

EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

Web Sitesi Nedir?



Microsoft Türkiye

Açık Akademi

Web Uygulamaları ve Web Uygulamalarında Kullanılan Teknolojiler

Web uygulamaları Internet Explorer ve benzeri web tarayıcıları aracılığı ile eriştiğimiz uygulamalardır. Web Uygulamaları statik, yani sabit içerikli olabileceği gibi içeriği dinamik de olabilir. Sabit içerikli siteler belli aralıklarla yazılımcı tarafından güncellenir. Dinamik içerikli web sayfalarının kullanıcılara sunmuş olduğu içerikler ise belli kriterlere göre ya da kullanıcıya göre değişiklik gösterip kullanıcı ile etkileşime girebilir.

Facebook'u gözünüzün önüne getirin, Facebook'a girerken belli bir parola ve kullanıcı adı belirtip bize özel içeriğin karşımıza çıkmasını sağlıyoruz. Facebook'un içinde gezerken istediğiniz bilgiyi arayıp istediğiniz kullanıcının profilini ziyaret edebiliyorsunuz. Facebook'un belirttiğimiz özelliklerini göz önüne getirdiğimizde Facebook'un tam anlamıyla bir dinamik site olduğunu anlamak çok da zor olmayacaktır. Şimdi de gözünüzün önüne köşedeki bakkalımızın sitesini getirelim. Site bize sadece bakkalın adresini, satılan ürün çeşitlerini ve iletişim bilgilerini içeriyor. Bu sitede yazılımcı güncelleme yapmadığı sürece, siz, her ziyaretinizde aynı içerik ile karşılaşacağınız olacaksınız. Buradan da anlaşılacağı gibi köşe bakkalımızın sitesi statik bir sitedir.

Statik sitelerde genellikle, sadece **HTML (HyperText Markup Language)** teknolojisi kullanılırken, dinamik sitelerde ise içerisinde HTML'i de barındıran **ASP.NET, ASP,PHP** gibi web programlama teknolojileri de kullanılabilir. Bu teknolojilerden PHP ve ASP, ASP.NET teknolojisine göre daha eski teknolojilerdir ve uzun yıllardır kullanılmaktadırlar. Adının ilk duyulmasından sonra oldukça popüler olan **PHP**, geçen sürede bu özelliğinden hiçbir şey yitirmeden bugüne kadar geldi. **ASP** ise Microsoft tarafından Internet dünyasındaki çeşitli gelişmelerin ardından, 1996 yılında 1.0 sürümü ile duyuruldu. Genelde **VBScript** kodları ile yazılan ASP uygulamaları da, kısa zamanda hak ettiği ilgiyi buldu ve geniş bir kullanıcı kitlesine yayıldı. ASP teknolojisi, versiyon 3.0'a kadar geliştirilmeye devam edildi. 2000 yılında duyurulan ASP 3.0, "Klasik ASP"nin son sürümü oldu.

Microsoft, ASP'nin alt yapısındaki bazı problemler ve eksikliklerden dolayı, klasik ASP'yi geliştirmeyi bıraktı ve .NET platformu üzerinde çalışacak, web teknolojilerinde yeni bir yaklaşım olan ASP.NET'i geliştirdi. ASP.NET, .NET platformunda yer alan birçok dille yazılabilen, tam olarak nesneye yönelimli yeni bir teknoloji olarak ortaya çıktı. ASP.NET bugünkü yapısıyla, yazılım geliştiricilere çok kısa sürede, ileri seviye web uygulamaları geliştirebilmelerini sağlayan, sunucu taraflı (yazılan kodların web sunucusunda çalıştırıldığı) bir teknolojidir.

Web uygulamaları geliştirilirken, farklı teknolojilerden yararlanılabilmektedir. HTML ile sadece web sayfalarının nasıl görüntüleneceği belirlenirken, istemci ve sunucu tarafındaki bazı teknolojiler de kullanılarak, web sayfalarının nasıl çalışacağı, ziyaretçilerle nasıl etkileşimde bulunacağı belirlenebilir. Web programcısı, profesyonel web uygulamaları geliştirebilmek için, birden fazla teknolojiye hakim olmalıdır. Genel olarak **HTML, JavaScript** ve sunucu taraflı PHP, ASP ve ASP.NET gibi en az bir teknolojiye hakim olmak yeterlidir.

HTML ve CSS ayrı bir eğitim olarak Açık Akademi'de yer almaktadır. ASP.NET içeriğine başlamadan önce bu eğitimi bitirmeniz faydalı olur.

JavaScript

JavaScript, **istemci tarafında** çalışan ve amacı HTML'in statik yapısına zenginlik kazandırmak olan, bir script dilidir. HTML gibi statik bir dünyadan dinamik bir dünyaya geçiş, Microsoft'un da bu teknolojiye ilgi duymasına sebep oldu. Microsoft, JavaScript diline gerek yapı, gerek işlevsellik açısından çok benzeyen ve **JScript** adını verdiği bir dil geliştirdi. Aşağıda bir web sayfasına "Merhaba Dünya" yazılmasını sağlayan basit bir JavaScript kodu bulunmaktadır.

```
<script type="text/javascript">
  document.write("Merhaba Dünya");
</script>
```

Web Sayfalarını Programlama

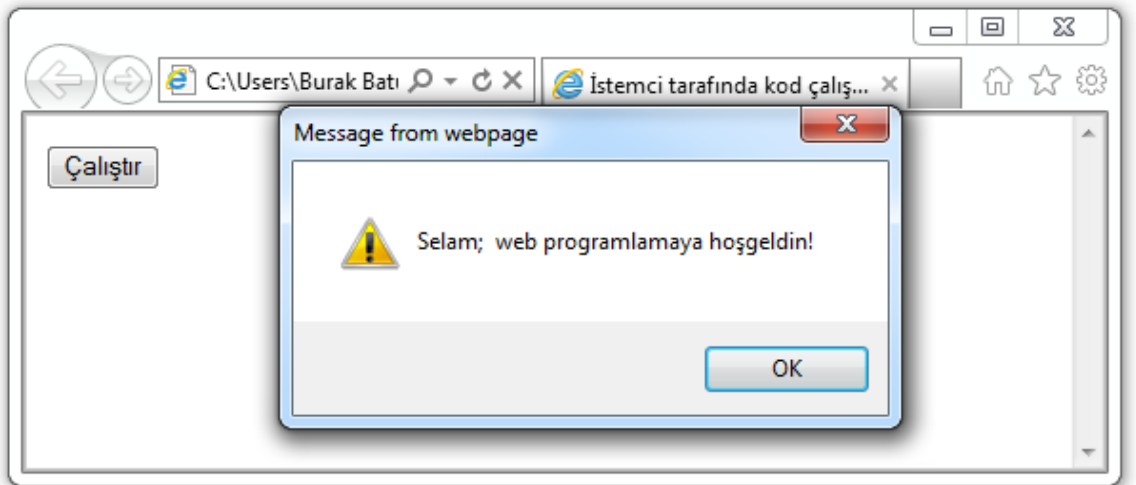
Web sayfalarının çalışma mantığı, masaüstü uygulamalara göre çok farklıdır. Masaüstü uygulamalar tek bir bilgisayar üzerinde çalışır. Çalışma esnasında oluşan çıktılar, ekran görüntüleri ve değişiklikler yine aynı bilgisayar üzerinde olmaktadır. Web sayfalarında ise çok farklı bir durum söz konusudur. Web sayfaları sunucu üzerinde tutulmakta ve çalıştırılmaktadır. Çalışma sonucunda HTML kodları üretilmekte ve gelen istekler sonucunda ziyaretçilerin bilgisayarında görüntülenmektedir. Böyle bir çalışma modelinde web programcısı olarak sunucu tarafında ve istemci tarafında bazı işlemler yapılabilir. Web uygulamaları geliştirme sürecinde, çeşitli diller ve teknolojiler kullanılır. Bu yapıların bazıları, sadece istemci tarafında çalışabilirken, bazıları ise sadece sunucu tarafında çalışabilmektedir.

İstemci Tarafı Programlama (Client-Side Programming)

Bir web sayfası hazırlanırken, bazı işlemlerin istemci tarafında yapılması sağlanabilir. Örneğin, ziyaretçinin bilgisayarındaki tarihi ve saati sayfaya yazdırmak, tarayıcı tipine göre işlem yapmak veya bir butona basıldığında ziyaretçiye bir uyarı mesajı göndermek gibi işlemler istemci tarafında yapılabilecekler birer örnektir. Bu tip işlemlerde yazılan kodlar istemci tarafında, yani tarayıcı üzerinde yorumlanmakta ve çalıştırılmaktadır. İstemci tarafında programlama yapılırken kullanılan temel dil JavaScript dilidir. HTML ve CSS ise yine istemci tarafında yorumlanırlar.

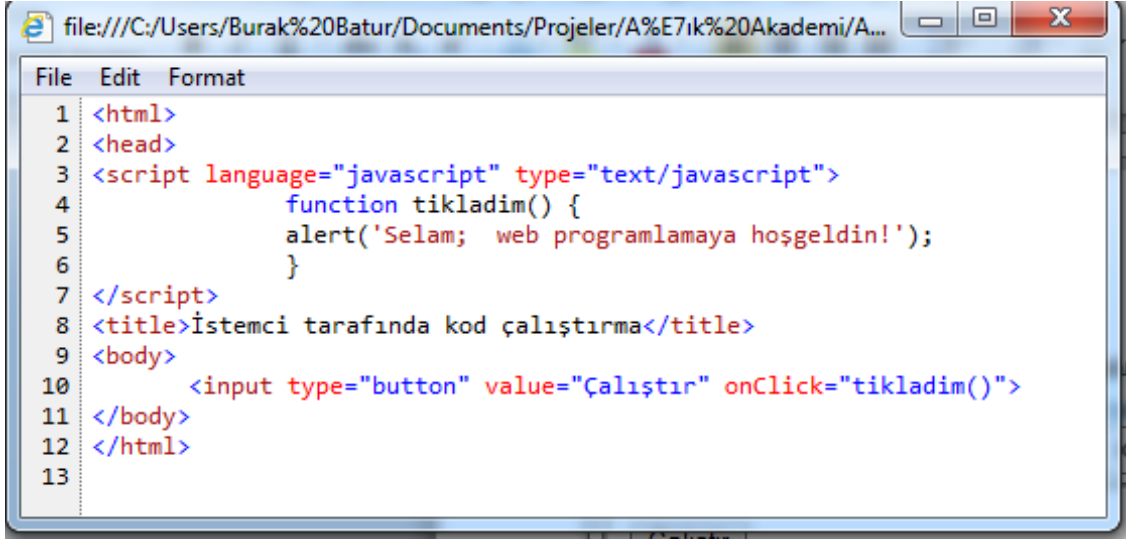
Aşağıdaki örnekte, "Selam; web programlamaya hoşgeldin! " şeklinde bir metnin, butona basılınca gösterilmesini sağlayan kodlar bulunmaktadır.

```
<html>
<head>
<script language="javascript" type="text/javascript">
    function tikladim() {
        alert('Selam; web programlamaya hoşgeldin!');
    }
</script>
<title>İstemci tarafında kod çalıştırma</title>
<body>
    <input type="button" value="Çalıştır" onClick="tikladim()">
</body>
</html>
```



Örnekteki kodlar bir dosyaya yazılıp, çalıştırdıktan sonra, Çalıştır butonuna tıklanıldığında, ekran görüntüsündeki gibi bir mesaj kutusu görülür. Burada yazılan JavaScript kodları istemci tarafında çalıştırılarak

üretileen çıktı kullanıcıya görüntülenmiştir. Eğer sayfanın boş bir yerinde fareye sağ tıklanıp **Kaynak Kodunu Göster (View Source)** seçilirse, yazılan JavaScript kodlarına erişilebilir.



```
file:///C:/Users/Burak%20Batur/Documents/Projeler/A%20Akademi/A...
File Edit Format
1 <html>
2 <head>
3 <script language="javascript" type="text/javascript">
4     function tikladim() {
5         alert('Selam; web programlamaya hoşgeldin!');
6     }
7 </script>
8 <title>İstemci tarafında kod çalıştırma</title>
9 <body>
10     <input type="button" value="Çalıştır" onClick="tikladim()">
11 </body>
12 </html>
13
```

İstemci tarafında kod yazmanın en önemli dezavantajlarından birisi, yazılan kodu saklayamamaktır. Burada uyarı penceresi çıkarmak yerine daha önemli işlemler yapılabileceği düşünülürse, yazılacak kodların ziyaretçiler tarafından görüntülenmesi oldukça sakıncalı olabilir. Bazen bu kodların görüntülenebilmesi güvenlik açıklarına bile sebep olabilmektedir. Bu nedenle uygulamalarda önemli işlemlerin istemci tarafında yapılması tercih edilmemektedir.

Sunucu Tarafı Programlama (Server-Side Programming)

Sunucu tarafı programlama daha farklı bir yapı içerisinde çalışmaktadır. İstemci tarafı programlamada yazılan kodlar kullanıcıya gönderilerek, kullanıcının bilgisayarındaki tarayıcı üzerinde çalıştırılır ve sonuç yine tarayıcı üzerinde görüntülenir. Sunucu tarafı programlama da ise, yazılan kodlar kesinlikle istemciye gönderilmemekte ve sadece sunucu üzerinde tutulmaktadır. İstemciden sunucuya gelen isteğe göre kodlar sunucu üzerindeki ilgili programlar aracılığıyla çalıştırılır ve sonuç olarak üretilen HTML kodları ile sayfanın içerikleri istemciye gönderilir. İstemci tarafında sadece HTML kodları ve istemci tarafı script kodları görüntülenebilmektedir.

Sunucu tarafında PHP, ASP veya ASP.NET gibi bir teknoloji ile yazılmış olan kodlar, sunucu üzerindeki bazı yapılar tarafından çalıştırılır. Örneğin ASP ile geliştirilen kodlar sunucu tarafında bulunan **ASP.DLL** tarafından ele alınır ve HTML çıktı oluşturulur. Benzer olarak PHP de sunucu tarafı bir programlama dilidir ve o da sunucu üzerindeki PHP motoru tarafından çalıştırılır.

ASP.NET sunucu tarafı bir yazılım geliştirme teknolojisidir ve yazılan kodlar sunucu tarafında kullanılan .NET tabanlı dilin derleyicisi ile derlenip, ortak dil çalışma zamanı (CLR - Common Language Runtime) tarafından çalıştırılır. Elde edilen HTML çıktı istemciye gönderilir. Aşağıda ASP.NET ile hazırlanan bir sayfa örneği yer almaktadır.

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<script runat="server">

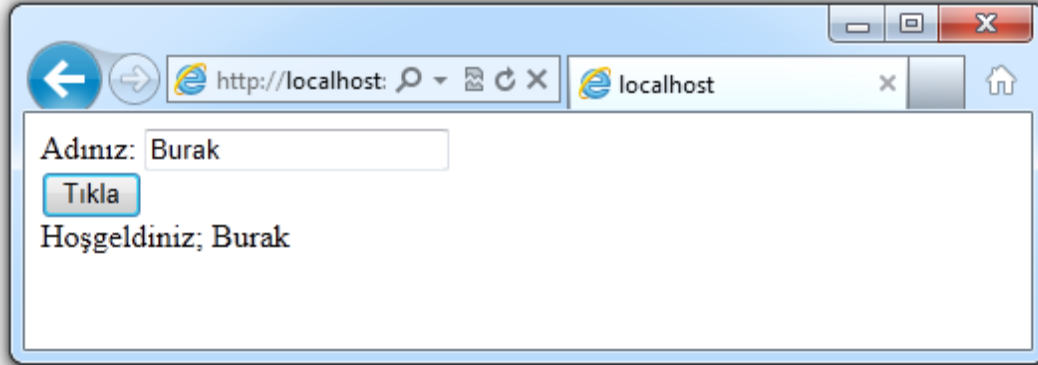
    protected void Button1_Click(object sender, EventArgs e)
    {
        Labellsim.Text = "Hoşgeldiniz; " + TextBoxIsim.Text;
    }
}
```

```
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>

      Adınız:
      <asp:TextBox ID="TextBox1sim" runat="server"></asp:TextBox>
      <br />
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Tıkla" />
      <br />
      <asp:Label ID="Label1sim" runat="server" Text="Label"></asp:Label>
    </div>
  </form>
</body>
</html>
```

Örnek Visual Studio ortamında çalıştırıldığında TextBox1sim metin kutusuna (<asp:TextBox> ile belirtilen kısım) bir metin yazılıp Tıkla butonuna tıklandığında, metin kutusu içerisine yazılan yazılar Label1sim isimli metin alanına (<asp:Label> ile belirtilen kısım) aktarılacaktır.



Buradaki önemli hususlardan biri de sayfanın kaynak kodlarıdır. Sayfa üzerindeyken fareye sağ tıklayıp **Kaynak Kodunu Göster (View Source)** seçildiğinde aşağıdaki gibi bir sonuç elde edilir.

Görüldüğü gibi yazılan kodlar ile ortaya çıkan kodlar farklıdır. Ayrıca `<script runat="server"></script>` etiketleri arasında yazılan C# kodları da hiçbir şekilde görünmemektedir.

IIS (Internet Information Services)

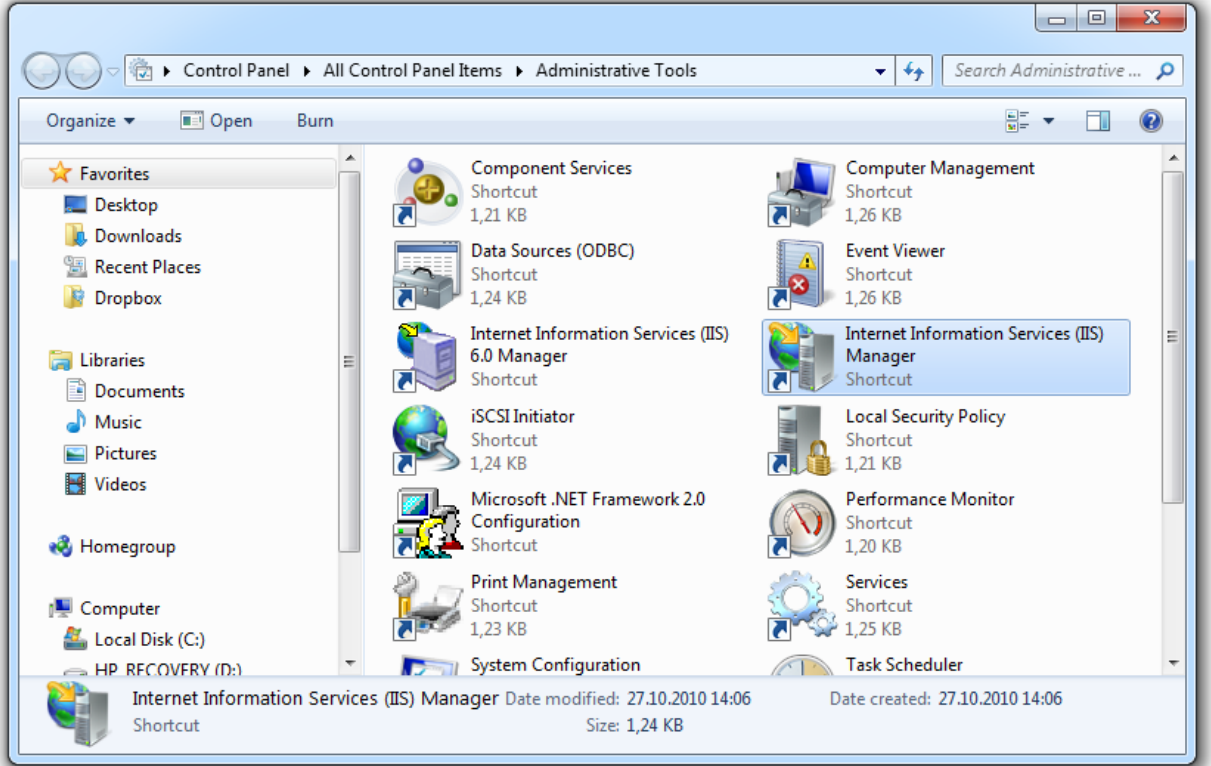
Internet Information Services, web uygulamalarının yayınlanması için web uygulamalarını barındırıp bunları dışarıya sunacak olan sunucu servislerinden biridir. ASP.NET uygulamaları da yayınlanmak için IIS içerisinde barındırılıyor olmalıdır. IIS dışarıdan gelen çağrıları yanıtlayabilmek için varsayılan olarak 80 nolu portu dinler ve gelen talepleri uygun altyapıya devredip işleterek dışarıya HTML çıktısını gönderir. Biz de geliştirmiş olduğumuz ASP.NET uygulamalarını yayımlamak için IIS üzerinde gerekli konfigürasyonları yaptıktan sonra, uygulamamızı IIS üzerinde barındırıyoruz. Bir ASP.NET uygulamasını IIS üzerinden yayımlamak istiyorsak, IIS üzerinde sanal dizin adını verdiğimiz yapıları kullanmak zorundayız.

IIS'te Sanal Dizinler

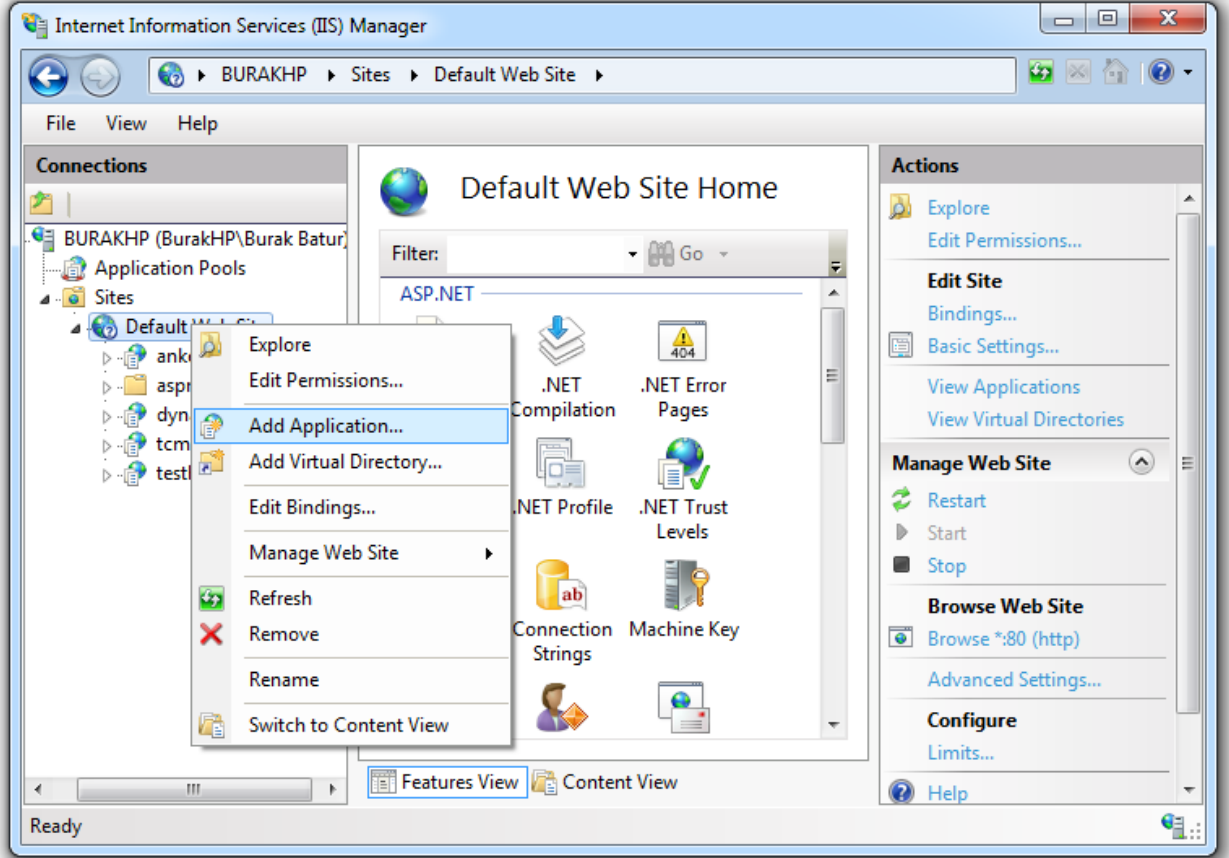
Sanal Dizinler adından da anlaşılacağı üzere aslında var olmayan ve web tabanlı bir uygulamanın dışarıya sunulduğu, IIS üzerinde bulunan yapılardır. ASP.NET uygulamaları da dışarıya sunulabilmek için bir sanal dizin içerisinde yer alıyor olmalıdır. Yeni bir Sanal Dizin oluşturma işlemi aşağıdaki gibi gerçekleştirilebilir.

Yeni Bir Sanal Dizin Oluşturmak

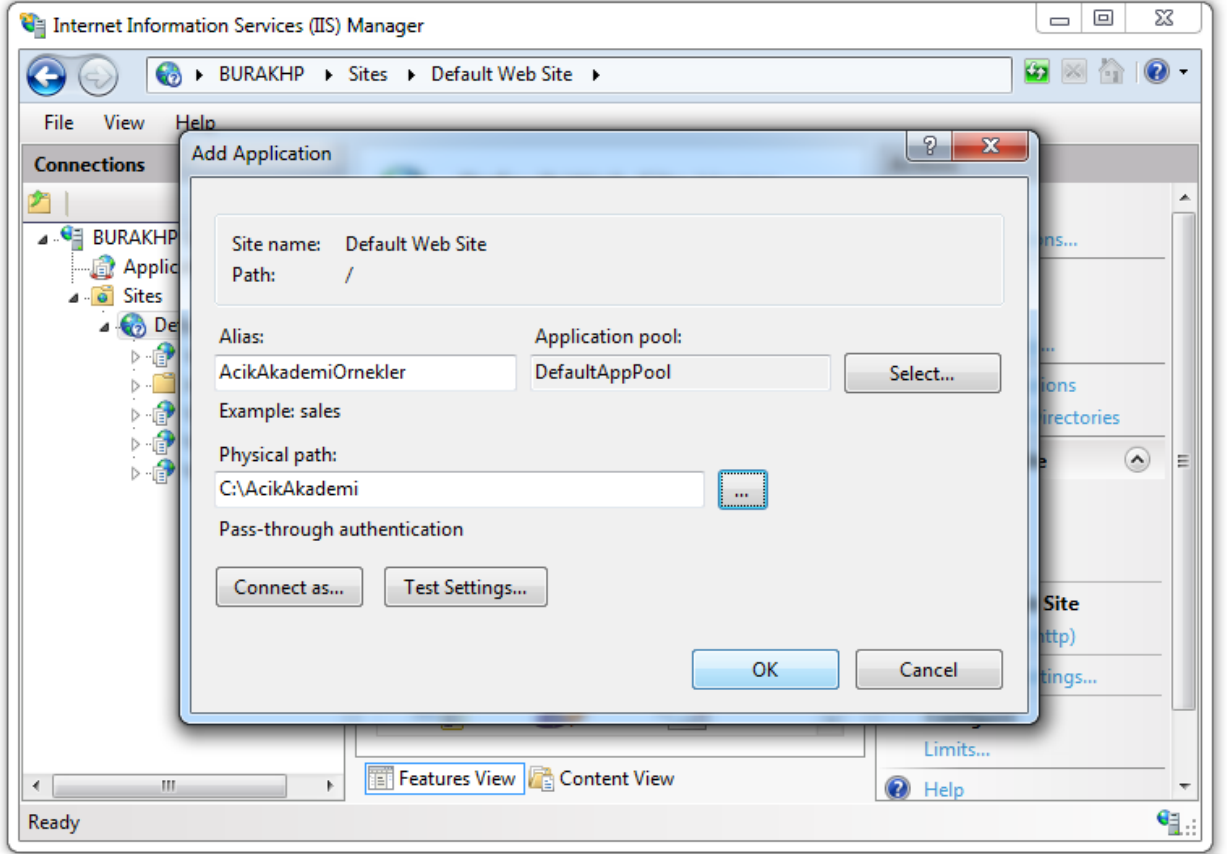
Yeni bir sanal dizin oluşturmak için ilk olarak **Denetim Masası**'ndan **Yönetimsel Araçlar** bölümüne geçilip **Internet Information Services (IIS)** kısayoluna tıklanarak IIS'in yönetim penceresi açılmalıdır.



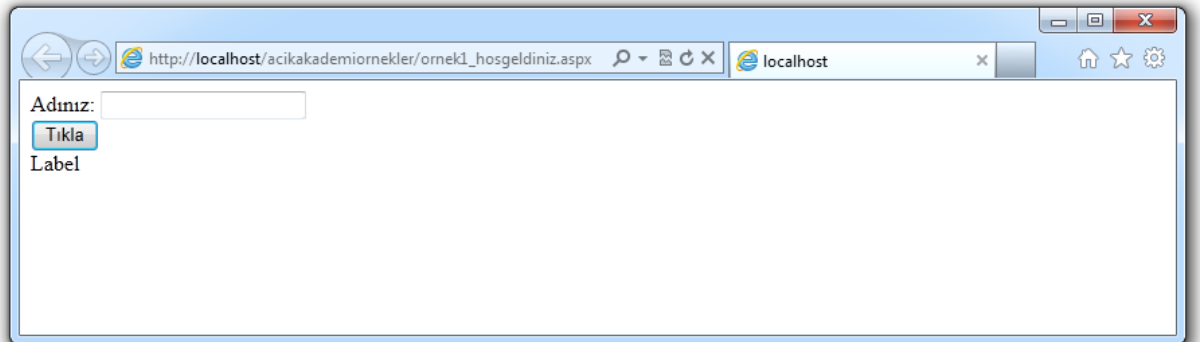
Yönetimsel Araçlar bölümünden IIS'e çift tıklandığında IIS'in yönetim penceresi açılıyor olacaktır. Bu pencere aracılığı ile IIS üzerinde bulunan siteler görülüp, özellikleri ayarlanabilir ya da yeni bir site eklenebilir. Bir siteye yeni bir sanal dizin eklemek için site üzerinde mouse'ın sağ tuşu ile tıklanıp açılan menüden **Add Application (Web Uygulaması Ekle)** seçeneği seçilmelidir.



Yukarıdaki işlemten sonra sanal dizinin takma adı (alias) ve fiziksel dosyaların saklanacağı yer soruluyor olacaktır. Takma isim, sanal dizinin dışarıdan görülecek olan ismi olacaktır. Takma isim (Alias) bölümüne istediğimiz ismi yazdıktan sonra fiziksel yol (Physical Path) bölümünden de fiziksel dosyalarımızın, yani ASP.NET dosyalarımızın, bulunduğu bölümü seçiyoruz.



İçeriğin fiziksel olarak depolanacak olduğu klasör de seçilip ok tuşu ile sihirbaz sonlandırılır. Bu adımdan sonra, fiziksel dizine istenilen ASP.NET dosyaları eklenip çalıştırılabilir. Siz de bir önceki bölümde belirtmiş olduğumuz ASP.NET kodlarını, bir notepad'a yapıştırıp, ornek1_hosgeldiniz.aspx şeklinde bir isimle kaydettikten sonra buraya ekleyebilirsiniz. Sayfayı çalıştırmak için bir Internet Explorer penceresi açın ve URL bölümüne http://localhost/acikakademiornekler/ornek1_hosgeldiniz.aspx yazın.



EĞİTİM :

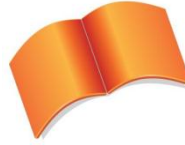
**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

Visual Studio İle Web Projesi
Oluşturma



Microsoft Türkiye

Açık Akademi

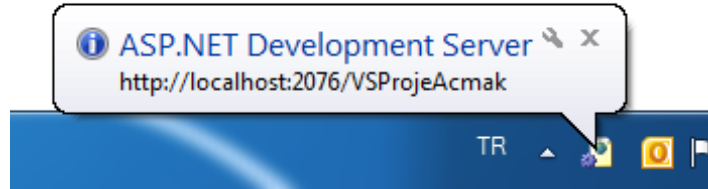
Visual Studio ile Web Projesi Oluřturma

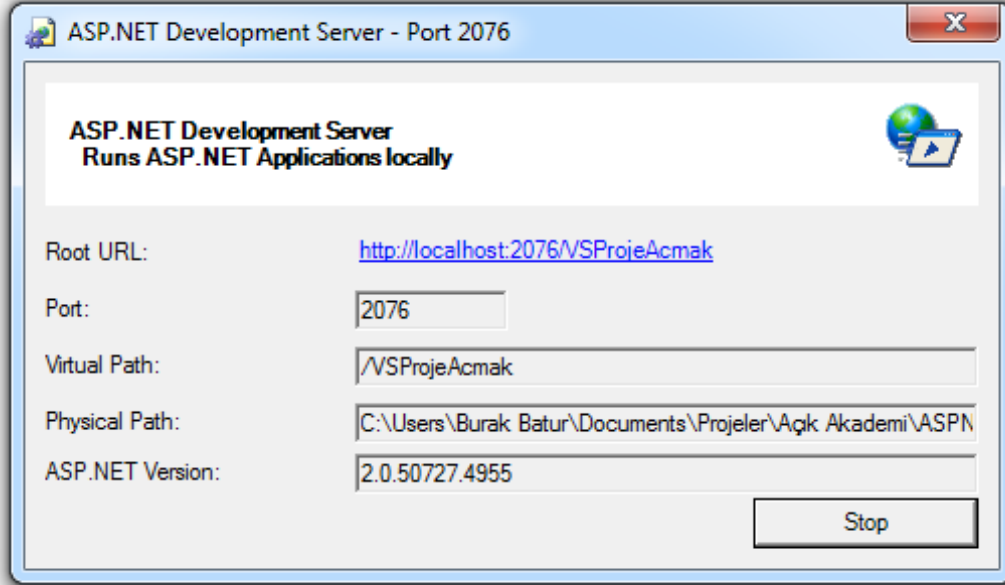
Visual Studio ortamında bir web sitesi projesinin oluřturulması için birden fazla yol bulunmaktadır. Yazılımcının ihtiyalarına gre belirlenen bu yolların farklı zellikleri, avantajları ve dezavantajları bulunabilmektedir. Visual Studio’da yeni bir web sitesi projesi amak için **File** mensnden **New** seeneėindeki **Web Site** semelisiniz. Aılan pencereden projenin tipi, konumu ve kullanılacak dili semelisiniz. Bu noktada oluřturulacak olan projenin konumunu belirlemek nemli bir seim olacaktır. Visual Studio ortamında File System, **HTTP** ve **FTP** olmak zere  farklı yol ile proje oluřturulabilmektedir. Ařaėıda bu yollar detaylı bir řekilde incelenmiřtir.

File System

.NET Framework’n 2.0 srmnden itibaren, **SDK** (Software Development Kit) ile birlikte gelen **ASP.NET Development Server**, IIS’e baėımlı kalınmadan web uygulamalarının alıřtırılabilmesini saėlamaktadır. Visual Studio aracı ile btnleřik alıřan bu web sunucu uygulaması sayesinde bilgisayarda IIS kurulu olmasa dahi uygulama yapılandırılıp alıřtırılabilmektedir.

File System yntemi, bilgisayarın fiziksel dizininde saklanacak bir web projesi oluřturulmasını saėlar. Aılan proje IIS gibi bir web sunucusuna baėımlı olmayacaėı için bilgisayarlar arasında kolayca tařınabilecektir. Zira IIS’e baėımlı kalınması durumunda projenin bir bařka bilgisayarda alıřtırılabilmesi için gerekli yapılandırma iřlemlerinin ele alınması gerekecektir. Bu yntem ile bir proje geliřtirilirse, belirlenen fiziksel bir klasr ierisinde yeni bir web sitesi projesi oluřturulacaktır. Proje alıřtırıldıėında ise Visual Studio’nun kendi web sunucusu devreye girecektir. Bu web sunucusu, sadece yerel (local) makineden gelen istekleri cevapladıėı iin gvenlik aıėı oluřurmamakla birlikte, IIS’e oranla daha performanslı alıřmaktadır. nk; bu sunucu sadece yazılım geliřtirme amalı bir kullanım iin geliřtirilmiřtir. Proje alıřtırıldıėında bu sunucu alıřtırılmakta ve grev ubuėunun (taskbar) saat kısmında bir ikon ıkmaktadır. Bu ikonun zerine ift tıklanırsa o an alıřan proje ile ilgili bilgilere eriřilebilir. Ařaėıda bu iřlem ile ilgili ekran grntleri bulunmaktadır.





- File System ile oluşturulan siteler, gerekli durumlarda IIS üzerinde açılacak sanal bir dizine taşınabilir ve buradan da çalıştırılabilir.
- ASP.NET Development Server ile web sitesi yayımlama yapılamamaktadır.

HTTP

HTTP, önceki sürümlerinde de kullanılan klasik site oluşturma yöntemidir. Yeni bir web sitesi oluşturulurken bu yöntem kullanılırsa, Visual Studio verilen isme uygun olarak IIS üzerinde bir sanal dizin (virtual directory) oluşturur ve sitenin IIS üzerinden çalışmasını sağlar. Bir sunucuda yayınlanacak olan web sitesi IIS üzerinde barındırılacağı için bu yöntem ile proje geliştirmek daha gerçekçi bir ortam sunar. Fakat projenin başka bir bilgisayara taşınması esnasında bazı yapılandırma işlemleri yapmak gerekecektir.



Uygulama geliştirmek için bir de **Remote Site** seçeneği vardır. Bu seçenek kullanılarak **HTTP** ile uzak bir site üzerinden çalışmak mümkün olmaktadır. Ancak uzak makinede **FrontPage Server Extension**'lerinin kurulu olması gerekmektedir.

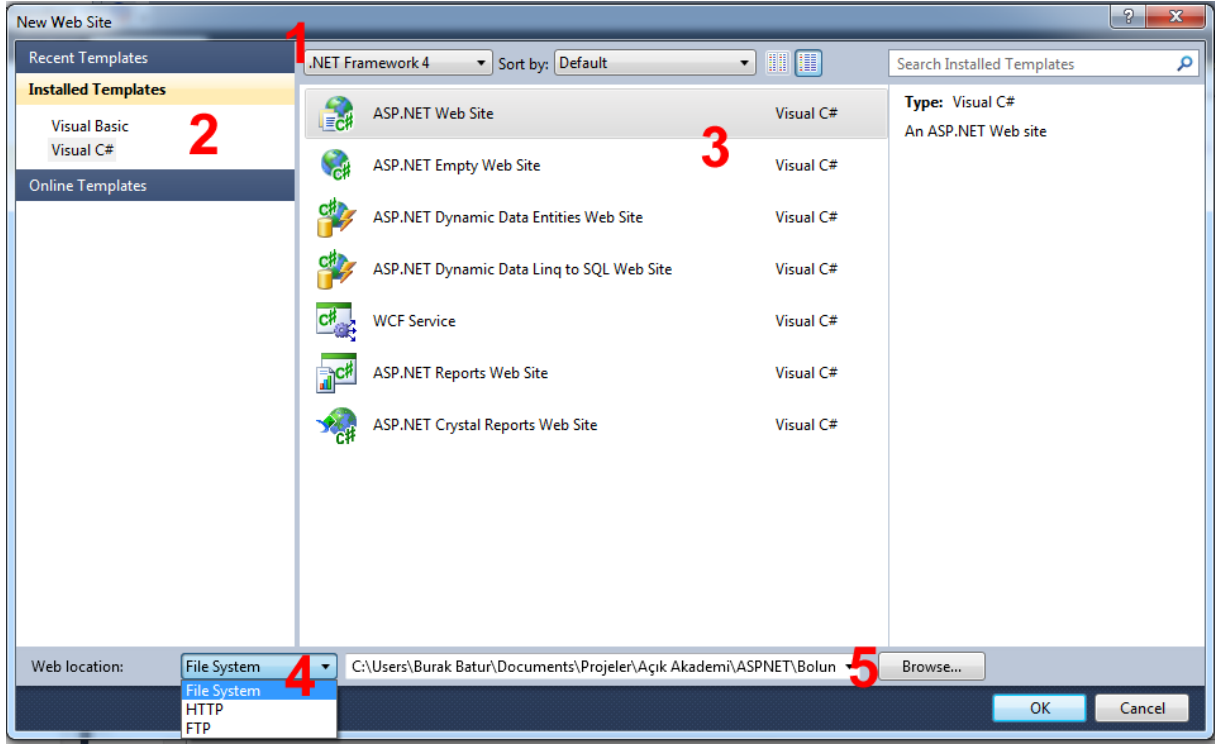
FTP

Bu yöntemde site belirtilen bir FTP (File Transfer Protocol) adresi üzerinde oluşturulacaktır. Bu yöntem ile çalışırken uygulama dosyaları sunucu üzerinde oluşturulduğu için, dosyaları yeniden ftp üzerinden yüklemeye gerek yoktur. Bu şekilde geliştirilen bir proje üzerinde çalışırken devamlı bir Internet bağlantısına ihtiyaç duyulmaktadır. Bu zorunluluk dışında, ayrıca, yapılan değişiklikler siteye anında yansıtılacağı için bazı eksiklikler ve oluşabilecek hatalar sayfaları ziyaret eden kişiler tarafından da görülebilecektir. Bu nedenle bu uygulama geliştirme sürecinde daha dikkatli çalışmak gerekir.

Visual Studio ile Yeni Bir Web Projesi Oluşturmak

Bir web projesi hazırlanırken bu üç yoldan uygun olanı seçilebilir. Genellikle geliştirme aşamasında sunduğu avantajlar ve kolaylıklar açısından File System yöntemi web uygulamalarında tercih edilmektedir.

Visual Studio’da yeni bir web sitesi projesi oluşturmak için **File** menüsündeki **New** sekmesinden **Web Site** seçilir. Açılan pencereden ise proje ile ilgili detaylar seçilecektir. Aşağıdaki şekilde bu pencere bulunmaktadır.

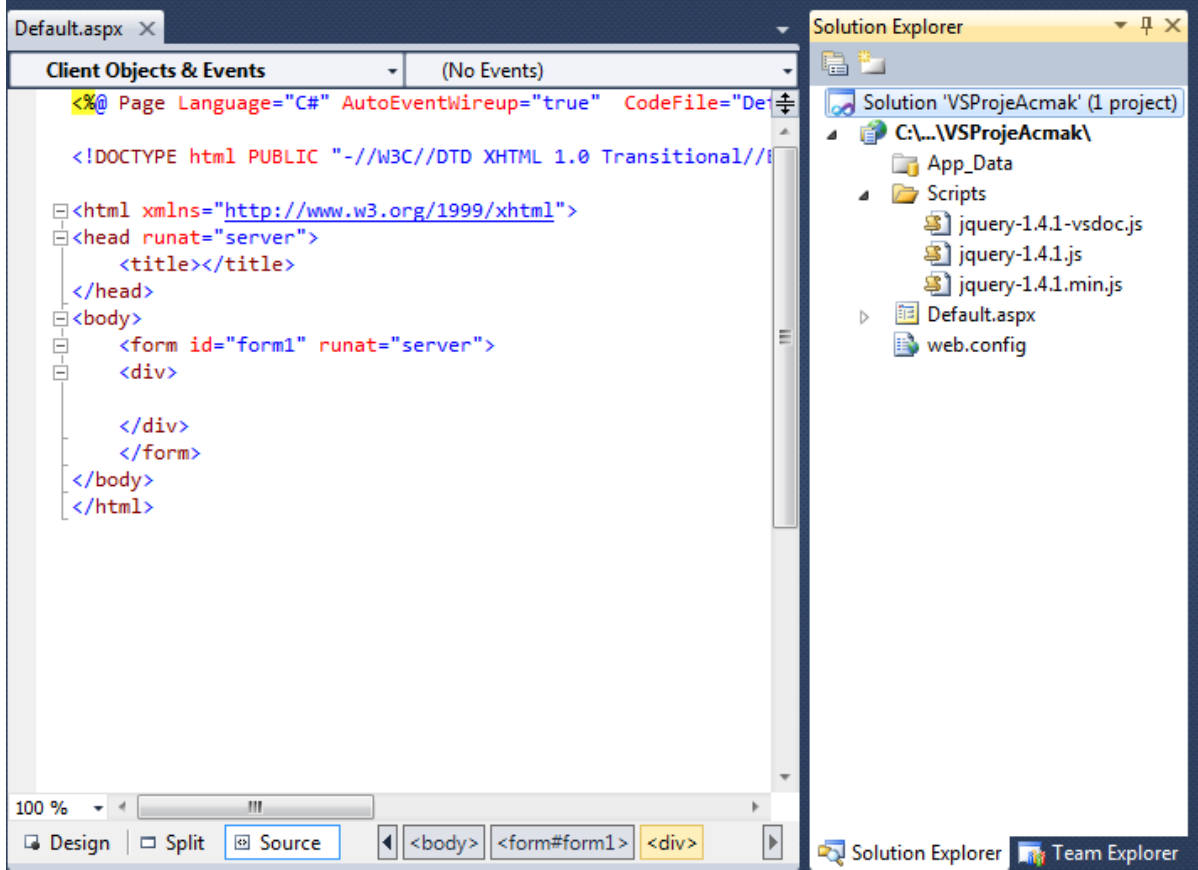


Şekildeki numaralandırılmış kısımlar ile ilgili olarak;

- 1- **Proje sürümü:** Projenin hangi .NET Framework sürümüne göre oluşturulacağı seçilir. Yapılacak seçime göre farklı proje şablonları (template) getirileceği gibi proje açıldığında Toolbox’taki kontroller ve projenin yapısı da ilgili sürüme göre güncellenecektir.
- 2- **Dil (Language):** Projenin hangi dil ile geliştirileceği belirlenir. Visual Studio ile birlikte kurulan .NET destekli diller bu listede gelmektedir. Seçtiğiniz dil ile geliştirme yapılabilecekler şablonlar 3 numaralı alanda listenecektir.
- 3- **Proje şablonu:** Visual Studio için kurulmuş olan proje şablonlarından hangisinin kullanılacağı seçilir. Standart bir web uygulaması açmak için **ASP.NET Web Site** seçeneği seçilir.
- 4- **Konum (Location):** Projenin hangi yöntem ile açılacağı belirlenir. Yukarıda anlatılan **File System**, **HTTP** veya **FTP** seçeneklerinden biri seçilmelidir.
- 5- **Projenin kaydedileceği konum:** Projenin fiziksel veya sanal olarak hangi konumda kaydedileceği belirlenir. Burada Browse (Gözet) tuşu kullanılarak daha görsel bir şekilde seçim de yapabilirsiniz.

Uygun seçimler yapılarak standart bir web projesi açılacak olunursa Visual Studio proje için gerekli dosyaları ekleyecek ve yapılandırmaları yapacaktır. Örneğin **.NET Framework 4.0** sürümü üzerinde **ASP.NET Web Site** şablonunu kullanarak **File System**’de **C#** dilini kullanarak bir proje açılarak, Solution Explorer’da projeye eklenen dosya ve klasörler aşağıdaki resimde olduğu gibi görülebilir. Oluşturulan web sitesi içerisinde **Default.aspx** ve bu dosya ile birlikte çalışacak olan **Default.aspx.cs** dosyası bulunmaktadır. aspx dosyası **Design** (Tasarım) ve **Source** (HTML Kodları) olmak üzere iki kısımdan oluşur. Bu iki kısımda da aslında aynı işler yapılmaktadır. Design kısmında web sayfası görsel öğeler kullanılarak hazırlanırken, Source kısmında web

sayfası HTML kodları yazılarak hazırlanır. Design kısmından görsel öğeler kullanarak web sayfası hazırlandığında, Visual Studio arka planda gerekli HTML kodlarını kendisi üretmektedir. Proje içerisinde veri dosyalarının saklanabilmesi için **App_Data** ve JavaScript dosyalarının saklanması için **Scripts** klasörleri eklenir. Yine uygulama ile ilgili önemli ayarları ve bilgileri saklayan **web.config** dosyası da projeye eklenmiş olarak gelmektedir. Aşağıdaki resimde Default.aspx sayfasının Source kısmı ve projedeki dosyalar görülmektedir.



EĞİTİM :

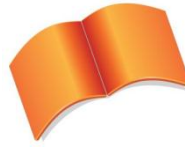
**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

ASP.NET Çalışma Modeli



Microsoft Türkiye

Açık Akademi

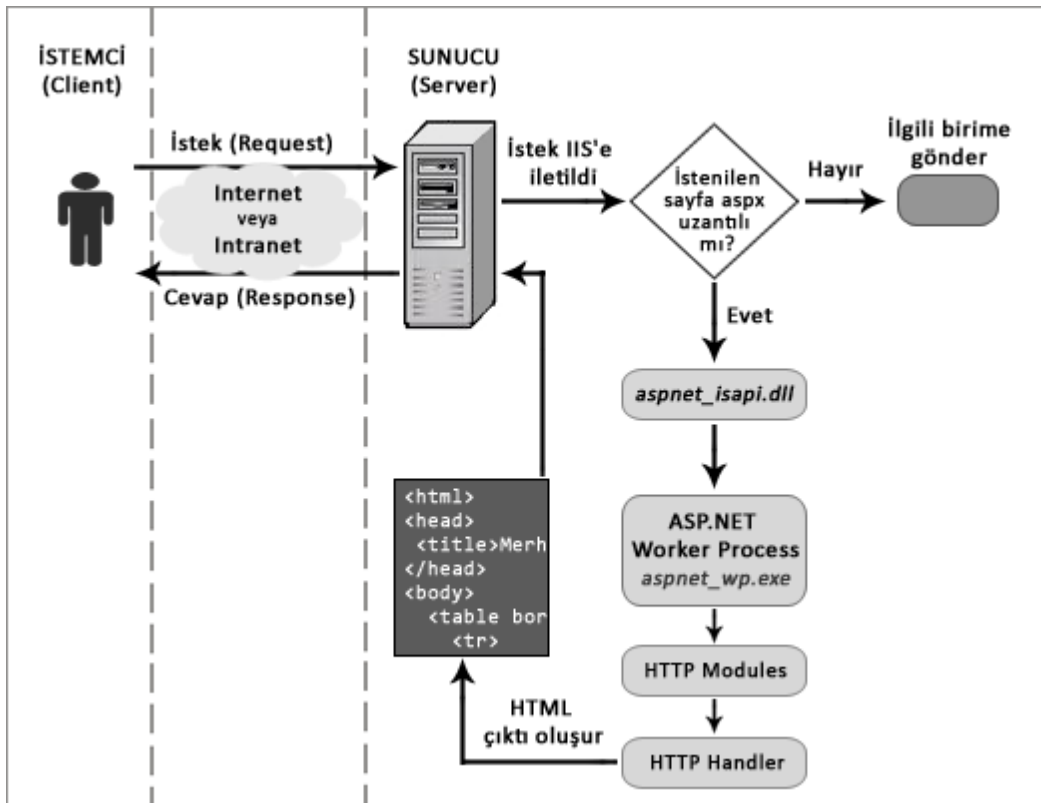
ASP.NET Çalışma Modeli

ASP.NET'in çalışma mantığı, ASP ve PHP gibi betik (script) tabanlı programlama ortamlarına göre oldukça farklıdır. ASP.NET ile bir web uygulamasının geliştirilmesinde büyük kolaylıklar sağlanırken, çalışma zamanında hissedilir düzeyde performans artışı gözlemlenebilir.



www.shutterstock.com · 77850235

Bir web uygulamasındaki web sayfasına istekte bulunulduğunda, sunucu tarafında sayfa üretilir ve HTML çıktısı üretilerek istemciye gönderilir. Tabii ki bu, olayın basit anlamdaki tarifidir. Oysa ki işin arka yüzünde, ASP.NET'in oldukça gelişmiş bir çalışma modeli bulunmaktadır. Sonuçta web uygulaması, istemci ve sunucu arasındaki istekler (request) ve cevaplardan (response) oluşacağı için; bir web sayfasına yapılan isteklerin ne şekilde ele alınacağını bilmek, oldukça önemli olacaktır.



- Bir ASP.NET sayfasına istemciden gönderilecek ilk istek, her zaman **GET** metodu ile URL'den yapılacaktır. Bu istek, internet veya intranet üzerinden ilgili sunucuya iletilecektir.
- Web sayfalarına yapılan istekler, varsayılan olarak 80 numaralı porttan (80 numaralı port numarası standart değerdir, fakat portun numarası değiştirilebilir) yapılacağı için, sunucu bilgisayar bu isteği web sunucusuna, yani IIS'e iletir. IIS'in temel görevi kendisine gönderilen isteklerin yönetilmesini ve cevaplanmasını sağlamaktır.
- Bu noktada IIS; kendisine talep edilen dosyanın uzantısına bakarak, bu dosyayı hangi uygulamanın veya ortamın ele alacağını belirleyecek ve bu birime isteği aktaracaktır. Zira htm, asp, aspx, php, jpg vb. gibi dosyalara yapılan isteklerin her biri sunucu tarafında farklı şekilde ele alınacaktır. Örneğin; htm, jpg, gif gibi içeriği sabit olan dosyalara gelen isteklerde IIS, bu dosyaların disk üzerindeki konumundan alınarak, istemciye aynen taşınmasını sağlayacaktır.
- Fakat asp veya aspx gibi uzantısı olan dosyalar; sunucu tarafında çalıştırılacak kodlar içerdiği için, belirli bir birimin ya da birimlerin isteği ele alması gerekecektir. Örneğin **asp** uzantılı bir dosyaya gelen talep, IIS tarafından **asp.dll** Isapi (Internet Server API) uzantısına gönderilecektir. Burada asıl incelenecek durum ise bir aspx dosyasına gelen bir isteğin, IIS tarafından nasıl yönetileceğidir. **.aspx** uzantılı bir sayfaya istek geldiğinde, IIS dosyanın uzantısını algılayıp, bu isteği **aspnet_isapi.dll** isimli dosyaya yönlendirecektir.
- **aspnet_isapi.dll** dosyası, unmanaged (yönetilemez) bir kütüphane olduğu için .NET Framework ile bütünleşik şekilde çalışamayacaktır. Bu noktada bu dll dosyasının amacı; istemciden gelen istek ile ilgili tüm bilgileri toplamak ve bu bilgilerle birlikte isteği, named pipe aracılığıyla **ASP.NET Worker Process'e** (**aspnet_wp.exe**) iletmek olacaktır.
- ASP.NET Worker Process; gelen isteğin .NET Framework, dolayısıyla Common Language Runtime (CLR) tarafından ele alınmasını ve bu sürecin yönetilmesini sağlar (Eğer web sunucusunda bir ASP.NET uygulaması çalışırken Görev Yöneticisi'ndeki İşlemler sekmesine bakılırsa, **aspnet_wp.exe** görülebilir).
- Bu süreçte **HTTP Module**'ler uygulama ile ilgili temel işlemleri (kimlik doğrulaması, session bilgilerinin yönetilmesi, OutputCache mekanizmasının çalışması, rol ve profil yönetimi gibi işlemler) gerçekleştirir.
- **HTTP Handler** ise sayfanın nesne örneğinin oluşturulması, kodların çalıştırılması ve sayfanın HTML kodlarının üretilmesi işlemlerini yapacaktır.
- Üretilen HTML kodları, aynı yolu takip ederek istemciye gönderilecektir.

EĞİTİM :

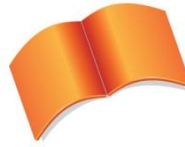
**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

ASP.NET Sayfasının Yapısı



Microsoft Türkiye

Açık Akademi

ASP.NET Sayfasının Yapısı

Önceki teknolojilere nazaran bir ASP.NET sayfasının yapısında oldukça önemli yenilikler bulunmaktadır. Eski teknolojilerde yazılan sunucu tabanlı kodlar HTML etiketleriyle birlikte tutulmaktaydı. HTML etiketleri ile birlikte yazılan bu kodlar karmaşalara yol açabiliyor ve uygulama geliştiricinin işini zorlaştırabiliyordu. ASP.NET ile birlikte gelen yeni sayfa modelinde ise artık HTML kodlarının ve sunucuda çalışacak kodların birbirinden ayrıştırılması sağlandı. Sayfanın HTML kodları ve kullanılacak kontrollerin tanımlamaları **.aspx** uzantılı bir dosyada saklanırken, sunucuda çalıştırılacak C# kodları ise **.aspx.cs** uzantılı bir dosyada saklanmaktadır (VB.NET ile hazırlanan bir projede bu dosyanın uzantısı **.aspx.vb** olacaktır). Her ne kadar kodlar birbirinden ayrıştırılsa da sunucu tarafında sayfa çalıştırıldığında kodlar bir bütün olarak ele alınır ve bu şekilde çalışması sağlanır. Yapı itibarıyla kodların birbirinden ayrıştırılması gerek geliştirme aşamasında gerekse hata denetleme-ayıklama aşamasında uygulama geliştiricilerin işlerini oldukça kolaylaştırmıştır. Basit olarak oluşturulacak bir ASP.NET sayfasının içerisindeki kodlar aşağıda görülmektedir.

Default.aspx dosyasının içeriği:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>Untitled Page</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        </div>
      </form>
    </body>
  </html>
```

Default.aspx.cs dosyasının içeriği:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

Görüldüğü gibi HTML ve C# kodları birbirinden ayrı şekilde saklanmaktadır. Sayfa yapısı ile ilgili olarak göze çarpan en önemli noktalardan bir tanesi sayfanın arka planda aslında bir sınıf (class) olarak tutulmasıdır. Nesne tabanlı bir ortamda çalışıldığı için bu ortamda hazırlanan tüm sayfalar birer sınıf olarak tutulacaktır. Dolayısıyla bir sayfaya gelen her istekte sayfanın sunucuda bir nesne örneği oluşturulacaktır. Yine olay tabanlı bir ortamda çalışıldığı için sayfa nesnesinin kendisine özel olayları (event) bulunacaktır. Sayfa oluşturulurken adım adım olayları gerçekleştirerek sonunda HTML kodlara dönüştürülür (render) ve istemciye gönderilir. İşlem yapılan

sayfa sunucuda nesne olarak ele alındığı için sayfanın nesne örneği, belleğe (RAM) alınır. Sayfa ile ilgili işlemler bitirildiğinde ise nesne örneği bellekten kaldırılır. Bu nedenden dolayı da bir web sayfasındaki bilgilere diğer bir web sayfasından normal şartlarda erişmek mümkün olmamakta ve bu işlemi gerçekleştirebilmek için bazı yollar ve teknikler kullanılmaktadır.

HTML kodlarının bulunduğu aspx uzantılı dosya incelendiğinde sayfanın **Page** direktifi ile başladığı görülür. Bu tanımlama aslında dosyanın bir ASP.NET sayfası olduğunu belirler. Page direktifinin sahip olduğu nitelikler sayfa için gerekli olan çalışma zamanı ve geliştirme zamanı davranışlarını belirleyen değerler içerir. Örneğin **CodeFile** niteliği ile sayfa için çalışma zamanında değerlendirilecek ve özellikle arka plandaki kodların (code-behind) tutulacağı dosyanın adını belirtir. **Inherits** isimli nitelik ile üretilecek olan çalışma zamanı sayfasının hangi tipten türetileceği, **Language** niteliği bu sayfada hangi dil ile geliştirme yapılacağı, **AutoEventWireup** niteliği ile sayfaya ait olay metodlarının otomatik olarak bağlanıp bağlanılmayacağı belirlenmektedir.

<html>, <head> ve <body> gibi standart HTML etiketlerinin bulunduğu sayfadaki en önemli kısımlardan birisi de **<form>** etiketidir. Standart bir ASP.NET sayfasında her zaman için <form> etiketi bulunmalıdır. Sunucu kontrollerinin tamamı mutlaka bu etiket içerisinde yer almalıdır. <form> etiketi ile ilgili söylenecek diğer önemli bir nokta da mutlaka **runat="server"** tanımlamasını içermesidir. Bu sayfa içerisinde HTML etiketleri dışında ASP.NET sunucu kontrollerinin de tanımlamaları yapılabilir. **<asp:** ile başlayan tanımlamalar aslında birer sunucu kontrolü tanımlamasıdır. Bunun dışında yine eski ASP teknolojisinde olduğu gibi HTML etiketlerin arasına **<% %>** işaretleri arasında C# veya VB.NET gibi bir dil ile sunucuda çalışacak kodlar yazılabilir (Fakat bu tip bir kullanımın zorunlu olmayan durumlarda yapılması önerilmez!).



ASP.NET sayfasının sayfa içeriği ve C# kodları birbirinden farklı dosyalarda saklanabileceği gibi daha önceki teknolojilerde olduğu gibi aynı sayfa içerisinde de saklanabilir. Ancak bu yöntem yazılımcı ile tasarımcının paralel çalışmasını zorlaştırdığı ve kod karmaşası oluşturabileceğinden dolayı önerilmez.

Sunucu tarafında çalışacak kodların yer aldığı .aspx.cs uzantılı code-behind dosyasında ise .NET tabanlı bir programlama dili ile yazılacak kodlar yer alır. Burada daha önceden de bahsedildiği gibi sayfanın sınıf olarak tasarımı yer alır. Yazılan sınıfın mutlaka **System.Web.UI** isim alanı altında yer alan **Page** sınıfından kalıtılması gerekir. Visual Studio'da açılan her code-behind dosyasında bir de **Page_Load** adında bir metod eklenmiş olarak gelir. Bu metod sayfanın yaşam döngüsü içerisinde yer alan ve sayfa ilk yüklendiği esnada çalıştırılacak olay metodudur. Yine sayfanın kendine ait diğer olayların metodları, sayfaya eklenecek sunucu kontrollerinin olay metodları, programcının kendi yazacağı metodlar, değişkenler (field) ve özellikler (property) bu kısımda yer alabilir. Böylece sayfada yer alacak HTML tanımlamaları ve sunucu tarafında çalışacak kodlar birbirinden ayrıştırılmakta ve programcının daha derli toplu şekilde işlemlerini yürütebilmesi sağlanmaktadır. Sayfanın yapısı ile ilgili en önemli durumlardan birisi de sayfanın yaşam döngüsüdür. Bu konu bir sonraki başlıkta detaylı şekilde ele alınacaktır.

ASP.NET mimarisinde her sayfanın çalışma zamanında (run-time) derlenmesi (compile) söz konusudur. ASP.NET uygulamalarının en önemli özelliklerinden birisi sunucu tarafında çalıştırılacak kodların derlenmiş şekilde saklanabilmesidir. Bu özellik sayesinde uygulamanın çok daha hızlı çalıştırılabilmesi mümkündür. Zira sunucu tarafında çalışacak olan kodlar C# gibi bir dilde değil IL (Intermediate Language) gibi bir aradilde saklanacak ve bu da daha hızlı bir çalışma şekli sunacaktır. Hazırlanan bir web projesinde sunucuda çalışacak kodları içeren dosyalar, direkt olarak derlenmiş şekilde sunucuda saklanabilir. **Web Uygulamalarının Dağıtımı** başlıklı konuda bu işlemin detayları anlatılacaktır. Uygulama dosyaları sunucuda derlenmiş şekilde saklanmasa dahi ASP.NET bir dosyaya gelen ilk istekte dosyanın derlenmiş halini oluşturup sistemde saklar. Bu dosyaya sonradan gelen isteklerde ise dosya tekrar derlenmez ve böylece istemciye daha hızlı cevap verilebilir.

ASP.NET sayfalarında iki türlü kodlama yöntemi bulunmaktadır: **code-behind kodlama**, **inline kodlama**. Önceki sayfalarda yer alan örnekte de kullanılan code-behind yöntemindeki mantık, HTML kodlarının ve sunucuda çalışacak kodların farklı iki dosyada saklanması esasına dayanır. Fakat ASP ve PHP gibi teknolojilerde olduğu gibi ASP.NET'te de sunucuda çalışacak kodların HTML kodları ile içiçe yazılması da sağlanabilir. **Inline kodlama** adı verilen bu yöntemde tek bir .aspx dosyasında hem HTML kodları hem de sunucuda çalışacak C# gibi bir dil ile yazılmış kodlar saklanabilir. Fakat inline kodlama ASP.NET'te pek kullanılan bir yol değildir. Code-behind kodlama çoğu noktada büyük faydalar sağladığı için bu yöntem kullanılmaktadır.

Code-behind yönteminde sunucuda çalışacak kodları içerisinde barındıran ayrı bir fiziki dosya söz konusudur. Aşağıda bu şekilde hazırlanmış bir sayfanın kod parçalarını saklayan iki dosya görülmektedir.

Default.aspx (HTML elementleri ve sunucu kontrollerinin tanımlamaları)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click"/>
  </form>
</body>
</html>
```

Default.aspx.cs (Sunucuda çalışacak kodlar)

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write("Code-behind yöntemi ile kodlanmış sayfadasın...");
    }
}
```

Görüldüğü gibi kodlar birbirinden ayrıştırılmış ve iki farklı dosyada saklanmıştır. Burada iki dosyayı birbirine bağlayan noktalar sınıfın adı ve code-behind dosyasının adıdır. Default.aspx sayfasındaki **Page** direktifi içerisindeki **CodeFile** niteliği arka planda hangi dosyaya bağlanılacağını, **Inherits** niteliği ise bu dosya içerisindeki hangi sınıftan miras alma işleminin yapılacağını belirler.

ASP.NET sayfalarında kullanılan bir diğer yöntem de inline kodlamadır. Buradaki esas ise tüm kodların tek bir dosyada saklanmasıdır. Aşağıda bu şekilde oluşturulan bir sayfanın kodları görülmektedir.

Default.aspx (HTML elementleri, sunucu kontrollerinin tanımlamaları ve sunucuda çalışacak kodlar)

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
```

```
{
    Response.Write("Inline kodlama ile oluşturulmuş sayfadasın.");
}
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click"/>
    </form>
</body>
</html>
```

Yukarıdaki örnek incelendiğinde, sunucu tarafında çalışacak kodların **<script runat="server">** şeklinde açılmış bir blok içerisine yazıldığı görülebilir. <script> etiketi içerisine runat="server" niteliğinin eklenmesi bu kodların sunucu tarafında ele alınacağını bildirmektedir.

EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

ASP.NET Dosya ve Klasör
Tipleri



Microsoft Türkiye

Açık Akademi

ASP.NET Dosya Tipleri

ASP.NET ile geliştirilen uygulamalarda, site ziyaretçilerinin görüntüleyebileceği temel dosya tipi **aspx** olmakla birlikte, yine uygulamalar içerisinde sıklıkla kullanılan başka dosya tipleri de bulunmaktadır. ASP.NET uygulamalarında kullanılan dosya tiplerinden bazıları ve işlevleri aşağıda detaylı bir şekilde açıklanmıştır.

aspx: Temel işlemlerin yapılacağı ve kullanıcıların görüntüleyebileceği dosyalardır. Metin tabanlı bir dosya biçimi olup, web formlarının oluşturulmasında kullanılır. aspx dosyaları, HTML kodlarını, sunucu kontrollerini ve kullanıcı kontrollerini içerir. HTML kısmında sayfa içerisindeki içeriklerin nerede ve nasıl bulunacağı belirlenir. Bu kısımda HTML etiketleri dışında ASP.NET sunucu ve kullanıcı kontrolleri de tanımlanabilir. aspx sayfalarında <script> kısımları içerisinde C#, VB.NET veya .NET ortamındaki farklı bir dil ile kodlar yazılabileceği gibi, bu kod kısımları farklı dosyalarda da tutulabilir.



.aspx dosyaları ile birlikte çalışacak olan C#, VB.NET vb. kodlarının farklı bir dosyada saklanması (code-behind kodlama) durumunda kullanılan dil C# ise dosya uzantısı **aspx.cs**, VB.NET ise dosya uzantısı **aspx.vb** şeklinde sağlanır. Bu dosyalar içerisinde sadece sunucuda çalışacak kodlar yer alabilir.

- **ascx:** aspx sayfaları içerisinde kullanılacak kullanıcı kontrolü (user control) dosyalarıdır. Yapısı bir aspx dosyası ile hemen hemen aynıdır. .aspx dosyalarında bulunan <html>, <head> ve <body> gibi HTML elementleri ascx dosyalarında yer almamalıdır. Bu dosyalar tek başlarına çalıştırılmazlar, sadece aspx veya master uzantılı dosyaların içinde bir kontrol olarak kullanılabilirler.
- **master:** ASP.NET 2.0 ve sonraki sürümlerde aspx sayfaları için temel şablon olarak kullanılan dosyalardır. MasterPage adı verilen bu sayfaların genel yapısı aspx sayfaları ile hemen hemen aynıdır.
- **asax:** Uygulamanın çalıştığı süre içerisinde uygulamayla ilgili bazı olayları (uygulamanın ilk başlaması, uygulamada hata oluşması olayları gibi) yakalamak için kullanılan dosya türüdür. Yakalanan olaya göre işlemler yapılmasını sağlar. Uygulamalarda varsayılan olarak **Global.asax** dosyası kullanılır.
- **config:** Projenin kullanacağı genel ayarları içeren XML tabanlı bir dosyadır. Uygulamanın genelinde geçerli olacak ayarlar bu dosyada saklanmaktadır. Bu dosya uygulamada **web.config** olarak isimlendirilmektedir. Uygulamanın ana dizinindeki web.config dosyası uygulama ile ilgili genel ayarları, bir klasör içerisinde oluşturulan web.config dosyası da, o klasörde geçerli olacak ayarları saklamaktadır.
- **sitemap:** Menü kontrollerinin çalışması için gerekli bilgileri içeren XML tabanlı bir dosyadır.
- **skin:** Sunucu kontrollerinin görünümünün ayarlanabileceği stil dosyalarıdır. Bu dosyalar **App_Themes** adındaki özel bir klasör içerisine oluşturulan tema klasörlerinde saklanmaktadır.
- **cs:** İçerisinde sunucuda çalıştırılacak sınıfların, yapıların (struct) vb. Tiplerin oluşturulabileceği kod dosyasıdır. C# dili ile geliştirilen tipler için dosya uzantısı **cs** olurken, VB.NET dilinde ise bu dosyanın uzantısı **vb** olmalıdır. Oluşturulan sınıf ve diğer tiplere uygulamanın her yerinden erişilmesi isteniliyorsa bu dosyalar **App_Code** isimli özel bir klasör içerisinde tutulmalıdır.
- **dll:** Derlenmiş kod kütüphanesi dosyalarıdır. Daha önceden hazırlanan kod kütüphaneleri **Bin** isimli özel bir klasöre eklenerek bu kütüphane içerisindeki tipler (sınıflar, yapılar, enum sabitleri gibi) kullanılabilir.
- **resx:** Uygulama içerisinde metin, resim, ikon, medya içerikli dosyaları saklayabilen XML formatındaki bir dosya şeklindedir. İçerisinde anahtar kelime-değer (key-value) çiftleri şeklinde saklanan bu verilere

uygulamadaki dosyalardan erişilebilmektedir. Özellikle farklı diller için geliştirmeler yapılırken bu kaynak (resource) dosyası kullanılmaktadır.

- **asmx**: Web servisleri dosyalarıdır. Web servisleri farklı platformlardaki, farklı uygulamaların kullanabildiği servislerdir. aspx sayfalarında olduğu gibi asmx dosyalarının da arka tarafında C# ve VB .NET gibi dillerle yazılmış kodlar yer almaktadır.



.aspx uzantılı dosyalar ile **.master** ve **.ascx** uzantılı dosyalar temel olarak aynı yapıdadırlar. .aspx uzantılı dosyaların arka planında çalışan .aspx.cs uzantılı dosyalar olduğu gibi;

- .master dosyaların arka planında çalışan **.master.cs**
- .ascx dosyalarının arka planında çalışan **.ascx.cs**

uzantılı dosyalar da bulunmaktadır. Bu dosyalar yine sunucuda çalışacak olan kodları içermektedir.

ASP.NET Klasör Tipleri

ASP.NET uygulamalarında özel amaçları olan ve belirli tipteki dosyaları içerebilen klasörler bulunmaktadır. Bu klasörlerin isimleri ve ne amaçla kullanıldığını aşağıda görebilirsin.

- **Bin**: İçerisinde web uygulamasında kullanılacak derlenmiş .NET assembly dosyalarını (genellikle dll dosyaları) bulundurmaktadır. Bir web sayfası çalışırken, sunucu ilk olarak bu klasör içerisindeki kod kütüphanelerine bakarak sayfaları arayacaktır.
- **App_Code**: Bin klasörüne benzer bir görev üstlenmekle beraber, derlenmemiş kod dosyalarını içerisinde tutar. Bu klasörde **.cs**, **.vb** gibi sınıf ve kod dosyaları, **.wsdl** veya **.xsd** türünde dosyalar bulunabilir. Çalışma zamanında bu klasördeki kodlar derlenir ve sonraki isteklerde derlenmiş kodlar çalıştırılır.
- **App_Themes**: Web uygulaması içerisinde tanımlanan ve sayfalardaki elementler tarafından kullanılan temaları saklar. Bu temaların içerisinde de **.skin** ve **.css** dosyaları bulunur.
- **App_Data**: Uygulama içerisinde kullanılabilecek olan **SQL Server veritabanları (mdf)**, **Microsoft Access (mdb veya accdb)** ve **XML** gibi veri ile ilgili dosyaları içerir.
- **App_LocalResources**: Sayfa bazında kaynak (resource) tanımlanırken kullanılan dosyaları saklar. Genellikle yerelleştirme (localization) işlemlerinin yapılacağı kaynak dosyalarını saklamak için kullanılır.
- **App_GlobalResources**: Bir web uygulamasındaki her dosya tarafından erişilebilen genel (global) kaynakların saklanması için kullanılır.

EĞİTİM :

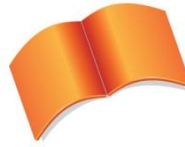
**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

**Web Sayfalarında Olay
Kullanımı**



Microsoft Türkiye

Açık Akademi

Web Sayfalarına Kontrol İlave Etmek

Web sayfalarında olayların nasıl ele alındığını anlatmadan önce ilk olarak bir web sayfasına nasıl kontrol ekleneceği üzerinde durmak mantıklı olacaktır. Web sayfalarına kontrol ilave etmenin birkaç farklı yolu vardır. Bunlardan ilki ve en kolay olanı Visual Studio ortamında kontrolü ToolBox'tan sürükleyip sayfa içerisindeki istenilen bir yere bırakmaktır ya da ToolBox'ta istenilen kontrol üzerine çift tıklayarak o anda imlecin bulunduğu yere kontrol eklenebilir.

Visual Studio ortamında sayfaya kontrol ilave etmenin bir diğer yolu da HTML tarafına geçip kod yazarak kontrol eklemektir. Eğer isterseniz, HTML kodları tarafında da ToolBox üzerindeki herhangi bir kontrolü sürükleyip istenilen yere bırakarak tasarıma devam edebilirsiniz.

[Buraya sen de dene şeklinde bir şey ekleyebiliriz, ilk olarak bir tane label'ı design tarafında sürükleyip bıraktıralım, sonra aynısını html tarafında yaptıralım. Sonra farklı isimde bir label'ı html tarafında yazdıralım.]

Web Sayfalarında Olay (Event) Kullanımı

.NET tabanlı dillerde, olay kavramı uygulamaların önemli temellerinden birini oluşturmaktadır. Bu durum, hem web uygulamaları, hem de Windows uygulamaları için geçerlidir.

Olaylar (events) uygulamanın çalışması sırasında bir düğmenin tıklanması, bir formun açılması ya da bir seçeneğin değiştirilmesi gibi eylemlerden sonra istenilen bir metodun çalışmasını sağlayan sınıf (class) üyeleridir. **“.NET'te her şey bir nesnedir”** sözüne uygun olarak sayfaya bir buton sürüklenip bırakıldığında, aslında .NET Framework uygulama geliştiricileri tarafından yazılmış olan **Button** sınıfından bir nesne örneği oluşturulmuş olur. Uygulamayı geliştiren kişi, bu nesne örneğinin görsel kısmını görür. Ayrıca ID, Text gibi değiştirilebilen ve elde edilebilen birçok özellik, yine bu sınıfın üyeleridir. Visual Studio üzerinde sayfadaki bir butona çift tıklayarak yazılan metodun, çalışma zamanında kullanıcı tarafından tıklandığında çalışmasını garanti eden ise Click olayıdır. Butona çift tıklandığı anda ilgili butonun html koduna **OnClick = “MetotAdi”** şeklinde bir atama yapılır.

ASP.NET sunucu kontrollerinin birçok olayı bulunabilmektedir. Bu olaylardan bazıları programcının birçok işlemi gerçekleştirebilmesini sağlar. Bu olaylardan sadece bir tanesi ise kontrolün varsayılan olayıdır. Örneğin Button kontrolünün varsayılan olayı Click, DropDownList kontrolünün varsayılan olayı ise SelectedIndexChanged'dir. Visual Studio ortamında bir kontrolün üzerine çift tıklandığında otomatik olarak varsayılan olay için bir metod yazılır. Bir kontrole ait olay metodunun genel yapısı aşağıdaki gibidir.


```
public void kontrolAdi_OlayAdi(object sender, EventArgs e)
{
}
}
```

Oluşturulan metodun adı standart olarak kontrolunAdi_OlayinAdi şeklinde yazılmaktadır. Örneğin btnTikla isimli bir butonun Click olayı için btnTikla_Click şeklinde bir metod oluşturulur. Bu metod ismi değiştirilebilir. Bu özel metodun iki tane parametresi vardır. Bunlardan ilki **object** türündeki **sender** parametresidir. sender, olayın

üyesi olduğu nesnedir. Bir başka deyişle olayın tetiklenmesine neden olan kontroldür. Örneğin buton kontrolünün olayında sender parametresi tıklanan butonun nesne örneğini object tipinden getirir. İkinci parametre ise; **EventArgs** türündeki **e** parametresidir. Bu parametre ise o olay ile ilgili oluşan bazı önemli bilgileri içerir. e parametresi taşıdığı özellikler (property) aracılığıyla metoda önemli bilgiler taşıyabilmektedir. Bu olaya göre bazen sorgudan etkilenen kayıtların sayısı olabileceği gibi, bazen de yapılan işlemin durdurulmasını sağlayan bir boolean değer olabilir. Kontrol olaylarının metodlarında ilk parametre genelde object tipinden sender olur, fakat ikinci parametre EventArgs dışında farklı tiplerde olabilmektedir.



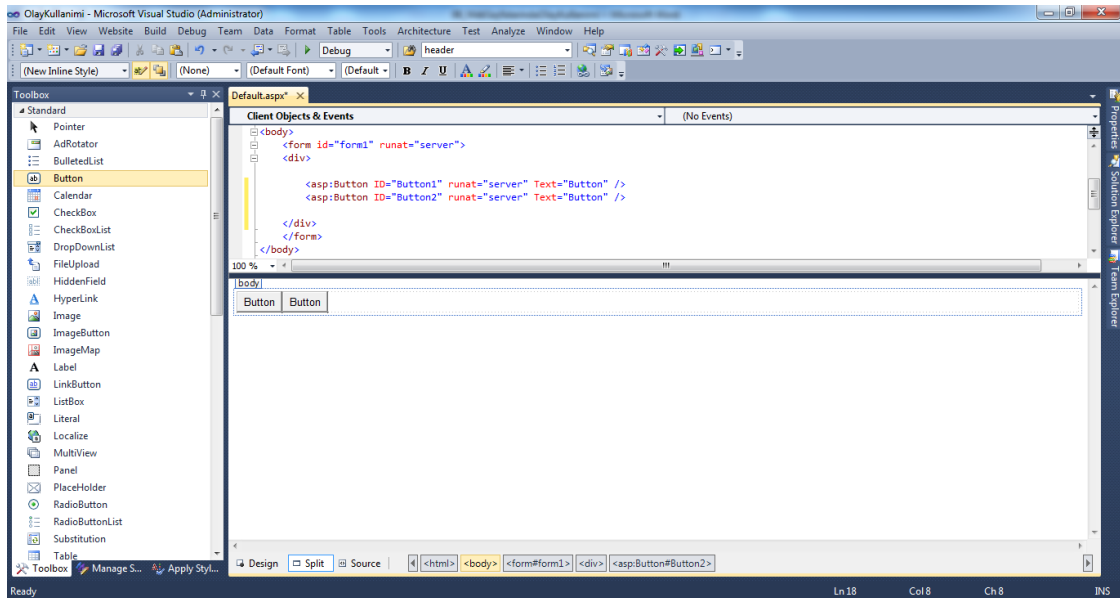
Bir kontrol olayının metodundaki ikinci parametre EventArgs tipinden olduğunda bu parametre önemli hiçbir bilgi taşımaz! Fakat olayın metoda bağlanmasını sağlayan temsilci tipinden dolayı bu parametrenin aktarılması zorunludur. Eğer bir olay metodunun ikinci parametresi EventArgs dışındaki bir tipten ise bu parametrenin oldukça önemli veriler taşıdığını unutmamak gerekir.

Visual Studio'da Designer ekranında, bir kontrole çift tıklayarak o kontrolün varsayılan olayına ulaşabilir ve ilgili metoduna istenilen kodlar yazılabilir. Bir kontrolün olaylarına ulaşmanın diğer bir yolu da; istenilen kontrolün özelliklerini görüntüleyip, Properties penceresindeki  simgesine tıklamaktır. Böylece listelenen olaylardan herhangi birini seçerek yine Visual Studio'nun otomatik oluşturacağı olay metodu içerisinde gerekli işlemler yapılabilir.



Her kontrolün birden fazla olayı olabilir. Bir kontrole çift tıklandığında ele alınacak olay, o kontrolün varsayılan olayıdır. Button kontrolü için varsayılan olay **Click** iken, diğer kontroller için farklı olaylar olabilir.

Gelin birlikte bir Button kontrolünün Click olayını ele alalım. Visual Studio üzerinde bir tane Web Projesi açalım ve sayfaya iki tane Button ekleyelim. Button'ların ID özellikleri üzerinde herhangi bir değişiklik yapmayarak Button'ların ID'lerini Button1 ve Button2 olarak bırakalım. Sayfamız dizayn ve HTML kısımları aşağıdaki gibi görünüyormalıdır.



Şimdi ilk olarak soldaki, yani ilk eklemiş olduğumuz Button1 isimli Button'a çift tıklayalım ve Click olayını ele alıp çalışacak olan metodu Visual Studio bizim için oluştursun. Her şey varsayılan şekilde gerçekleştiği için Visual Studio bizim için aşağıdaki şekilde bir kod yazacaktır.

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


Gördüğün gibi Visual Studio **Button1_Click** isimli bir metod oluşturdu ve Button1 tıklandığında bu metodun içerisine yazılacak olan kod çalışacaktır. Peki bunu nereden anlıyoruz? Şimdi Button1'in HTML kodlarının Button'a çift tıklanmadan önceki halini gözümüzün önüne getirelim. Kodlar aşağıdaki gibiydi. Yani Button2'nin şu andaki haline benzemekteydi.

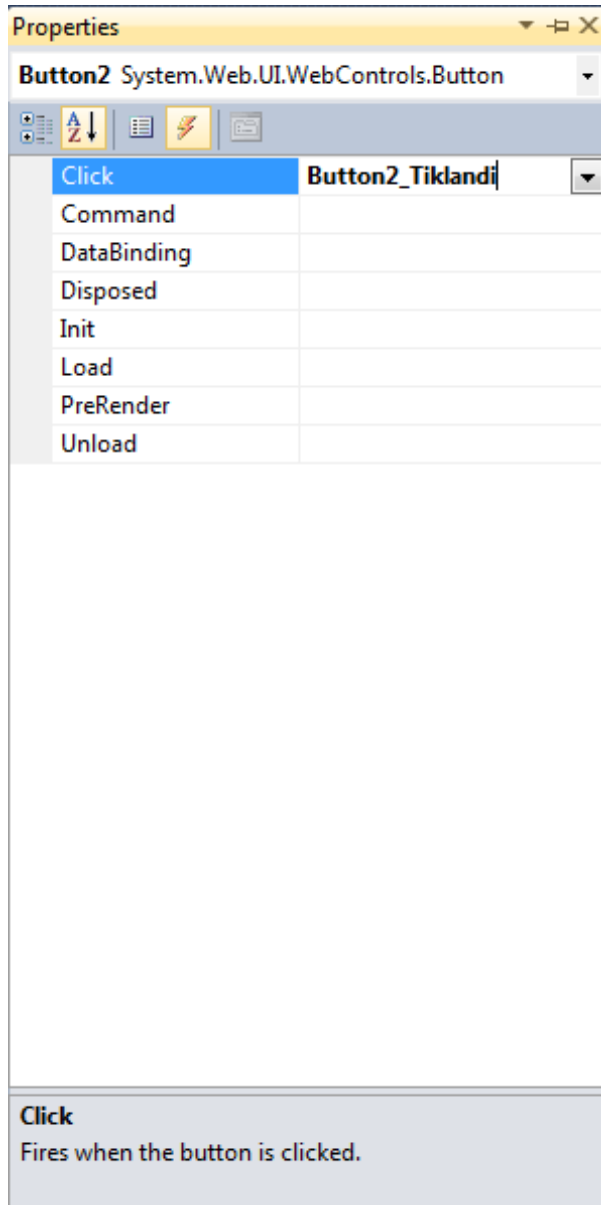
```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

Şimdi bir de Button1'e çift tıklandıktan yani Click olayı ele alındıktan sonraki HTML kodlarını gözümüzün önüne getirelim.

```
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
```

Gördüğün gibi onclick isminde bir etiket eklenmiş ve tahmin edeceğin gibi Button'a tıklandığında çalışacak olan metodun adı da onclick'in içerisinde **onclick="Button1_Click"** şeklinde belirtilmiş. Yani Button1'e tıklandığında çalıştırılacak olan metodun hangisi olduğunu buraya bakarak anlayabiliriz. Bu örnekte sadece iki tane Button var, hangi Button'un Button1 hangisinin Button2 olduğunu kolaylıkla anlayabiliyoruz. Daha karmaşık bir durumda yani daha kalabalık bir sayfada Button isimlerini daha anlamlı vermek gerekeceği gibi olay tetiklendiğinde çalışacak olan metodların adını da aynı şekilde daha anlamlı bir biçimde vermek istenebilir. Bu durumda nasıl bir yol izleyeceğimizi de örnekleyelim.

Button2 üzerine tıklayalım ve özellikle penceresine geçelim. (Özellikle penceresi açılmadıysa Button2'ye tıkladıktan sonra F4 tuşuna basabilirsin.) Button2'nin özellikler penceresinde olaylar (Events) bölümüne geçelim. (Olaylar bölümüne geçmek için hatırlayacağın üzere  simgesine tıklıyorduk.) Olaylar bölümünde Button kontrolünün Click dışında da olaylarının olduğu görülebilir. Bunlardan istediklerimizi ele alıp çalışacak olan metodu oluşturabiliriz. Click olayının yanındaki metin kutusuna istediğimiz ismi yazıp Enter tuşuna basalım ve Visual Studio bizim için metodu oluştursun. Metod ismi olarak **Button2_Tiklandi** diyebiliriz. Aşağıdaki resimde Button2'nin özellikler penceresini görüyorsun.



Button2'nin özellikler penceresinde, olaylar bölümünde Click olayına istenilen isim belirtildikten sonra Visual Studio bizim için metodu oluşturacaktır. Metod oluşturulduktan sonra Visual Studio tarafından otomatik olarak kod tarafına yönlendirilip Metodun içeriğini belirtmemiz sağlanacaktır. Belirttiğimiz isimdeki metod kod tarafında aşağıdaki şekilde görünecektir.

```
protected void Button2_Tiklandi(object sender, EventArgs e)
{
}
```

Web Sayfalarının Olay Tabanlı Yaşam Döngüsü

Her web sayfası sunucu tarafında bir .Net sınıfıdır. Çünkü içeriklerinde başka sınıflara ait örnekler, üye metodlar, olaylar, özellikler vb... vardır. Bir .NET sınıfında nasıl yapılandırıcı ve sonlandırıcı metodlar varsa web sayfalarında da benzer nitelikteki olaylar vardır. Şu ana kadar gerçekleştirmiş olduğumuz örneklerde Page_Load isimli bir metod dikkatnizi çekmiştir. Bu metodu standart bir .NET sınıfının içerisinde yer alan “**Yapılandırıcı**

Metod” gibi düşünebiliriz, yani sayfa her yüklendiğinde bu metod içerisine yazılacak olan kodlar çalışacaktır. Bir ASP.NET sayfasında Page_Load gibi bir kaç tane daha olay vardır ve sayfanın oluşturulma evrelerinde bu olaylar belli bir sıra ile tetiklenir. Bu olayları ele almak için kod tarafına aynı isimde ve Page_Load metodu ile aynı imzada bir metod oluşturmak yeterlidir. Aşağıdaki kod parçasında Page_Load metodunu görüyorsunuz, az sonra bahsedecek olduğumuz Page_Init olayını ele almak için yapman gereken tek şey, bu metodun aynısını oluşturmak ve ismini Page_Init olarak değiştirmektir. Burada daha hızlı kod yazmak için Page_Load’ı kopyalayıp daha sonra aynı sınıf içerisine yapıştırarak ismini değiştirebilirsiniz.

```
protected void Page_Load(object sender, EventArgs e)
{
}
}
```

Bir web sayfasının yaşam döngüsünde ele alınabilecek olaylar düşünüldüğünde aşağıdaki sıralama ile ilerleyebiliriz. Burada belirtilen sıralama, olayların tetiklenme sırasıdır. Yani, sayfa oluşturulurken buradaki olaylardan ilk olarak Page_PreInit son olarak da Page_Unload olayları tetikleniyor. Burada **Page_PreInit** ve **Page_Init** olayları kontrollerin nesne örneklerinin oluşturulması sırasında çalışan olaylardır. Sayfa üzerindeki kontrollerin nesne örnekleri oluşturulmadan önce müdahale edilecek olan işlemler bu adımda gerçekleştirilmelidir. **Page_Load** kontrollerin nesne örneği oluşturulduktan sonra tetiklenen olaydır ve Visual Studio her sayfada varsayılan olarak bu özelliği ele almıştır. Bu olayın tetiklenmesi ile birlikte kontrollerin nesne örnekleri oluşturulmuş demektir ve kontrollerin özelliklerine erişilip gerekli işlemler gerçekleştirilebilir. **Page_PreRender** ise sayfanın HTML çıktısı oluşturulmadan hemen önce tetiklenecek olan olaydır, bu adımda HTML çıktısına kod yazarak müdahale edebilirsiniz. **Page_UnLoad** sayfaya ait Html çıktısı üretildikten sonra çalışır. Bu nedenle burada HTML çıktısına müdahale etme şansı artık kalamamaktadır. Dolayısıyla, sayfa içerisinde kullanılan başka managed kaynakların kapatılması ve sisteme iade edilmesi için ideal bir olay metodudur.

- Page_PreInit
- Page_Init
- Page_Load
- Button için Click / Change Olay Metodları (Eğer tetiklenmişler ise)
- Page_PreRender
- Page_UnLoad

Burada dikkat edilirse **Page_Load** ve **Page_PreRender** olayları arasında sayfa üzerindeki bileşenlere ait **Click** ve **Change** olaylarının yer aldığı görülmektedir.

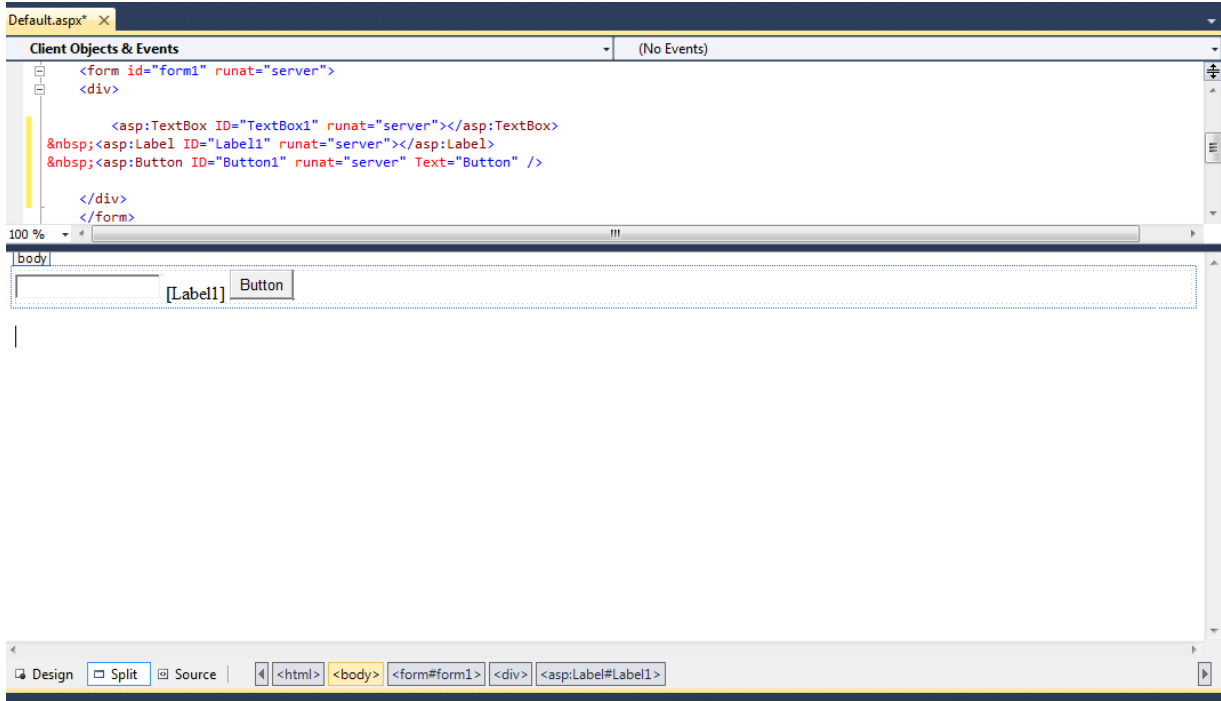


Bir web sayfasının yaşam döngüsünde kontrollerin click/change olayları, sayfa ilk talep edildiğinde tetiklenmez. Kullanıcı ilk talebi sonrasında istediği sayfayı gördükten sonra kontrollere ait olay metodlarını tetikletebilir.

Talep edilen her web sayfası, sunucu tarafında bir nesne olarak üretilir, çalıştırılır, HTML çıktısı alınır ve yok edilir (Dispose).

Web sayfasına ait olay metodları ile (ki bunların yaşam döngüsü içerisinde önemli bir rolü vardır), sayfanın Page direktifinde yer alan **AutoEventWireup** niteliğinin değeri arasında önemli bir ilişki vardır. Sayfaya ait olay metodlarının otomatik olarak bağlanmasını sağlamak için AutoEventWireup özelliğinin değerinin true olması gerekmektedir. Bu özellik False olarak ayarlandığında sayfaya ait olay metodları otomatik olarak ele alınmaz, bu durumda yazılım geliştirici kendi metodlarını yazarak bunu sayfanın page direktifinde yaşam döngüsünde yer alan istediği olaya bağlar ve çalıştırılmasını sağlar.

Şimdi bir örnekle konuyu pekiştirelim. Visual Studio üzerinde yeni bir web projesi açalım ve sayfa üzerine bir tane **TextBox** bir tane **Label** ve bir tane de **Button** kontrolü sürükleyip bırakalım. Label kontrolünün Text özelliğini boşluk olarak ayarlayalım ve ilk başta hiç bir şey görüntülemesin. Buradaki amacımız senin de anladığın gibi Button'a tıkladığında TextBox'ın içindeki yazının Label'a kopyalanması. Projeyi açıp üzerinde gerekli eklemeleri yaptıktan sonra Visual Studio'da aşağıdaki gibi bir görünüm ile karşılaşmalısın.



Şimdi bir önceki konuda konuştuğumuz gibi Button'un Click olayını ele alalım ve aşağıdaki kodları yazalım. (Click olayını ele almak için Button'a çift tıklayabilirsin.)

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

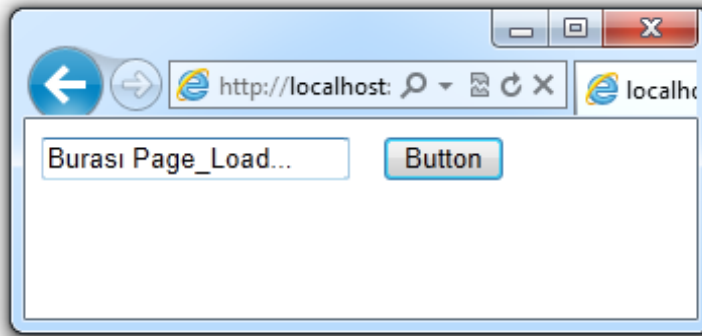
Bu kodları yazdıktan sonra ctrl+f5 tuş kombinasyonu ile uygulamayı çalıştırıp test edebiliriz. Açılan sayfada TextBox'a bir şeyler yazalım ve Button'a tıklayalım. Bu işlemin ardından TextBox'a yazmış olduğumuz yazının aynen Label'a kopyalandığını göreceğiz.



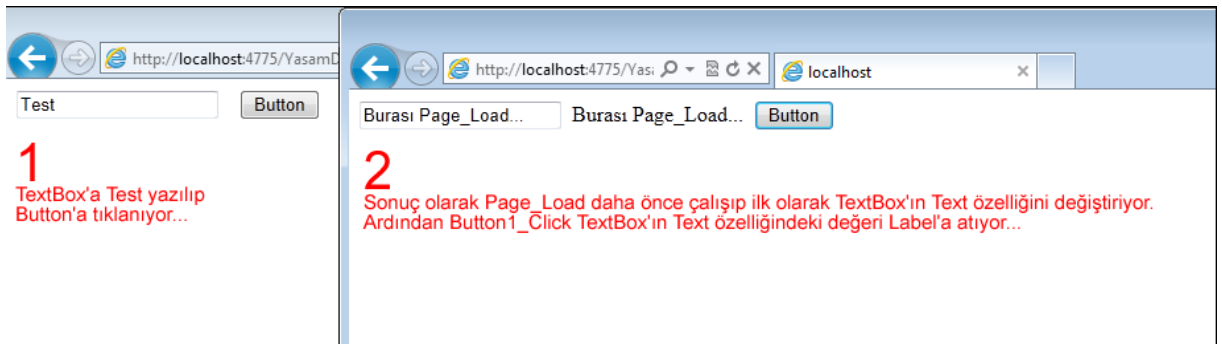
Şimdi sayfanın şu anda boş olan Page_Load bölümüne gidelim ve aşağıdaki kodu yazalım. Bu adımdan sonra da yine yukarıdaki gerçekleştirmiş olduğumuz işlemi gerçekleştirelim. Bakalım sonuçta bir değişme olacak mı?

```
protected void Page_Load(object sender, EventArgs e)
{
    TextBox1.Text = "Burası Page_Load...";
}
```

Sayfayı ilk çalıştırdığımız anda aşağıdaki görünüm ile karşılaçağız. Page_Load kısmına yazılan kodlar çalışacak ve TextBox'ın Text özelliğini "Burası Page_Load..." şeklinde değiştirecek.



Bir önceki örnekte olduğu gibi TextBox'a Test yazıp Button'a tıklandığında ise aşağıdaki sonuç ile karşılaşacaksınız. Sanki istediğimiz olmadı. Biz Button'a her tıkladığımızda sayfa yeniden sunucuya gönderiliyor ve sayfanın nesne örneği yeniden oluşturuluyor. Sayfanın nesne örneğinin yeniden oluşturulması ile birlikte daha önce bahsetmiş olduğumuz olaylar sırası ile yeniden tetikleniyor. Page_Load olayının kontrollerin olaylarından daha önce tetiklendiğini hatırlıyor olmalısın. Bu örnekte de görüldüğü gibi Button'a tıklandığında Button1_Click metodundan önce Page_Load metodu çalıştırılıyor ve ilk olarak TextBox'ın Text özelliğine "Burası Page_Load..." değeri atanıyor ardından Button1_Click çalışarak bu değeri Label1'in Text özelliğine atıyor.



Peki sayfa her yüklendiğinde TextBox'ın Text özelliğini boşaltmak isteseydik nasıl bir yol izlerdik. Senin de tahmin edeceğin gibi Page_Load'da "Burası Page_Load..." yazmak yerine " " yazmak yeterli olacaktı. Ancak burada yine bir problem söz konusu, bu sefer bir önceki örnekten hatırlayacağın gibi Label'da da her zaman boşluk karakteri atanacak. Bu problemin önüne nasıl geçebiliriz? Keşke sayfanın ilk sefer yüklenmesi ile sayfa üzerindeki kontroller aracılığı ile sunucuya geri gönderilmesini algılayabiliyor olsaydık değil mi? Üzülmeysin, sıradaki konu bunu, yani **PostBack** kavramını ele alıyor...

EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

PostBack Kavramı



Microsoft Türkiye

Açık Akademi

PostBack Kavramı

PostBack, ASP.NET ile web teknolojilerinde kullanılmaya başlanan bir kavramdır. Bir ASP.Net sayfası içerisinde, bir olayın tetiklenmesi sonucunda sayfanın, içerisindeki bilgilerle birlikte sunucuya gönderilmesi ve sunucuda yeniden üretilen sayfanın geri getirilmesi işlemine **PostBack** denilmektedir. Burada tetiklenen olay bir butona tıklanması veya açılır listeden bir seçeneğin seçilmesi olabilir. Bir .aspx sayfasına yapılan ilk talep dışındaki bütün taleplerde sayfa postback işlemini gerçekleştirmiş olur.

Bir sayfanın ilk istek ile mi, PostBack işlemi sonucunda mı çağırıldığını kontrol etmek için, **Page** sınıfının **IsPostBack** özelliği kullanılır. Aşağıda **Page.IsPostBack** ifadesinin Page_Load içerisinde örnek kullanımı yer almaktadır.

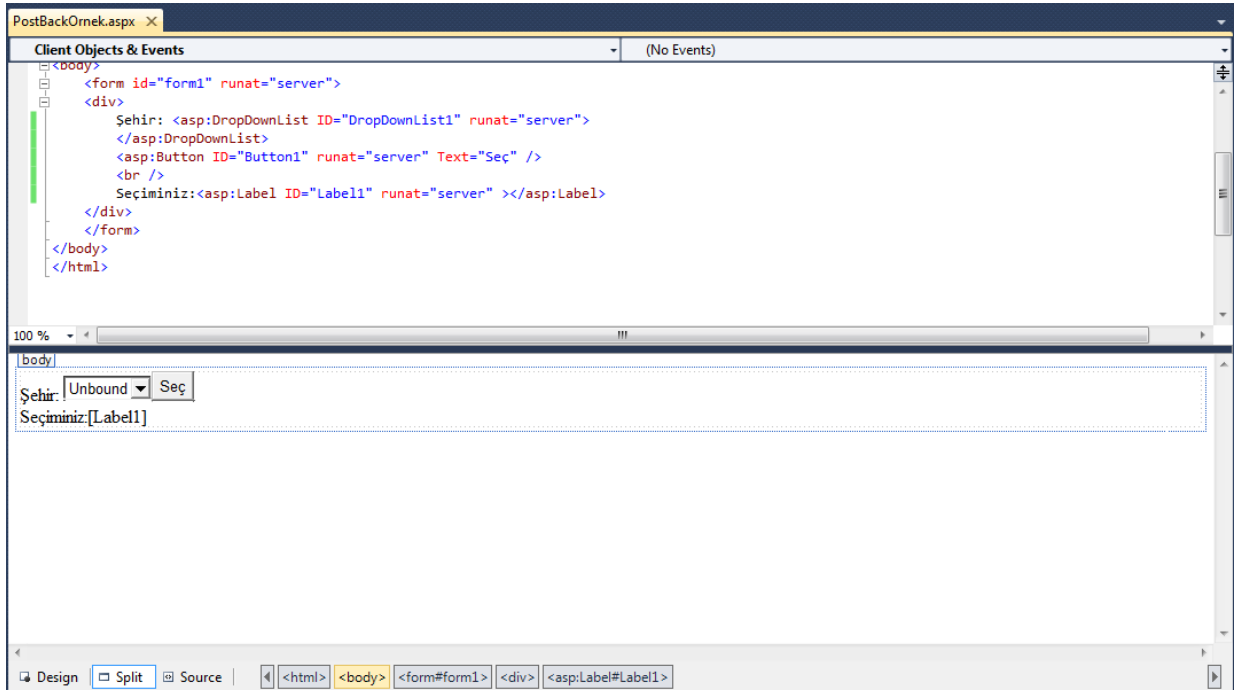
```
public void Page_Load(object sender, EventArgs e)
{
    if(Page.IsPostBack == false) // sayfa postback yapılmamış ise
    {
        // sayfa ilk sefer yüklendiğinde gerçekleştirilmesi gereken işlemler
    }
}
```

Page.IsPostBack ifadesi boolean bir değer döndürecektir. Eğer sayfa ilk defa çalışıyorsa **False**, PostBack işlemi sonucunda çalışıyorsa **True** değeri döner. IsPostBack özelliği ve PostBack kavramını daha iyi anlayabilmek için yeni bir web sitesi projesi açalım. Proje içerisinde yer alan bir sayfanın Page_Load metoduna aşağıdaki kodları yazalım. Design tarafına geçip sayfaya bir tane Button ekledikten sonra uygulamayı çalıştıralım.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Sayfa PostBack yaptı mı : ");
        Response.Write(Page.IsPostBack.ToString());
    }
}
```

Sayfa ilk çalıştırıldığında oluşacak olan metin, **“Sayfa PostBack yaptı mı : False”** şeklinde olacaktır. Sayfa üzerindeki Butona tıklandığında, buton sayfadaki bilgileri POST metoduyla sunucuya göndereceği için; sayfada PostBack işlemi gerçekleşecek ve sayfadaki yeni metin **“Sayfa PostBack yaptı mı : True”** şeklinde olacaktır.

Bir sayfaya yapılan ilk talepten sonraki taleplerde PostBack işleminin gerçekleştiğini gördük. Şimdi birlikte ufak bir uygulama geliştirerek PostBack kavramını daha iyi anlayalım. Bir önceki açmış olduğumuz projeye yeni bir sayfa ekleyelim. Sayfa üzerinde bir tane DropDownList kontrolü, bir tane Label ve bir tane de Button kontrolü ekleyeceğiz. Burada amacımız; DropDownList içerisinde bir kaç tane şehir listeleyp, kullanıcıya bunlardan birini seçtirmek ve ardından seçimi Label’da görüntülemek olacaktır. İlk olarak sayfamıza gerekli kontrollerimizi ekleyelim. Sayfaya gerekli kontroller eklendikten sonra sayfanın görünümü aşağıdaki resimdeki gibi olacaktır.



Sayfanın HTML kısmını hazırladıktan sonra sıra geldi kod tarafındaki işlemleri yapmaya. Şu anda projeyi çalıştırıp sayfaya göz attığınızda DropDownList kontrolünün içinde herhangi bir öğe olmadığını göreceksiniz. Bu gayet doğal bir durum çünkü DropDownList'i oluşturduk ancak içine herhangi bir öğe eklemedik. İlk adım olarak sayfanın Page_Load metodu içerisinde DropDownList'in öğelerini dolduralım. Burada örneği kısa tutmak adına sadece üç tane şehir ekliyoruz ancak ilerideki konulardaki bilgileri de edindikten sonra bir veritabanında saklı olan şehirleri de kolaylıkla DropDownList'e ekleyebilirsiniz. Biz şimdilik sabit olarak 3 tane şehir ekleyeceğiz.

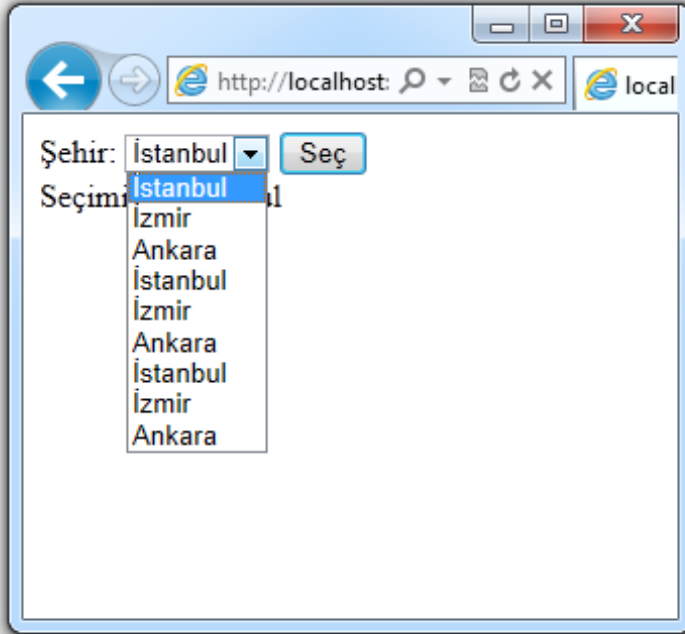
```
protected void Page_Load(object sender, EventArgs e)
{
    DropDownList1.Items.Add("İstanbul");
    DropDownList1.Items.Add("İzmir");
    DropDownList1.Items.Add("Ankara");
}
```

Bu kodları Page_Load'a ekledikten sonra sayfayı çalıştırırsan artık DropDownList'in boş gelmediğini göreceksin. Şimdi sıra geldi Button'a tıklandığında Label'a seçilen ilin yazdırılması. Bu işlem için Button'un click olayını ele almamız gerekiyor, daha önceki bölümlerden de hatırlayacağınız gibi Visual Studio'da Button'a çift tıklayarak Click olayını ele alabiliyorduk. Button'a çift tıklayalım ve Visual Studio bizim için gerekli metodu oluştursun. Metodun içerisine aşağıdaki kodları yazarak kullanıcının seçtiği değeri Label'a aktarabiliriz.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = DropDownList1.SelectedItem.ToString();
}
```

Kodlardan da anlayacağınız gibi DropDownList'te kullanıcının seçmiş olduğu değeri SelectedItem özelliği ile elde edebiliyoruz. Bu özelliği ToString() metodu ile string tipine dönüştürdüğümüzde direkt metni elde ediyoruz ki bu da istediğimiz sonuç. Uygulamayı şimdi çalıştırıp herhangi bir şehri seçip Button'a tıkladığında seçilen şehir Label'a gelecektir. Şehir seçmeye devam etmek isteyip DropDownList'i bir kez daha genişletirsen garip bir durum olduğunu fark etmelisin. Şehirler DropDownList'e bir kez daha eklenmiştir. Şimdi bir şehri daha seçip Button'a tıklayıp ardından DropDownList'e göz attığında şehirlerin biraz daha arttığını göreceksin. Bu döngü bu

şekilde sürüp gidecektir. Sonuç olarak 3 tane şehir olması gereken yerde her tıkta artan ve kendini tekrar eden şehirler topluluğu yer almaktadır.

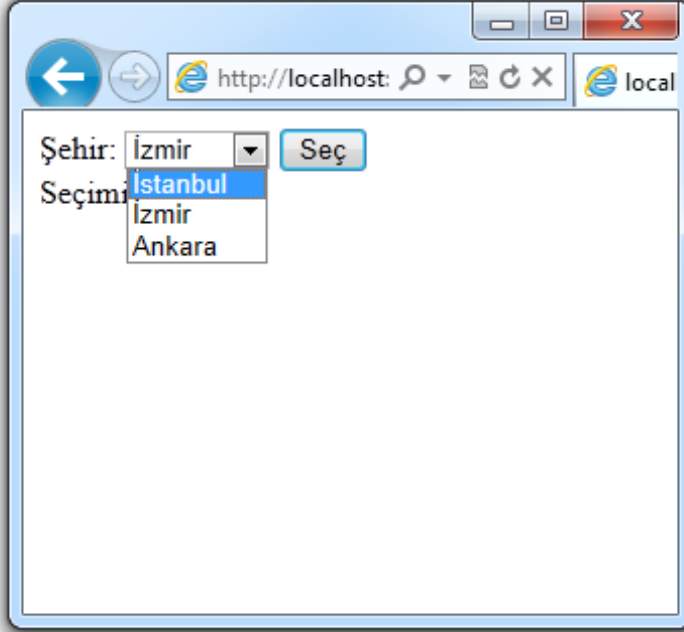


Burada sayfa, sunucuya her gönderildiğinde kontroller dahil bütün sayfa içeriği yok edilip yeniden oluşturulmaktadır. Buna rağmen dropDownList'in önceki talepten kalan verileri hatırlaması, yeni gelen taleplerde de aynı verileri yeniden üzerine eklemesinin sebebi **VIEWSTATE**'dir. Bir kontrole eklenen verilerin bir sonraki talepte ve sayfa oluşumunda hatırlanmasının nedeni; viewstate nesnesidir. Viewstate nesnesi, arka planda istemci ile sunucu arasında gizli bir nesne olarak verileri taşır. Yukarıdaki ikiye-üç... katlanmanın sebebi de budur. Bütün kontrollerin **EnableViewState** özellikleri varsayılan olarak **true**'dur. İstenirse **false** yapılarak kontrollerin önceki talepte kendisine yüklenen verileri hatırlamasının önüne geçilebilir. (TextBox, RadioButton ve CheckBox kontrolleri hariç)

DropDownList'e öğe ekleyen kısım Page_Load olay metodudur. Butona basılarak veya başka bir yolla sayfa sunucuya her gönderilişinde o kod çalışır ve listeye aynı elemanları sürekli ekler. Bunun önüne geçmenin yolu ise; Page_Load'daki listeye öğe ekleme kodunun sadece sayfaya yapılan ilk talepte çalışmasını sağlamak ve sonraki taleplerde de çalışmasını engellemektir. **Page.IsPostBack** özelliğini kullanarak bu kontrolü yapmak sorunu çözecektir. Şimdi Page_Load metoduna geri dönelim ve kodları aşağıdaki hale getirelim. Dikkatle incelediğinde burada bir **if** yapısının kullanıldığını ve sayfanın ilk sefer mi yükleniyor olduğu, yoksa sunucuya PostBack metodu ile geri mi gönderildiğinin tespitinin yapıldığını fark edeceksin. Burada söylenmek istenilen "Sayfa PostBack değilse ilgili işlemi yap" ifadesidir.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownList1.Items.Add("İstanbul");
        DropDownList1.Items.Add("İzmir");
        DropDownList1.Items.Add("Ankara");
    }
}
```

Sayfa ilk talep edildiğinde Page.IsPostBack özelliği false döneceğinden, kontrol if bloğuna girecek ve dropDownList'e istenilen elemanlar eklenecektir. Sayfaya yapılan diğer taleplerde Page_Load olay metodu çalışmasına rağmen, Page.IsPostBack özelliği true döneceği için kontrol if bloğuna girmeyecek ve dropDownList'e elemanların yeniden eklenmesi önlenmiş olacaktır ve problem çözülecektir.



EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

**Web Uygulamalarının
Gelişimi**

Konu :

Web Uygulamalarının Dağıtımı



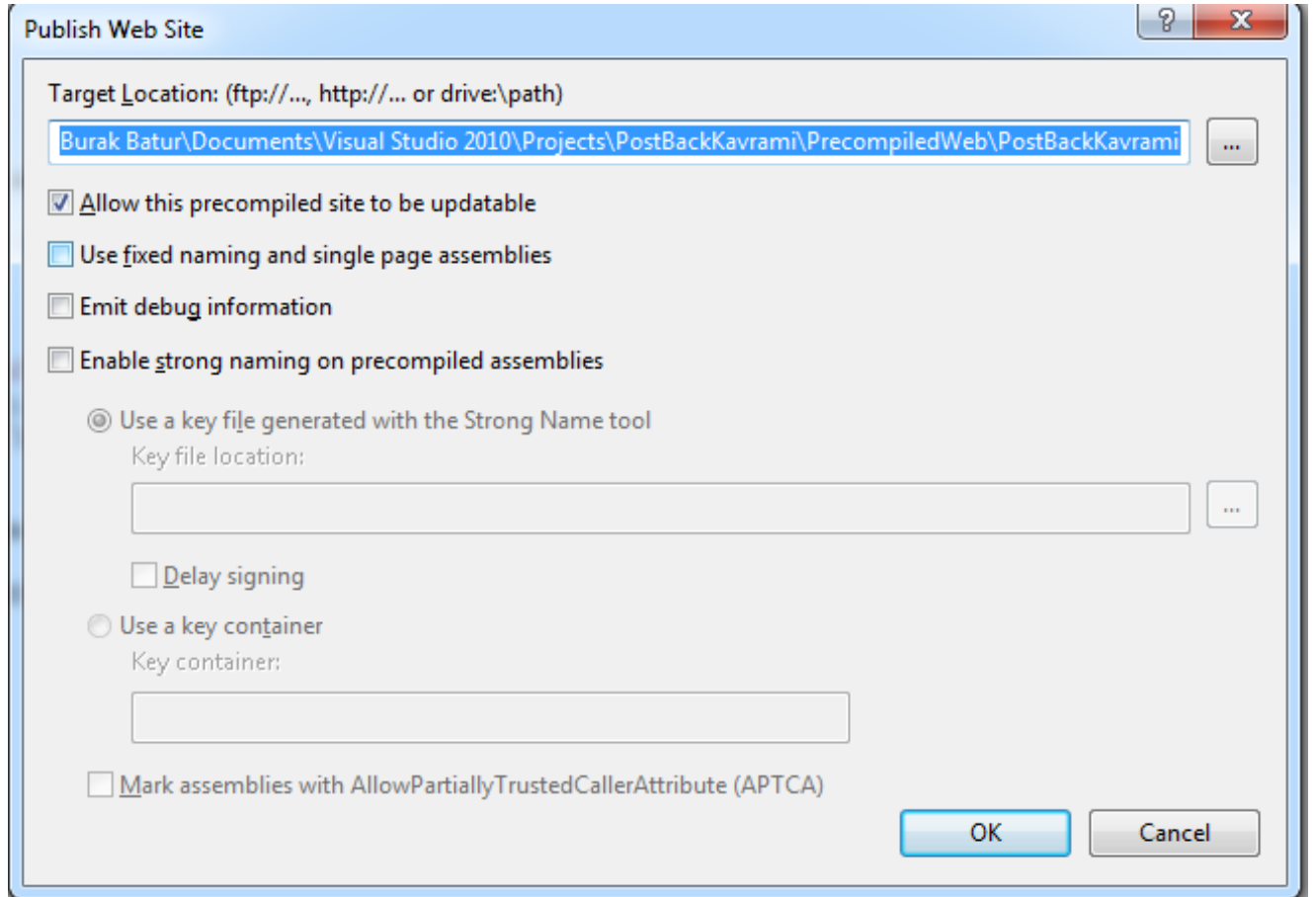
Microsoft Türkiye

Açık Akademi

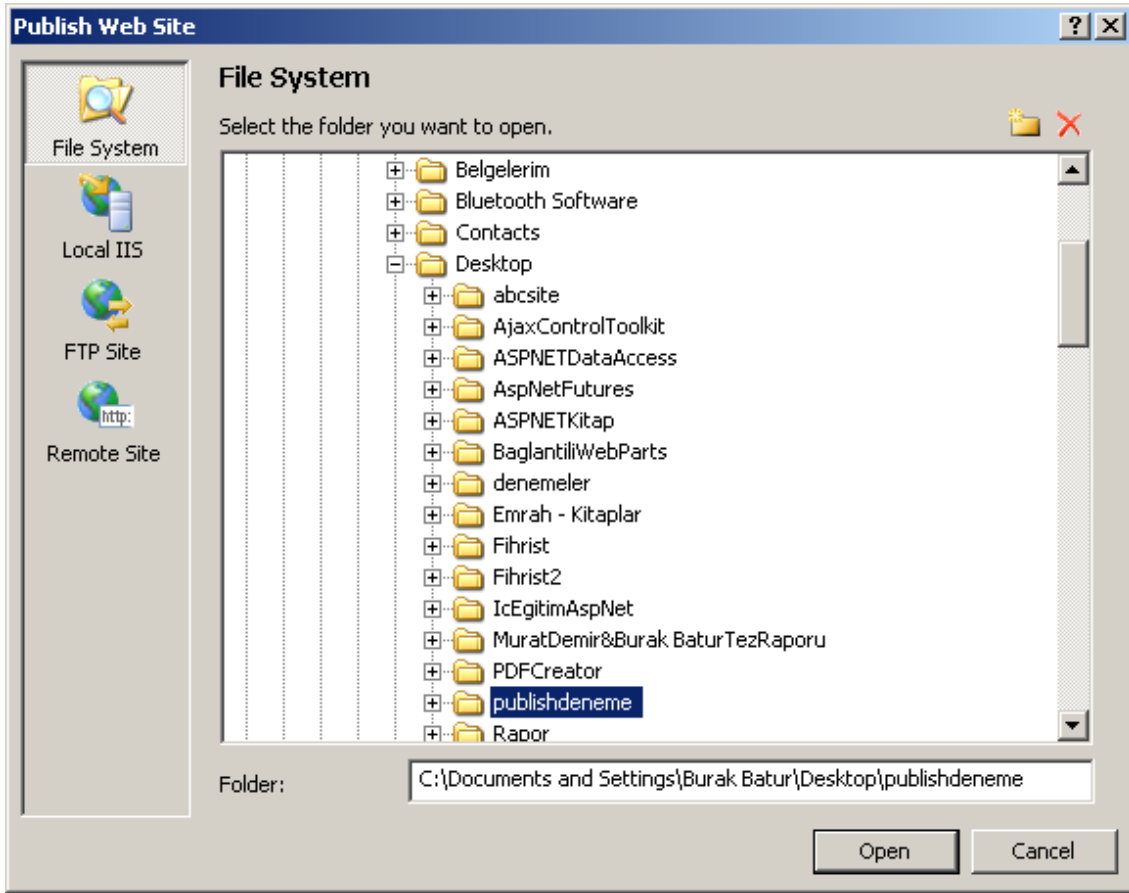
Web Uygulamalarının Dağıtımı

Web uygulamalarının çalışması ve dağıtılması masaüstü programlarından biraz farklıdır. Bu farklılık çalışma mimarisinin farklı olmasından kaynaklanmaktadır. Önceki konulardan da hatırlanacağı üzere web uygulamalarının çalışması için bir tane web sunucu servisine ihtiyaç duyulmaktaydı. Bu servis, http protokolü üzerinden gelen çağrılara göre web sitelerini çalıştırıp gerekli çıktıları yine http protokolünü kullanarak iletliyordu. ASP.NET uygulamalarını çalıştıracak olan servis hatırlanacağı üzere Internet Information Services (IIS) olarak adlandırılmaktadır. Dolayısı ile geliştirilen bir ASP.NET projesinin yayınlanması için IIS kullanılacaktır. ASP.NET içerisinde yer alan özel klasörler yardımıyla web uygulamalarının dağıtımı da oldukça kolaylaştırıldı. Herşeyi kendi özel klasörünün içerisine koyduktan sonra sitenin tamamını web sunucusundaki klasörüne kopyalayıp yapıştırarak web sitesi yayınlanabilir. Evet bu kadar basit! Çalışma anında tüm parçalar kendi özel klasörlerinden alınarak derlenir ve kullanıcıya HTML çıktısı gönderilir.

Visual Studio ortamında geliştirdiğiniz bir ASP.NET uygulamasını Visual Studio'nun araçları ile dağıtmak için Visual Studio'nun **Build** menüsü altında yer alan **Publish Web Site** seçeneğini kullanabilirsiniz. Menüden seçenek seçildikten sonra aşağıdaki resimde yer alan yeni bir pencere açılacak ve bu alandan gerekli ayarlar ve sitenin yayınlanacak olduğu alan belirtildikten sonra dağıtım işlemi gerçekleştirilecektir.



Resimden de görüldüğü gibi gerekli yolu en üstte bulunan metin kutusuna yazıp OK tuşuna bastıktan sonra site yayınlanacaktır. Sitenin yayınlanacak olduğu yolu yazmak yerine seçmek daha uygun olacaktır. Bu işlem içinde metin kutusunun hemen yanında bulunan üzerinde ... bulunan butona tıklanarak web sitesinin yayınlanacak olduğu alan seçilebilir. Bahsedilen butona tıklanıldıktan sonra aşağıdaki resimde yer alan görünüm ile karşılaşılacaktır.



Resimde görülen ekran yardımı ile sitenin yayınlanacağı alan seçilebilir. Bu alanda da resimden görüldüğü üzere dört farklı alana yayınlama yapılabilmektedir. Bunlardan ilki olan **File System** seçeneği ile daha önceden bulunan ya da yeni oluşturulan herhangi bir klasöre dosyalar gönderilecektir. **Local IIS** seçeneği ile birlikte uygulamanın geliştirildiği bilgisayardaki IIS'in içindeki herhangi bir yere dosyalar kopyalanabilmektedir. **FTP Site** seçeneği ile FTP ayarları kullanıcı adı ve parolası belirtilen bir FTP alanına yayınlama yapılabilirken, **Remote Site** seçeneği ile uzak bir siteye yayınlama yapılabilir.

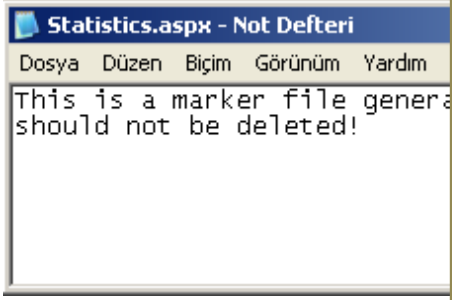


Visual Studio'nun araçları kullanılırken web sitesi yayınlaması yapılıyorsa hedef olarak gösterilen klasörün içeriğinin boş olmasına ya da önemli bir şey olmamasına dikkat edilmelidir çünkü hedef klasöre veriler gönderilirken ilk olarak klasörün içindeki tüm veriler silinecek ve yenileri kopyalanacaktır.

Visual Studio üzerinden yayınlama yapıldığında site ön-derlenmiş (**PreCompiled**) olarak yayınlanacaktır.

Visual Studio üzerinden uygulama dağıtılırken uygulamanın dağıtılacak olduğu yer dışında birkaç ayar daha yapılabilmektedir. İlk Resim incelendiğinde bu seçenekler görülmektedir. Bu seçeneklerden en üstte yer alan **Allow this precompiled site to be updateable** seçeneği ile sitenin yayınlandıktan sonra güncellenip güncellenemeyeceği belirleniyor. Site, bu seçenek seçili iken yayınlanırsa, yayınlandıktan sonra aspx sayfaları güncellenebiliyor olacaktır. Ancak, bu seçenek pasif durumda ise site yayınlandıktan sonra aspx tarafında bulunan HTML kodları bile dll'lerin içine eklenip sitedeki kod güvenliği maksimum düzeyde sağlanmış olacaktır.

Allow this precompiled site to be updated
kodlarındaki aspx sayfalarından biri NotePad ile



Yayınlama ayarları yapılırken belirlenebi
assemblies seçeneğidir. Bu seçenek seçildiğinde
birinin farklı birer assembly'e atılacağı ve bu işlemin
belirtiliyor. Bu seçenek kullanılarak sayfa bazında

Yayınlama yapılırken belirlenecek son iki
precompiled assemblies seçeneğidir. Enable
derlenirken oluşturulan assembly'lerin Strong Name
ile de debug bilgilerinin hedef sunucuya gönderilmesi

lananan

page
in her
ılacağı

ng on
e site
nation

EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

Web Kontrolleri

Konu :

Web – HTML Kontrolleri



Web Kontrolleri

ASP.NET ile geliştirilen web sayfalarında kullanılan, önemli işlevleri ve özellikleri olan ve birçok işlemin kolay bir şekilde yapılmasını sağlayan hazır kontroller bulunmaktadır. Bu kontroller değişik amaçları yerine getirmekte ve uygulamalarda birçok kolaylık sağlamaktadır. ASP.NET'te kullanılabilecek web kontrolleri, **sunucu kontrolleri** ve **HTML (istemci) kontrolleri** olmak üzere iki sınıfa ayrılmaktadır.

Sunucu kontrolleri, üstlenmiş oldukları görevleri sunucu tarafında gerçekleştirirken, HTML kontrolleri üstlendikleri görevleri istemci tarafında gerçekleştirir. Duruma farklı bir açıdan baktığımızda, sunucu kontrollerinin aslında birer .NET sınıfından türediğini görebiliriz. Doğal olarak, diğer tüm .NET sınıfları gibi kontroller de sunucu tarafında çalışacaktır. ASP.NET sayfalarına yazacak olduğunuz .NET (C#, VB.NET vs.) kodları içerisinden sunucu kontrollerine erişip dinamik olarak özelliklerini ayarlayabilirsiniz. Ancak HTML kontrolleri üzerinde işlem yapmanız gerektiğinde istemci tarafında çalışacak Javascript gibi bir dile ihtiyacınız olacaktır. Daha önceki bölümlerde bahsedilen örnekleri gözünüzün önüne getirdiğinizde bazı kontrollerin özelliklerine C# kodu içerisinden eriştiğimizi hatırlayacaksınız. Daha önceki örneklerimizde yoğun olarak TextBox, Label ve Button gibi kontrolleri kullandık ve bu kontrollerin Text özelliklerine erişip Button'un Click olayını ele alarak işlem gerçekleştirdik. Açıklamadan da anladığınız gibi TextBox, Label ve Button birer sunucu kontrolüdür.

HTML ve Sunucu Kontrollerinin tamamına, Visual Studio içerisindeki **Toolbox** (Araç Kutusu) penceresinden ulaşılabilir. Bu kontroller, Toolbox içerisinde görevlerine ve özelliklerine göre oluşturulmuş farklı gruplar içerisinde toplanmıştır. Bu gruplar şunlardır:

Standart: Temel web programlama işlemlerini yerine getirebilecek olan kontrollerin yer aldığı gruptur.

Data: Veri ile ilgili işlemler için kullanılan kontrollerin yer aldığı gruptur.

Validation: Veri doğrulaması ve veri kontrolü için kullanılan kontrollerin yer aldığı gruptur.

Navigation: Site içi dolaşımın sağlanması için kullanılan kontrollerin yer aldığı gruptur.

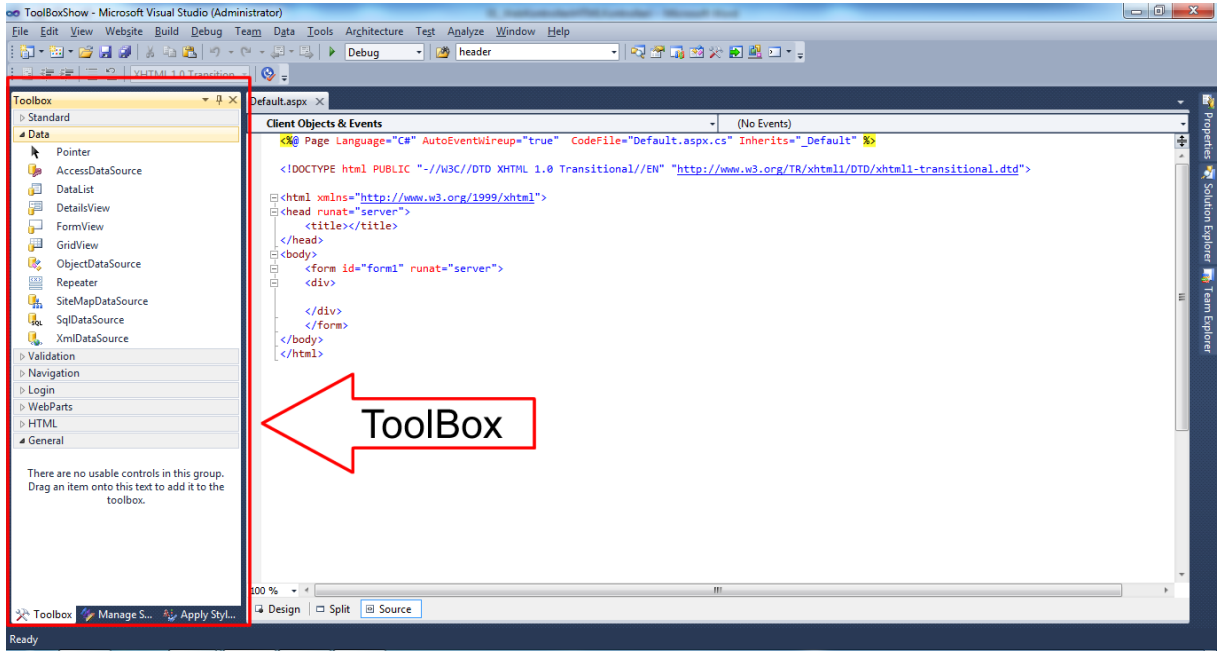
Login: Üyelik sistemi için kullanılan kontrollerin yer aldığı gruptur.

WebParts: Uygulamanın yönetilebilen ve kişiselleştirilebilen parçalara ayrılmasını sağlayan gruptur.

HTML: ASP.NET kontrolleri dışındaki sıklıkla kullanılan HTML kontrollerinin bulunduğu kontrol grubudur.

General: Tüm bunlar dışında kalan, genelde sonradan eklenen kontrollerin yer aldığı gruptur.

Visual Studio'da ToolBox penceresine erişmek için küçük bir örnek yapalım. Yeni bir Visual Studio açıp, Visual Studio içerisinde bir Web Site projesi oluşturalım. Web Site projesi oluşturduktan sonra, Visual Studio varsayılan olarak bir tane sayfa oluşturmuş ve HTML tarafı düzenlenebilir olarak sayfayı karşınıza getirmiş olmalıdır. ToolBox diye adlandırmış olduğumuz bölüm aşağıdaki resimde sol tarafta bulunan ve yukarıda belirtilen kriterlere göre kontrolleri içeren bölümdür.



Yukarıdaki resimde ToolBox'ı görüyorsunuz, resimden ya da kendi Visual Studio'nuzdan dikkat edebileceğiniz gibi, tüm pencerelerde olduğu gibi ToolBox'da da kapatma düğmesi vardır. Bu düğmeye tıklandığında, ToolBox kapanacaktır ve siz yeniden açana kadar kapalı kalacaktır. Kapatılmış ToolBox'ı yeniden açmak için, Visual Studio'nun **View** menüsü altında yer alan **ToolBox** seçeneğini seçebilirsiniz. Hadi şimdi ToolBox'u kapat ve yeniden açmayı dene!

HTML Kontrolleri

HTML kontrolleri ASP.NET öncesinde de kullanılabilen ve istemci tarafı olarak çalışan kontrollerdir. Bu kontroller HTML ile tasarlanmış olan sayfalara dinamik katmak adına, istemci tarafı bir takım fonksiyonları çalıştırmak ve kullanıcı ile sayfanın etkileşimini arttırmak için yazılım geliştiriciler tarafından kullanılmaktadır. HTML kontrolleri ASP.NET ile uygulama geliştirilirken de kullanılıp, istemci tarafı işlemler yaptırılabilir. Visual Studio'nun araç kutusunda, HTML kontrolleri ayrı bir tabda yazılım geliştiricilere sunulmaktadır. HTML kontrolleri Visual Studio ortamında kullanıp, özellikleri Visual Studio aracılığı ile ayarlanabilir. Şimdi en fazla kullanılan HTML kontrollerini inceleyelim.

Button

Button kontrolü, tıklanıldığında istemci tarafında bir fonksiyonu çalıştırmak için kullanılabilecek bir kontroldür. Kontrolü sayfaya sürükleyip bırakıp üzerinde çift tıklanıldığında Visual Studio'nun HTML tarafında bir tane JavaScript fonksiyonu oluşturduğu ve sayfa çalıştırıldığında bu fonksiyonun içerisine yazılan kodların sayfa PostBack olmadan istemcide çalıştırıldığı görülecektir.

Reset ve Submit

Reset kontrolü, tıklanıldığında sayfadaki metin kutularının içeriğini temizleyecek olan kontroldür. Bu kontrol adından da anlaşılacağı üzere kullanıcılardan veri girilmesi istenilen bir sayfada kullanıcının formu temizlemesi için kullanılabilir.

Submit kontrolü ise Reset'in tersi olarak formda tamınlanan işin yapılmasını sağlayacak olan kontroldür. Kullanıcının görüş belirttiği bir sitede, bu iki kontrol birlikte kullanılabilir. Örneğin; Kullanıcı Submit'e basarsa veriler e-mail olarak gönderilir, kullanıcı Reset'e basarsa sayfadaki kontrollerin içleri boşaltılır.

Text, TextArea ve Password Kontrolleri

Bu üç kontrol aslında sunucu tarafındaki TextBox'ın görevlerinden üçünü HTML tarafında üstlenmektedir. Text kontrolü kullanıcıdan veri almak için kullanılmakta ve kullanıcının girmiş olduğu veriler diğer kullanıcılar tarafından görüntülenmektedir. Password kontrolünde ise kullanıcının girmiş olduğu veriler tarayıcının parola karakteri şeklinde görüntülenerek girilen verilerin üçüncü şahıslar tarafından okunmasının önüne geçilmiş olur. TextArea kontrolü ise TextBox kontrolünün TextMode özelliğinin MultiLine olarak ayarlanmış hali gibi düşünülenilir. Bu kontrolle, kullanıcıdan birden fazla satırda veri kabul edilebilmektedir.

Not: TextBox Kontrolü Sunucu Kontrolleri başlığı altında ele alınmaktadır.

File

File kontrolü istemciden sunucuya dosya transfer işlemlerinde, istemcide bulunan dosyanın yerinin elde edilmesi için kullanılabilecek bir kontroldür. Kontrol sayfaya eklendikten sonra tıpkı sunucu kontrollerinde bulunan ve bu kontrolle aynı işlemi yapan FileUpload kontrolü gibi görünüyordur olacaktır.

CheckBox

HTML kontrollerinde bulunan CheckBox kontrolü, sunucu kontrollerinde yer alan CheckBox kontrolü ile aynı amaçla kullanılabilecek bir kontroldür. Tabi bu kontrol sunucu tarafında bulunana göre daha az özellik içermektedir ve dolayısıyla sunucu tarafta çalışmasına oranla kullanımı oldukça azdır.

Radio

Radio kontrolü, sunucu kontrolleri arasında yer alan RadioButton ile aynı amaçla kullanılabilecek bir kontroldür. Tıpkı HTML kontrollerinde bulunan CheckBox kontrolünde olduğu gibi, Radio kontrolünün de RadioButton'a oranla ASP.NET uygulamalarında kullanımı oldukça azdır.

Hidden

Hidden kontrolü sunucu tarafında bulunan HiddenField kontrolü ile benzerlik gösteren ve sayfanın arka planında istemcide veri saklanmasına yarayan kontroldür. Kontrolün value özelliğine atanan veriler sayfa çalıştırıldığında kullanıcılara gösterilmeyerek arka planda saklanır ve bu veriler üzerinde işlemler yapılabilir.

Table

Table kontrolü sayfaya eklendiğinde, tablo oluşturan bir kontroldür. Günümüzde, Visual Studio, tablo eklemede çok fazla kolaylıklar sağlamaktadır. Table kontrolünü buradan sürükleyip bırakmak yerine Visual Studio'da sayfanın HTML kısmının, Design tarafına geçildiğinde görünür olan Table menüsünü kullanmanızı tavsiye ederiz.

Image

Belirtilen kaynaktaki resmi görüntüleyecek olan kontroldür. Sayfaya eklendiğinde kodlara bakılacak olursa standart bir etiketi ile benzerlik göstermektedir. Bu kontrol, ASP.NET sunucu kontrolleri arasında yer alan Image kontrolü ile aynı işi yapabiliyor olmasına rağmen, ASP.NET sunucu kontrolleri arasındaki Image kontrolünü kullanıyor olmak, görüntülenecek resmin ve özelliklerinin dinamik olarak ayarlanmasını sağlayacağı için daha mantıklı olacaktır.

Select

ASP.NET sunucu kontrolleri arasında yer alan DropDownList ile benzerlik gösteren Select kontrolü, kullanıcılara verileri açılır liste halinde sunmak için kullanılabilecek bir kontroldür. İstemci taraflı olayları

tetiklemek ve sadece istemci tarafında işlem yapılmak istenildiğinde, Select kontrolü tercih edilebilir ve kontrol kullanılarak istemci taraflı bir takım fonksiyonlar ile istenilen işlemler yapılabilir.

Div

Div kontrolü içerisinde HTML etiketleri, HTML kontrolleri ve ASP.NET sunucu kontrolleri barındırabilen sayfa tasarımında önemli görevler üstlenebilen bir kontroldür. Div kontrolü Javascript gibi istemci taraflı çalışabilen script dilleri ile de kullanılarak değerleri çalışma zamanında ayarlanabileceğinden, sayfaya dinamizm katması adına oldukça kullanılan ve günümüz tasarım metodlarına çok önemli yer tutan bir kontroldür.

EĞİTİM :

**WEB UYGULAMALARI VE
WEB KONTROLLERİ**

Bölüm :

Web Kontrolleri

Konu :

Sunucu Kontrolleri – Standart
Kontroller



Microsoft Türkiye

Açık Akademi

Sunucu Kontrolleri (Server Controls)

ASP.NET uygulamalarında sunucu tarafındaki işlemlerin gerçekleşmesinde rol alacak kontrollere sunucu kontrolleri denir. Toolbox içerisinde HTML grubu dışında yer alan tüm kontroller, birer sunucu kontrolüdür. Sunucu kontrolleri, sayfanın Source kısmında tanımlanırken sunucu tarafında çalışacağı **runat="server"** ifadesi ile belirtilir. Sunucu kontrollerini Visual Studio'nun Toolbox'ında yer alan gruplara göre ele alıyoruz. Bu grupların işlevsellik açısından herhangi bir amacı olmayıp sadece kullanım kolaylığı için oluşturulduğunu hatırlatmakta fayda var. Siz de isterseniz Visual Studio üzerinde kendi gruplarınızı oluşturabilirsiniz.

Standart Kontroller (Standard Controls)

Web uygulamalarında temel işlemleri gerçekleştirmek için kullanılan kontroller, Standart Kontroller grubunda toplanmıştır. Burada bulunan kontroller, ASP.NET projelerinde en fazla ihtiyaç duyulan ve dolayısıyla en fazla kullanılan kontrollerdir. Bu kontroller ve kontrollerin önemli özelliklerinin açıklamalarına kontrol başlıkları altında devam edelim.

Label

Label kontrolü, kullanıcılara bir metin göstermek için kullanılacak olan kontroldür. Görüntülenmek istenen metin, tasarım anında Label'ın **Text** özelliğine atanabileceği gibi eğer istenilirse çalışma anında dinamik olarak da değiştirilebilir. Label'ın en fazla kullanılan özelliği, Text özelliğidir. Text özelliği, etiketin içinde görüntülenecek yazıyı tutacak olan özelliktir.



Label'ın Text özelliğini dinamik olarak değiştirmiyorsanız, yani tasarım anında belirttiğiniz değer proje çalıştırıldıktan sonra değişmeden sabit olarak kalıyorsa, Label kullanmak yerine, görüntülemek istediğin metni direkt olarak sayfaya yazabilirsin.

Sayfaya bir tane Label kontrolü sürüklenip bırakıldığında Visual Studio aşağıdaki kodları oluşturacaktır. İstenildiği takdirde Text özelliğine varsayılan olarak atanan Label değeri değiştirilip görüntülenmek istenen metin burada belirtilebilir.

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

TextBox

TextBox kontrolü kullanıcı ile web sitesinin etkileşimini sağlayan en temel kontroldür. TextBox, kullanıcıdan veri almak için kullanılan kontrollerden biridir. Kullanıcılar kendilerinden istenilen verileri, TextBox aracılığı ile belirtirler. TextBox kullanılarak, kullanıcılardan metin tabanlı bilgiler istenilebilir ve kullanıcıların TextBox kontrollerine girmiş olduğu veriler, çalışma zamanında alınıp kullanılabilir. TextBox kontrolleri, kullanıcıdan veri alma amacı için tasarlanmasına rağmen eğer istenilirse veri görüntülemek için de kullanılabilir. Bu durumda dinamik olarak, tıpkı Label'da olduğu gibi, TextBox'ın Text özelliğinin belirlenmesi gerekmektedir. Kullanıcıların TextBox'a girmiş olduğu verileri alıp kullanmak için de kullanılacak olan özellik **Text** özelliğidir. TextBox'ın Text özelliğinin yanında, projelerde çok kullanılan bir kaç özelliğini daha açıklayalım;

TextMode Özelliği

TextBox kontrolü, değişik amaçlara hizmet etmek için üç farklı modda kullanılabilir. Bu modlar **SingleLine**, **MultiLine** ve **Password** modlarıdır. TextBox'ın Text özelliği içerisinde yer alan metin, belirtilen moda göre farklı şekillerde kullanıcılara gösterilebilir. Bu modların detaylarını birlikte inceleyelim:

SingleLine: Bu modda metin tek satır halinde görüntülenir. TextBox'un varsayılan modu da SingleLine'dır. Dikkat edilecek olursa sayfaya sürüklenip bırakılan, yani varsayılan ayarlarında olan bir TextBox kontrolüne

sadece tek satır veri girilebilir. Bu mod kullanıcıdan ad, soyad vb. gibi tek satırlık bilgilerin alınması için uygundur.

MultiLine: TextMode özelliklerinden, MultiLine seçili ise TextBox'ta bulunan metin, birden fazla satırda görüntülenebilir. TextBox kontrolünün TextMode özelliği, MultiLine olarak ayarlandığında, TextBox'ın görünümü değişecek ve bir tane kaydırma çubuğu görünür olacaktır. Artık, TextBox'a bir satırdan daha fazla veri girilebilir ve Enter ile bir alt satıra inilebilirsiniz. MultiLine modu, adres bilgisi veya bir ürünün açıklama bilgisi gibi, birden fazla satır gerektiren alanlarda kullanılabilir.

Password: Kullanıcıdan, şifre bilgisi gibi başka kişiler tarafından görülmesinde sakınca olan bilgiler istenirken, TextBox'ın TextMode özelliği Password olarak ayarlanmalıdır. TextMode, Password olarak ayarlandığında, TextBox'a girilen veriler tarayıcıya özgü bir karakter olarak görüntülenir ve bu alana girilen veri başkaları tarafından görülemez.

ReadOnly Özelliği

TextBox, veri alma amaçlı değil de, veri görüntülemek amaçlı gösteriliyorsa, TextBox'ta bulunan metnin değiştirilmesi istenilmez. Bu durumda, ReadOnly özelliği, True olarak ayarlanarak, kullanıcıların TextBox'ta bulunan veriyi, sadece okunabilir olarak görüntülemesi sağlanabilir.

MaxLength Özelliği

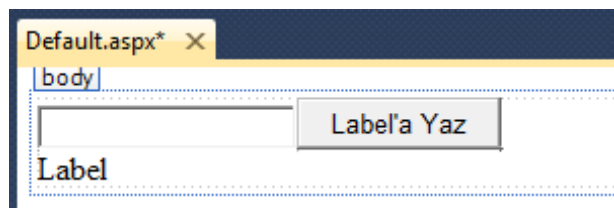
TextBox'a girilecek olan verinin maksimum boyutu söz konusu ise, MaxLength özelliğinde, bu boyut belirtilerek, kullanıcının belirtilen boyuttan daha fazla veri girmesi engellenmiş olur. Veritabanı uygulaması ile çalışacak olan uygulamalarda MaxLength özelliği kullanılarak, kullanıcının TextBox'ın bağlı olduğu alanın boyutunu aşmaması sağlanarak, çalışma zamanında alınabilecek olan hatalar engellenmiş olur.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

Button

Button kontrolü üzerine tıklanıldığında bir iş yaptırmak için kullanılacak kontroldür. Çalışma zamanında Button kontrolü üzerine tıklanıldığında, **Click** olayı tetiklenir ve bu olayın yakalanması gerekir. Click olayının olay yakalayıcısına, Button'un üzerine tıklanıldığında yapılacak olan iş veya işler tanımlanır. Button'un Click olayının olay yakalayıcısı, Visual Studio ortamında Button üzerine çift tıklanılarak oluşturulabileceği gibi özellikler bölümünden de oluşturulabilir. Web sitelerinde her Button üzerinde, Button'un yapacak olduğu işi tanımlayan bir ifade yer alır. Bu ifade Button'un **Text** özelliğinde belirtilmektedir.

Button kontrolünün kullanımını örneklemek için, şu ana kadar bahsettiğimiz TextBox ve Label kontrollerini de kullanarak şöyle bir örnek yapalım; Visual Studio ortamında yeni bir tane Web Sitesi projesi açıp ana sayfaya bir tane Label, bir tane TextBox ve bir tane de Button sürükleyip bırakalım. Amacımız Button'a tıklanıldığı zaman TextBox'a girilen değer Label'a yazdırılması. Bu işlem için hazırlanan web sayfasının tasarımı aşağıdaki resimde görüldüğü gibidir.



Form tasarımı yapıldıktan sonra ilk olarak Button'a çift tıklanılıp Click olayı ele alınmalıdır. Button'a çift tıklayarak Click olayını ele alalım ve Visual Studio'nun bizim için oluşturduğu metodun içerisine aşağıdaki kodları yazalım.

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

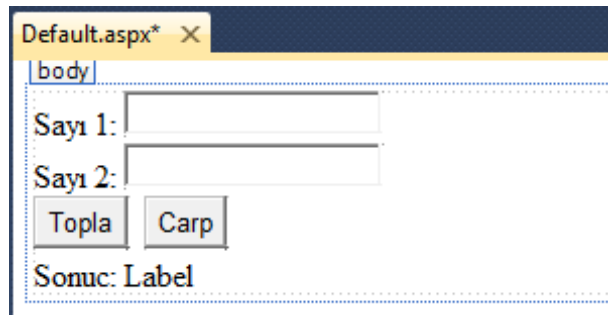
Kodlardan anlayacağın üzere yapılan işlem Button'a tıklandığında TextBox'un içerisinde yazılan metnin Label'a kopyalanma işlemidir. Bunu, programlama dilinde yeniden açıklayacak olursak, TextBox'un Text özelliğindeki değer Label'ın Text özelliğine kopyalanacaktır diyebiliriz.

Burada ufak bir hatırlatma yapmak faydalı olacaktır. Button'a çift tıkladığımızda Button'unumuzun ismini değiştirmeyip Button1 olarak bıraktıysak Visual Studio Button1_Click adında bir metod oluşturacaktır. Button'a tıklandığında Button1_Click isimli metodun (Olay yakalayıcı diye de adlandırabiliriz) çalışacağını belirtmesi için HTML tarafında Button'un kodları aşağıdaki hale getirilmiştir. Kodları incelediğinizde Click olayında Button1_Click isimli olay yakalayıcısının çalıştırılacağı açık bir şekilde görülüyor.

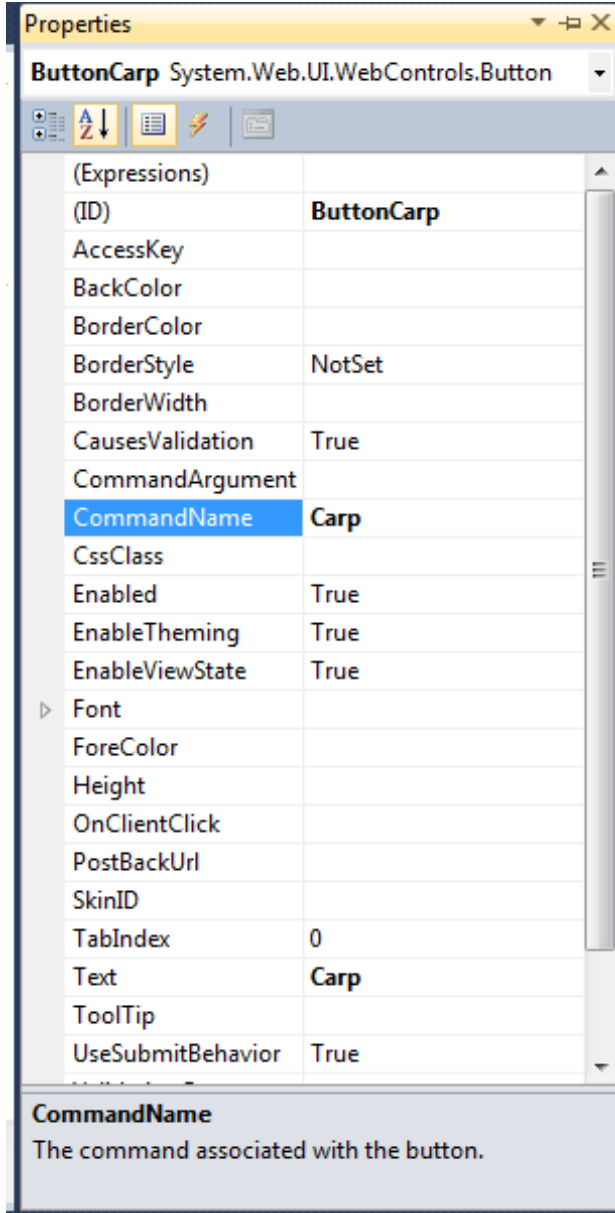
```
<asp:Button ID="Button1" runat="server"
OnClick="Button1_Click" Text="Label'a Yaz" />
```

CommandName Özelliği

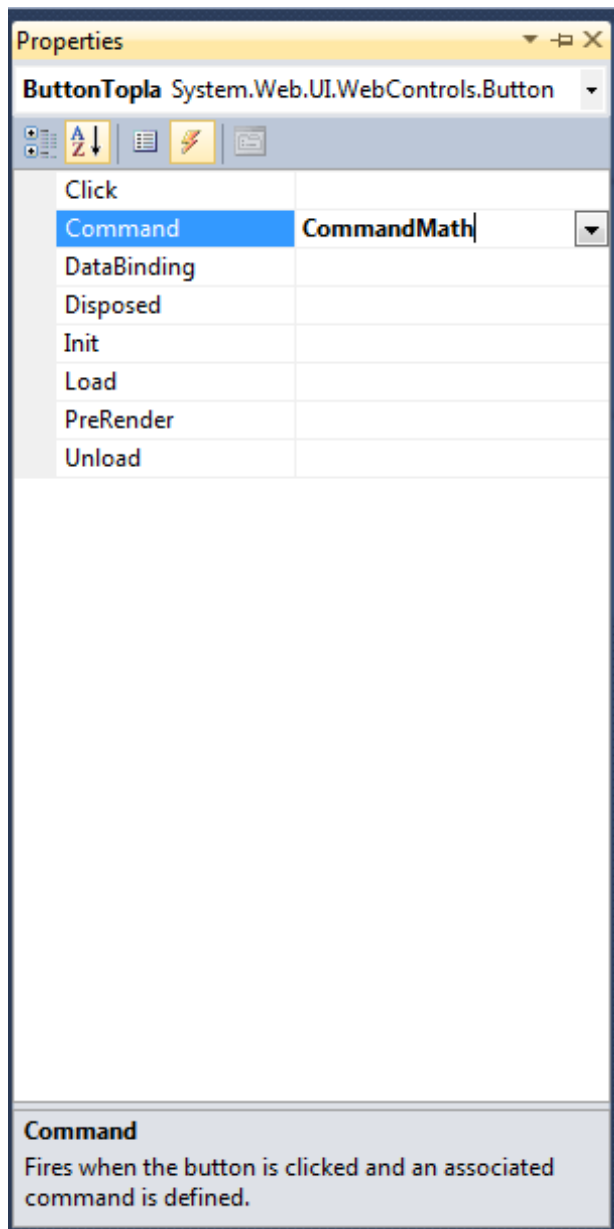
Bir tane olay yakalayıcısı ile farklı Button'ların olaylarını ele almak istediğimizi düşünelim. Burada amaç gerçekleştirilmek istenen işleri tek bir metod içerisinde toplamak ve belirtilen parametreye göre farklı işlemler yapmaktır. Daha derli toplu kod yazmak için kullanılan bu yöntemi tercih ettiğimizde, hangi Button'dan hangi işlemin gerçekleştirileceği bilgisini **CommandName** özelliği aracılığı ile alıyoruz. İleride konuşacak olduğumuz Data Kontrollerinde sık sık kullanılan bu özelliği daha iyi kavramak için bir tane örnek düşünelim. Örneğimizde iki tane TextBox, bir tane Label ve iki tane Button kontrolü yer alacak. Buttonlardan biri toplama diğeri ise çarpma işlemini yapacak. Ancak, burada Button'ların Click olaylarını ele almak yerine, **Command** olaylarını ele alıp, tek metod içerisinde belirtilen parametreye göre işlem yapıyor olacağız. Hangi Button'un toplama, hangi Button'un çarpma işlemini yapacağını ise Buttonlara CommandName özelliği ile belirteceğiz. Şimdi ilk adım olarak Visual Studio'yu açalım ve bir tane WebSite projesi oluşturduktan sonra sayfaya belirttiğimiz kontrolleri sürükleyip bırakalım. Görünümü resimdeki gibi gerçekleştirebilirsiniz.



TextBox'ların ID'lerine yeni bir değer atamadık. Yani, varsayılan olarak TextBox1 ve TextBox2 olarak kaldı. Label'da aynı şekilde varsayılan olarak Label1 olarak bırakıldı. Resimden de göreceğin gibi Button'ların Text özellikleri değiştirildi ve ID'leri de ButtonTopla ve ButtonCarp olarak belirlendi. Şimdi sıra geldi CommandName özelliklerini belirtmeye. İlk olarak, ButtonTopla'ya tıklayalım ve özellikler penceresinden CommandName özelliğini **Topla** olarak belirleyelim. Diğer Button için de CommandName özelliğini **Carp** olarak belirleyelim.



Şimdi sıra geldi Button'ların Command olaylarını ele almaya. Daha önceki konulardan da hatırlayacağın gibi, bu işlemi özellikler penceresinde olaylar sekmesine geçerek yapıyorduk. İlk olarak ButtonTopla'da yukarıda da gördüğün özellikler penceresinde olaylar sekmesine geçelim ve Command olayının adını **CommandMath** olarak belirleyelim. Command olayının adını belirtip Enter'a bastığında Visual Studio, olay yakalayıcı metodu senin için oluşturacaktır. Tahmin edeceğin üzere, adı da CommandMath olarak belirtilmiştir. Şimdi diğer Button için de aynı işlemi yapalım. Ama dikkatli olalım, burada da Command özelliğinin adı **CommandMath** olmalıdır. Çünkü her ikisine tıklandığında da aynı metodun çalışmasını istiyoruz.



Tasarım tarafındaki işlemleri bitirdikten sonra, HTML koduna dönüp baktığımızda form etiketi aşağıdaki gibi olmalıdır. Burada dikkat edeceğin noktalar; Kontrol ID'leri ve özellikleri boşluk ve alt satıra inme etiketleri aynı olmasa da uygulamanın çalışmasını etkilemeyecektir.

```
<form id="form1" runat="server">
<div>
Sayı 1:
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
Sayı 2:
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
<br />
<asp:Button ID="ButtonTopla" runat="server" Text="Topla"
CommandName="Topla"
oncommand="CommandMath" />&nbsp;  
<asp:Button ID="ButtonCarp" runat="server" Text="Carp" CommandName="Carp"
oncommand="CommandMath" />
<br />
Sonuc:
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

```
</div>  
</form>
```

Şimdi sıra geldi, Button'lara tıklandığında çalışacak olan C# kodunun yazılmasına. Bu işlem için kod tarafına geçelim ve CommandMath isimli metodun içeriğini aşağıdaki kodlarla dolduralım.

```
protected void CommandMath(object sender, CommandEventArgs e)  
{  
    int sayi1 = int.Parse(TextBox1.Text);  
    int sayi2 = int.Parse(TextBox2.Text);  
    int sonuc=0;  
    switch (e.CommandName)  
    {  
        case "Topla":  
            sonuc = sayi1 + sayi2;  
            break;  
        case "Carp":  
            sonuc = sayi1 * sayi2;  
            break;  
    }  
    Label1.Text = sonuc.ToString();  
}
```

Kodları adım adım inceleyelim. İlk olarak TextBox'lardan gelen değerlerin tamsayı tipine tip dönüşümleri gerçekleştiriliyor. Ardından olay yakalayıcının e parametresi üzerinden CommandName özelliği sorgulanıp Topla'mı yoksa Carp'mı olduğu sorgulanıyor ve her seçime göre farklı bir sonuç üretiliyor. Son olarak da Label'a elde edilen sonuç yazılıyor. Şimdi kafamızdaki soruyu gündeme getirelim: Neden Click olaylarını ele alıp iki metod kullanmadık? Bu işlem bize nasıl bir avantaj sağladı?

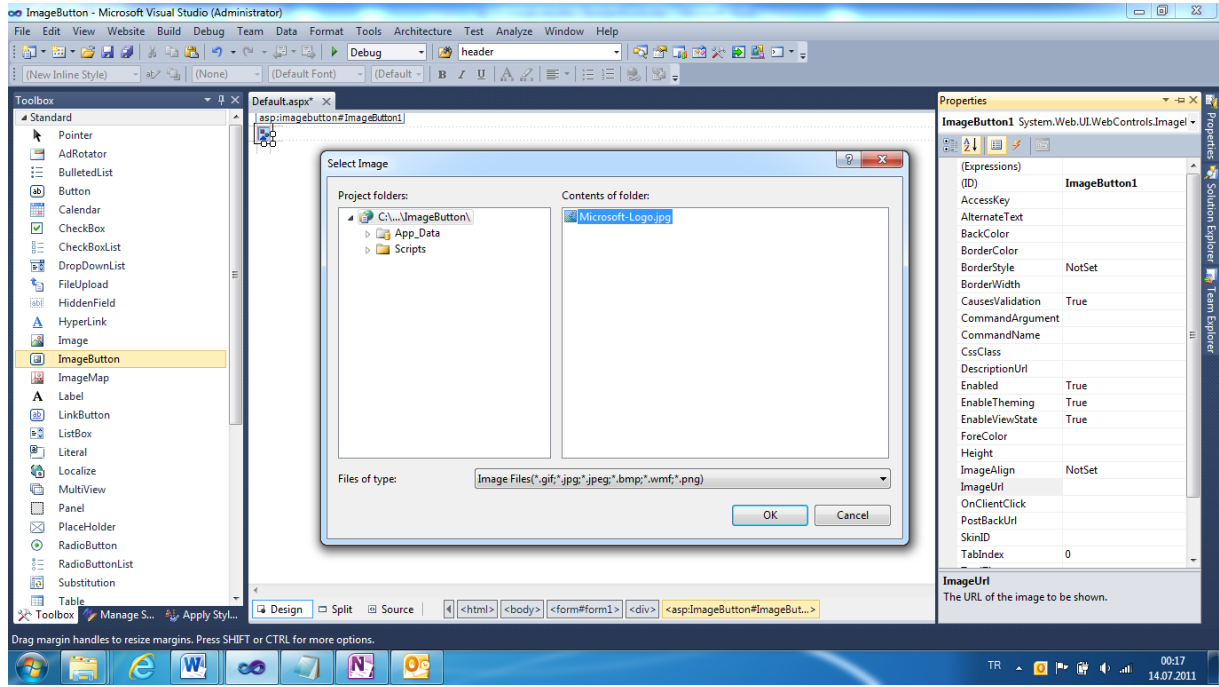
Kodları dikkatle incelersen tip dönüşümü işlemlerinin işlem tipinden bağımsız olarak birer kere yapıldığını görüyoruz, eğer toplama ve çarpma için iki ayrı metod kullansaydık bu kodları iki defa yazacaktık ve yine Label'a sonuç yazdırma işlemini de iki defa yazmış olacaktık. Command olayını ele alarak aynı kodların birden fazla yazılmasının önüne geçmiş olduk, hem daha az kod yazdık hem de kodlarımız daha okunaklı ve derli toplu oldu. Şimdi sen de uygulamayı çalıştır ve test et.

LinkButton

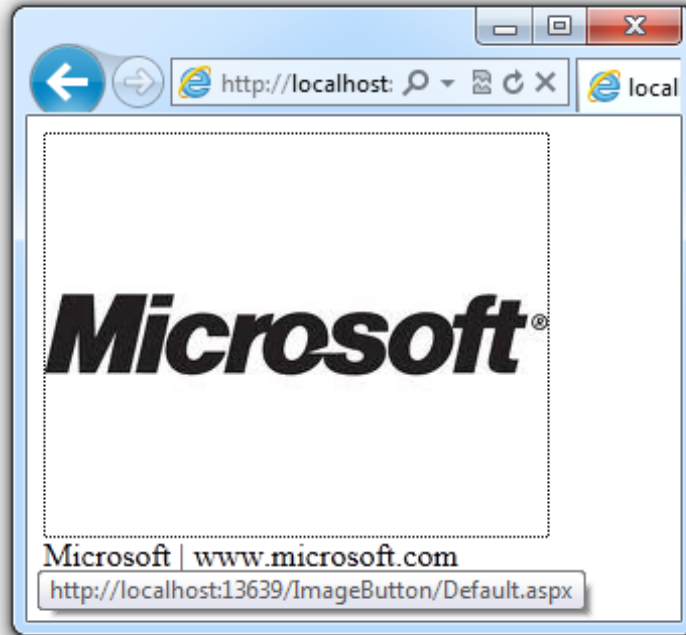
LinkButton kontrolü, kullanımı ve işlevselliği açısından Button kontrolü ile benzerlik göstermektedir. Ancak görünümü Button kontrolünden farklıdır. LinkButton kontrolü, sitelerde daha güzel tasarımlar gerçekleştirmek için kullanılabilir. Bu kontrol, normal bir bağlantı (Link) gibi görünür. Ancak, tıpkı Button gibi çalışır.

ImageButton

ImageButton kontrolü de Button ve LinkButton gibi çalışır. Bu kontrol ile istenilen bir resim kullanılarak, resmin üzerine tıklandığında, sunucu tarafında bir olayın tetiklenmesi sağlanabilmektedir. Bu kontrolün kullanımını basit bir şekilde örnekleyelim; Bir sayfaya bir tane ImageButton ve bir tane de Label kontrolü sürükleyip bırakalım. ImageButton'un **ImageUrl** özelliği, projede bulunan herhangi bir resim dosyası seçilerek ayarlandıktan sonra, Button'un Click olayı tetiklendiğinde çalışacak olan metod içerisinde, Label'ın Text özelliğine, daha önceden tanımlanmış bir metni yazdıralım. ImageButton'un ImageURL özelliğini resimde gördüğün gibi ayarlayabilirsin.



Uygulama çalıştırılıp test edildiğinde belirtilen resim'e tıklanıldıktan sonra Label'a daha önce belirtilen metnin yazıldığını göreceksin.

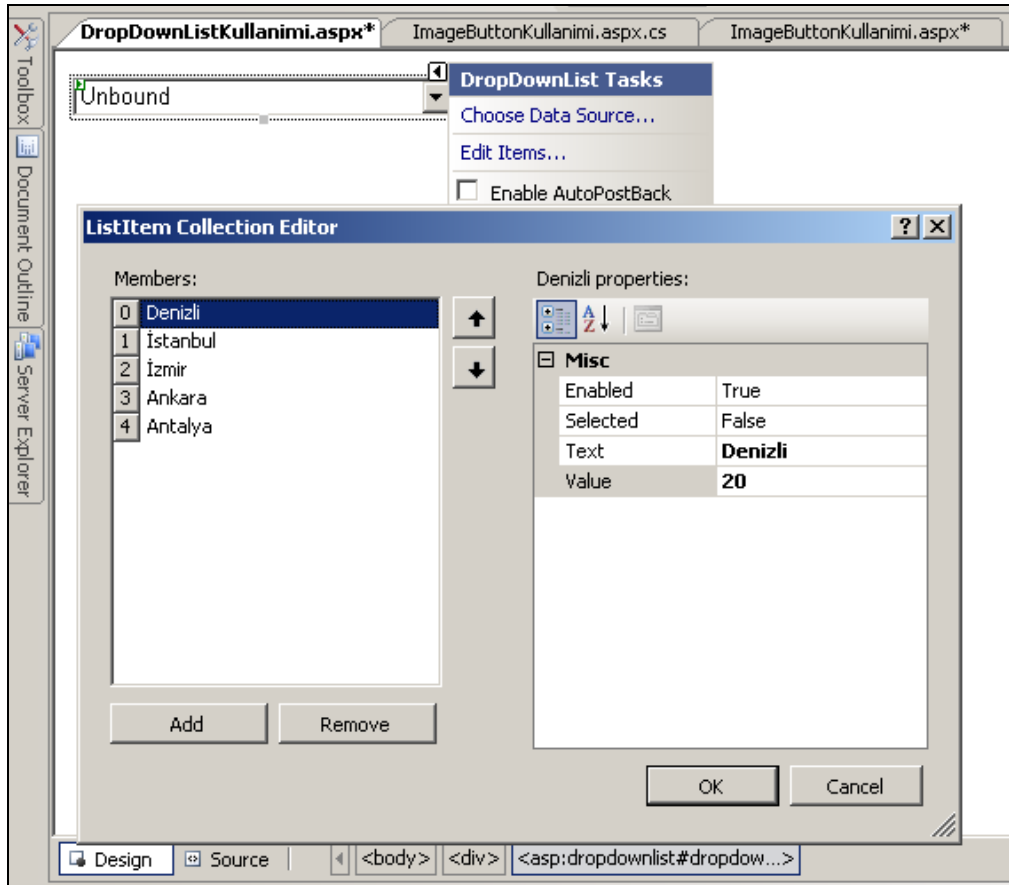


HyperLink

HyperLink kontrolü HTML'deki `...` etiketi ile benzerlik göstermektedir. HyperLink kontrolünün **NavigateUrl** özelliğinde, bir hedef sayfa belirtilir ve kontrolün üzerine tıklanıldığında sunucu tarafında bir olay tetiklenmeden direkt olarak belirtilen sayfaya yönlendirilir. HyperLink kontrolünün nereyi işaret ettiği **Text** özelliği ile kullanıcıya bildirilebilir. Eğer istenilirse, HyperLink kontrolünün **ImageUrl** özelliğinde bir resim belirtilerek kontrol tıpkı ImageButton'da olduğu gibi kullanılabilir.

DropDownList

DropDownList kontrolü bir grup öğeyi kullanıcıya bir açılır liste halinde sunarak kullanıcının bu öğelerden birini seçmesine olanak tanıyan kontroldür. DropDownList'te görüntülenecek olan öğeler, bir veri kaynağı üzerinden getirilebileceği gibi, kod tarafından çalışma anında ya da HTML tarafından durağan olarak da eklenebilir. HTML tarafından öğe eklemek için Visual Studio'nun araçlarını kullanılabilirsin. Visual Studio ortamında herhangi bir sayfaya bir tane DropDownList kontrolü sürüklenip bırakıldığında, DropDownList kontrolünün SmartTag'i açılıp **Edit Items** bağlantısına tıklanılarak yeni öğeler eklenebilir veya daha önce eklenmiş olan öğeler silinip güncellenebilir.



Bu işlemten sonra HTML kodlarına göz atıldığında DropDownList kontrolü içerisinde her öğenin bir ListItem olarak eklendiği görülecektir. ListItem nesnesi, DropDownList veya ListBox gibi kontrollerle birlikte kullanılabilen, değer ve görüntülenecek Text taşıyan nesnedir. Yukarıdaki resimden de görüleceği üzere ListItem içerisinde şehir ve plaka kodu rahatlıkla tutulabilmektedir.

DropDownList nesnesinde seçim değiştirildiğinde sayfanın PostBack olması iseniyorsa, SmartTag'den Enable Auto PostBack seçeneği seçilmelidir. Bu özellik aktif değilse, seçim değişse dahi sayfa PostBack olmayacaktır. Eğer seçim değiştiğinde herhangi bir kod çalıştırılmayacaksa sayfanın gereksiz yere PostBack olmaması için bu özellik pasif konumda bırakılmalıdır.

ListBox

ListBox kontrolü verileri kullanıcıya liste halinde sunup kullanıcının listelenen öğelerden istediği öğeleri seçmesine olanak tanıyan kontroldür. ListBox kontrolüne öğe eklemek, DropDownList kontrolünde olduğu gibidir. Dolayısıyla, aynı yollar ve aynı yöntemlerle bu kontrole de veri getirilebilir.

SelectionMode Özelliği

ListBox kontrolü ile kullanıcının birden fazla seçeneği seçmesi sağlanabilir. Böyle bir gereksinimin olduğu durumlarda SelectionMode özelliği Multiple olarak belirlenebilir ve kullanıcının birden fazla öğeyi seçmesi sağlanmış olur.



DropDownList ve ListBox gibi kontrolleri veritabanındaki bilgiler ile doldurulması bir sonraki bölümde örneklenecektir.

CheckBox

Kullanıcının herhangi bir şeyi işaretlemesi için kullanılabilecek kontroldür. CheckBox kontrolü genellikle bir durumu belirlemek için kullanılır. Örneğin; Bir kullanıcıya, kayıt formunda, mail listesine kayıt olmak isteyip istemediği sorulabilir ya da pek çok üyelik formunda gördüğümüz gibi “Sözleşmeyi okudum” tarzı bir onay almak için de kullanılabilir.

CheckBoxList

Kullanıcılara tek bir durum değil de birden fazla durum arasından istediklerini işaretletmek gerektiğinde birden fazla CheckBox kullanımına ihtiyaç duyulacaktır. Birden fazla CheckBox kullanmak hem sayfanın tasarlanması hem de CheckBox’ların yönetimi açısından pek kolay olmayan bir durumdur. Bu sebeple birden fazla durum söz konusu olduğunda, CheckBoxList kontrolü tercih edilerek daha kolay bir şekilde yapılmak istenilen iş yapılmış olur. CheckBoxList’in bir güzel tarafı da verilerin ListBox ve DropDownList gibi dinamik olarak doldurulabilir olmasıdır. Bu özellik kullanılarak veritabanında bulunan durum veya özellikler kullanıcıya kolaylıkla seçtirilebilir.

RadioButton

Bir grup durumdan, sadece bir tanesininin seçilmesi isteniyorsa RadioButton kontrolü kullanılabilir. RadioButton kontrolleri gerçek hayatta web sitelerinde sıklıkla kullanılan kontrollerdir. Örneğin; Bir kullanıcı kayıt formu göz önüne getirildiğinde, bu kayıt formunda kayıt olan kullanıcının cinsiyeti genellikle sorulur ve kullanıcının listelenen cinsiyetlerden sadece birini seçmesi gerekmektedir.

RadioButton kontrolü kullanılırken bilinmesi gereken en önemli özelliği **GroupName** özelliğidir. Sayfaya sürüklenen RadioButton’lardan aynı amaca hizmet edecek olanları aynı grup içerisine alınmalıdır ki aralarından sadece bir tanesi seçilebiliyor olsun. RadioButton’lar sayfaya ilave edildikten sonra GroupName özelliklerine bir atama yapılmadan sayfa çalıştırıldığında tüm RadioButton’ların aynı anda seçilebiliyor olduğu görülecektir ki bu da problemli bir sayfa anlamına gelmektedir. GroupName özelliği belirtilmezse cinsiyet belirtilirken bir kişi hem Erkek hem de Bayan seçeneklerini aynı anda seçebilir.

GroupName özelliği ile farklı amaçlara hizmet edecek olan RadioButton grupları da tasarlanabilir. Kullanıcı kayıt formu örneği yeniden göz önüne getirilip kullanıcıdan hem cinsiyet hem de yaş aralığı bilgisi istenildiği düşünüldüğünde; GroupName özelliğinin ne amaçla kullanılabileceği daha iyi anlaşılacaktır. Böyle bir durumda cinsiyet seçilmesi amacı ile sayfaya ilave edilen RadioButton’ların GroupName’leri “cinsiyet”, yaş aralığı için kullanılacak olanların GroupName’leri ise “yas” olarak belirlenip iki gruptan da sadece birer seçeneğin seçilmesi sağlanabilir.

RadioButtonList

RadioButtonList kontrolü, CheckBoxList kontrolüne benzer şekilde öğeleri kullanıcılara RadioButton olarak sunar ve aralarından bir tanesinin seçilmesi sağlar. Bu kontrolün verileri de diğer liste kontrollerinde olduğu gibi veritabanından getirilebilir.



RafioButtonList ve CheckBoxList kontrollerinde bulunan **RepeatDirection** özelliği ile listelenen veriler yatay (Horizontal) ya da dikey (Vertical) olarak listelenebilir. Varsayılan yön dikeydir.

BulletedList

Sayfada bir grup verinin madde işaretli bir liste halinde görüntülenmesi isteniyorsa BulletedList kontrolü kullanılabilir. BulletedList kontrolünde listelenecek olan veriler tasarım anında belirlenebildiği gibi çalışma zamanında dinamik olarak da belirlenebilmektedir. BulletedList kontrolünün **BulletStyle** özelliği ile madde işaretinin değiştirilmesi sağlanabilmektedir hatta **CustomImage** seçeneği seçildiğinde **BulletImageUrl**'de belirtilen resim, madde işareti olarak kullanılabilir.

HiddenField

HiddenField kontrolü sayfa içerisinde arka planda bilgi taşımak için kullanılabilecek bir kontroldür. Value özelliğine atılan değer sayfanın HTML çıktısı ile birlikte gösterilir ve gerek duyulduğu anda oradan okunarak kullanılabilir.

Literal

Sayfaya, çalışma anında HTML sözcüklerinin eklenebilmesine olanak sağlayan sunucu kontrolüdür. Literal kontrolünün Text özelliğine, istenilen metin girilerek çalışma anında çıktının HTML olarak yorumlanması sağlanır. Örneğin `Kalın Yazı` şeklindeki bir içerik Text özelliğine atanarak, Kalın Yazı metninin kalın bir şekilde görünmesi sağlanmış olur. Literal kontrolü, Text özelliğinde bulunan metni üç farklı modda görüntüleyebilmektedir. Bunlardan ilki olan **Transform** sayfayı talep eden web tarayıcısının kullanmış olduğu protokole göre bir çıktı üretirken, **PassThrough** modu ise sayfayı talep eden web tarayıcısına göre çıktı üretir. Mode özelliği **Encode** olarak ayarlandığında ise Text özelliğinde bulunan metin, direkt HTML'deki karşılığı olarak görüntülenir. Örneğin `Kalın Yazı` metni mode özelliği Encode olarak ayarlandığında web tarayıcısına `>Burak>` şeklinde gönderilerek web tarayıcısının bu kodları yorumlaması engellenir ve çıktı `Kalın Yazı` şeklinde üretilir.

Burada dikkat edilmesi gereken mode Encode'dur. Bir web sitesi yaptığınızı ve bu web sitesinde HTML dersleri anlattığınızı hayal edin. `` etiketini anlatmak istiyorsunuz ve sayfanın kodlarına `` etiketi yazdığınızda sayfada `` etiketinden itibaren her yer kalın oluyor. Bu durumun önüne geçmek için etiketi sayfaya yazdırırken bir Literal kontrolü kullanıp mode'unu da **Encode** olarak belirtirseniz sorun çözülecektir. Bu problemi görmek ve çözümü örneklemek için hemen bir Visual Studio açıp, sayfaya Literal kontrolü sürükleyip PageLoad'da da Literal'e HTML kodları atayarak deneme yapabilirsiniz.

Calendar

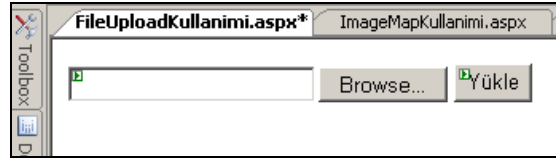
Geliştirilen uygulamalarda kullanıcılara bir tarih seçtirilmek istenildiğinde Calendar kontrolü kullanılabilir. Varsayılan olarak Calendar kontrolü ile kullanıcılara bir gün seçtirilebilir ama **SelectionMode** özelliğinden bir haftanın ya da bir ayın da seçilmesi sağlanabilir.



Calendar kontrolündeki yıllar arasında geçiş zorluğu ve her ay değişiminden sonra sayfanın PostBack olması sebebi ile yazılım geliştiriciler AJAX Control Toolkit içerisinde yer alan Calendar kontrolünü tercih etmeye başlamışlardır.

FileUpload

Web sitesinin bulunduğu sunucuya dosya yüklemek gerektiğinde, FileUpload kontrolünü kullanılabilirsin. Bu kontrol ile birlikte, dosya yükleme işlemi, gerekli yazma izinleri verildiğinde oldukça kolay bir şekilde gerçekleştirilebilmektedir. Kontrolün kullanımı aşağıdaki örnek ile birlikte açıklayalım. Visual Studio ortamında bir web sitesi projesi içinde yeni bir tane sayfa ekleyip sayfaya bir tane FileUpload ve bir tane de Button kontrolü sürükleyip bırakalım. Burada amaç FileUpload kontrolü ile yüklenecek olan dosyayı bulmak ve ardından da Button'un Click olayına yazılacak olan kodlarla dosyayı sunucuya yüklemektir. Sayfayı resimde görüldüğü gibi tasarlayalım.



Button'un click olayında çalışacak olan olay yakalayıcı metoda aşağıdaki kodları yazalım. Kodlardan da anlayacağın gibi yapılan işlem, dosyayı Files klasörü altına upload etmek.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        FileUpload1.SaveAs(Server.MapPath("Files/" + FileUpload1.FileName));
    }
}
```

Kodlar incelendiğinde ilk olarak bir if bloğunda FileUpload kontrolünün **HasFile** özelliği kullanılarak yükleme yapmak için herhangi bir dosyanın seçilip seçilmediği belirlendi. Eğer dosya seçili ise dosyayı sunucuya yüklemek için çalışacak olan kodlar, if bloğu içine yazıldı. If bloğu içindeki kodlara dikkat edildiğinde ise FileUpload kontrolünün **SaveAs** metodu ile seçili olan dosyanın belirtilen hedefe kaydedilmesi sağlandı. Burada dikkat edilmesi gereken bir diğer metod ise **Server** sınıfının **MapPath** metodudur. Bu metod ise sayfanın bulunduğu sunucudaki yolunu döndürür. MapPath metodu parametre olarak sayfanın bulunduğu yola eklencek olan değeri de alır. Görüleceği üzere burada, FileUpload kontrolünün **FileName** özelliği kullanılarak, dosyanın, Files isimli klasörün içine, orijinal isminde kaydedilmesi sağlanmaktadır.



Sunucuya dosya yüklenebilmesi için sunucuda gerekli yazma izinlerinin bulunması gerekmektedir, aksi halde hata alınacak ve dosya yüklenemeyecektir.

Panel

Panel kontrolü kendi içine kontrol alarak bu kontrollerin belli bir alanda gruplanmasına olanak tanıyan sunucu kontrolüdür. Aynı grupta olması istenilen kontroller bir Panel kontrolü içerisine koyularak bir takım ayarların Panel üzerinden yapılmasına olanak sağlar.