EĞİTİM :

ASP.NET AJAX

Bölüm :
ASP.NET AJAX'a Giriş

Konu :

ASP.NET AJAX'a Giriş

Internet'in hızlanması ve tarayıcı özelliklerinin gelişmesi ile birlikte yazılım dünyası gerek dağıtma kolaylığı gerekse bakım kolaylığı nedeni ile web tabanlı uygulamalara doğru yönelmeye başladı. ASP.NET ile birlikte Web tabanlı uygulamalar dünya çapında artmaya başladı ve pek çok kurum projelerini ASP.NET ile geliştirme yoluna gitti, çünkü ASP.NET, yazılım geliştiricilere pek çok yenilik ve pek çok kolaylık sağlamaktaydı. Web tabanlı uygulamaların artması ile beraber, uygulamaların dezavantajları da ortaya çıkmaya başladı ve kullanıcılar bu dezavantajları gündeme getirmeye başladı. Peki bu dezavantajlar nelerdi? Olaya son kullanıcı penceresinden bakıldığında Web tabanlı uygulamaların kullanımı pek kolay olmamaktaydı. Örneğin DropDownList'ten seçilen bir şehrin ilçelerinin başka bir DropDownList'te yüklenmesi için sayfa olduğu gibi yenileniyor ve kullanıcı odaklandığı noktayı kaybediyordu veya borsadaki hisse senedinin değerini takip eden bir kullanıcı sayfa yenilendiğinde hisse senedinin yerini kaybedecek ve belki de ciddi anlamda bir maddi kayba uğrayacaktır. Böyle durumlar ve benzerleri göz önüne alındığında son kullanıcılar haksızdı denilemez. Olaya bir de yazılım geliştirici penceresinden bakılırsa web uygulamalarında asenkron işlemler yaptırmak pek kolay olmuyordu. Sayfa içerisinde gerçekleştirilecek olan en küçük bir işlemde bile, sayfa, olduğu gibi PostBack olarak sayfanın tamamı sunucuya gönderiliyor. Dolayısıyla sayfanın tamamı sunucudan geri yükleniyordu. Performans açısından pek de iç açıcı olmayan bu gibi durumların önüne geçmek için, arka planda pek çok işlem yapılması gerekiyor. Ancak, zaman zaman gerçekleştirilen işlemler dahi yeterli olmayıp, bazı şeylerin göz ardı edilmesi gerekiyordu. Yazılım geliştiriciler teknik anlamda performans arttırıcı işlemler uygulamanın yanı sıra son kullanıcılardan gelen şikayet ve talepleri de karşılamak zorunda kalıyor ve durum bazen içinden çıkılamaz bir hal alıyordu.

Web tabanlı uygulamalardaki genel sorun, Web tabanlı uygulamaların masaüstü uygulamalarının performansının ve kullanım kolaylığının gerisinde kalmasıydı. Kullanıcılardan gelen genel talepler de bu doğrultudaydı. Kullanıcı taleplerinin dikkate alınması ile birlikte yazılım dünyasında Web tabanlı uygulamalar da yenilenmekte ve kullanıcılarını memnun etmeye başlamaktaydı. Web tabanlı uygulamalardaki yeniliklerin temelini **A**senkron **JA**vascript and **X**ml (**AJAX**) adı verilen teknoloji oluşturmaktaydı. AJAX ile adından da anlaşılacağı üzere Web tabanlı uygulamalarda asenkron işlemler yapılmasına olanak tanınmakta ve güncellenmeler sırasında sayfanın parçalı olarak güncellenebilmesi sağlanabilmektedir. AJAX kullanıcıların deneyimlerini de değiştirerek, kullanıcılara Web de alışık olmadıkları bir deneyim kazandırmaktadır. Artık borsadaki hisse senedinin değerini takip eden bir kullanıcı, anlık olarak fiyat değişimlerini animasyonlu bir şekilde görebilmekte ve en önemlisi de sayfanın tamamı sunucuya gönderilmediği için ekrandaki görüntü kaybolmamakta ve dolayısıyla kullanıcı odaklandığı yeri kaybetmemektetir. Kullanıcı memnuniyetlerinin üst seviyeye çıkmasını sağlayan AJAX, her geçen gün hayran kitlesini arttırmaktadır. Kullanıcıların memnun olduklarını gören pek çok kuruluş da kendi uygulamalarını AJAX ile güçlendirip kullanıcılarına mükemmel bir deneyim sunmaktadırlar. AJAX'ı kullanan kuruluş sayısı her geçen gün artmakla birlikte yakın zaman içerisinde AJAX kullanmayan kuruluş sayısı, sayılabilir hale gelecektir.

ASP.NET AJAX

Bir siteye AJAX desteği kazandırmak için JavaScript kodu yazmak, XmlHttpRequest nesnesinin kullanımına hakim olmak gerekmektedir. Gerekli bilgilere sahip olunduktan sonra da küçük işlemler için oldukça uzun kod yazma işlemleri yazılım geliştiricileri bekliyor olacaktır. Microsoft bu duruma el atarak yazılım geliştiricileri ASP.NET AJAX ile tanıştırdı ve basit işlemler için çok fazla iş gücü harcamanın önüne geçilmiş oldu.

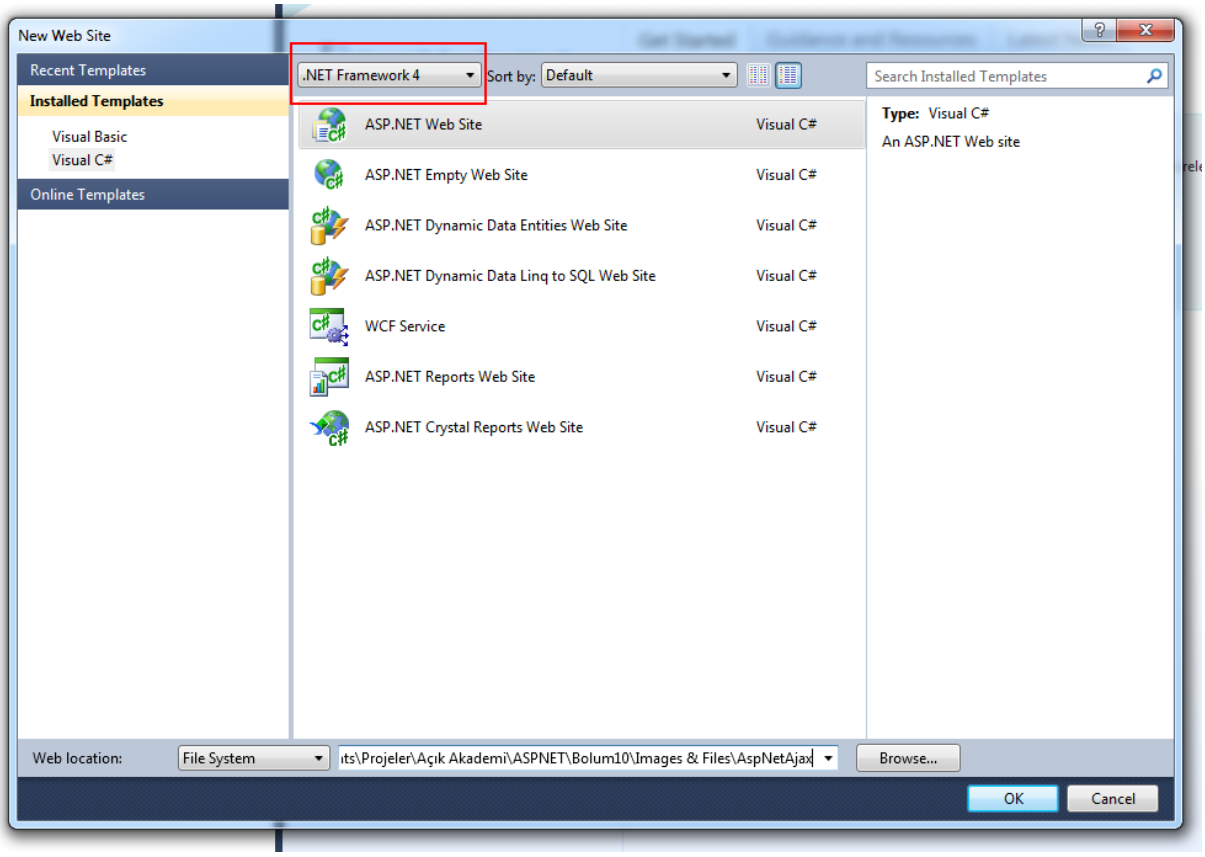
ASP.NET AJAX ile birlikte yazılım geliştiricler tarayıcı uyumluluğunu kontrol etmek zorunda değillerdir. Çünkü; ASP.NET AJAX'ın sunmuş olduğu altyapıda yer alan script kütüphanesinde kontrollerin gerçekleştirilmesi için tüm kodlar yazılmış durumdadır ve ASP.NET AJAX ile sunulan diğer nesneler kullanıldığında zaten uyumuluk kontrolü gerçekleştirilmektedir.

ASP.NET AJAX ile sunulan kontrollerle, tek satır kod yazmadan kontrolleri sürükleyip bırakarak asenkron olarak işlem yapabilen ve kullanıcı deneyimlerini arttıran uygulamalar geliştirmek oldukça kolay bir hal almıştır.

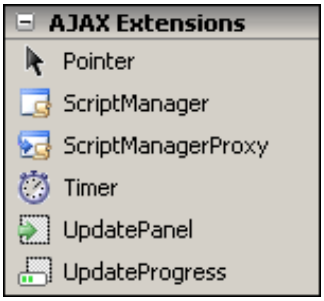
ASP.NET'in 2.0 ve daha üst versiyonları için kullanılabilen ASP.NET AJAX'ı Visual Studio 2005 ile birlikte kullanmak için ASP.NET AJAX'ın resmi web sitesi olan http://www.asp.net/ajax/ adresi ziyaret edilip Downloads bölümünden ASP.NET AJAX indirilip kurulmalıdır. ASP.NET 3.5 ve daha yeni sürümler, Visual Studio 2008/2010 veya Visual Studio'nun daha yeni versiyonları ile birlikte AJAX uygulaması geliştiriliyorsa, ekstra işlem yapmaya gerek olmaksızın zaten uygulamalarda ASP.NET AJAX kullanılabiliyor olacaktır.

Visual Stuio 2008/2010 veya daha yeni bir Visual Studio sürümünde geliştirme yapıyorsanız ASP.NET AJAX'ı kullanmanız için herhangi bir ek yazılım kurmanıza gerek yoktur. ASP.NET'in 3.5 sürümünden itibaren ASP.NET'in bir parçası haline gelen ASP.NET AJAX, Visual Studio ortamında .NET Framework 3.5 veya daha güncel bir sürüm üzerinde web projesi açıldığında aktif ve kullanılabilir olarak karşınıza çıkacaktır.

ASP.NET AJAX'ın aktif olduğu bir proje geliştitmek istiyorsanız, Visual Studio ortamında yeni web sitesi projesi açarken Framework 3.5 veya daha yeni bir sürüm olduğuna dikkat edin. ASP.NET AJAX'ın aktif olarak kullanıldığı bir proje açarken ekran aşağıdaki gibi olmalıdır.



ASP.NET AJAX'ın aktif olduğu bir proje açtığınızda araç kutusunda yeni bir sekmenin eklendiğini fark edeksiniz. Eklenen sekme ASP.NET AJAX tarafından sunulan kontrolleri içermektedir.



ASP.NET AJAX ile birlikte yazılım geliştiricilere toplam beş adet yeni kontrol sunulmaktadır. Bu kontrollerden **ScriptManager** ve **ScriptManagerProxy** sayfa içerisinde bulunan diğer kontrollerin kullandıkları scriptleri yönetmekle sorumlu iken, **UpdatePanel**, parçalı güncelleme işleminin gerçekleştirilmesine olanak tanır. **UpdateProgress**, sayfanın bir alanı güncellenirken görüntülenip güncelleme işlemi bittikten sonra kaybolarak, "Güncelleniyor..." tarzı bilgilerin kullanıcıya belirtilmesine olanak tanır. **Timer** ise web dünyasında eksikliği hissedilen belli zaman aralıklarında bir takım işlemleri gerçekleştirmeye olanak tanıyarak bu alandaki soruna çözüm bulmuştur.

EĞİTİM :

ASP.NET AJAX

Bölüm :
Kontroller İle Çalışmak

Konu :
Kontroller İle Çalışmak

ScriptManager

ScriptManager, sayfada bulunan istemci taraflı scripteleri yönetmekle görevlidir. Sayfa içerisine kullanılacak olan Script kütüphanelerini sayfaya yükleyerek diğer kontrollerin bu scriptleri kullanmasına olanak tanır. ScriptManager kontrolü sayfa içerisinde gerçekleşecek olan parçalı güncelleme olaylarının da gerçekleştirilmesini sağlayacak olup sayfa içerisinde bulunan parçalı güncelleme olaylarında aktif görev üstlenen UpdatePanel kontollerini de yönetir.

Sayfa içerisinde herhangi bir ASP.NET AJAX kontrolü kullanılmadan önce ScriptManager kontrolü mutlaka eklenmiş olarak bulunmaldır. Çünkü; Script Manager, AJAX Kontrollerinin yönetimi ve çalışmasından sorumludur. Sayfaya ScriptManager kontrolü eklendikten sonra özelliklerine göz atılacak olursa **EnablePartialRendering** isimli bir özellik ile karşılaşılacaktır. Varsayılan değeri True olarak ayarlanmış olan EnablePartialRendering özelliği, sayfada parçalı güncelleme işleminin aktif olup olmayacağını belirler. ScriptManager'ın özelliklerinden **AsyncPostBackTimeout** özelliği ise asenkron çağrılardaki zaman aşımı süresinin saniye cinsinden belirlenebildiği özelliktir.

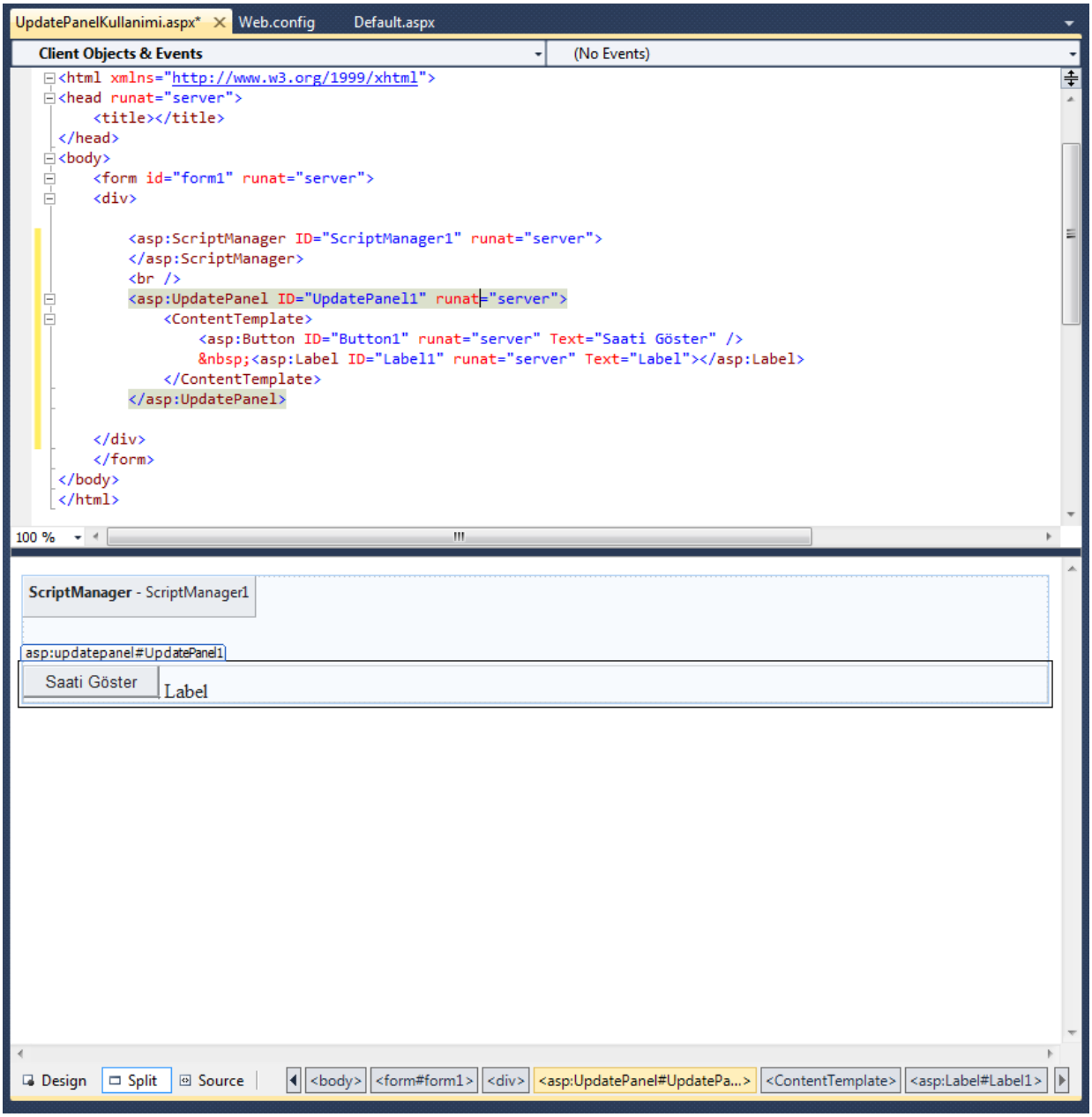
ScriptManagerProxy

Bir sayfaya sadece bir tane ScriptManager kontrolü eklenebilir ancak ScriptManager MasterPage'e eklendiği ve MasterPage'den türetildiği bir durumda türeyen sayfada ScriptManager'a bir özellik eklenmek istendiğinde ne olacaktır. Bu gibi durumları ele almak için ScriptManagerProxy adındaki kontrol kullanılıp sayfaya özgü ayarlar gerçekleştirilebilir.

UpdatePanel

UpdatePanel kontrolü sayfayı bölümlendirip parçalı güncelleme işlemi gerçekleştirilmesine olanak tanıyan kontroldür. UpdatePanel kontrolü ASP.NET sunucu kontrolleri arasında yer alan Panel kontrolüne benzetilebilir. UpdatePanel kontrolü tıpkı Panel kontrolü gibi diğer kontrolleri taşıyabilir ancak gerek yapısı gerekse kullanım amacı ile Panel kontrolünden farklıdır. UpdatePanel'deki amaç sayfayı bölümlendirip, her bölümün güncellemesini ayrı ayrı gerçekleştirebilmektir. UpdatePanel kontrolünün içerisine eklenen kontroller UpdatePanel'in **ContentTemplate**'i içerisinde yer alır. ContentTemplate bölümü UpdatePanel'in içeriğinin belirlendiği bölümdür. Bir UpdatePanel'in ContentTemplate'ini oluşturmak için HTML kaynak kodu tarafına geçilip kod yazmak seçeneği kullanılabileceği gibi Visual Studio ortamında dizayn tarafında herhangi bir kontrol UpdatePanel'in içine sürüklenip bırakılabilir. UpdatePanel içerisine ilk bırakılan kontrolün ardından Visual Studio ContentTemplate'i oluşturup UpdatePanel içerisine bırakılan kontrolü ContentTemplate içerisine ekleyecektir.

İlk ASP.NET AJAX örneği ile UpdatePanel'i kullanmaya başlayalım. ilk olarak ASP.NET AJAX'ın aktif olduğu bir web sitesi projesi açılıp bir sayfaya **ScriptManager** kontrolü ekleyelim. Hatırlayacağın üzere diğer AJAX kontrollerinin çalışması için ScriptManager kontrolünü kullanılacak olan diğer ASP.NET AJAX kontrollerinden önce-daha üste- sayfaya eklemek gerekiyordu. ScriptManager kontrolünün eklenmesinin ardından sayfa içerisine bir tane UpdatePanel eklenip, UpdatePanel içerisine bir tane Label ve bir tane de Button kontrolü ekleyelim. Gerekli işlemlerin ardından sayfanın görünümü aşağıdaki gibi olmalıdır.



Visual Studio tarafından oluşturulan kodlar ise aşağıdaki gibidir. Görüleceği üzere ContentTemplate oluşturulmuş ve kontroller oluşturulan ContentTemplate içerisine eklenmiştir.

```
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                    Text="Saati Göster" />
                 <asp:Label ID="Label1" runat="server"
Text="Label"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
```

Şu anki düşünülen senaryoda Button'a tıklanıldığında Label'a o anki sistem saatinin yazdırılması hedeflenmektedir. Bu sebeple Button'un click olayı ele alınıp oluşturulan metoda aşağıdaki kodları yazalım.

```
    protected void Button1_Click(object sender, EventArgs e)
    {
```

```
        Label1.Text = DateTime.Now.ToString();
    }
```

Görüleceği üzere yazılan kodlar daha önceki örneklerde kullanılan kodlardan farksızdır. Kodlarda sistem saati Label kontrolüne yazılmaktadır. Kullanıcılar Button'a tıkladıklarında sayfanın tamamı PostBack olmayacak sadece UpdatePanel'in içi sunucuya gönderilip gerekli işlemlerin ardından bu alanın sonucu geri getiriliyor olacaktır. Daha önceki deneyimler göz önüne alındığında Button kontrolüne tıklama işleminin ardından genel beklenti sayfanın tamamının PostBack olup daha sonra Label'da sistem saatini görmek olacaktır ancak şu anki örnekte böyle bir şey olmayacaktır. Sadece UpdatePanel'in içeriği yenilenecektir. Durumun böyle olduğunun son kullanıcı penceresinden bakıldığında en önemli kanıtı, Internet Exploerer'ın ProgressBar'ında herhangi bir hareket olmamasıdır. Bahsettiğimiz durumu görmek için hemen yazdığın uygulamayı çalıştırıp Button'a tıklayabilirsin. Nasıl? Artık sen de günümüzde pek çok sitede bulunan teknoloji ile kod geliştirmeye başladın...

Proje çalıştırıldığında saatin gösterildiği alanlar değişecek ve ProgressBar'da herhangi bir değişiklik olmayacaktır. Yani daha doğrusu tekrar vurgulamak gerekirse sayfa tamamen PostBack olmayacaktır. Yukarıdaki gibi bir örnekte kullanıcı memnuniyeti arttırılarak sayfanın görünümü bir saniyeliğine bile olsa kaybolmuyor ve kullanıcı beyaz ekranla karşı karşıya kalmıyor. Kullanıcı sadece Button'a tıkladığında değişmesini beklediği alana odaklanıyor ve diğer etkenler nedeni ile dikkat dağınıklığı oluşmuyor.

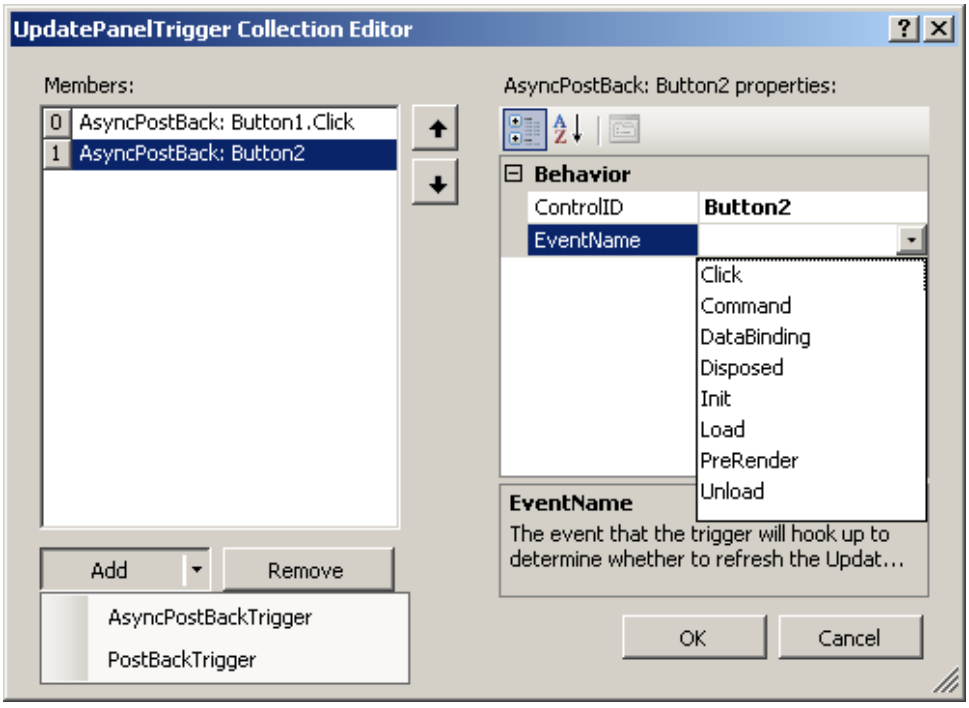
Örnek tekrar göz önüne getirilecek olursa UpdatePanel içerisine eklenen Button'a tıklanıldığı zaman asenkron bir isteğin başlatılığı hatırlanacaktır. Bunun nedeni UpdatePanel'in **ChildrenAsTriggers** özelliğinin True olarak atanmaış olmasıdır. ChildrenAsTriggers özelliği UpdatePanel içerisine eklenen herhangi bir öğenin başlatacağı çağrının UpdatePanel'in güncellenmesini tetikleyip tetiklemeyeceğini belirtir, eğer özellik True olarak atanırsa ekstra bir işlem yapılmasına gerek olmadan UpdatePanel içerisindeki kontrollerin olaylarını tetikliyor olmak yeterlidir.

Triggers Koleksiyonu

UpdatePanel'in içeriğinin güncellenmesini sağlamak için güncelleme işlemini başlatacak olan kontroller ve kontrollerin olayları UpdatePanel'in Trigger koleksiyonuna eklenmelidir. Trigger koleksiyonuna eklenen kontrollerin belirtilen olayları tetiklendiği anda UpdatePanel'in içeriği de güncellenmeye başlar ve olayların, olay yakalayıcılarına yazılan kodlar işletilir.

UpdatePanel içerisine eklenen kontrollerin tetikleyici olarak Trigger koleksiyonuna eklenmesi mecburi değildir. ChildrenAsTriggers özelliği True olarak ayarlandığında zaten UpdatePanel içerisinde bulunan kontroller içeriğin güncellenmesini başlatıyor olacaktır.

Trigger koleksiyonuna eklenenen kontrollerin hem asenkron çağrı başlatması hem de sayfayı olduğu gibi sunucuya gönderip senkron çağrı başlatması mümkün olmaktadır. UpdatePanel'in özelliklerinden Triggers'a tıklanılıp açılan diyalog penceresi aracılığı ile yeni bir Trigger eklenebilir. UpdatePanel'in Trigger koleksiyonun listeleneceği diyalog penceresi aşağıda yer almaktadır.



Yukarıdaki pencere kullanılarak soldaki **Add** butonu ile hem asenkron hem de senkron tetikleyici eklenebilir. UpdatePanel'e trigger eklerken yukarıdaki resimde de görülen sağ taraftaki Behavior bölümü aracılığı ile güncelleme işlemini başlatacak olan kontrol seçilir. **ControlID** özelliği ile güncelleme işlemini gerçekleştirecek olan kontrol **EventName** ile de kontrolün hangi olayı tetiklendiğinde güncellenme işleminin başlatılacağı belirlenir.

UpdatePanel'i Dışarıdaki Bir Kontrol İle Güncellemek

UpdatePanel'in içeriği UpdatePanel içerisinde bulunan bir kontrol aracılığı ile güncellenebileceği gibi Trigger eklenerek UpdatePanel dışında yer alan bir kontrol ile de güncellenebilir. Durumu küçük bir örnekle açıklamak için daha önceki örnekte yer alan UpdatePanel'in hemen altına yani UpdatePanel'in dışına bir adet Button atılıp, UpdatePanel'in Trigger koleksiyonuna da eklenilen Button asenkron Trigger olarak eklendikten sonra UpdatePanel'in kodları aşağıdaki gibi olacaktır.

```
    <form id="form1" runat="server">
    <div>

        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <br />
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                    Text="Saati Göster" />
                 <asp:Label ID="Label1" runat="server"
Text="Label"></asp:Label>
            </ContentTemplate>
            <Triggers>
                <asp:AsyncPostBackTrigger ControlID="ButtonTetikle"
EventName="Click" />
            </Triggers>
        </asp:UpdatePanel>

    </div>
    <asp:Button ID="ButtonTetikle" runat="server" Text="Saati Güncelle"
        onclick="Button1_Click" />
    </form>
```

Yukarıdaki kodlarda yer alan ButtonTetikle isimli Button'a tıklanıldığı zamanda asenkron olarak Label'a o anki saat yazılıyor olacaktır. Burada ButtonTetikle isimli Button'un Click olayına da Button1'in Click olayı için hazırlanan metodun bağlanmış olduğunu fark etmiş olmalısın.

Sayfa İçerisinde Birden Fazla UpdatePanel Kullanımı

Bir sayfa içerisinde birden fazla UpdatePanel kontrolü kullanmak için sayfa içerisine birkaç tane UpdatePanel kontrolü sürükleyip bırakıp içlerini doldurmak yeterli olacaktır ancak gerek performans gerek de kullanılabilirlik için birkaç işlem daha gerçekleştirmek gerekmektedir. Sayfa içerisinde birden fazla UpdatePanel kullanıldığında bilinmesi gereken en önemli şey UpdateMode özelliğidir. **UpdateMode** özelliği **Always** ve **Conditional** olmak üzere toplam iki adet değer almaktadır. Varsayılan değeri Always olan UpdateMode özelliği UpdatePanel'in içeriğinin sayfada herhangi bir güncelleme olayı olduğu zaman mı yoksa sadece Trigger Koleksiyonu içerisinde belirtilen olaylardan biri veya UpdatePanel içindeki kontrollerden birinin bir olayı tetiklendiği andamı güncelleneceği belirleniyor. Adından da anlaşılacağı üzere Always her zaman, Conditional ise Triggers Koleksyonu içerisindeki bir olay tetiklendiği anda güncelleneceğini belirtmektedir. Durumu örneklemek için bir sayfa içerisine ScriptManager'ın ardından iki adet UpdatePanel ve içlerine de birer tane Button ve Label ekleyelim. Button'ların her ikisinin de gerçekleştireceği işlemler aynıdır ve her ikiside iki Label'a da o anki tarihi yazıyor olacaktır. Gerekli işlemler gerçekleştirildikten sonra sayfanın içeriği aşağıdaki gibi olacaktır.

```
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <br />
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button1" runat="server" Text="Saat'i Güncelle"
OnClick="Button1_Click" />
                 
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
        <br />
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button2" runat="server" Text="Saati Güncelle"
OnClick="Button1_Click" />
                 
                <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
```

Yukarıdaki kodlarda yer alan Button'lara tıklanıldığı zaman devreye girip çalışacak olan Button1_Click isimli metodun içeriği de aşağıdaki gibidir.

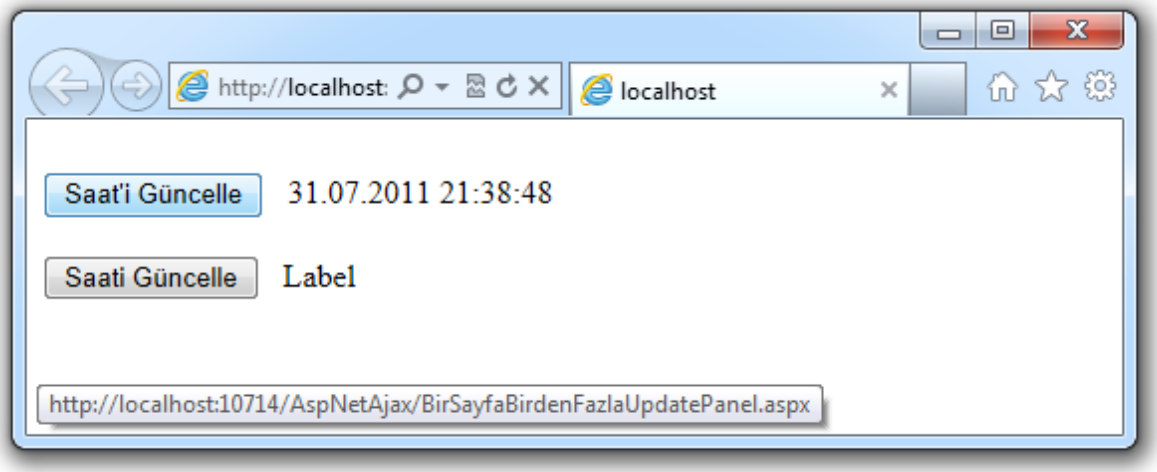
```
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = DateTime.Now.ToString();
        Label2.Text = DateTime.Now.ToString();
    }
```

Kodlardan görüleceği üzere her iki Label'a da sistem saati yazdırılmaktadır. Sayfa içerisinde yer alan UpdatePanel'lerin ikisi de varsayılan ayarlarında olduğu için bir UpdatePanel'de bir işlem gerçekleştirilirken diğeride güncelleniyor olacaktır. Yani başka bir deyişle bir UpdatePanel içindeki Button'a tıklanıldığı zaman her iki Label'a aynı değer yazılıyor olacaktır. Bu durum şu andaki senaryoda bir problem oluşturmuyor gibi görünse

de olaya performans açısından bakıldığında çok hoş olmayan bir durumdur. Bir sayfada on – onbeş UpdatePanel kullanıldığı düşünüldüğünde birinin içinde gerçekleştirilecek bir işlem için diğerlerininde güncellenmesi demek, sayfanın olduğu gibi güncellenmesi anlamına gelebilir. Böyle bir durumda da AJAX kullanmanın fazla bir esprisi kalmayacaktır. Bu sebeple UpdatePanel'lerin UpdateMode özellikleri gerekmedikçe Always olarak bırakılmamalıdır. Yukarıdaki örnekte altta yer alan UpdatePanel2 isimli UpdatePanel'in UpdateMode özelliği Conditional olarak ayarlanıp aşağıdaki hale getirildikten sonra UpdatePanel2 sadece içerisindeki Button'a tıklanıldığı zaman güncelleniyor olacaktır.

```
...
<asp:UpdatePanel ID="UpdatePanel2" UpdateMode="Conditional" runat="server">
...
```

UpdatePanel2'in UpdateMode özelliği, Conditional olarak ayarlandıktan sonra üstte UpdatePanel1 içerisinde yer alan Button'a tıklandığında sadece UpdatePanel1'in içeriğinin güncellendiği ancak alttaki Button'a tıklandığında ise her iki UpdatePanel'in de güncellendiği görülecektir çünkü UpdatePanel1'in UpdateMode özelliği hala varsayılan değer olan Always olarak ayarlı durumdadır.

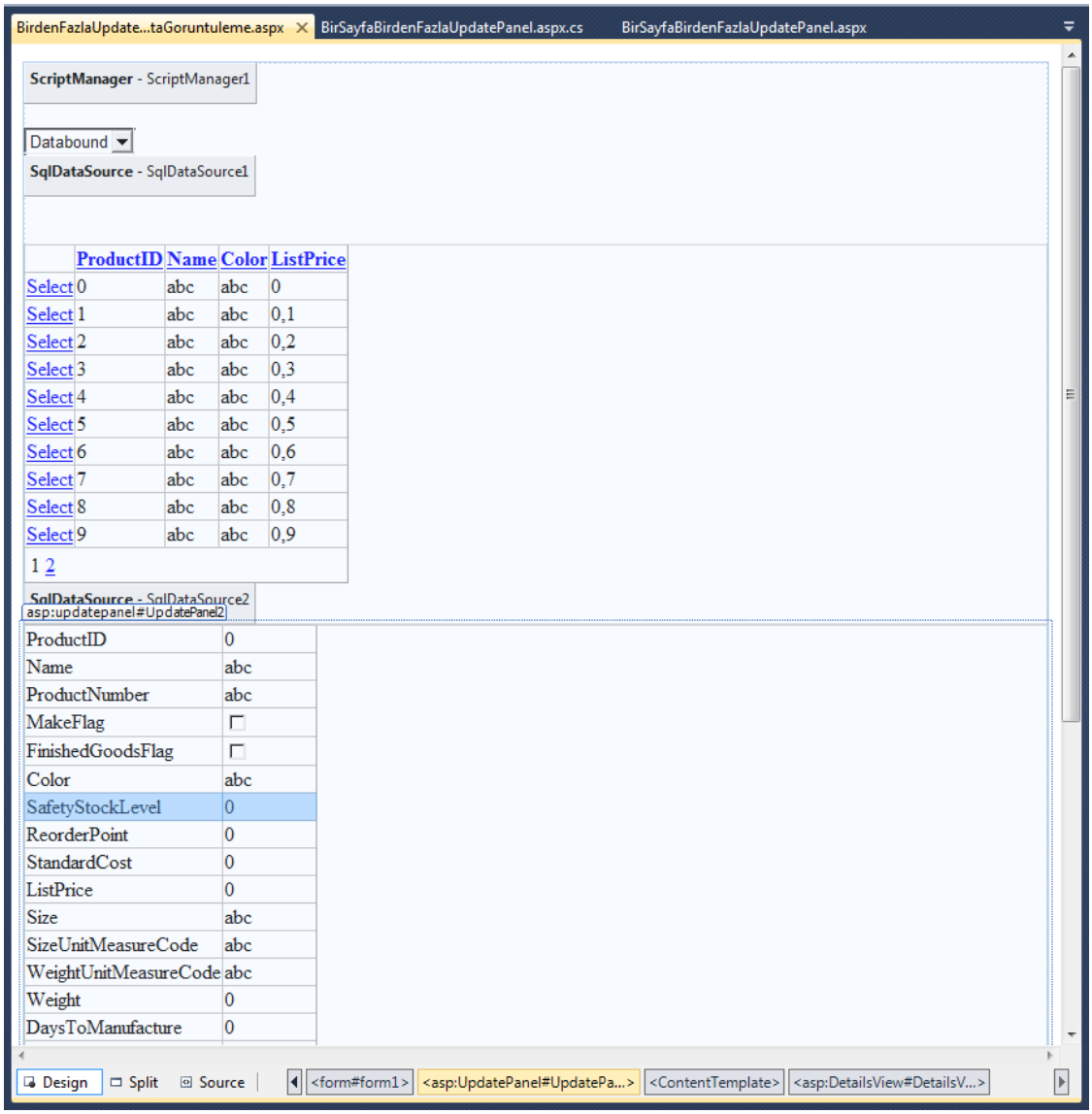


Sayfada UpdatePanel'lerin dışına bir alana, bir tane de Label ekleyip Button'lara tıklandığında yeni eklenen Label'ın Text özelliğine de sistem saati yazdırılmak istenildiğinde Button'lara tıklanıldığında yeni eklenen yani UpdatePanel'ler dışında yer alan Label'da bir değişiklik olmadığı gözlemlenecektir çünkü asenkron çalışan alanlar sadece asenkron çalışan alanlara müdahele edebilir. Sayfaya yeni eklenen Label'ın Text özelliğinin de güncellenebilmesi için sayfanın PostBack olması gerekmektedir.

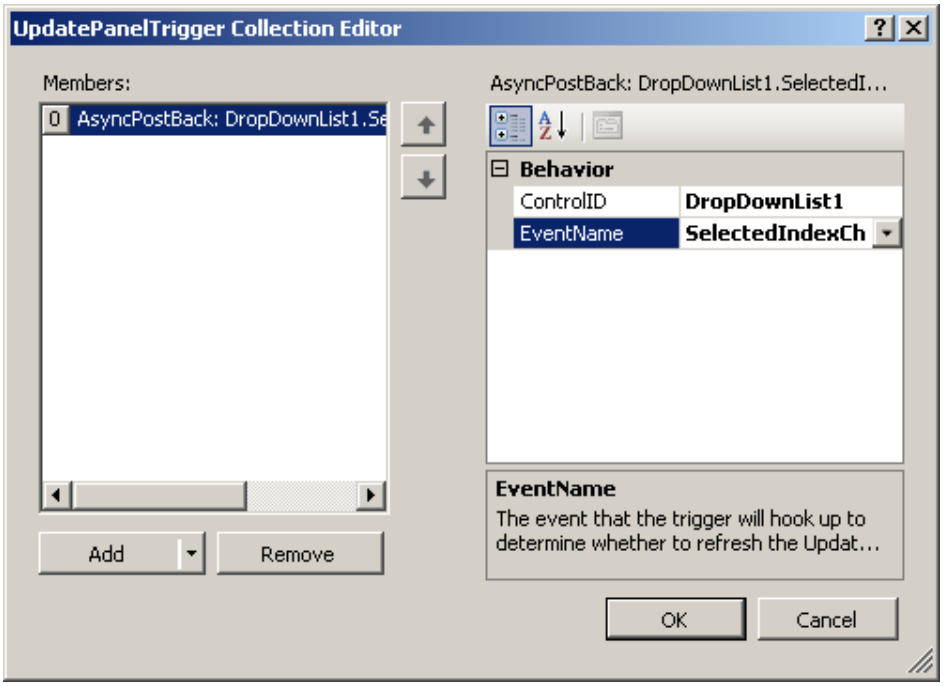
Asenkron alanlardan sadece asenkron çalışan alanlar içerisinde yer alan kontrollere müdahele edilebilir. UpdatePanel'ler dışında yer alan diğer kontrollerin özelliklerinin çalışma zamanında değiştirilmesi için sayfanın PostBack olması gerekmektedir.

Sayfa içerisinde birden fazla UpdatePanel kullanımını gerektirecek durumlardan biri de master – detay formları olarak gösterilebilir. Daha gerçekçi bir örnek olması adına AdventureWorks veritabanında yer alan alt kategoriler, ardından bu alt kategorilerin altında yer alan ürünler ve seçilen ürünlerin detaylarının gösterileceği bir form tasarlayalım. Alt kategoriler bir DropDownList kontrolü içerisinde ve seçilen alt kategori altında yer alan ürünler de GridView kontrolü içerisinde gösterilecektir. GridView kontrolünde seçilen ürünün detayı ise DetailsView kontrolü içerisinde yer alıyor olacaktır. Bu senaryoda iki adet UpdatePanel yeterli olacaktır. İlk UpdatePanel, GridView kontrolünü taşıyacak olup sadece DropDownList kontrolünün seçili olan değeri değiştiğinde güncellenecektir. İkinci UpdatePanel ise DetailsView kontrolünü taşıyacak olup sayfada herhangi

bir değişiklik olduğunda devamlı güncelleniyor olacaktır. Sayfanın genel tasarımı aşağıdaki resimdeki gibi olacaktır.



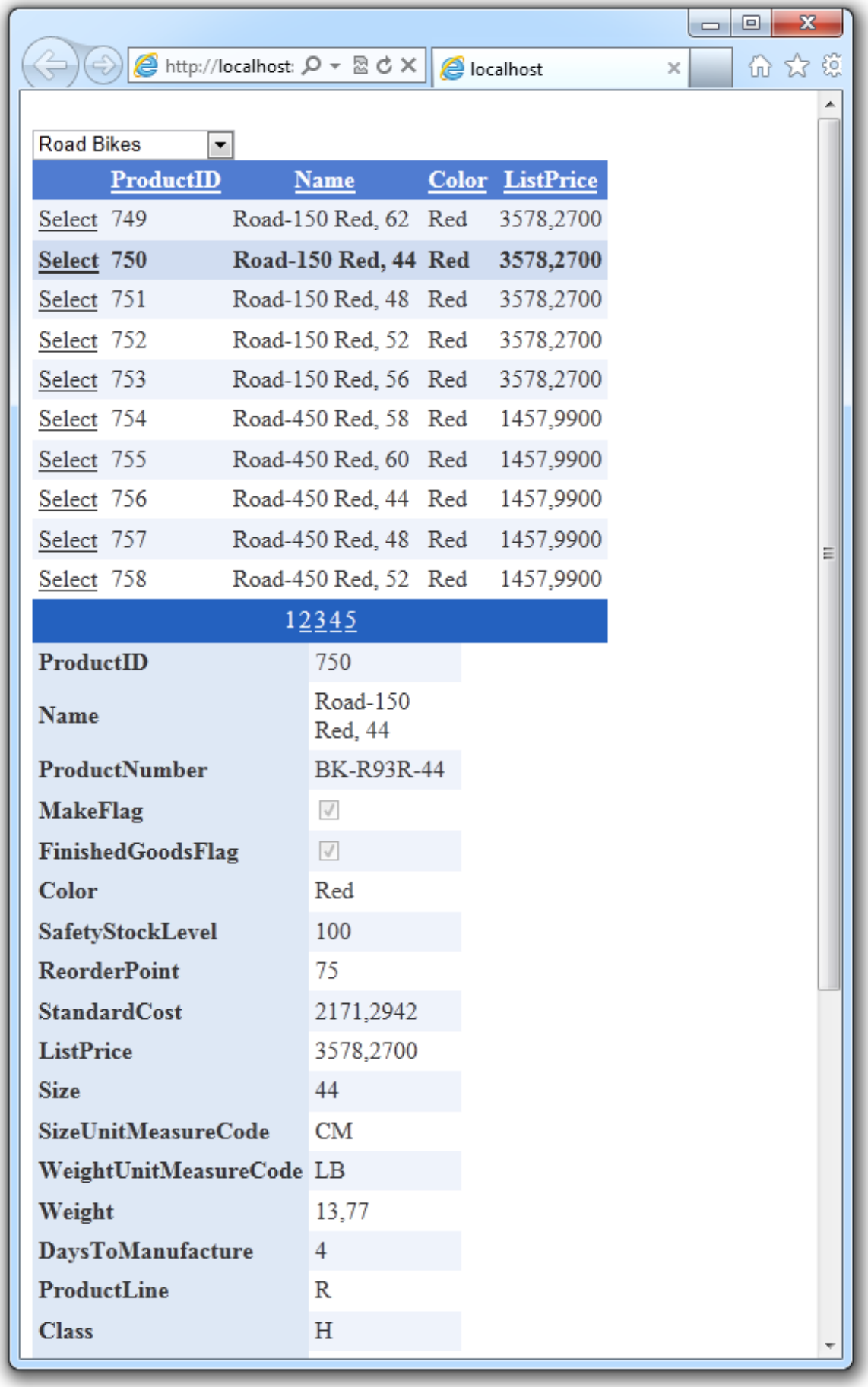
En üstte DropDownList yer alıyor, onun altında bir UpdatePanel içerisinde GridView ve en alta da başka bir UpdatePanel içerisinde DetailsView yer alıyor. Şu andaki durumda SqlDataSource kontrolü ve bu kontrolün avantajları kullanılarak kod yazmadan hedeflenen işlemler gerçekleştirilmektedir. Ancak, DropDownList'e tıklandığı zaman, sayfa PostBack olarak istenilen işlemleri gerçekleştirmektedir. Çünkü DropDownList herhangi bir UpdatePanel içerisinde yer almıyor. Sayfa yukarıdaki gibi tasarlandıktan sonra, sıra geldi UpdatePanel'lerin ayarlarını gerçekleştirmeye. GridView'i taşıyan ilk UpdatePanel'e Trigger eklenerek işleme başlanılabilir. UpdatePanel'in özellikler penceresinden Triggers'a tıklanıp açılan pencere aracılığı ile de aşağıdaki resimde görüldüğü gibi DropDownList'in SelectedIndexChanged olayı, ilk UpadatePanel'e Trigger olarak eklenmektedir.



Şimdi sıra geldi, bu UpdatePanel'in UpdateMode özelliğini Conditional olarak ayarlamaya. Bu UpdatePanel sadece DropDownList'te seçili olan değer değiştiğinde ve kendi içerisinde yer alan GridView'in herhangi bir olayı tetiklendiğinde güncellensin isteniliyor. Bu sebeple UpdatePanel'in UpdateMode özelliği Conditional olarak ayarlanmalıdır. Sayfanın tasarımının görüntülendiği resme dikkat edildiğinde GridView üzerinden seçim, sıralama ve sayfalama yapılabileceği farkedilmiş olmalıdır. Yani kullanıcıya sadece on adet veri görüntülenecek ve bunlardan birinin seçilmesi sağlanacaktır. Tabi kullanıcı sayfa değiştirip diğer verilere erişmekte özgürdür.

DetailsView kontrolünü taşıyan ikinci UpdatePanel'in UpdateMode özelliği ise Always olarak bırakılmalıdır. Çünkü; Hem DropDownList'te seçili olan değer değiştiğinde, hem de GridView üzerindeki bir olay tetiklendiğinde seçili olan değer değişeceği için bu UpdatePanel'in içeriği mutlaka güncellenmelidir. Bu sebeple ikinci UpdatePanel üzerinde herhangi bir işlem gerçekleştirilmiyor.

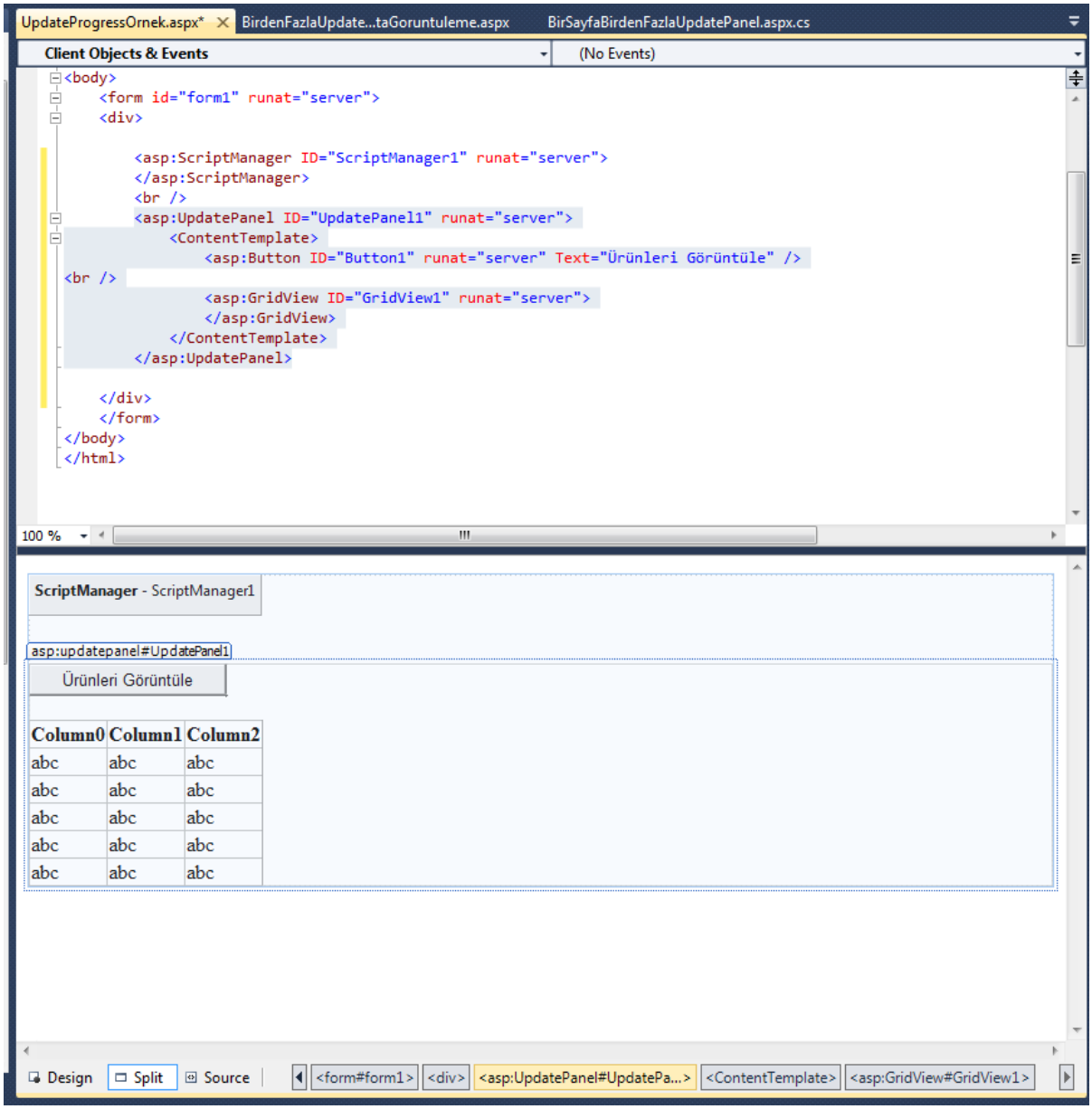
Web tabanlı uygulamalarda, tasarım da çok önemli olduğu için GridView ve DetailsView kontrollerini varsayılan tasarımları ile bırakmayıp SmartTag'lerinden Auto Format bağlantısı aracılığıyla birer tane birbirini tamamlayacak sitil seçilip uygulama çalıştırıldığında aşağıdaki görünüm ile karşılaşılıyor olacaktır.


http://localhost:
localhost
Road Bikes
ProductID
Name
Color
ListPrice
Select 749 Road-150 Red, 62 Red 3578,2700
Select 750 Road-150 Red, 44 Red 3578,2700
Select 751 Road-150 Red, 48 Red 3578,2700
Select 752 Road-150 Red, 52 Red 3578,2700
Select 753 Road-150 Red, 56 Red 3578,2700
Select 754 Road-450 Red, 58 Red 1457,9900
Select 755 Road-450 Red, 60 Red 1457,9900
Select 756 Road-450 Red, 44 Red 1457,9900
Select 757 Road-450 Red, 48 Red 1457,9900
Select 758 Road-450 Red, 52 Red 1457,9900
1 2 3 4 5
ProductID 750
Name Road-150 Red, 44
ProductNumber BK-R93R-44
MakeFlag
FinishedGoodsFlag
Color Red
SafetyStockLevel 100
ReorderPoint 75
StandardCost 2171,2942
ListPrice 3578,2700
Size 44
SizeUnitMeasureCode CM
WeightUnitMeasureCode LB
Weight 13,77
DaysToManufacture 4
ProductLine R
Class H

UpdateProgress

Ajax uygulamalarında sayfa PostBack olmadığı için, kullanıcılara, güncellenmekte olan alanın güncellendiği bilgisini belirtmek gerekmektedir. Sayfada bir hareket olmadığı için kullanıcılar işlemlerinin gerçekleşmediğini düşünüp birkaç kere daha aynı işlemi tekrarlamak isteyebilirler. Bu gibi durumların önüne geçmek için UpdateProgress kontrolü kullanılabilir. UpdateProgress kontrolü, bağlandığı UpdatePanel güncellenmeye başladıktan belirli bir zaman sonra görüntülenir ve güncelleme işleminin sona ermesinin ardından ekrandan kaybolur. UpdateProgress içerisinde "Lütfen Bekleyiniz..." tarzı yazılar kullanılabileceği gibi, pek çok yazılım geliştirici hareketli resimleri tercih ederek kullanıcı deneyimini üst seviyeye taşıyacak olan modelleri tercih etmektedir.

UpdateProgress kontrolünün kullanımını örneklemek için yeni bir sayfa oluşturalım. Sayfaya ScriptManager'ın ardından bir tane UpdatePanel eklendikten sonra, UpdatePanel içerisine bir adet GridView ve bir adet'te Button kontrolü ekleyelim. Button'a tıklanıldığında gerçekleştirilecek işlem son derece basittir. AdventureWorks veritabanında bulunan Production isimli tablodaki –eğer sisteminizde AdventureWorks yoksa farklı bir veri tabanı ile de çalışabilirsiniz- verileri getirmek. Tasarladığımız form aşağıdaki gibi görünüyor olmalıdır.



Button'un Click olayı tetiklendiğinde çalışacak olan kodlar ise aşağıda yer almaktadır. Burada en dikkat etmen gereken nokta kalın punto ile belirttiğimiz bölümdür.

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlConnection baglanti = new SqlConnection();
        baglanti.ConnectionString = @"Data Source=.;
                                    Initial Catalog=AdventureWorks;
                                    Integrated Security=True";
        SqlCommand komut = new SqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "Select ProductID,Name,ListPrice FROM
Production.Product";

        DataTable dt = new DataTable();
        SqlDataAdapter adapter = new SqlDataAdapter(komut);

        //Uygulama 5 saniye bekletiliyor!
        System.Threading.Thread.Sleep(5000);
```

```
        adapter.Fill(dt);

        GridView1.DataSource = dt;
        GridView1.DataBind();
    }
```

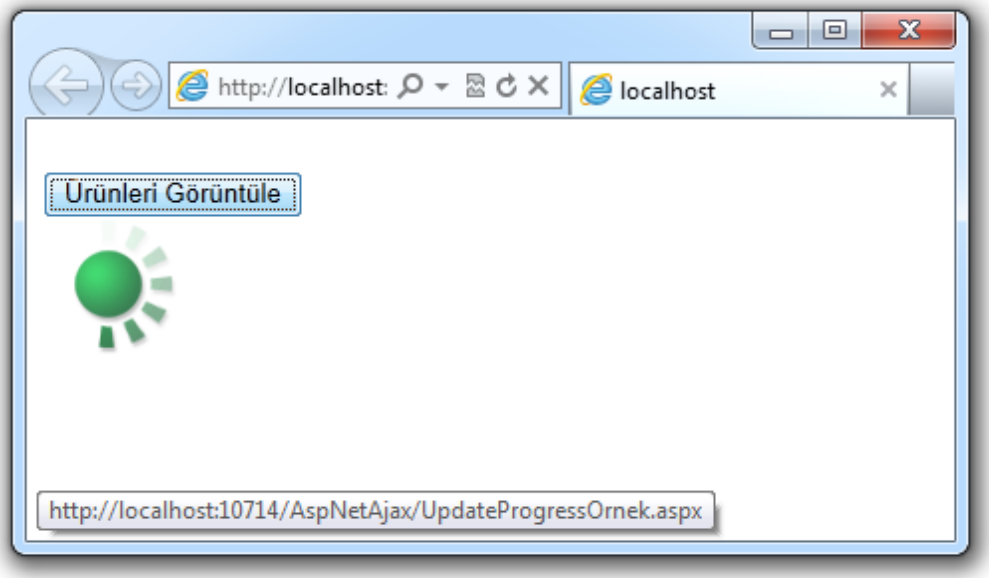
Yukarıdaki kodların çalışabilmesi için **System.Data** ve **System.Data.SqlClient** adres uzayının sayfaya referans edilmiş olması gerekmektedir.

Kodlar, daha önce "Veriye Erişim" bölümünde ele alınan kodlar ile benzerlik göstermektedir. Ancak, bu bölümde, yukarıdaki kod bloğunda kalın puntolar ile gösterilen uygulamayı beş saniye bekletecek ekstra bir kod daha yer almaktadır. Bu kodun amacı uzun süren bir işlemi simüle etmektir. Sayfa bu şekildeyken çalıştırıldığında Button'a tıklanıldıktan sonra beş saniye içinde ekranda herhangi bir hareket olmayacaktır ve veriler getirildikten sonra aniden ekrana basılacaktır. Standart bir son kullanıcı normal bir durumda bu beş saniye boyunca beklemeyecek ve aynı Button'a birkaç defa daha basacaktır. Kullanıcıyı bilgilendirmek için UpdateProgress kontrolü kullanılmalıdır. UpdateProgress kontrolü aşağıdaki görüldüğü gibi kullanılabilir.

```
<form id="form1" runat="server">
    <div>

        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <br />
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                    Text="Ürünleri Görüntüle" />

<br />
<asp:UpdateProgress ID="UpdateProgress1" runat="server"
      AssociatedUpdatePanelID="UpdatePanel1">
            <ProgressTemplate>
                <img alt="" src="gears.gif"
                style="width: 68px; height: 68px" />
            </ProgressTemplate>
      </asp:UpdateProgress>

                <asp:GridView ID="GridView1" runat="server">
                </asp:GridView>
            </ContentTemplate>
        </asp:UpdatePanel>

    </div>
    </form>
```

Görüleceği üzere UpdateProgress, direkt UpdatePanel'in içine eklenerek kullanılabilmektedir. Bu aşamada **AssociatedUpdatePanelID** özelliğini belirtmeye gerek yoktur. Ancak, sayfa içerisinde birkaç UpdatePanel ve dolayısıyla birkaç tane UpdateProgress varsa, bu özellik ile UpdateProgress'in hangi UpdatePanel'e bağlanacağı belirlenmiş oluyor. UpdateProgress kontrolünün kodları inecelendiğinde, ProgressTemplate adında bir şablonu olduğu görülecektir. ProgressTemplate'de diğer şablonlar gibi düşünülebilir. Yani, içerisine herhangi bir içerik alabilmektedir. Örnekteki içerikte de güncelleme sırasında bir resmin görüntülenmesi sağlanmaktadır. UpdateProgress kontrolünün önemli özelliklerinden birisi de **DisplayAfter** özelliğidir. DisplayAfter özelliği içine bir tamsayı alır ve UpdateProgress'in belirtilen UpdatePanel güncellemeye başladıktan kaç milisaniye sonra görüntüleneceği belirtilir. UpdateProgress'in **DynamicLayout** adında bir özelliği daha vardır. True ya da False değerlerinden birini alabilen bu özellik ile UpdateProgress'in sayfaya dinamik olarak mı ekleneceği yoksa yerinin

ayrılacağı mı belirtiliyor. Varsayılan değeri True olan DynamicLayout özelliği, genellikle varsayılan değerinde bırakılarak UpdateProgress için boş bir alan ayrılmasının önüne geçilmiş olur. Yukarıdaki kodlar da eklendikten sonra uygulama çalıştırıldığında aşağıdaki gibi bir görünüm ile karşılaşılıyor olacaktır.



Timer

Timer kontrolü masaüstü uygulamalarda çok fazla kullanılan bir kontroldür. Timer kontrollerindeki temel amaç bir işlemi belli zaman aralıklarında tekrarlamaktır. ASP.NET'te bir eksliklik olan Timer kontrolünün eksikliği ASP.NET AJAX ile giderildi. Artık ASP.NET tarafında da javascript kodu yazmadan belli zaman aralığında belli işlemleri tekrarlatmak mümkün. ASP.NET AJAX öncesinde yazılım geliştiriciler JavaScript kodu yazarak bu eksikliği gideriyorlardı ama artık, ekstra kod yazmaya gerek kalmadı.

Timer kontrolü ile yapılabilecek çok fazla şey vardır örneğin hisse senetlerinin değerlerini yarım dakikada bir sunucudan asenkron olarak getirebilecek bir web sitesi, oldukça işe yarar ki zaten şu anda pek çok banka bu tekniği kullanmaktadır ya da bir haber portalındaki haberler asenkron olarak yenilenip kullanıcılara daha güzel bir deneyim yaşatılabilir. Örnekler çoğaltılabilir, ancak Timer kontrolünün kullanımını örneklemek için daha önceki örneklerde de sıklıkla kullanılan Label kontrolü ile devam edelim.

Projemize yeni bir sayfa ekleyip, Timer'ı birlikte kullanalım. Timer kullanmak için sayfaya bir tane ScriptManager, ardından da bir UpdatePanel ekleyelim. UpdatePanel içerisine de bir Label ile bir tane de Timer kontrolü sürükleyip bırakalım. Timer kontrolünün **Interval** özelliği, kontrolün **Tick** olayının kaç milisaniye aralıklarla tetikleneciğinin belirtildiği özelliktir. Timer'ın Interval özelliğinde yer alan değer kadar aralıklarla tetiklenen Tick olayı ele alınıp gerekleştirilmesi istenlen işlemler oluşturulan metoda yazılır. Şu andaki hedef, kullanıcılara saati anlık olarak göstermek olsun. Bu durumda Timer'ın Interval değeri 1000 yani 1 saniye olmalıdır. Tick olayına ise bölümün başından beri alışık olunan kodları yazalım, yani anlık olarak saat bilgisini Label'a yazacak olan kodlar. Oluşturulan kodlar aşağıda yer almaktadır.

TimerKontrolu.aspx

```
<form id="form1" runat="server">
 <div>

     <asp:ScriptManager ID="ScriptManager1" runat="server">
     </asp:ScriptManager>
```

```
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                <asp:Timer ID="Timer1" runat="server" Interval="1000"
ontick="Timer1_Tick">
                </asp:Timer>
            </ContentTemplate>
        </asp:UpdatePanel>

    </div>
    </form>
```

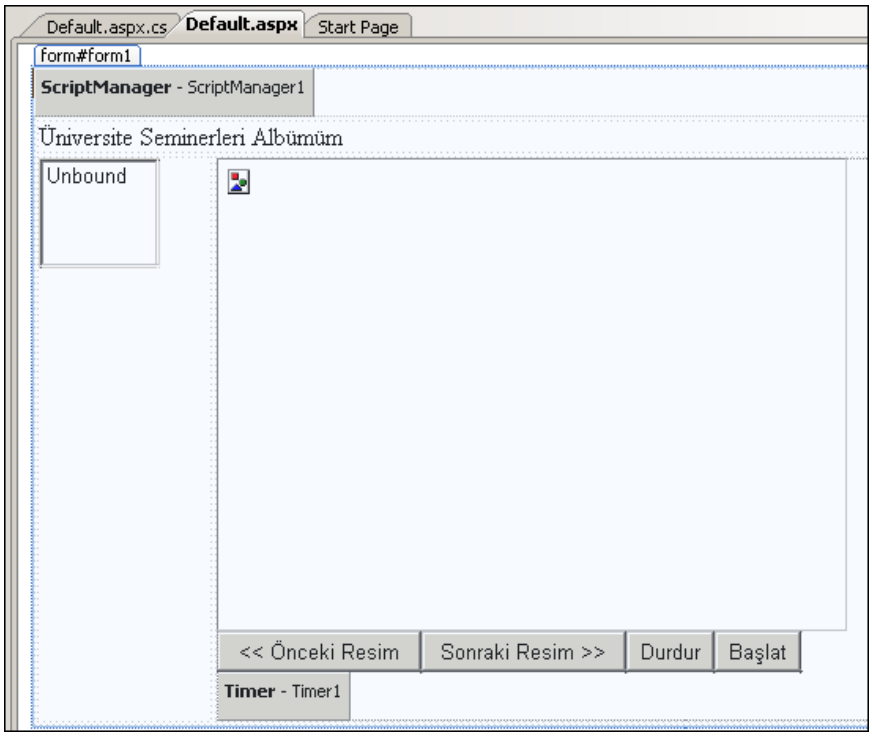
TimerKontrolu.aspx.cs

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        Label1.Text = DateTime.Now.ToString();
    }
```

Yukarıdaki kodlar çalıştırıldığında Label kontrolünün Text özelliği saniyede bir yenilenip o anki sistem saati ile değiştiriliyor olacaktır. Dolayısıyla son kullanıcılara güncel saat gösterilmiş olacaktır. Daha önce bu işlemi yapmak için bir miktar JavaScript kodu yazmak gerekmekteydi ama artık görüleceği üzere JavaScript yazmaya gerek kalmıyor.

Timer kullanılarak gerçekleştirilecek işlem sadece saat göstermek değildir. Daha önce gerçekleştirilmesi zor olan pek çok işlem Timer yardımı ile kolaylıkla gerçekleştirilebilmektedir. Yeni bir örnek ile güncel hayatta da pek çok yerde karşılaştığımız bir uygulamayı birlikte geliştirelim.

Timer kontrolü kullanılarak, güzel bir fotoğraf albümü uygulaması geliştirelim. Uygulamada, kök dizinde bulunan resimler adlı klasör içindeki, klasörler halinde bulunan resim albümlerinin içinde yer alan resimlerin, slayt gösterisi şeklinde gösterilmesini hedefliyoruz. Tasarlanan sayfada sol tarafta bir ListBox içerisinde resimler klasörü içinde yer alan albüm klasörleri listeleniyor olacaktır. Klasör adları ListBox'a çalışma zamanında dinamik olarak doldurulacaktır. Klasör isimlerini elde etmek için **System.IO** isimalanı içinde yer alan sınıflar kullanılıyor olacaktır. Resimler gösterilirken de yine ListBox içerisinde seçili olan albüme göre System.IO sınıfları kullanılarak belirtilen klasör içerisinden dinamik olarak gösterilecektir. Resimler, bir UpdatePanel içerisinde yer alan Image kontrolü içerisinde bir Timer aracılığı ile beşer saniyeliğine görüntüleniyor olacaktır. Hedeflenen amaca uygun tasarım aşağıdaki gibi gerçekleştirilebilir.



Form tasarımında solda bir adet DropDownList, sağ tarafta ise bir tane UpdatePanel ve UpdatePanel içinde de bir Image kontrolü, dört tane Button ve bir adet de Timer kontrolü yer almaktadır. Button'lar sırası ile önceki ve sonraki resme geçmek , gösteriyi durdurmak ve başlatmak için kullanılıyor olacaktır. Timer kontrolü ise her beş saniyede bir görüntülencek olan resmi değiştirecektir. Yukarıdaki tasarımı oluşturan kodlar aşağıda yer almaktadır.

```
<form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <table width="100%">
            <tr>
                <td colspan="2" align="left">
                    Üniversite Seminerleri Albümüm
                </td>
            </tr>
            <tr>
                <td width="15%" valign="top">
                    <asp:ListBox ID="listAlbumler" runat="server"
AutoPostBack="True"></asp:ListBox>
                </td>
                <td width="85%" valign="top">
                    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                        <ContentTemplate>
                            <asp:Image ID="imgResim" runat="server" Height="300px"
Width="400px" />
                            <br />
                            <asp:Button ID="btnOnceki" runat="server" Text="<<
Önceki Resim" Width="130px" OnClick="btnOnceki_Click" />
                            <asp:Button ID="btnSonraki" runat="server"
Text="Sonraki Resim >>" Width="130px"
                                OnClick="btnSonraki_Click" />
                            <asp:Button ID="btnDurdur" runat="server" Text="Durdur"
OnClick="btnDurdur_Click" />
                            <asp:Button ID="btnBaslat" runat="server" Text="Başlat"
OnClick="btnBaslat_Click" />
                            <asp:Timer ID="Timer1" runat="server" Interval="5000"
OnTick="Timer1_Tick">
                            </asp:Timer>
                        </ContentTemplate>
                        <Triggers>
                            <asp:AsyncPostBackTrigger ControlID="listAlbumler"
EventName="SelectedIndexChanged" />
                        </Triggers>
                    </asp:UpdatePanel>
                </td>
            </tr>
        </table>
    </div>
    </form>
```

Bu aşamada dikkat çeken noktalardan biri ListBox'ın UpdatePanel dışında yer alıyor olması ve UpdatePanel'e Asenkron PostBack Trigger olarak eklenmiş olmasıdır. ListBox'ın seçili olan öğesi değiştiğinde UpdatePanel'in içi de güncelleniyor olacaktır. Ancak, SelectedIndexChanged olayının tetiklenmesi için ListBox'ın AutoPostBack özelliği True olarak ayarlanmış olmalıdır.

Tasarımın gerçekleştirilmesinin ardından, ilk gerçekleştirilmesi gereken işlem ListBox'a resimler klasörü içinde yer alan albümlerin doldurulması olacaktır. Bu işlem için System.IO isimalanı içinde yer alan sınıfların kullanılacağından daha önce bahsedilmişti. Gerekli işlemleri gerçekleştirecek olan kodlar, PageLoad'a yazılıp albümler ListBox'a yüklenebilir.

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            string dizin = Server.MapPath("resimler") + "\\";
            DirectoryInfo di = new DirectoryInfo(dizin);
            DirectoryInfo[] klasorler = di.GetDirectories();
            foreach (DirectoryInfo album in klasorler)
            {
                listAlbumler.Items.Add(album.Name);
            }
        }
    }
```

Yukarıdaki kodlarda ilk olarak albümlerin bulunduğu klasörün yerini saklayacak olan string tipinde **dizin** adında bir değişken tanımlanıyor ve dizin değişkeninin işaret ettiği klasör içinde bulunan klasörler **DirectoryInfo** sınıfının **GetDirectories()** metodu ile elde ediliyor. Ardından, foreach döngüsü ile albüm adları ListBox'a ekleniyor. Bu işlemin ardından bir metod yazılıp seçili olan albüm içinde yer alan resimlerin getirilmesini sağlayalım.

```
void ResimGoster(string yon)
    {
        // Seçili olan albümdeki resimlerin listesini alıp bir diziye atılıyor
        string albumDizin =
            Server.MapPath("resimler") + "\\" + listAlbumler.SelectedItem.Text;

        DirectoryInfo di = new DirectoryInfo(albumDizin);
        FileInfo[] resimler = di.GetFiles("*.jpg");

        /* Her çağrıdan sonra dizi içinde bir sonraki resme gidileceği için resim
    numarası ViewState'de saklanıyor ve her işlem yapılmasında resim numarası
    değiştiriliyor. */

        int resimNo;
        if (ViewState["resimNo"] != null)
        {
            resimNo = (int)ViewState["resimNo"];
            // Eğer resimler ileri doğru okunacaksa
            if (yon == "ileri")
            {
                resimNo++;
                // Eğer dizideki son resime gelinirse baştaki resme dönülüyor
                if (resimNo >= resimler.Length)
                    resimNo = 0;
            }
            // Eğer resimler geri yönde olursa
            else
            {
                resimNo--;
                // İlk resime gelinmişse dizideki son resme dönülüyor
                if (resimNo < 0)
                    resimNo = resimler.Length - 1;
            }
            ViewState["resimNo"] = resimNo;
        }
        else
        {
            resimNo = 0;
            ViewState["resimNo"] = resimNo;
        }
        /* Son olarak resmin numarası alınıp imgResim adındaki kontrolde
    görüntüleniyor */

        imgResim.ImageUrl =
```

```
        "resimler\\" + listAlbumler.SelectedItem.Text + "\\" + resimler[resimNo];
    }
```

Yukarıdaki kodlarda yer alan ResimGoster isimli metodun görevi, almış olduğu yön parametresine göre gösterilen resmi değiştirip yeni bir resmi Image kontrolünde göstermektir. Tabi metodun ilk çağrısında daha önce gösterilen resim olmadığı için ilk resimden göstermeye başlamaktadır. Resim adları ise belirtilen albüm içinde bulunan .jpg uzantılı dosyalardan elde edilmektedir. Resimler görüntülenirken ilk olarak belirtilen albümde bulunan resimler bir diziye atılıyor ve ViewState'te hangi resmin görüntülendiği bilgisi tutuluyor. Ardından, diğer çağrılarda belirtilen yöne göre, ViewState'teki değer arttırılıyor veya azaltılıyor. Dizinin son veya ilk elemanına gelindiğinde ise tekrar dizinin başına veya sonuna dönülerek resimlerin devamlı gösterilmesi sağlanıyor. Metod yazıldıktan sonra, sayfa ilk yüklendiğinde çalıştırılmak üzere aşağıdaki kodlar PageLoad'a eklenmelidir. Aşağıdaki kodlar sadece sayfa ilk sefer yüklendiğinde çalışacak şekilde ayarlanmalıdır.

```
listAlbumler.SelectedIndex = 0;
ResimGoster("ileri");
```

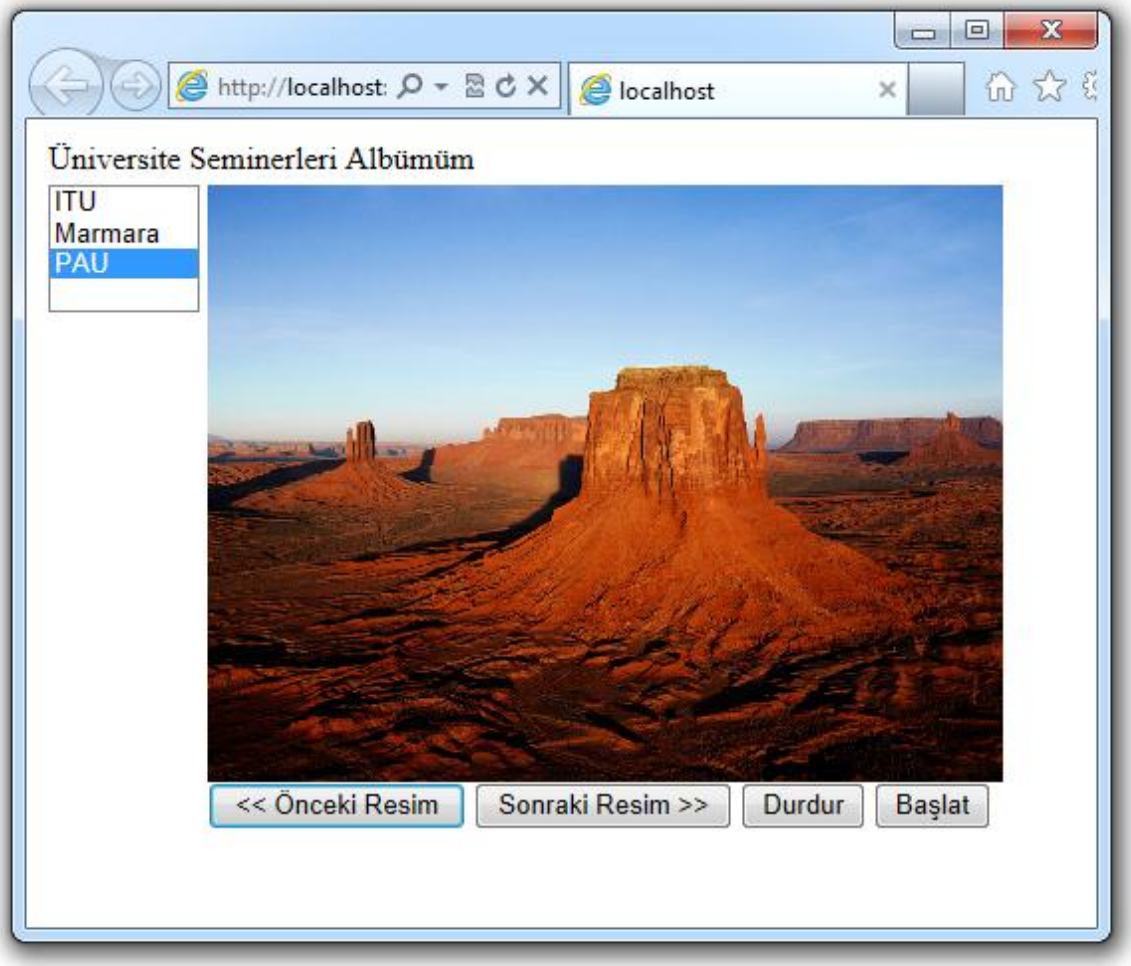
Yukarıdaki kod sayfa ilk yüklendiğinde çalışacak ve albümün ilk resmini Image kontrolü içerisinde görüntülüyor olacaktır. Diğer resimlerinde sırayla gelmesini sağlamak için Timer'ın Tick olayı ele alınıp ResimGoster metodu çağrılmalıdır.

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        ResimGoster("ileri");
    }
```

Şimdi sıra geldi Button'ların Click olaylarını ele alıp gerekli işlemleri gerçekleştirmeye. Durdur butonuna tıklantığında Timer pasif duruma alınacaktır, Başlat'da ise Timer aktif duruma alınacaktır. Sonraki ve önceki ise ResimGoster metodunu belirtilen yöne doğru yeniden çağıracaktır.

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        ResimGoster("ileri");
    }

    protected void btnOnceki_Click(object sender, EventArgs e)
    {
        ResimGoster("geri");
    }

    protected void btnSonraki_Click(object sender, EventArgs e)
    {
        ResimGoster("ileri");
    }

    protected void btnDurdur_Click(object sender, EventArgs e)
    {
        Timer1.Enabled = false;
    }

    protected void btnBaslat_Click(object sender, EventArgs e)
    {
        Timer1.Enabled = true;
    }
```

Yukarıdaki kodlar da eklendikten sonra, uygulamayı çalıştırılabilirsin. Uygulama çalıştırıldıktan sonra aşağıdaki resimde yer alan görünüm ile karşılacaksın. Tabi resimler klasörü içerisine yeni klasör oluşturup resim atmayı unutmamak lazım.



Microsoft