Resources.StreamResourceInfo(GelenVeri, Nothing), New
Uri("deneme.txt", UriKind.Relative)).Stream
```

```
Dim bitler(MetinStream.Length) As Byte
MetinStream.Read(bitler, 0, MetinStream.Length)
Dim Metin As New System.Text.StringBuilder
For Each bit As Byte In bitler
    Metin.Append(Convert.ToChar(bit))
Next
Dugme.Content = Metin.ToString
```

Her zamanki gibi yeni bir **Stream** yaratarak elimizdeki **GelenVeri** üzerinden ZIP dosyasında **deneme.txt**'yi alıyoruz. Stream'imizi bir **byte** dizisine aktardıktan sonra her bir byte'ı tek tek karaktere çevirerek bir **StringBuilder** yardımı ile sürekli bir metne dönüştürüyoruz. Son olarak elimizdeki metni de düğmemizin üzerine yazdırıyoruz.

Böylece harici bir ZIP dosyasını istemci tarafında asenkron olarak indirmiş ve sonrasında da dosya içerisindeki iki farklı resmi farklı **Image** nesnelerine aktarmış olduk. Bu kadarla kalmayıp aynı ZIP dosyası içerisindeki bir TXT dosyasından da metin alarak kullandık. Sonuç olarak istemci tarafından sunucuya sadece bir istek yolladık ve tek bir dosya aldık. Ayrıca aldığımız dosyanın sıkıştırılmış olduğunu da unutmamakta fayda var.

Hepinize kolay gelsin.

Daron YÖNDEM

Vista Gradientları XAML Kodları

Özellikle "developer" tabanlı olanlar için hazırlanan bir uygulamanın görsel arayüzüne süslmek hem bir "çin işkencesi" oluyor hem de ortaya zaten güzel bir ürün de çıkmıyor. Bu gibi durumlarda eğer projelerinizde bir tasarımcı ile çalışma şansınız da yoksa en azından internetteki hazır renk şemalarından faydalanabilir, birbiri ile uyumlu renkler üreten web sitelerinden faydalanabilirsınız.

Veya daha da pratik bir yol var, bir yerlerde beğendiğiniz renkleri "ödünç" alabilirsiniz :) Peki nerden? Hemen önünüzde Windows Vista duruyor, tasarımını da hiç fena sayılmaz :)

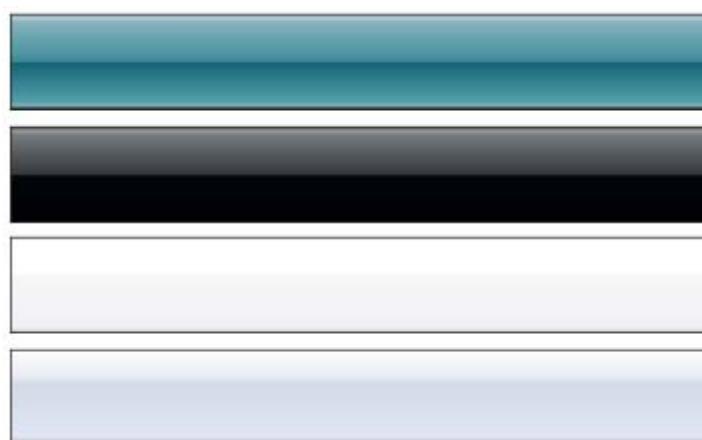
```
<Canvas
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="640" Height="375.082"
    Background="White"
    x:Name="Page">
    <Rectangle Width="572.131" Height="78.885" Canvas.Left="31.148" Canvas.Top="8"
    Stroke="#FF000000">
        <Rectangle.Fill>
            <LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">
                <LinearGradientBrush.GradientStops>
                    <GradientStop Color="#FF000000" Offset="0"/>
                    <GradientStop Color="#FF9AC6CF" Offset="0.0494537"/>
                    <GradientStop Color="#FF54A1AA" Offset="0.0714264"/>
                    <GradientStop Color="#FF146478" Offset="0.5"/>
                    <GradientStop Color="#FF408C9A" Offset="0.505493"/>
                    <GradientStop Color="#FF87B6C0" Offset="0.928574"/>
                    <GradientStop Color="#FFBCCDD7" Offset="0.950546"/>
                    <GradientStop Color="#FFAEBFCA" Offset="0.983521"/>
                    <GradientStop Color="#FFAEBFCA" Offset="1"/>
                </LinearGradientBrush.GradientStops>
            </LinearGradientBrush>
        </Rectangle.Fill>
    </Rectangle>
    <Rectangle Width="572.131" Height="78.885" Canvas.Left="31.148"
    Canvas.Top="99.784" Stroke="#FF000000">
        <Rectangle.Fill>
            <LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">
                <LinearGradientBrush.GradientStops>
                    <GradientStop Color="#FF000104" Offset="0"/>
                    <GradientStop Color="#FF02070B" Offset="0.494507"/>
                    <GradientStop Color="#FF33373D" Offset="0.494507"/>
                    <GradientStop Color="#FF757A7C" Offset="0.917587"/>
                    <GradientStop Color="#FFA0A1A3" Offset="0.956039"/>
                    <GradientStop Color="#FF48494A" Offset="1"/>
                </LinearGradientBrush.GradientStops>
            </LinearGradientBrush>
        </Rectangle.Fill>
    </Rectangle>
```

```

<Rectangle Width="572.131" Height="78.885" Canvas.Left="31.148"
Canvas.Top="189.929" Stroke="#FF000000">
<Rectangle.Fill>
<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">
<LinearGradientBrush.GradientStops>
<GradientStop Color="#FFD4D4D4" Offset="0"/>
<GradientStop Color="#FFF0F2F4" Offset="0.0659332"/>
<GradientStop Color="#FFF6F5F8" Offset="0.598907"/>
<GradientStop Color="#FFFFFFFF" Offset="0.609894"/>
<GradientStop Color="#FFFFFFFF" Offset="0.978027"/>
<GradientStop Color="#FFC7C7C7" Offset="0.994507"/>
<GradientStop Color="#FFC7C7C7" Offset="1"/>
</LinearGradientBrush.GradientStops>
</LinearGradientBrush>
</Rectangle.Fill>
</Rectangle>
<Rectangle Width="572.131" Height="78.885" Canvas.Left="31.148"
Canvas.Top="281.713" Stroke="#FF000000">
<Rectangle.Fill>
<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">
<LinearGradientBrush.GradientStops>
<GradientStop Color="#FFDFE4F4" Offset="0"/>
<GradientStop Color="#FFB8BCC2" Offset="0.0439606"/>
<GradientStop Color="#FFE0E6F4" Offset="0.0769196"/>
<GradientStop Color="#FFD4DBE8" Offset="0.648346"/>
<GradientStop Color="#FFE8ECF4" Offset="0.714279"/>
<GradientStop Color="#FFFFFFFF" Offset="1"/>
</LinearGradientBrush.GradientStops>
</LinearGradientBrush>
</Rectangle.Fill>
</Rectangle>
</Canvas>

```

İşte size Vista'daki gradientların Silverlight veya WPF ile kullanılabilecek XAML kodları. Yukarıdaki örnek Silverlight 1.0 uygulamasında dört farklı gradienti gösterebilmek için dört adet dikdörtgen kullandım.



Vista Gradientları

Silverlight UserControl'üm Design modunda Blend içinde mi? Yoksa gerçek hayatı mı?

Bugün sizlerle ufak fakat bence bir o kadar da değerli bir ip ucu paylaşacağım. Üzerinde çalıştığımız projelerden birinde hiç hoş olmayan bir sorun ile karşılaştık. Aslında sorunun nedeni Visual Studio ve Expression Blend içerisinde Silverlight projeleri düzenlenirken söz konusu projeler içerisindeki UserControl'lerin PageLoad ve Init kodlarının tasarım aşamasında da çalıştırılıyor olması. Ne demek istiyorum?

Örneğin **Detay** adında bir UserControl hazırladınız ve bunu da **Ana** adında bir UserControl'ün içerisinde yerleştirdiniz. Bu yerleştirme işlemini de XAML içerisinde namespace tanımlayarak yaptınız ki tasarımcı Blend içerisinde söz konusu UserControl'u rahatlıkla düzenleyebilsin. Özetle UserControl'lerinizin XAML kodları aşağıdaki gibi olsun;

[Ana.xaml]

```
<UserControl x:Class="SilverlightApplication27.Anा"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="400" Height="300" xmlns:SilverlightApplication27="clr-
  namespace:SilverlightApplication27"
  xmlns:d="http://schemas.microsoft.com/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">
  <Grid x:Name="LayoutRoot" Background="White">
    <TextBlock x:Name="Metin" Text="DENEME"/>
    <SilverlightApplication27:Detay Margin="83,61,217,139"
      d:LayoutOverrides="VerticalAlignment"/>
  </Grid>
</UserControl>
```

Gördüğünüz üzere diğer UserControl'ü almak üzere XAML NameSpace tanımı yapılmış ve ekrana da Detay adındaki UserControl yerleştirilmiş.

[Detay.xaml]

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  x:Class="SilverlightApplication27.Detay"
  d:DesignWidth="100" d:DesignHeight="100">

  <Grid x:Name="LayoutRoot">
    <Grid>
      <TextBlock HorizontalAlignment="Left" VerticalAlignment="Top" Text="TextBlock"
        TextWrapping="Wrap" x:Name="Metin"/>
    </Grid>
  </Grid>
</UserControl>
```

Yukarıdaki şekli ile tanımladığımız **Detay** adındaki **UserControl** içerisinde de sadece bir **TextBlock** bulunuyor. **Detay**'in kodunda bir şekilde **Page.Load** durumunda ekrandaki kontrollere veri bağlarsanız tüm bu işlemlerin Expression Blend içerisinde **Ana.XAML** açıldığında çalıştırıldığını göreceksiniz. Sonuç itibarı ile **Ana.XAML** içerisinde **Detay** kontrolünü koyduğumuz için Blend **Ana** adındaki **UserControl** içerisinde göstermek üzere **Detay**'ı çalıştırıp doğrudan sahneye yerleştiriyor. "Ne kadar güzel?" dedığınızı duyar gibiyim :) Aslında durum gerçekten hoş. Bu sistem sayesinde ana bir kontrole yerleştirilmiş **UserControl**'ler kendi **Page.Load**'ları da çalıştırılarak gerçek çıkış hallerindeki görüntüleri ile tasarımcıya gösteriliyorlar. Fakat ya **Page.Load**'da çalışan kod uygulamanın bir web sunucu üzerinden host edilmiş olmasını gerektiriyorsa? İşte tam da o noktada Blend çatlıyor :) doğal olarak....

Peki ne yapmak gerek?

Çözüm basit. Bizim **UserControl**'lerin kendilerinin Blend'de mi yoksa gerçekten çalışma zamanında da çalıştırıldıklarını algılamaları ve ona uygun işlem yapmaları gerekiyor. Örneğin bizim **detay** adındaki **UserControl** Blend tarafından açıldığında kendi içindeki **TextBlock**'e "DENEME" yazarken, gerçekten açıldığında ise sunucudan veri çekmeli.

[VB]

```
If Not ComponentModel.DesignerProperties.GetIsInDesignMode(Me) Then
    MyServis.GetNewsFromBlogAsync()
Else
    textBlock.Text = "Blend içerisinde blog ile bağlantı kurulamaz."
End If
```

Yukarıda gördüğünüz kod ile herhangi bir **UIELEMENT**'in Blend içerisinde Design modunda açılıp açılmadığını öğrenebiliyorsunuz. Örneğin bizim örneğimizde normal şartlarda web servisinden veri çeken kod eğer Blend içerisinde açılmış ise veri çekmek yerine uygun yere uygun mesajı yazıyor. Farklı örneklerde tasarımcı yardımcı olmak amacıyla **DataBind** ettiğiniz kontrolleri belki de tasarımcı için kod içerisinde veri yaratıp databind edebilirsiniz. Oysa uygulama çalıştığında gerçek veri kaynağına yönelebilir.

GetIsInDesignMode metoduna parametre olarak herhangi bir **UIELEMENT** verdığınızda size Design modunda olunup olunmadığına dair Boolean bir değer döndürüyor.

Hepinize kolay gelsin ;)

Daron YÖNDEM

www.Daron.Yondem.com

Volkan ALBAYRAK

www.volkanalbayrak.blogspot.com