

EĞİTİM :

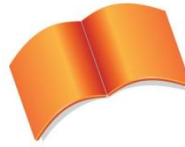
**GÜVENLİK VE ÜYELİK
YÖNETİMİ**

Bölüm :

Güvenlik ve Yetkilendirme

Konu :

**Windows Tabanlı Kimlik
Denetimi**



Microsoft Türkiye

Açık Akademi

ASP.NET ile geliştirilen uygulamalar hayata geçirildiğinde herkese açık olacaktır. Web sitesi geliştirmenin temel mantığı da budur. Aslında bir web sitesi tasarlarken web sitesine daha fazla ziyaretçi çekecek olan tasarımlar üzerine yoğunlaşılır ve sitelerin trafiğinin artırılması için çaba gösterilir. Hatta pek çok web sitesine çeşitli reklâmlar koyularak, siteye gelen ziyaretçilerin bu reklâmlara tıklaması sağlanıp, ziyaretçiler üzerinden gelir elde edilmeye çalışılır. Bu kadar geniş kitlelere açık olan bir uygulama geliştirmenin avantajı olduğu kadar dezavantajı da vardır. Site yayına verdikten sonra sitede bulunan her kullanıcı, eğer bir güvenlik ve yetkilendirme sistemi yoksa her sayfaya erişebilecektir. HTML ile bir site geliştiriliyorsa bu pek problem değildir. Ama ASP.NET gibi içeriğin dinamik olarak yönetilebildiği bir teknoloji ile uygulama geliştirildiğinde, sitede bulunan her ziyaretçinin her sayfaya erişebilmesi pek doğru olmayacaktır. İşte bu noktada güvenlik kavramı önem kazanmaktadır. ASP.NET öncesinde bu işlemler, durum yönetimi teknikleri ile yapabilmekteydi ama ASP.NET'in sunmuş olduğu güvenlik ve yetkilendirme modeli ile sitenin güvenlik politikasını yönetmek hem daha güvenli hem de daha kolay olacaktır.

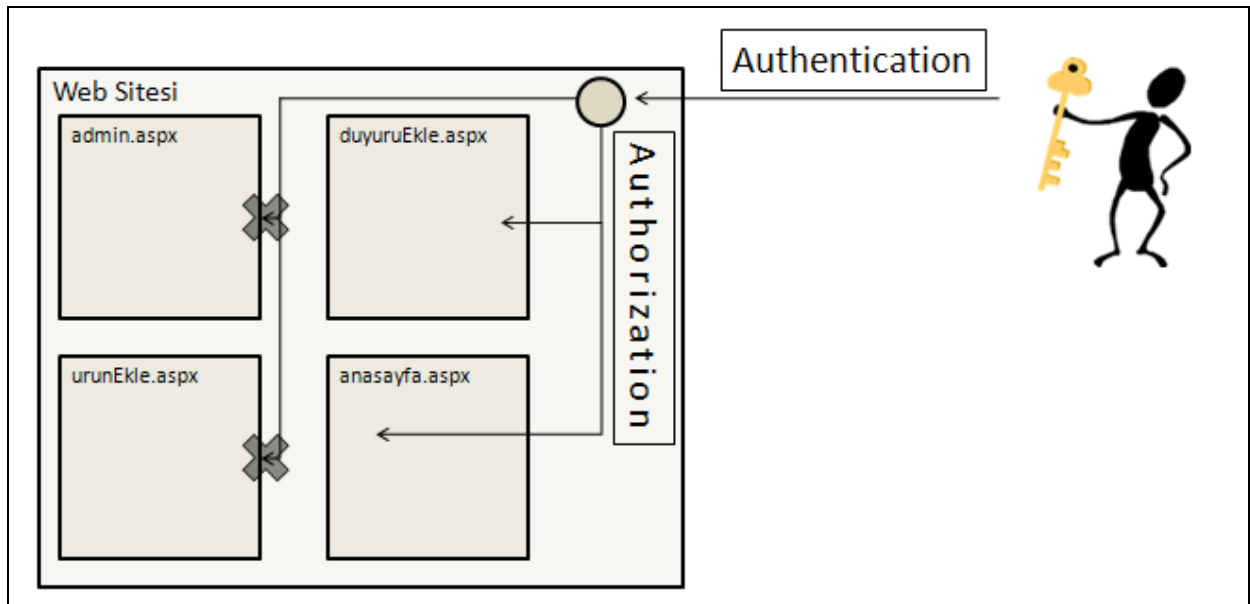
Authentication ve Authorization

ASP.NET'teki güvenlik modellerinden önce ilk olarak Authentication ve Authorization kavramları üzerinde duralım.

Authentication: Authentication sözcüğünün anlamı kimlik denetimidir. Aslında kimlik denetimi sözcüğü bu kelimeyi yeterli miktarda açıklamaktadır. Bir kullanıcının siteye erişim yetkisi olup olmadığının belirlenmesine Authentication (Kimlik Denetimi) denilmektedir. Kimlik denetiminde, kullanıcının site içerisinde nerelere erişim yetkisi olduğu belirlenmez. Sadece kullanıcının siteye erişip erişemeyeceği belirlenir.

Authorization: Bu sözcük de, izin/yetki anlamına gelmektedir. Kullanıcı kimlik denetiminden (Authentication) geçtikten sonra, nerelere erişim hakkı olduğu bu kavramla belirlenmektedir.

Aşağıdaki resim incelendiğinde Authentication (Kimlik Denetimi) ve Authorization (Yetkilendirme) kavramları daha net anlaşılabilir.



Resim incelendiğinde siteye giriş yapmak isteyen kullanıcı ilk olarak kimlik denetiminden geçerek sisteme kabul ediliyor ancak sistem içerisinde görüldüğü gibi her istediği yere giriş yapamıyor.

ASP.NET Güvenlik Modları

ASP.NET yazılım geliştiricilere üç farklı tipte güvenlik modu sunmaktadır. Bunlar, Windows Tabanlı Güvenlik, Form Tabanlı Güvenlik ve Passport.

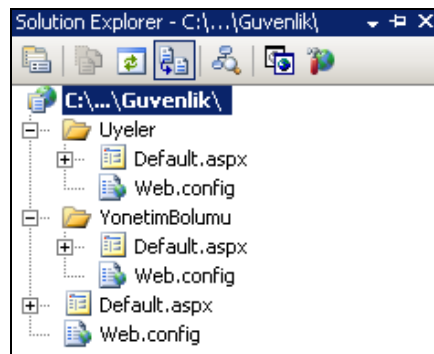
Windows Tabanlı Güvenlik: Uygulama IIS düzeyinde korunur ve Windows veya bağlı olunan domaindeki kullanıcılar üzerinden güvenlik sağlanır. Windows tabanlı güvenlik genellikle bir kurum içinde çalışan Internet'e açılmayan intranet'te çalışacak olan uygulamalar için kullanılır. Mevcut sistemde bulunan kullanıcıların uygulamaya erişmesi denetlenmiş olur.

Form Tabanlı Güvenlik: Bu modda güvenlik, ASP.NET formları üzerinden sağlanıyor. Kullanıcılara birer kullanıcı adı ve şifre atanarak, ve gerekli yerlerde bu şifre denetlenerek güvenlik sağlanır. Form Tabanlı Güvenlik, Internet uygulamaları için biçilmiş kaftandır. ASP.NET 2.0 ve sonrasında, Form Tabanlı Güvenlikte kullanılmak üzere yazılım geliştiricilere hazır kontroller sunulmuştur. Sürükle ve bırak yöntemi ile çok kolay bir şekilde dakikalar içinde uygulamada bu kontroller kullanılabilir. SQL Server Express desteği ile tüm ayarlar otomatik olarak yapılmakta ve kullanıcı bilgileri veri tabanında saklanabilmektedir.

Passport: Bu modda uygulama Microsoft tarafından sağlanan merkezi güvenlik sistemi ile korunabilmektedir. Bu sistem pek çok kullanıcı tarafından kullanılan Messenger'da oturum açmaktan ya da Hotmail hesabı kontrol edilirken kullanılan sistemden farklı değildir. Siteye giriş yapacak olan kullanıcıların herhangi bir Microsoft Passport'u olması yeterlidir. Bu sistemi kullanmak için Microsoft'tan bu hizmetin satın alınmış olması gerekmektedir.

Güvenlik Ayarını Yapmak

Daha önceki bölümlerden hatırlayacağın gibi ASP.NET'deki uygulama ayarları, web.config adındaki XML dosyasında saklanmaktadır. Tahmin edeceğin üzere güvenlik ayarları yapılırken de web.config dosyası kullanılıyor olacaktır. Güvenlik ayarlarının nasıl yapıldığına değinmeden önce, ilk olarak web.config üzerinde biraz duralım. Web.config dosyası sadece bulunduğu klasör ve bulunduğu klasörün altında bulunan klasöre etki eder. Aşağıdaki resim göz önüne alındığında en altta bulunan web.config dosyası sitede bulunan tüm dosyalara ve alt klasörlere etki eder. Üyeler klasörü içinde bulunan web.config dosyası, üyeler klasörü içinde bulunan dosya ve klasörlere etki edecektir. Peki aynı ayarlar hem en üstte bulunan web.config'de hem de alt klasörlerde bulunan web.config'de varsa ne olacak? Bu gibi durumlarda hiyerarşik olarak en altta bulunan dosyada yer alan ayarlar geçerli olacaktır. Resimde görülen site için güvenlik ayarı yapılmak istenildiğinde sadece web.config dosyalarında bu ayarları yapmak yeterli olacaktır. Örneğin Yönetim Bölümüne, sadece bu alana giriş izni olan kullanıcıların girmesi istendiğinde, web.config'e bir iki satır kod yazarak tüm klasör için güvenlik ayarları yapılmış olacaktır. Keza, üyeler bölümü için de iş bu kadar basittir.



Aşağıdaki kod parçası Web.config üzerinde güvenlik ayarlarının belirlenmesini örneklemektedir.

```
<system.web>
... ..
<authentication mode="Windows|Forms|Passport|None">
</authentication>
... ..
</system.web>
```

Web.config içinde güvenlik ayarları <system.web> altında bulunan <authentication> düğümünden yapılabilmektedir. Bu alanda sitenin hangi güvenlik modunu kullanacağı belirlenmektedir. Güvenlik modlarından herhangi birini seçerek, sitenin seçilen modla korunmasını sağlamış oluruz. Eğer None seçeneği seçilirse, site için herhangi bir güvenlik politikası kullanılmaz. Şimdi teker teker bu güvenlik modlarını ele almaya başlayalım.

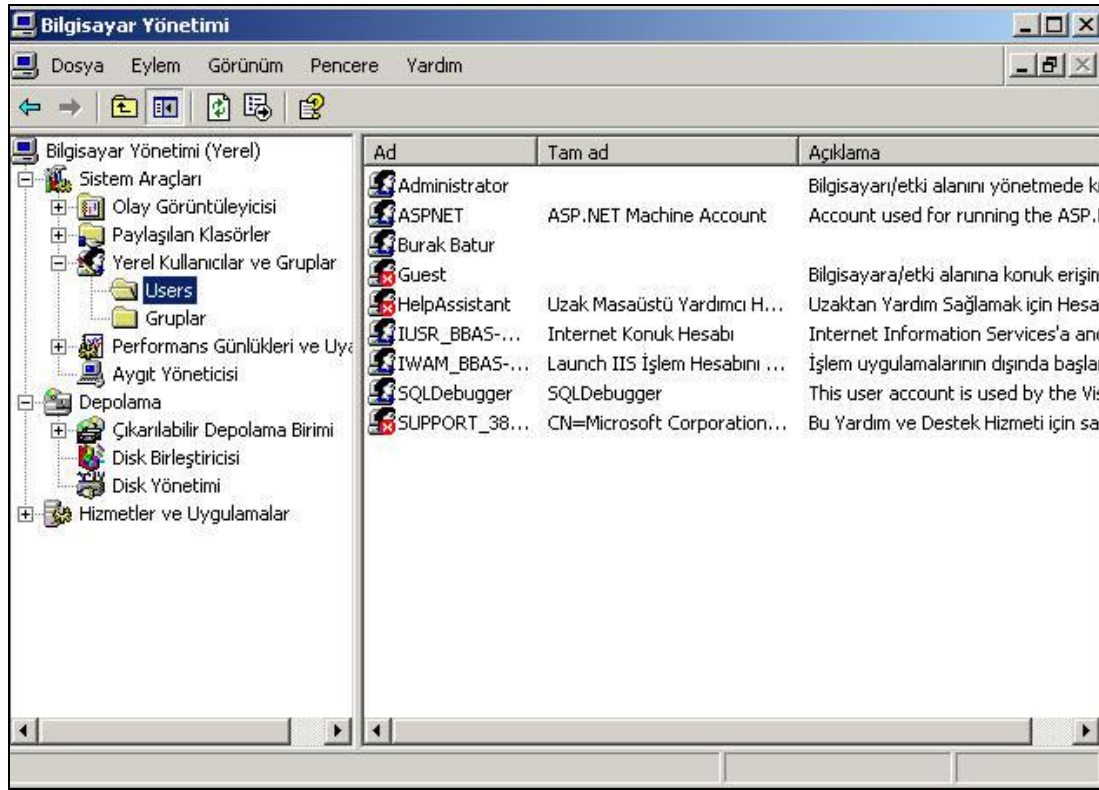
Windows Tabanlı Güvenlik

Windows Tabanlı Güvenlik, intranet uygulamaları yani kurum içerisinde çalışan uygulamalar için daha uygundur. Windows Tabanlı Güvenlikte kullanıcılar uygulamanın çalıştırıldığı makine üzerinde bulunan kullanıcılardan ya da uygulamanın çalıştığı, dahil olduğu domainden sorgulanmaktadır. Bu güvenlik modunda, IIS, kullanıcının bilgilerini denetler ve eğer bu adım başarısız olursa, kullanıcının, kullanıcı adı ve şifresini girebileceği bir ekran çıkartılır.

Şimdi birlikte Windows Tabanlı Güvenlik'in daha iyi anlaşılabilmesi için ufak bir örnek geliştirelim. Küçük bir örnek yapmak için ilk olarak bilgisayarımıza bir kaç tane kullanıcı ekleyeceğiz. Normal şartlar altında, kurumların bilgi işlem çalışanları tarafından kurumun etki alanına eklenecek olan kullanıcıları biz kendi bilgisayarımızda yer alan Windows işletim sistemi altına ekleyeceğiz. Şu anda yapacağımız işlem aslında gerçek bir ortamın simülasyonundan daha farklı değildir.

Bu işlemi yapmak için:

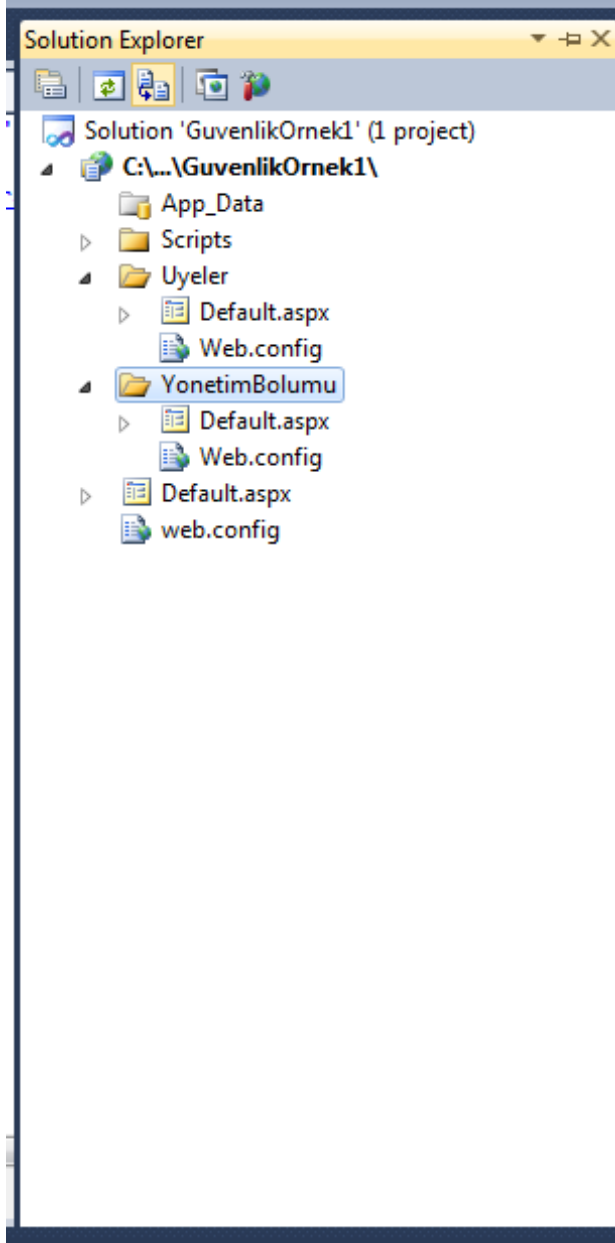
Başlat→Denetim Masası→Yönetimsel Araçlar→Bilgisayar Yönetimi yolu ile Bilgisayar Yönetimi penceresini açılıp burada Yerel Kullanıcılar ve Gruplar düğümü genişletip Kullanıcılar bölümüne geçilir.



Users'ın üzerine sağ tıklayıp “yeni kullanıcı” seçildiğinde, yeni bir tane pencere açılacaktır. Bu alandan kullanıcının bilgilerini doldurup Kullanıcının ilk oturumda şifre değiştirmesini gerektirmeyecek şekilde kutucukta yer alan işareti kaldıralım ve Oluştur Butonunu ile sisteme yeni kullanıcı ekleyelim.

Artık WinDeneme adlı yeni oluşturulan kullanıcıyı denemelerimizde kullanılabiliyoruz. Şimdi bu kullanıcının görüntüleyebileceği bir sayfa tasarlayalım ve web.config'in açıklandığı bölümde, bahsedilen proje üzerinden ilerleyelim. İsterseniz ilk olarak bu projeyi oluşturalım. Yeni bir Web sitesi projesi açmak için Visual Studio'yu

açıp yeni bir Web Sitesi Projesi açalım. Hemen ardından aşağıdaki resimdeki gibi olacak şekilde, gerekli klasör ve sayfaları oluşturalım. Sayfa içeriklerini istediğin gibi doldurabilirsin.

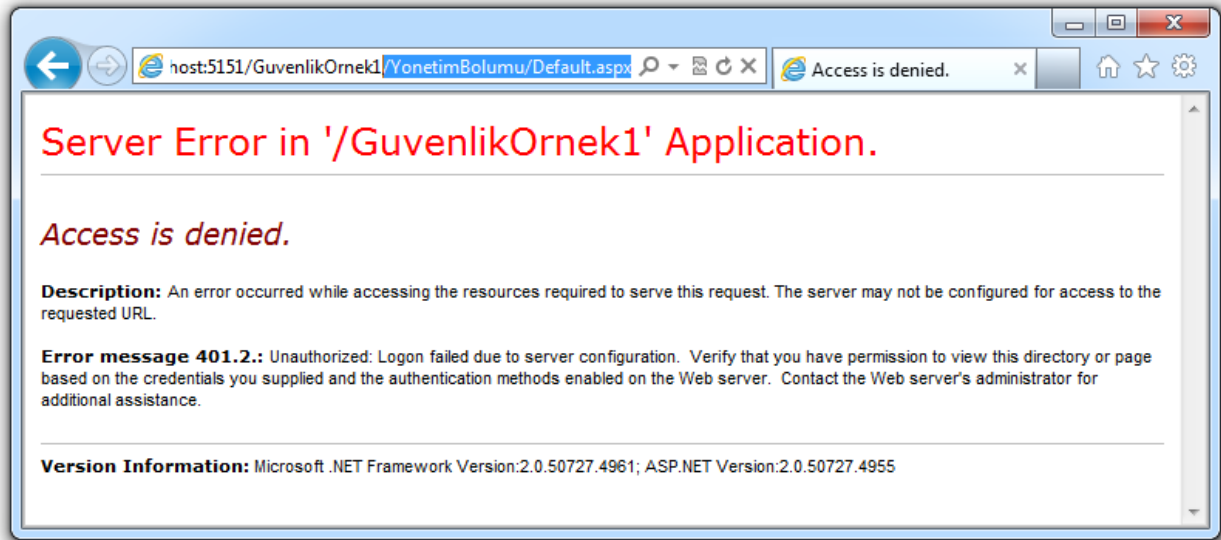


Resmi incelediğinde, Yönetim Bölümüne sadece sisteme giriş yapan kullanıcıların girmesini isteyebilirsin. Bu işlem için ilk olarak klasörün içinde bulunan web.config'de, tüm kullanıcıların buraya girmesini engelleyen aşağıdaki kodu yazalım.

```
<system.web>
... ..
  <authorization>
    <deny users="*" />
  </authorization>
... ..
```

```
</system.web>
```

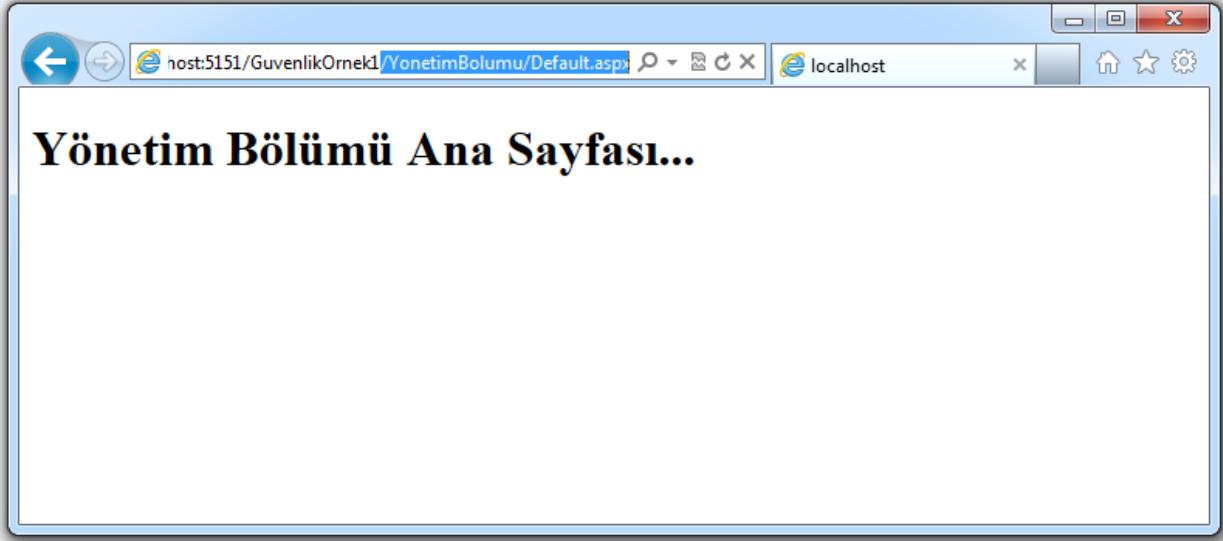
Yukarıda yer alan kodlar ile Yönetim Bölümüne tüm kullanıcıların girmesi engellenmiş oluyor. Uygulama çalıştırılıp ilgili klasör içinde bir sayfaya erişilmeye çalışıldığında aşağıdaki hata mesajı ile karşılaşılacaktır.



Daha önce oluşturulan WinDeneme ve sistemimde bulunan Burak Batur kullanıcısı için bu klasöre erişme izni verilmek istendiğinde Yönetim Bölümü içerisinde bulunan web.config dosyasına aşağıdaki kodlar ekleniyor olmalıdır.

```
<system.web>
... ..
<authorization>
    <allow users="BilgisayarAdi\Burak Batur"/>
    <allow users="BilgisayarAdi\winDeneme"/>
    <deny users="*/>
</authorization>
... ..
</system.web>
```

Bu ayarlarla iki kullanıcıya da Yönetim Bölümüne erişim hakkı verilmiş oldu. Burak Batur kullanıcısı ile Yönetim Bölümüne yeniden erişilmeye çalışıldığında aşağıdaki görüntü ile karşılaşılacaktır.



Authorization Düğümü

Authorization düğümü ile sisteme giriş yapan kullanıcıların, nerelere erişip erişmeyeceğinin ayarları yapılabilir. Bu düğüm içerisinde kullanılabilecek iki tane daha alt düğüm vardır. Bu iki düğüm, kullanıcının erişim yetkileri için kullanılmaktadır.

Allow: Allow düğümü ile ilgili alana erişim yetkisi verilecek olan kişi ve gruplar belirtiliyor.

Deny: Deny düğümü ile ilgili alana erişimi yasaklanan kişi ve gruplar belirtiliyor.

Allow ve Deny düğümlerinde kullanılacak özellikler ise aşağıda açıklanmaktadır.

Users: Bu alana kullanıcı isimleri belirtiliyor. İlgili işleme göre burada belirtilen kullanıcıların hakları belirlenir.

Roles: Bu alana rol gruplarının isimleri girilerek, o rol grubunda bulunan kullanıcıların tümünün hakları belirtilir.

Verbs: Bu alana HTTP iletim metodlarının isimleri belirtilerek, ilgili alan üzerinde o metodun hakları belirtilir.

Yukarıdaki örnekte, web.config dosyasında authorization düğümü içerisinde ayarlama yapılan ilk bölümdeki kodlara dikkat edildiğinde, aşağıdaki kod bloğundaki gibi bir ifadenin yer almakta olduğu görülmektedir.

```
<deny users="*" />
```

Burada * karakteri kullanılarak tüm kullanıcıların sisteme girmesi yasaklanmış oldu. * karakteri az önceki cümleden de anlaşılacağı gibi tüm kullanıcılar anlamına gelmektedir. Bu özellik Users'ta kullanılabileceği gibi Roles'de de kullanabilirdi.

İfade içinde * karakteri yerine ? karakteri de kullanabilirdi yani ifade aşağıdaki gibi olabilirdi.

```
<deny users="?" />
```

Bu ifade oturum açmamış olan yani tüm anonim kullanıcıları yasakla anlamına gelmektedir. Böylece ? karakteri, anonim kullanıcılar anlamına gelmektedir. Uygulamalarda ? karakteri sıklıkla kullanılmaktadır. Eğer rol

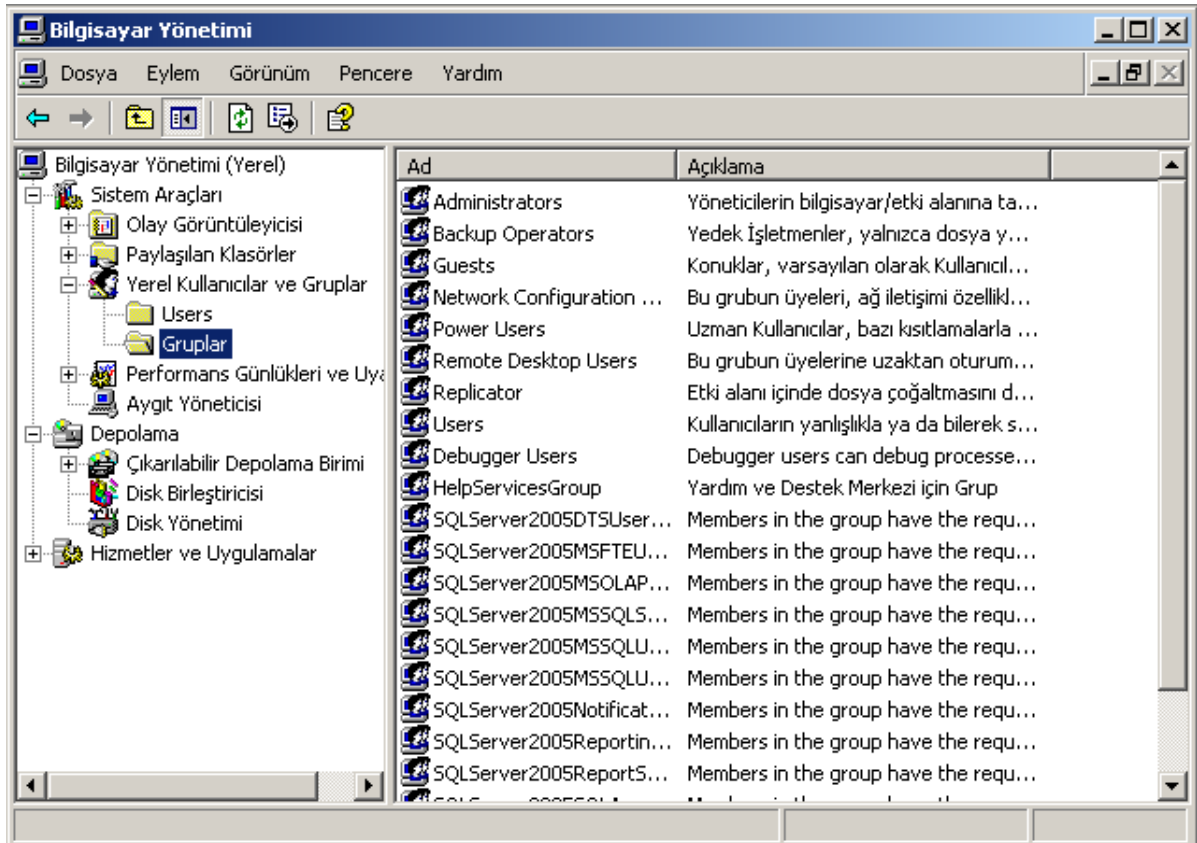
tabanlı bir sistem geliştirilmiyorsa yani sistemde sadece tek bir yetki varsa sadece `<deny users="?" />` ile güvenlik ayarını yapmak yeterli olacaktır.

Bir Rol Grubunun Hakkını Belirlemek

ASP.NET'te kullanıcı hakları belirlenebileceği gibi bir rolün hakları da belirlenebilmektedir. Bu işlemi gerçekleştirmek için sisteme rol eklenebilir.

Bu işlemi yapmak için:

Başlat→Denetim Masası→Yönetimsel Araçlar→**Bilgisayar Yönetimi** yolu ile Bilgisayar Yönetimi penceresi açılıp burada Yerel Kullanıcılar ve Gruplar düğümünü genişletildikten sonra Gruplar bölümüne geçilir.



Gruplar'ın üzerine sağ tıklayıp "Yeni Grup" seçildiğinde yeni bir pencere açılacaktır. Bu alandan Grubun bilgilerini doldurup Oluştur butonuna tıklanıldığında sisteme yeni bir grup eklenecektir.

Sisteme eklenen rol **<allow roles="SiteYonetici" />** şeklinde kullanılabilir. Bu kullanımdan sonra SiteYonetici rolüne dahil olan tüm kullanıcılara ilgili alana giriş izni verilecektir.

HTTP İletim Metotlarının Yetkilerini Yönetmek

Kullanıcılar ve Gruplar üzerinde yetkilendirme tanımlanabileceği gibi HTTP iletim metotları üzerinde de yetkilendirme yapılabilmektedir. Bu işlem daha önce açıklandığı gibi **Verbs** düğümü ile yapılabilmektedir.

```
<deny verbs="GET, DEBUG" />
```

Yukarıdaki kod bloğu incelendiğinde HTTP protokolü üzerinden siteye gelecek olan GET ve DEBUG istekleri bu site için çalıştırılmayacaktır. Verbs özelliği içinde kullanılabilecek olan HTTP iletim metotları POST, GET, HEAD, ve DEBUG'dır.

EĞİTİM :

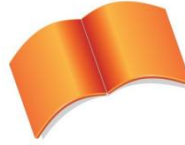
**GÜVENLİK VE ÜYELİK
YÖNETİMİ**

Bölüm :

Güvenlik ve Yetkilendirme

Konu :

Rol Grubu Hakları



Microsoft Türkiye

Açık Akademi

Form Tabanlı Güvenlik

Hatırlayacağın üzere Windows Tabanlı Güvenlik'in intranet uygulamaları için uygun bir çözüm olacağını daha önce belirtmiştik. Internet uygulamaları için, uygun çözümün Form Tabanlı Güvenlik olduğundan da bahsetmiştik. Windows Tabanlı Güvenlik'te her kullanıcı için domainde veya uygulamanın çalışacak olduğu makine üzerindeki Windows'da bir kullanıcı oluşturmak gerekmektedir. Ama, binlerce kişinin erişeceği bir sistemde, kullanıcıları teker teker Windows'da oluşturmak pek de mantıklı bir çözüm değildir. Form Tabanlı Güvenlik'te kullanıcılar uygulama bazında oluşturulup, gerekli kimlik denetimi de uygulama bazındaki bilgilere göre gerçekleştirilir veya kullanıcılar herhangi bir veri tabanı sisteminde saklanıp kimlik denetimi ilgili veri tabanından da gerçekleştirilebilir. ASP.NET uygulamalarında Form Tabanlı Güvenlik kullanmak için uygulama ayarlama dosyasında (web.config) gerekli ayarları yapmak gerekmektedir.

```
<system.web>
... ..
<authentication mode="Forms">

    <forms loginUrl="login.aspx" name="AspNetGüvenlikDemo"
    defaultUrl="default.aspx" />

</authentication>
... ..
</system.web>
```

Yukarıdaki kod bloğuna dikkat edildiğinde, en fazla dikkat çeken şey authentication modunun **Forms** olarak ayarlandığı olmaktadır. Bu söz dizimi ile, sitenin güvenlik modunun Form Tabanlı Güvenlik olduğu bildirilmektedir. Hemen alttaki satırda ise Form Tabanlı Güvenlik'in özel ayarları yapılmaktadır. **LoginUrl** ile kimlik denetimi gereken bir yere erişilmeye çalışıldığında, kullanıcının otomatik olarak kullanıcı adı ve parola girmesi gereken, yönlendirilecek olan sayfanın yolu belirtilmekte; **name** alanında ise kullanıcının sistemine atılacak olan Cookie'nin adı belirtilmektedir. **defaultUrl** ile de, kullanıcı, direkt login sayfasına girdiğinde, geçerli kullanıcı adı ve şifreyi belirttikten sonra varsayılan olarak yönlendirileceği sayfanın yolu belirtilir. Aşağıdaki tabloda Forms etiketi içerisinde gerçekleştirilecek olan diğer ayarlar yer almaktadır.

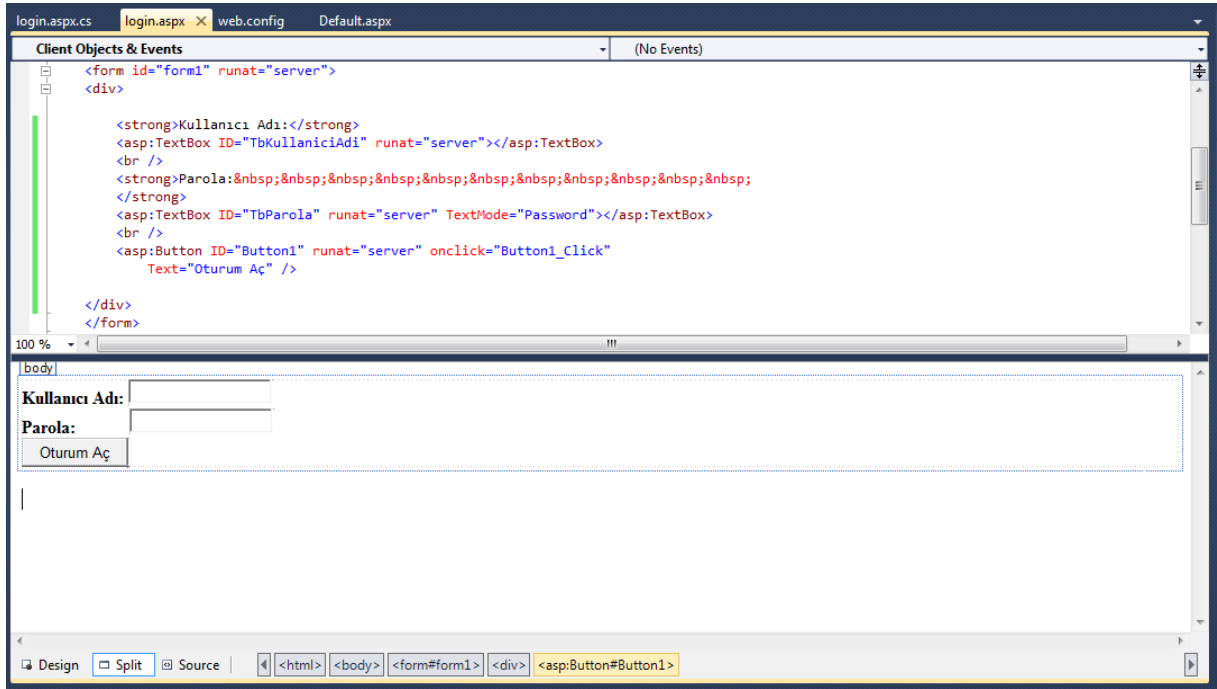
Özellik	Açıklama
protection	Kullanıcının sistemine atılacak olan Cookie'nin korunma modu belirtiliyor. Burada mümkün olan toplam dört tane seçenek vardır. Bunlar: All: Cookie hem data doğrulama hem de şifreleme algoritmaları ile korunur. Varsayılan mod budur. None: Cookie'ye herhangi bir koruma algoritması kullanılmaz. Encryption: Cookie şifreleme algoritması ile korunur ama veri doğrulama aktif değildir. Validation: Veri doğrulama aktiftir ama şifreleme aktif değildir.
path	Kullanıcının sisteminde Cookie'nin atılacak olduğu yer belirtilebilir.
timeout	Kullanıcının sistemine atılan Cookie'nin dakika cinsinden ne kadar geçerli olacağı belirtiliyor. Varsayılan değer 30'dur.
cookieless	Cookie kullanılıp kullanılmayacağı belirtiliyor.
domain	Cookie'lerin gönderilecek olduğu domain belirtiliyor.
slidingExpiration	Kullanılan Cookie'nin zaman aşımı süresinin her çağrıdan sonra yenilenip

	yenilenmeyeceği belirtiliyor. Varsayılan değer false'tur.
enableCrossAppsRedirect	Uygulanan güvenliğin başka bir uygulama tarafından da kullanılıp kullanılmayacağı belirtiliyor. Örneğin bir kaç tane site tasarlandığı düşünölsün. Kullanıcılar bunların tamamında tek bir parola kullanabilmektedir, bu durumda kullanıcılar sitelerin herhangi birinden oturum açtığında bu bilgi diğer sitelerde de kullanılabilsin yani kullanıcı yeniden oturum açmak zorunda bırakılmasın var olan oturum diğer siteler için de kullanılabilsin.(Bu durum Cookie adları aynı olduğu durumlarda geçerlidir.)
requireSSL	Güvenlik bilgileri iletilirken SSL bağlantısı kullanılıp kullanılmayacağı belirtiliyor.

Bir örnekle Form Tabanlı Güvenliği daha iyi anlatalım. Visual Studio ortamında yeni bir tane Web sitesi projesi açalım ve ilk olarak aşağıdaki kod bloğunda görölen kodları uygulama ayarlama dosyasına (web.config) ekleyerek, az önce açıkladığımız gibi hem sitenin Form Tabanlı Güvenlik ile çalışmasını sağlayalım hem de oturum açmamış olan kullanıcıların sisteme erişmesini engelleyelim.

```
<system.web>
<authentication mode="Forms">
    <forms loginUrl="login.aspx" name="AspNetGüvenlikDemo"
        defaultUrl="default.aspx" />
</authentication>
... ..
<authorization>
    <deny users="?" />
</authorization>
... ..
</system.web>
```

Yukarıdaki <forms> etiketine dikkat edildiğinde **loginUrl**'i login.aspx olarak belirlendi. Kullanıcılar bu siteye erişmeye çalıştıklarında eğer oturum açmamışlarsa login.aspx'e otomatik olarak yönlendirileceklerdir. Bu durumda bir tane de login.aspx sayfası tasarlanması gerekmektedir. Siteye bir tane login.aspx adında bir sayfa ekleyelim ve sayfaya iki TextBox bir tane de Button sürükleyip bırakalım. Gerçekleştirmiş olduğumuz tasarım aşağıdaki resimdeki gibi olacaktır. Çok basit bir kullanıcı doğrulama ekranı geliştiriyoruz. Kullanıcının, kullanıcı adını ve parolasını isteyip, doğru girdiği senaryoda oturum açacağız.



Button'un Click olayını ele alıp aşağıdaki kodları da Button1_Click isimli metoda ekleyelim.

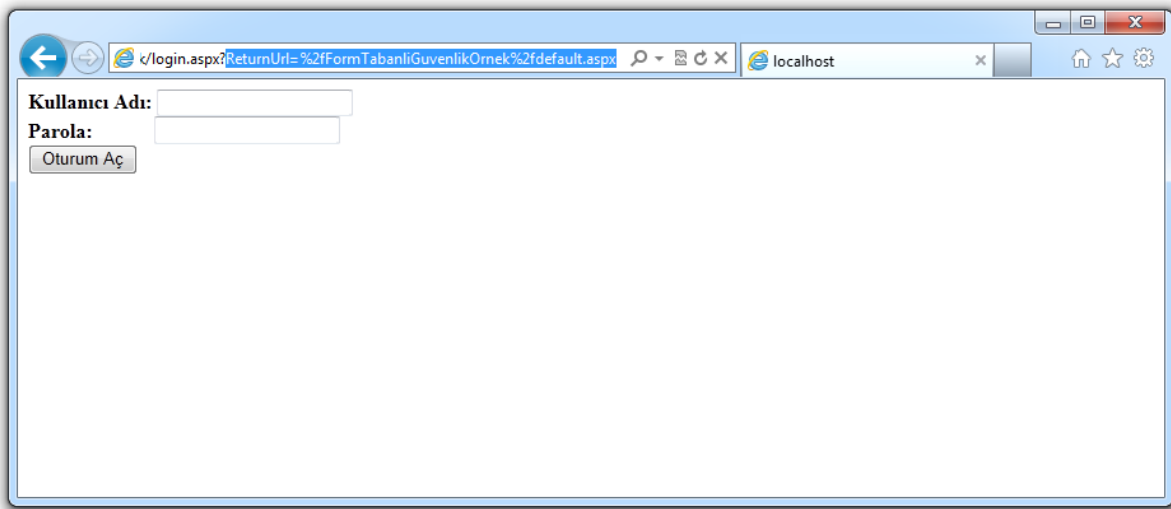
```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TbKullaniciAdi.Text == "Burak" && TbParola.Text == "1111")
    {
        FormsAuthentication.RedirectFromLoginPage(TbKullaniciAdi.Text, true);
    }
    else
    {
        Response.Write("Yanlış giriş yaptınız...");
    }
}
```

Yukarıdaki kod bloğu, kullanıcının girdiği verileri sorguluyor. Eğer kullanıcı adı Burak ve parola da 1111 ise kullanıcı doğru giriş yapmış olacaktır ve **FormsAuthentication** sınıfının **RedirectFromLoginPage** metodu ile kullanıcı, kendisini login sayfasına yönlendiren bir önceki sayfaya geri yönlendirilecektir. Eğer kullanıcı direkt olarak login.aspx'e geldiyse, defaultUrl'de belirtilen sayfaya yönlendirilecektir. RedirectFromLoginPage metodunun içindeki parametrelerin ilki, kullanıcı adını; ikincisi ise cookie oluşturulup oluşturmayacağı bilgisini alır. Bu işlemleri yaptıktan sonra, uygulama default.aspx sayfasından çalıştırıldığında, kullanıcı, direkt olarak login.aspx'e yönlendirilecektir çünkü henüz oturum açılmamıştır. Kullanıcı login.aspx'te gerekli bilgileri girdikten sonra default.aspx sayfasına yönlendirilecektir. Çünkü; login.aspx'e buradan yönlendirilmişti.

Projeyi çalıştırdıktan sonra login.aspx sayfasındayken, URL'e dikkat edildiğinde returnUrl'in burada yer alıyor olduğu görülecektir.



FormsAuthentication sınıfını burada yer aldığı şekilde kullanmak için `System.Web.Security` isimalanı, using bloklarında sayfaya eklenmiş olmalıdır.



Gerçekleştirilen örnekte, kullanıcının, kullanıcı adı ve parolası sayfanın içinde saklanmaktadır. Bu da güvenlik açısından pek doğru bir durum değildir. Eğer istenilirse, kullanıcı bilgileri, konfigürasyon dosyasında (web.config) depolanabilir. Kullanıcı bilgilerini web.config dosyasında saklamak, Web Form'da saklamaktan kesinlikle daha güvenlidir. Çünkü; web.config dosyalarına web üzerinden erişmek mümkün değildir. Internet Explorer adres çubuğuna SiteAdi/web.config yazılıp web.config dosyasına erişilmeye çalışıldığında bu dosyanın gösterilmesinin yasak olduğunu belirten bir uyarı ile karşılaşılır. web.config dosyasında <forms> etiketinin ayarları aşağıdaki gibi değiştirilerek, kullanıcının bilgileri buraya eklendiğinde, web.config'in authentication bölümünün son hali aşağıdaki gibi olacaktır.

```
<system.web>
...
<authentication mode="Forms">
  <forms loginUrl="login.aspx" name="AspNetGüvenlikDemo"
    defaultUrl="default.aspx" >

    <credentials passwordFormat="Clear" >
      <user name="Burak" password="1111"/>
    </credentials>

  </forms>
</authentication>
...
</system.web>
```

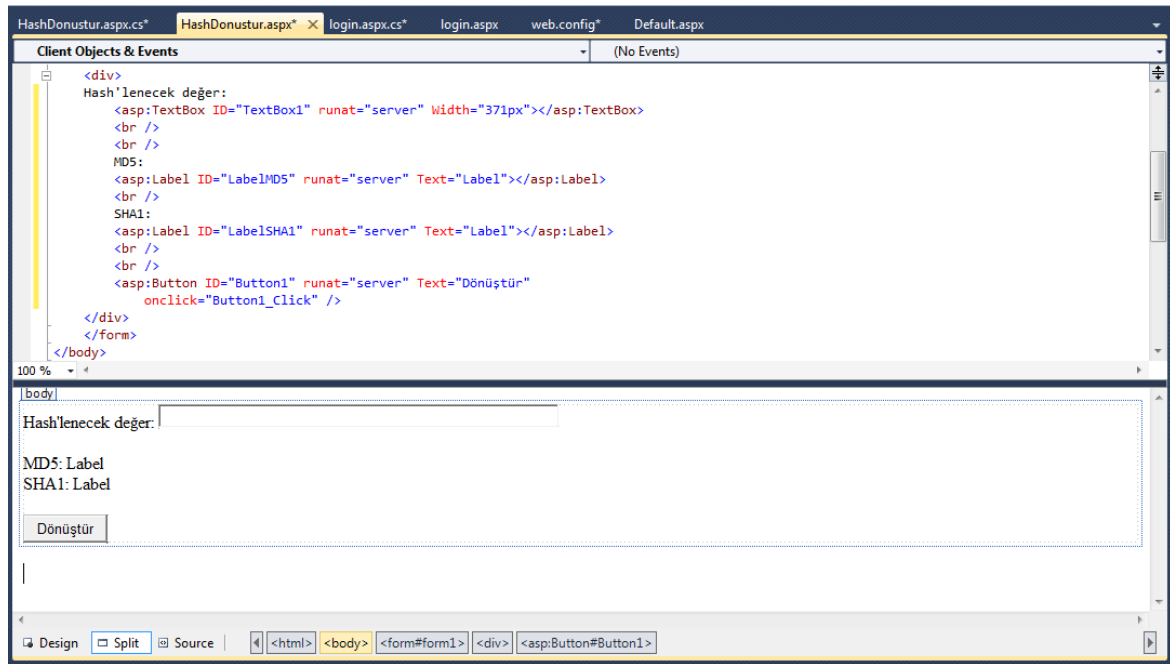
Kullanıcı bilgileri web.config dosyasına eklendi. Kod, dikkatle incelendiğinde, forms etiketi altında bulunan **Credentials** elemanı altında, kullanıcılar eklendiği görülebilir. **passwordFormat** alanını da, **Clear** olarak ayarlayarak, parolanın da clear text olarak saklanacağı belirtilmiştir. Bu alanda, istenilen kadar kullanıcı eklenebilir. Şimdi sıra geldi Button'un click olayında yazılan kodu güncellemeye, Login.aspx.cs'i açıp Buttonun click olayına gerekli kodlar yazıldığında ilgili metodun son hali aşağıdaki gibi olacaktır.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FormsAuthentication.Authenticate(TbkullaniciAdi.Text,TbParola.Text))
    {
        FormsAuthentication.RedirectFromLoginPage(TbkullaniciAdi.Text,true);
    }
    else
    {
        Response.Write("Yanlış giriş yaptınız...");
    }
}
```

```
}  
}
```

Kodları dikkatlice incelediğinde, sadece if'in içinde bulunan ifadenin değiştirildiğini fark etmiş olmalısın. Kullanıcı adı ve parola web.config'e eklendiğine göre, artık bu bilgileri sayfa içerisinde tutmanın bir anlamı olmayacaktır. **FormsAuthentication** sınıfının **Authenticate** metodu web.config'deki bilgilere göre oturum açma işlemini gerçekleştirir. İlk parametre olarak kullanıcı adını, ikinci parametre olarak ise parolayı alır. Eğer doğru kullanıcı adı ve parola kombinasyonu web.config dosyasında var ise bu metot geriye true değeri döndürür. Uygulama çalıştırılıp test edildiğinde web.config'de bulunan kullanıcı adı ve parola kombinasyonu ile giriş yapıldığında uygulamanın sorunsuz çalıştığı görülecektir.

Bu işlemlerden sonra site biraz daha güvenli hale geldi. Ama, herhangi bir kişi, bir şekilde web.config dosyasını ele geçirirse, sitede yer alan tüm kullanıcıların, kullanıcı adı ve parolalarına erişebiliyor olacaktır. Site daha güvenli hale getirilmek istenilebilir. web.config dosyasını ele geçiren bir kişi kullanıcı adını görse bile parolayı görmemeli ki sisteme buradaki kullanıcı hesaplarını kullanarak giriş yapamasın. web.config dosyasını gören bir kişinin buradaki parolaları görmesini engellemek için parolalar web.config dosyasına hashlenmiş olarak kayıt edilebilir. .NET Framework yazılım geliştiricilere bu alanda kullanmak için iki adet Hash formatı sunmaktadır. Bunlar; **SHA1** ve **MD5**. İkisinin de ortak özelliği, buradaki bilgilere göre gerçek şifreye geri dönülemez olmasıdır. Yani, herhangi biri bu verileri alsa bile, bir şey yapamıyor. Herhangi bir değerin SHA1 veya MD5 ile hashlenmiş halini elde etmek için **FormsAuthentication** sınıfının **HashPasswordForStoringInConfigFile** metodu kullanılabilir. Bu metodu kullanımını daha iyi anlayabilmek adına aşağıdaki örnek ile ilerleyelim. Forms Authentication anlatılırken geliştirilen örnek ile devam ederek yeni bir tane sayfa ekleyelim ve sayfa içine bir TextBox, iki Label ve bir tane de Button sürükleyip bırakalım. Bu işlemlerden sonra sayfanın görünümü aşağıdaki gibi olacaktır.



Aşağıda görülen kodları da Button'un Click olayına ekleyelim.

```
protected void Button1_Click(object sender, EventArgs e)  
{  
    LabelMD5.Text =  
        FormsAuthentication.HashPasswordForStoringInConfigFile(TextBox1.Text,  
        "MD5");  
}
```



```
LabelSHA1.Text =  
FormsAuthentication.HashPasswordForStoringInConfigFile(TextBox1.Text,  
"SHA1");  
}
```



Yukarıda kodları bulunan uygulamayı test etmek için geçici olarak authorization düğümünde yer alan deny ifadesini kaldırabilirsin. Bu şekilde daha hızlı ve kolay test etme şansına sahip olacaksın.

FormAuthentication sınıfındaki HashPasswordForStoringInConfigFile metodu herhangi bir stringin belirttiğimiz formattaki karşılığını döndürmektedir. Metodun kullanımı oldukça kolaydır. Metodun ilk parametre olarak hashlenecek olan değeri, ikinci parametre olarak da hangi formatta hashleyeceği bilgisini almaktadır. TextBox'a 1111 değeri girilip Button'a tıklanıldığında 1111 değerinin her iki formattaki karşılığı ekrana yazılacaktır.

Şimdi sıra geldi web.config dosyasında hangi algoritmayı kullanalım sorusuna. MD5 daha hızlı çalışırken SHA1 daha yavaş çalışıyor ama çözülmesi çok daha zor ihtiyaca göre istenilen algoritma kullanılabilir. Devam eden örnekte SHA1 tercih edilmiştir ve web.config dosyasındaki **credentials** bölümünü aşağıdaki gibi ayarlanmıştır.

```
<system.web>  
...  
...  
<authentication mode="Forms">  
  <forms loginUrl="login.aspx" name="AspNetGuvencilikDemo"  
    defaultUrl="default.aspx" >  
  
    <credentials passwordFormat="SHA1">  
  
      <user name="Burak"  
        password="011C945F30CE2CBAFC452F39840F025693339C42"/>  
  
    </credentials>  
  
  </forms>  
</authentication>  
...  
</system.web>
```

Görüldüğü üzere, geliştirilen uygulama daha güvenli bir hale getirildi. Kullanıcı adları ve parolaları, uygulama ayarlama dosyasında kolayca depolanabilmektedir. Ama, istenilirse bu bilgiler herhangi bir veri tabanı

uygulanmasında da depolanabilir. Kullanıcıları, herhangi bir veritabanında depolamak için yapılacak tek değişiklik tahmin edileceği üzere login.aspx'te olacaktır. Kullanıcı adı ve parola sorgulanıp eğer doğru ise kullanıcı sisteme kabul edilecektir.

Dosyaya Özel Güvenlik Ayarı

Şu anda kadarki yapılan ayarlar, sitenin tamamı için geçerlidir. Siteye gelen tüm kullanıcılar, herhangi bir sayfaya erişmek istediklerinde, güvenlik kontrolünden geçmelidirler. Ancak, bazı durumlarda kullanıcıların bazı sayfalara oturum açmadan erişmesi istenilebilir. Bu durum bir örnekle açıklanacak olursa, online alışveriş sitesi tasarlanıldığını düşünelim. Ziyaretçiler oturum açmadan ürünleri listeleyip özelliklerini görebilmeli ama ürün almaya karar verdiklerinde sipariş vermek için oturum açmalıdırlar. Bu bölümde anlatılan, şu ana kadar geliştirilen örneğe geri dönüldüğünde, bir önceki bölümde yeni bir tane sayfa ekleyip, girilen metnin hashlenmiş karşılığını kullanıcıya gösterilmişti. Şimdi, sadece bu sayfa için kullanıcıların oturum açması istenilsin. Diğer sayfalar, tüm ziyaretçilere açık olsun. Bu işlem için ilk olarak uygulama ayarlama dosyasından <authorization> </authorization> bölümüne yazılan kodlar kaldırılmalı ve sitenin tüm sayfaları tekrar anonim kullanıcılar için açık hale getirilsin.

```
<authorization>
<deny users="?" /> (Bu satırı kaldırıyoruz!)
</authorization>
```

Aşağıdaki kodlar ile de sadece HashDonustur.aspx sayfası için oturum açmayı gerektirecek olan, aşağıda görülen kodları <configuration>.....</configuration> tagleri arasında uygulama ayarlama dosyasına ekleyelim.

```
<location path="HashDonustur.aspx">
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</location>
```

<location> ile hangi sayfa için güvenlik ve yetkilendirme ayarı yapılacağı belirleniyor. <location> kod bloğu arasında bulunan kodlar ise bölümün başlarından beri bahsedilen kodlardan farklı değil.

EĞİTİM :

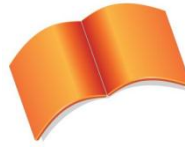
**GÜVENLİK VE ÜYELİK
YÖNETİMİ**

Bölüm :

Üyelik Yönetimi

Konu :

Üyelik Yönetimi



Microsoft Türkiye

Açık Akademi

Web sitelerinde Form Tabanlı Güvenlik kullanıldığında, bir üyelik yönetim sistemi yazmak gerekli olmaktadır. ASP.NET'in ilk sürümünde bu işlemler oldukça zor olmaktadır. ASP.NET 2.0 ile birlikte hayatımıza giren yeni yapılar ile birlikte üyelik yönetim sistemleri oldukça kolay ve hızlı bir şekilde geliştirilebilmektedir. Herhangi bir sitede kullanılacak bir üyelik yönetim sisteminde standart olarak bulunması gereken yapılar; kullanıcı eklemek, kullanıcıların kendi bilgilerini yönetmesi, şifre hatırlatma servisi ve güçlü bir rol yönetimi olarak tanımlanabilir. ASP.NET ile kullanıcılara sunulan Üyelik Yönetimi servisinde, bu yapıların tamamı mevcut ve kullanımı kolaydır. Üyelik yönetim servisi kullanılmadan önce ilk olarak site, üyelik yönetim servisinin kullanımı için ayarlanmalıdır.

Üyelik Yönetimi Kullanmak İçin Siteyi Ayarlamak

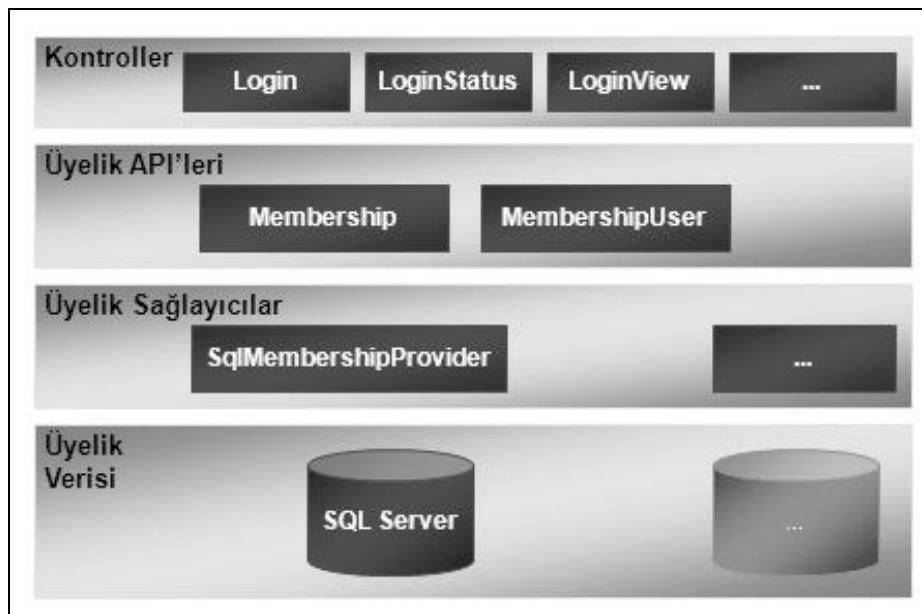
Üyelik yönetim sistemi kullanılmak istenen bir sitenin ayarlanmasına, uygulama ayarlama dosyasından (web.config) başlanmalıdır. Üyelik yönetim sistemini kullanmak için site, Form Tabanlı Güvenlik kullanacak şekilde ayarlanmalıdır. Şimdi yeni bir proje açıp adım adım bir site geliştirelim. İlk olarak Visual Studio üzerinde yeni bir web sitesi projesi oluşturalım. Sitenin güvenlik tipini Form tabanlı olarak ayarlamak için aşağıda görülen kodlarda olduğu gibi web.config'deki authentication düğümünü düzenleyelim.

```
<system.web>
... ..
<authentication mode="Forms">
  <forms defaulturl="default.aspx"></forms>
</authentication>
... ..
</system.web>
```

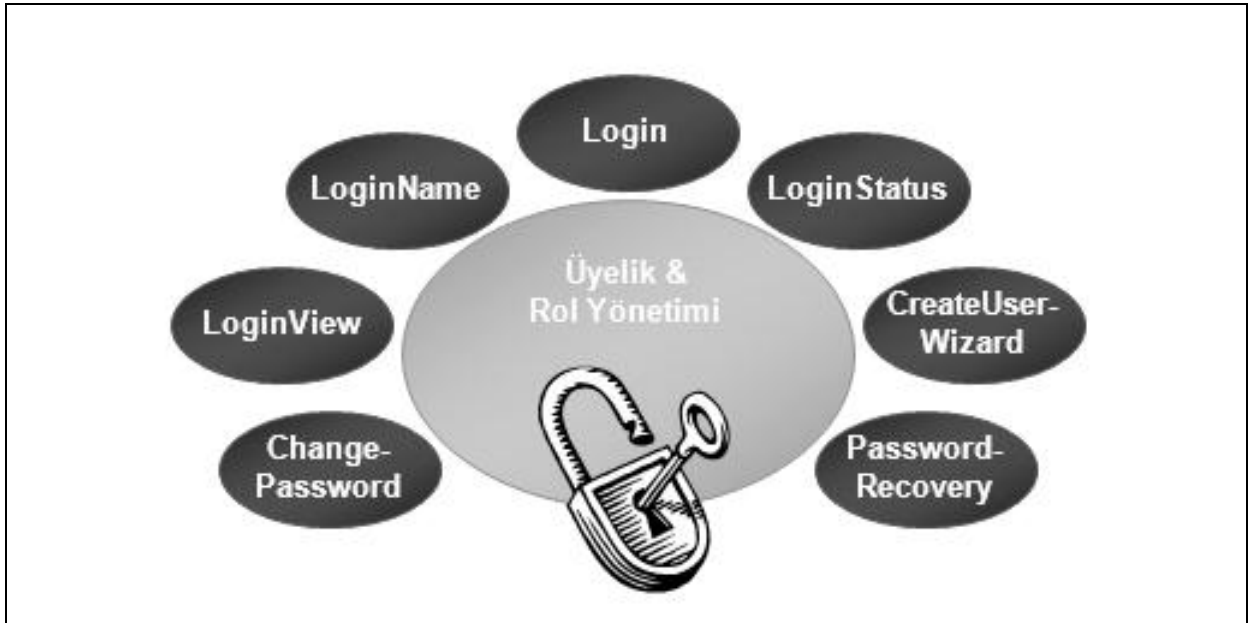


Üyelik Yönetim Sistemi Form Tabanlı Güvenlik ile birlikte kullanılabilir. Form Tabanlı Kullanım için gerekli olan ayarlar Güvenlik ve Yetkilendirme bölümünde ele alınmaktadır.

Sitede üyelik yönetim sistemini kullanmak için bu kadar ayar yeterli olacaktır. Bu kadar ayarlamadan sonra ilerleyen sayfalarda ele alacağımız kontroller sürüklenip bırakılarak üyelik yönetim sistemi kullanılabilir olacaktır. Bu kontroller ve arka planda kullanılan yapılar aşağıdaki resimde yer almaktadır.



Yukarıdaki resimde ASP.NET içerisinde yer alan üyelik yapısı görülmektedir. Görüldüğü üzere, en üstte **Login Kontrolleri** olarak adlandırılan kontroller bulunmaktadır. Bu kontroller ile, bir siteye üyelik yönetim yeteneği kazandırılabilir. Yazılım geliştiriciler, sitede üyelik yönetimi kullanmak için sadece Login kontrollerine bağımlı değildir. Şekilde, Login kontrollerinin hemen altında yer alan API'ler sayesinde de, üyelik yönetiminin sağladığı imkânlar kullanılabilir. Resimde bir kademe alta göz atıldığında da **SqlMembershipProvider** adında bir veri sağlayıcısı dikkat çekiyor. Bu veri sağlayıcısı, üyelik bilgilerinin SQL Server'da tutulduğu durumlarda SQL Server'a erişim sağlayan sınıfları içinde barındırmaktadır. Resmin en altında da üyelik verisinin tutulduğu alanlar listelenmiştir. Üyelik bilgilerini SQL Server'da saklayan yazılım geliştiriciler için işin ne kadar basit olacağı ilerleyen sayfalarda görülüyor olacaktır. Ancak, diğer veri kaynaklarında veri saklamak isteyen kullanıcılar için, işler o kadar da kolay değildir. SQL Server dışındaki veri kaynaklarında üyelik bilgilerinin saklanması için .NET Framework içinde tanımlanmış olan hazır üyelik sınıflarının, sağlayıcıların vb. yeniden yazılması gerekecektir.



Kontrollere göz atılacak olursa, kontrollerin isminden ne iş yaptıkları rahatlıkla anlaşılabilmektedir. Örneğin; **Login** kontrolü, sisteme daha önce kayıt olmuş bir kullanıcının oturum açma bilgilerini isteyerek sisteme giriş yapmasını sağlıyor. **ChangePassword** kontrolü, kullanıcıların parolalarını değiştirmelerine olanak tanıyor. **PasswordRecovery** kontrolü ise, unutulmuş parolaların hatırlatılması konusunda bir görev üstlenmiş. Sisteme yeni bir kullanıcı eklemek için kullanılacak olan kontrol ise **CreateUserWizard** olacaktır. **LoginStatus** kontrolü, siteye gelen kullanıcının oturum durumunu görüntüleyen kontroldür. Kullanıcının çevrimiçi veya çevrim dışı olmasına göre kullanıcıya farklı bağlantılar çıkararak kullanıcının durum değiştirmesini sağlıyor. **LoginName** kontrolü o an oturum açmış olan kullanıcının, kullanıcı adını taşıyor ve uygulama içinde herhangi bir yerde kullanılabilir. **LoginView** kontrolü de, **LoginStatus** kontrolüne benzer bir amaç için kullanılıyor. **LoginView** kontrolünün kullanım amacı **LoginStatus**'e göre oldukça geniştir. Bu kontrolde, kullanıcının oturum durumuna göre farklı içerik görüntülenebildiği gibi, ayrıca kullanıcının rol bilgisine göre de farklı bir içerik görüntülenebiliyor. **LoginStatus** kontrolü, farklı gruptan kullanıcılara farklı içerik sunmak için tasarlanmış bir kontroldür. Yazılım geliştiriciler, ASP.NET'in ilk sürümlerinde bu işlemi yapmak için panellerle boğuşmak zorundalardı. Ama artık bu işlemler oldukça kolay bir şekilde yapılabilir.

Login Kontrolleri İle Çalışmak

Kullanıcı Oluşturmak

Siteye kullanıcı eklemek için **CreateUserWizard** kontrolü kullanılmalıdır. CreateUserWizard kontrolü pek çok sitede olduğu gibi kullanıcıların sisteme kendilerini eklemesi için kullanılabileceği gibi, site yöneticileri tarafından yeni kullanıcı eklemek için de kullanılabilir.



CreateUserWizard kontrolünü yukarıda bahsedilen farklı amaçlar için kullanmak ufak bir ayarın değiştirilmesine bağlıdır. CreateUserWizard'ın özelliklerinden biri ile eklenen kullanıcının, eklendikten hemen sonra oturum açması da sağlanabiliyor. Bu özelliği aktif hale getirerek kullanıcının kendisinin kayıt olması sağlanır ve kullanıcı kayıt olduktan hemen sonra oturum açarak, sitenin hizmetlerinden faydalanmaya başlayabilir. Ancak, site yöneticisi tarafından eklenen bir kullanıcının, eklendikten hemen sonra oturum açması pek kullanışlı olmayacaktır. Görüldüğü üzere, kontrol, bu oranda özelleştirilebiliyor. CreateUserWizard kontrolü, diğer Login kontrolleri gibi Visual Studio içerisinde Toolbox'ta bulunan Login tabında yer almaktadır. Kontrol, Toolbox'tan sürüklenip bırakıldığında yukarıdaki gibi bir görüntü ile karşılaşılacaktır. Şu anda herhangi bir ayar yapılmadığı için kontrol, varsayılan ayarlarında görüntülenmektedir. Ama, bu görüntü tamamen kişiselleştirilebilmektedir. Bu görüntünün nasıl kişiselleştirileceği konusu açıklanmadan önce, kontrolü, bir sayfaya sürükleyip bırakarak nasıl çalıştığını incelemek mantıklı olacaktır. Bu işlem için, açmış olduğumuz projenin ana sayfasına bu kontrolü sürükleyip bırakalım ve Ctrl+F5 ile projeyi çalıştıralım. Proje çalıştırıldığında varsayılan görüntü ile karşılaşılacaktır. Gerekli bilgiler girilip Create User butonuna basıldığında, sayfa, uzunca bir süre arka planda işlem yapıp daha sonra da eğer herhangi bir hata oluşmadıysa ve SQL Server Express servisi çalışıyorsa kullanıcının başarılı bir şekilde oluşturulduğu bilgisini iletacaktır.



SQL Server Express servisi, geliştirmenin yapılacağı bilgisayarda çalışmıyorsa kullanıcı eklenemediği gibi varsayılan ayarlarla üyelik yönetim sistemi de kullanılamayacaktır. Çünkü; varsayılan ayarlar SQL Express'i işaret eder.

Create User butonuna tıklandıktan sonra, sayfa arka planda App_Data klasörünün içine bir tane veri tabanı dosyası oluşturup, oluşturulan kullanıcı bu veritabanının içine eklendi. Bu veritabanı dosyasını görmek için Solution Explorer'da, Solution'ın üzerinde sağ tıklayıp menüden Refresh Folder seçeneğini seçmek gerekiyor. Bu işlemden sonra Solution içerisinde bulunan dosyalar yenilenecek ve **App_Data** klasörü ile App_Data içerisinde bulunan **ASPNETDB.mdf** isimli dosya görünür olacaktır.

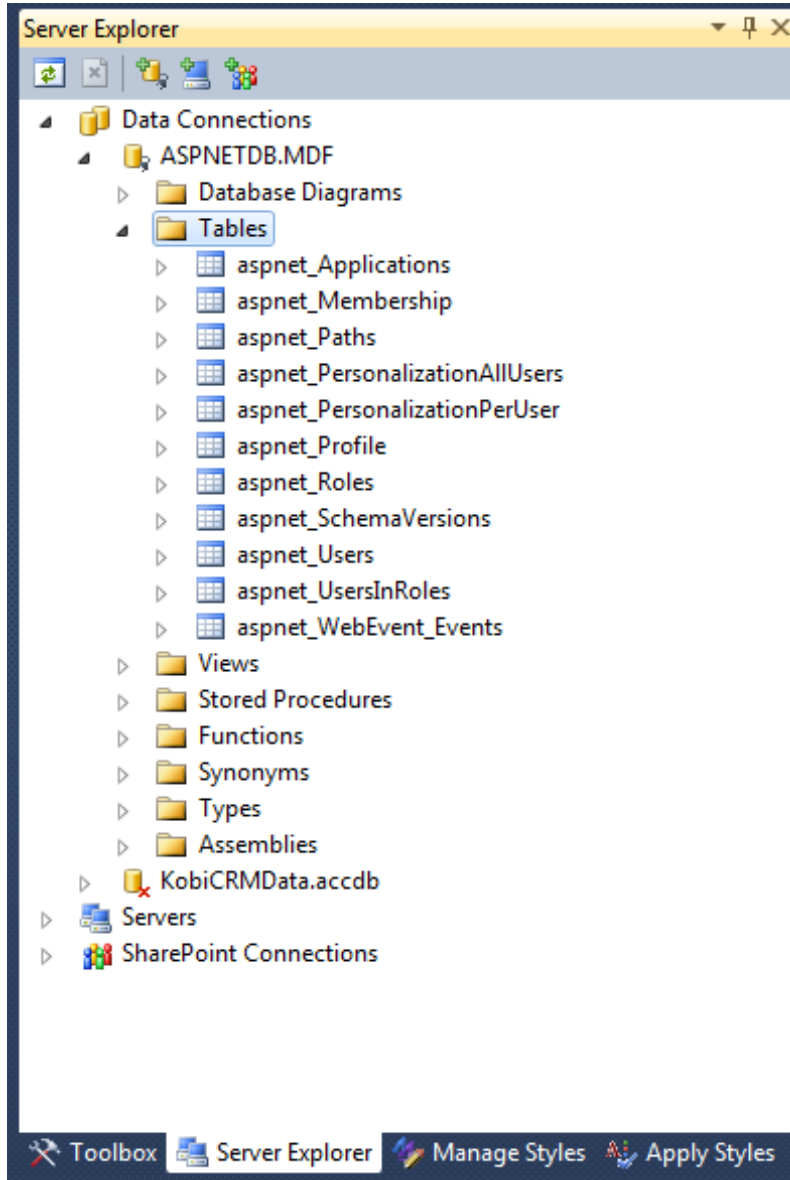


Kullanıcı eklerken alınabilecek olan hatalardan biri geçersiz parola olacaktır. ASP.NET üyelik yönetimi sisteminde parola oluşturmak için varsayılan güvenlik politikası gereği parola, minimum 7 karakterden oluşmalı ve minimum 1 tane alfabetik ve nümerik olmayan karakter içermelidir. Bu ayarlar da isteğe göre değiştirilebiliyor. Nasıl değiştirildiğini, ilerleyen sayfalarda göreceksin.

Eklenen kullanıcıyı görmek için **ASPNETDB.mdf** dosyasına çift tıklanabilir. Bu işlemten sonra Visual Studio, **Server Explorer** penceresi içinde ASPNETDB.mdf dosyasının içeriğini görüntüleyecektir.



Visual Studio üzerinde, veritabanına erişim gerektiren bir uygulama geliştiriliyorsa her seferinde pencere değiştirmek yerine Visual Studio içerisinde bulunan **Server Explorer** penceresi kullanılarak aynı ekranda hem veritabanının içeriği hem de uygulama görüntülenebilir. Böylece, daha hızlı bir şekilde uygulama geliştirilebilir.



ASPNETDB veritabanının içeriğine göz atıldığında, pek çok tablo, stored procedure, view, vb. nesne görülecektir. Bu nesnelerde, tüm üyelik bilgileri saklanırken, diğer nesneler de bu bilgilere erişim, yeni bilgi ekleme, bilgi güncelleme, bilgi silme gibi görevleri üstlenmişlerdir. Tablolarda hangi bilgilerin tutulduğu aslında isimlerinden anlaşılabilir. Aşağıdaki listede tabloların açıklamaları yer almaktadır.

Tablo Adı	Açıklaması
aspnet_Applications	Uygulamanın bilgilerinin saklanacak olduğu tablo.
aspnet_Membership	Parola, Güvenlik sorusu gibi üyelik bilgilerinin saklanacak olduğu tablo.
aspnet_Paths	Web Part kontrollerinin kullanıldığı sayfaların yolunun tutulacak olduğu tablo.
aspnet_PersonalizationAllUsers	Web Part kontrollerinin tüm kullanıcılar için varsayılan ayarlarının saklandığı tablo.
aspnet_PersonalizationPerUser	Web Part kontrollerinin kullanıldığı sayfalarda her kullanıcı için sayfanın diziliminin tutulacak olduğu tablodur.
aspnet_Profile	Kullanıcıya özgü profil bilgilerinin saklanacak olduğu tablo.
aspnet_Roles	Sitedeki tüm rollerin tanımlanacak olduğu tablo.
aspnet_Users	Kullanıcı bilgilerinin saklanacak olduğu tablo.
aspnet_UsersInRoles	Hangi kullanıcının hangi rol grubuna dâhil olduğu bilgisinin saklanacak olduğu tablo.

Bir tane CreateUserWizard kontrolü sayfaya eklenilip çalıştırıldığında arka planda neler olduğundan biraz bahsedikten sonra, biraz da CreateUserWizard kontrolünün kendisinden bahsedelim. Daha önceki sayfalarda CreateUserWizard kontrolünün tamamen kişiselleştirilebilir olduğundan bahsetmiştik. Şimdi, kontrolün özellikleri ile oynanarak, kontrolün üzerinde bulunan yazıların ve uyarıların nasıl kişiselleştirilebileceği hakkında konuşalım. Visual Studio ortamında kontrolün özelliklerine göz atıldığında pek çok özellik değerinden dolayı tanıdık gelecektir. Çünkü; varsayılan olarak görüntülenen yazıların her biri aslında birer özelliktir ve değiştirilebilir. Burada, bunların hepsinin teker teker nasıl değiştirileceğinden bahsetmek yerine bir kaç tanesinden bahsedip geriye kalanları sana bırakıyoruz. Örneğin; kontrolün en üstünde bulunan **User Name** olarak görüntülenen alana **UserNameLabelText** özelliğinden, **Password** yazan alana da **PasswordLabelText** özelliğinden erişilebilir. Diğer etiketler için de bu böyledir. CreateUserWizard kontrolünün kullanıcıya göstermiş olduğu hata mesajları da değiştirilebilir. Örneğin; kullanıcı, geçerli güvenlik politikasına göre uygun bir parola girmedikçe alacağı uyarı **InvalidPasswordErrorMessage** özelliği ile belirlenmektedir. Bu uyarı istenilen şekilde değiştirilebilir. Hatta bu özellik, belirlenen güvenlik politikasına göre dinamik olarak oluşturulabilmektedir. Hata mesajı dikkatle incelendiğinde, süslü parantezler içinde parametrelerle geçerli olan politikanın değerleri kullanıcıya gösterilebilmektedir.

Kullanıcı eklerken dikkat edilmesi gereken bir özellik parola girilirken minimum yedi karakter ve en az bir tane alfa-nümerik olmayan karakterden oluşan bir parola kombinasyonunun kullanıcıdan istendiğidir. Bu ayarların varsayılan ayarlarda bırakılması, sitenin güvenliği açısından her ne kadar olumlu bir durum olsa da zaman zaman bu ayarlar üzerinde oynama yapılması istenilebilir. Bir sitenin güvenlik politikası gereği minimum dört karakter yeterli iken, bazı siteler için varsayılan ayar olan yedi karakter bile yetmeyecektir. ASP.NET'te sitenin varsayılan ayarları machine.config dosyasından gelmektedir. Eğer makine düzeyinde bir ayar yapılmak isteniyorsa gidilecek olan yer **C:\WINDOWS\Microsoft.NET\Framework\vX.XXXX\CONFIG** içindeki machine.config dosyası olmalıdır. Bu dosya açılıp incelendiğinde hiç de yabancı olunmayan web.config dosyasına benzeyen bir dosya olduğu dikkatli gözlerden kaçmayacaktır. Dosya açılıp incelendiğinde değiştirmek istenilen alan, System.Web altında olan Membership düğümünde olacaktır. Bu alan dikkatlice incelenirse Membership'e bir tane sağlayıcı eklendiği ve varsayılan ayarların belirlendiği görülebilir. Aşağıdaki kod bloğu machine.config'de yer alan Membership sağlayıcısının (Provider) tanımlandığı bölümdür.


```

<membership>

<providers>

<add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider, System.Web,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"

connectionStringName="LocalSqlServer"
enablePasswordRetrieval="false"
enablePasswordReset="true"
requiresQuestionAndAnswer="true"
applicationName="/"
requiresUniqueEmail="false"
passwordFormat="Hashed"
maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="7"
minRequiredNonalphanumericCharacters="1"
passwordAttemptWindow="10"
passwordStrengthRegularExpression="" />

</providers>

</membership>

```

Yukarıdaki kodlara dikkat edilecek olursa minimum parola uzunluğunun (minRequiredPasswordLength) 7 olarak ayarlandığı, min alfa-nümerik olmayan karakter sayısının (minRequiredNonalphanumericCharacters) 1 olarak ayarlandığı görülecektir. Diğer özellikler de benzer şekilde ayarlanmıştır. Buradan bu ayarlar istenilen şekilde değiştirilebilir. Ancak unutulmamalıdır ki burada yapılacak olan ayarlar, bu makinede bulunan tüm sitelerde geçerli olacaktır. Bu durum çok fazla istenilen bir durum değildir. Hatta, mecbur kalınmadıkça bu dosyanın içeriği değiştirilmemelidir. O zaman tek çare kalıyor bu ayarları uygulamanın ayarlama dosyasında (Web.config) yapmak. Bu işlemi yapabilmenin yolu, web.config dosyasında yeni bir tane sağlayıcı tanımlamaktır. Ama, uygulama varsayılan ayarlarıyla çalıştırılmak istenildiğinde, varsayılan sağlayıcı yani **AspNetSqlMembershipProvider** kullanılmak istenecektir. Bu durumda yapılması gereken işlem varsayılan tarayıcıyı silip yeniden eklemek olacaktır. Burada silmek derken machine.config'den silmekten bahsetmiyoruz. Eğer bu sağlayıcı, machine.config'den silinirse doğal olarak özelliikle tanımlananlar dışında hiç bir sitede membership özellikleri çalışmayacaktır. Peki bu durumda nasıl silinecektir? ASP.NET'in çalışma mimarisi göz önüne getirildiğinde, ayarlar en üstten en alta doğru ayarlama dosyaları ile taşınmaktadır. Bir özellik bir alttaki ayarlama dosyasında geçerli değilse, bir üstteki ayarlar geçerli olur. Burada yapılacak olan işlem aslında machine.config'den gelen ayarları geçersiz kılıp yeniden tanımlamak olmalıdır. Bu işlem için web.config dosyasına aşağıdaki kodlar eklenebilir.

```

<system.web>
... ..
<membership>

  <providers>

    <remove name="AspNetSqlMembershipProvider" />

    <add connectionStringName="LocalSqlServer"
enablePasswordRetrieval="false"
enablePasswordReset="true"
requiresQuestionAndAnswer="true"
applicationName="/"
requiresUniqueEmail="false"
passwordFormat="Hashed"
maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="5"
minRequiredNonalphanumericCharacters="0"

```

```

passwordAttemptWindow="10"
passwordStrengthRegularExpression=""
name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider,
System.Web, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" />

</providers>

</membership>
... ..
</system.web>

```

Bu işlemi yapmanın en basit yolu machine.config'den ayarları kopyalayıp <remove> düğümünden sonra yapıştırmaktır. Yukarıdaki yeni ayarlara dikkat edilecek olursa minimum parola uzunluğu 5 ve minimum alfa-nümerik olmayan karakter sayısı da 0 olarak ayarlanmıştır.



Membersip sağlayıcısının ayarlarını değiştirmenin bir diğer yolu da IIS'in ayarları üzerinden bunu gerçekleştirmektir. IIS'de üzerinde çalışılacak olan siteye gelinir ve gerekli ayarlar yapılır. IIS üzerinden gerçekleştirilen ayarların da web.config'e yazılacağını unutmamak gerekir.

Siteye Kod Yazarak Kullanıcı Ekleme

Siteye kullanıcılar CreateUserWizard ile eklenebildiği gibi, Membership sınıfı kullanılarak da eklenebilmektedir. Bu işlem için yazılım geliştiricilere sunulan toplam dört adet aşırı yüklenmiş metod bulunmaktadır. Aşağıdaki tabloda metotların söz dizimleri görüntülenmektedir.

Membership.CreateUser(string username, string password)
Membership.CreateUser(string username, string password, string email)
Membership.CreateUser(string username, string password, string email, string passwordQuestion, string passwordAnswer, bool isApproved, out MembershipCreateStatus status)
Membership.CreateUser(string username, string password, string email, string passwordQuestion, string passwordAnswer, bool isApproved, object providerUserKey, out MembershipCreateStatus status)

Görüldüğü üzere bu metotlar yardımı ile dört farklı şekilde kullanıcı eklenebilmektedir. Bir örnekle bu kodları kullanalım. Örneğin daha anlaşılır olması adına, üçüncü satırdaki metodu tercih ediyoruz. Bu işlem için projemizde yeni bir web form oluşturalım ve bir tane Button sürükleyip bıraktıktan sonra, Click olayını ele alıp aşağıdaki kodları yazalım.

```

protected void Button1_Click(object sender, EventArgs e)
{
    MembershipCreateStatus durum;

    Membership.CreateUser("BurakBatur", "11111@w", "abc@def.com ",
        "deneme", "test", true, out durum);

    switch (durum)
    {
        case MembershipCreateStatus.DuplicateEmail:
            Response.Write("Belirtilen Email adresi daha önce
kullanılmış...");
            break;
        case MembershipCreateStatus.DuplicateProviderUserKey:
            break;
        case MembershipCreateStatus.DuplicateUserName:
            break;
        case MembershipCreateStatus.InvalidAnswer:

```

```

        break;
        case MembershipCreateStatus.InvalidEmail:
        break;
        case MembershipCreateStatus.InvalidPassword:
        break;
        case MembershipCreateStatus.InvalidProviderUserKey:
        break;
        case MembershipCreateStatus.InvalidQuestion:
        break;
        case MembershipCreateStatus.InvalidUserName:
        break;
        case MembershipCreateStatus.ProviderError:
        break;
        case MembershipCreateStatus.Success:
        Response.Write("kullanıcı oluşturuldu...");
        break;
        case MembershipCreateStatus.UserRejected:
        break;
        default:
        Response.Write("kullanıcı oluşturulamadı...");
        break;
    }
}

```

Bu alanda, son parametreye kadar yapılan işlemler genelde alışık olunan metot çağırma işlemidir. Son parametreye kadar olan parametreler, kullanıcı bilgilerini gönderirken kullanılıyor. Son parametre ise, out sözcüğü ile MembershipCreateStatus türünde bir değişken gönderip, bu değişkenin, değerini metottan alıp, program içinde kullanılmasını sağlıyor. **MembershipCreateStatus** .NET Framework içerisinde tanımlı olan bir numaralandırıcıdır (Enum). Kullanıcının oluşturulup oluşturulmadığı bilgisinin yanında, bir de kullanıcının neden oluşturulmadığı bilgisini içerir. Bu sayede, sitede işlem yapan kullanıcıya anlamlı hata mesajları gösterilmesini sağlar. Yukarıdaki kod bloğunda tüm durumlar yazılmıştır. Fakat, sadece iki tanesinde yapılacak işlem belirtilmiştir. Diğer alanların doldurulmasını sana bırakıyoruz. Site çalıştırılıp Button'a tıklanıldığında kullanıcı eklenecek ve herhangi bir hata yoksa Kullanıcı Oluşturuldu bilgisi ekrana yazılacaktır.



Yukarıdaki kodların çalışması için `using System.Web.Security;` kod bloğu sayfanın en üstündeki namespace referansları alanına eklenerek kullanılan sınıfların içinde yer aldığı NameSpace projeye dahil edilmelidir.

Kullanıcıların Oturum Açmasını Sağlamak

Şu ana kadar, site, Forms Authentication kullanacak şekilde ayarlandı ve siteye kullanıcı eklendi. Fakat, kimlik denetimi bölümü eklenmedi. İlk olarak uygulama ayarlama dosyasına (web.config) aşağıdaki kodlar eklenerek oturum açmamış olan kullanıcıların sisteme girmesini engelleyelim.

```

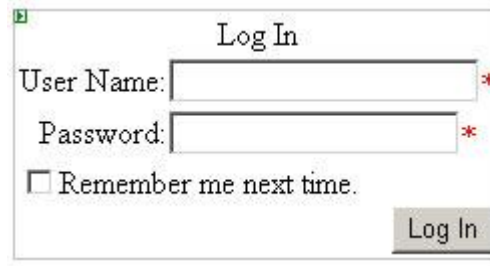
<system.web>
... ..
  <authorization>
    <deny users="?" />
  </authorization>
... ..
</system.web>

```

Login Kontrolü

Yukarıda bulunan kodlar web.config'e eklenip herhangi bir sayfa talep edildiğinde login.aspx sayfasının bulunamadığından bahseden bir hata mesajı ile karşılaşılacaktır. Bu gayet doğaldır çünkü hatırlanacağı üzere eğer ayarlar değiştirilmediyse kullanıcı geri dönüş URL'i ile birlikte **login.aspx** sayfasına yönlendirilecekti.

Yukarıdaki kodların eklendiği projeye bir tane Login.aspx sayfası eklenip içine de bir tane Login kontrolü sürüklenip bırakıldığında siteye gelen oturum açmamış kullanıcıların oturum açması sağlanabilecektir.



Projeye login.aspx eklenip içerisine Login kontrolü sürüklendikten sonra siteye erişmeye çalışan oturum açmamış olan kullanıcılar Login.aspx'e yönlendirilecek ve kullanıcı adı ile parola girebilecekleri bir alanla karşı karşıya kalacaklardır. Login kontrolü de tüm Login Web Server Kontrolleri gibi tamamen kişiselleştirilebilmektedir. İstenilirse tüm yazılar Türkçe olarak düzenlenebilir.

Kullanıcı, Login kontrolündeki **Log In** butonuna tıkladığında oturum açılacak ve otomatik olarak, login.aspx sayfasına yönlendirme yapan, ilk talep edilen sayfaya geri yönlendirilecektir.

Login kontrolünde dikkat edilmesi gereken bir özellik de **Remember me next time** yazılı olan CheckBox'tır. Eğer bu kutucuk işaretlenirse kullanıcının bilgisayarına bir Cookie atılacak ve kullanıcı bu siteye tekrar erişmeye çalıştığında atılan Cookie'den değerler okunarak kullanıcıdan tekrar kullanıcı adı ve parola bilgileri istenmeyecektir.

Sitelerde bulunan oturum açma sayfaları göz önüne getirildiğinde genellikle bu sayfalarda oturum aç linki dışında başka linkler de bulunur bunlardan biri **Siteye Üye Ol** ve bir diğeri de **Şifremi Unuttum** linkleridir. Bu linkler iyi tasarlanmış bir sitede bulunması gereken linklerdir. Bu işlemi yapmak için Login kontrolünün özellikler penceresi kullanılabilir.

Kullanıcıları Üye Ol sayfasına yönlendirecek olan özellikler **CreateUserIconUrl**, **CreateUserText** ve **CreateUserUrl** özellikleridir. Bu özelliklerden istenilen özellikler ayarlanıp kullanıcının tek bir linke tıklayarak Üye Ol sayfasına yönlendirilmesi sağlanmış olur. Aşağıdaki resim Visual Studio ortamında özellikler penceresinden bu özelliklerin ayarlanmasını göstermektedir.

CreateUserIconUrl	
CreateUserText	Yeni Kullanıcı
CreateUserUrl	~/Default2.aspx

Kullanıcı şifresini unuttuğunda hatırlaması için yönlendirilecek olduğu sayfayı belirlemek için **PasswordRecoveryIconUrl**, **PasswordRecoveryText** ve **PasswordRecoveryUrl** özellikleri kullanılmalıdır. Bu özelliklerden istenilen özellikler ayarlanıp kullanıcı kendisine şifre hatırlatacak olan sayfaya yönlendirilir. Aşağıdaki resim Visual Studio ortamında özellikler penceresinden bu özelliklerin ayarlanmasını göstermektedir.

PasswordRecoveryIconUrl	
PasswordRecoveryText	Şifremi Unuttum
PasswordRecoveryUrl	~/SifremiUnuttu

Yapılan bu ayarlardan sonra, Login kontrolü aşağıdaki görünümü almış olmalıdır.

Kod Yazarak Oturum Açmak

Kod yazarak oturum açmak için Login kontrolünün **Authenticate** olayı kullanılmalıdır. Eğer bu olayı yakalayacak olan metod tanımlı ise, Login kontrolü otomatik olarak oturum açmayıp Authenticate olayında yazılı olan kodları işletmeye çalışacaktır. Sayfaya eklenmiş olan Login kontrolünün Authenticate olay yakalayıcısı oluşturulup aşağıdaki kodlar ilave edildiğinde kullanıcı siteye kod tarafından kabul ediliyor olacaktır.

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    if (Membership.ValidateUser(Login1.UserName, Login1.Password))
    {
        FormsAuthentication.RedirectFromLoginPage(Login1.UserName, true);
    }
}
```



- Bu alanda Login kontrolü yerine iki TextBox bir tane de Button ile de aynı işlem yaptırılabilirdi ama görüldüğü üzere Login kontrolünün **UserName** ve **Password** özelliklerine kod tarafından erişilebilmektedir.
- **FormsAuthentication.RedirectFromLoginPage** metodu Güvenlik ve Yetkilendirme bölümünden hatırlanıyor olmalıdır. Bu metod, kullanıcı geçerli bilgileri belirtti ise kullanıcının oturum açmasını sağlayıp login sayfasına gelinen sayfaya geri dönülmesini sağlıyordu.

Oturum Açmış Kullanıcılarla Çalışmak

Oturum açmış kullanıcılarla çalışmak için hazır sunulan kontroller ile birlikte Kullanıcıların Parola Değiştirmesi, Oturum Durumlarını görmesi, Kullanıcı adlarının görüntülenmesi gibi işlemlere olanak tanıyan hazır kontroller de bulunmaktadır.

LoginName Kontrolü

LoginName kontrolü, adından da anlaşılacağı üzere oturum açan kullanıcının kullanıcı adını görüntülemek için kullanılmaktadır. Üyelik gerektiren sitelerde genellikle oturum açan kullanıcının kullanıcı ismi kendisine gösterilir. Bu işlem için, ASP.NET Login kontrollerinde bulunan LoginName kontrolü site içerisinde kullanılmak istenilen yere sürüklenip bırakılarak kolaylıkla kullanılabilir. Kontrol sürüklenip bırakıldığında Label kontrolü ile benzerlik gösterdiği görülmektedir. Bu kontrol de, diğer kontroller gibi kişiselleştirilerek görünüm ayarlarıyla oynanabilir. Aşağıdaki kod bloğunda LoginName kontrollünün kullanımı yer almaktadır.

```
Hoşgeldiniz
<asp:LoginName ID="LoginName1" runat="server" />
```

Bu kodların bulunduğu sayfa çalıştırılıp oturum açıldığında aşağıdaki görüntü ile karşılaşılacaktır.



Kullanıcı adı, LoginName kontrolü ile elde edilebileceği gibi kod tarafından **User.Identity.Name** kodları ile de elde edilip kullanılabilir.

ChangePassword Kontrolü

ChangePassword kontrolü ile oturum açmış kullanıcılar kendi parolalarını değiştirebilmektedir. Kontrol varsayılan olarak aşağıda görüldüğü gibi gelmektedir. Tüm Login kontrollerinde olduğu gibi, bu kontrolde de tüm özellikler tanımlanmış durumdadır ve değiştirilebilmektedir.

ChangePassword kontrolünde de gerekli olan doğrulama kontrolleri kullanılmış olup kullanıcının yanlış bir giriş yapması ya da TextBox'ları boş bırakması gibi hataların önüne geçilebilmiştir. Kontrol üzerinde yer alan tüm yazılar tamamen Türkçeleştirilebilmektedir. Bunlardan bir kaçına örnek vermek gerekirse; Üzerinde **Change Password** yazan ve tıklanıldığında eski parolayı yenisi ile değiştirecek olan Button'un Text özelliğini değiştirmek için **ChangePasswordButtonText** özelliği kullanılabilir veya en üstteki Başlık bilgisini değiştirmek için **ChangePasswordTitleText** özelliği kullanılabilir. Diğer özelliklerde benzer şekilde kontrolün özellikler penceresinden kolayca değiştirilebilmektedir.

Kontrolün nasıl çalıştığının görülmesi için; herhangi bir sayfaya bir tane ChangePassword kontrolü atılıp sayfa çalıştırılabilir. **Password** yazan bölüme kullanıcının aktif parolası girilmelidir. **New Password** yazan bölüme ve hemen altındaki TextBox'a da kullanıcının yeni parolası girilip **Change Password** butonuna tıklanılmalıdır. Bu işlemlerden sonra eğer herhangi bir hata oluşmadıysa Parolanın doğru bir şekilde değiştirildiğine dair bir uyarı alınacak ve **Continue** butonu görünür olacaktır. Continue butonuna tıklanıldığında sayfada hiç bir değişiklik olmamaktadır çünkü Continue butonuna tıklanıldığında oluşan olayı ele almak gerekmektedir.

Continue butonuna tıklanıldığında tetiklenecek olan olay **ContinueButtonClick** olayıdır ve bu olaya bir olay yakalayıcı yazılıp istenilen herhangi bir işlem yaptırılabilir. Aşağıdaki kodlar kullanıcıyı ChangePassword.aspx sayfasına yeniden yönlendirmektedir.

```
protected void ChangePassword1_ContinueButtonClick(object sender,
EventArgs e)
{
    Response.Redirect("ChangePassword.aspx");
}
```



ChangePassword kontrolünde eğer parola başarılı bir şekilde değiştirildiyse, kullanıcılar istenilen bir sayfaya otomatik olarak da yönlendirilebilmektedir. Bu işlem için kontrolün **SuccessPageUrl** özelliğinin ayarlanmış olması gerekmektedir.

Kod Yazarak Parola Değiştirmek

Kullanıcıların parolaları, ChangePassword kontrolü ile değiştirilebildiği gibi, istenirse **MembershipUser** sınıfından faydalanarak da değiştirilebilir. Bu işlem için MembershipUser türünde bir değişken tanımlanmalı ve sistemde bulunan herhangi bir kullanıcı buraya atanmalıdır. Daha sonra MembershipUser sınıfının ChangePassword metodu ile gerekli parola değiştirme işlemi yapılabilir. Bu konu, aşağıdaki örnek ile daha kolay anlaşılabilir. Geliştirmekte olduğumuz projemize bir tane sayfa ekleyip bu sayfaya da iki tane TextBox ve bir tane Button sürükleyip bırakalım. Sayfanın görünümü aşağıdaki gibi olmalıdır.

Düşünülen senaryoda o andaki oturum açan kullanıcının parolası değiştirilecektir. Ancak istenilirse kullanıcı adı bilinen herhangi bir kullanıcının parolası da değiştirilebilir. Sayfada bulunan butona tıklanıldığı zaman çalışacak ve kullanıcının parolasını değiştirecek olan kodlar aşağıda görüntülenmektedir.

```
protected void Button1_Click(object sender, EventArgs e)
{
    MembershipUser user =
    Membership.FindUsersByName(User.Identity.Name)[User.Identity.Name];

    bool ParolaDegistirildiMi =
        user.ChangePassword(TextBoxOldPassword.Text,
            TextBoxNewPassword.Text);

    if (ParolaDegistirildiMi)
    {
        Response.Write("Parola Değiştirildi...");
    }
    else
    {
        Response.Write("Parola Değiştirilemedi...");
    }
}
```

Kodlar satır satır incelendiğinde; ilk olarak **MembershipUser** tipinde bir değişken tanımlanıyor. Tanımlanan değişkeni kullanabilmek için tanımlanan değişkene sistemde bulunan bir kullanıcıyı atamak gerekiyor. Sistemde bulunan kullanıcılar da, **Membership** sınıfının **FindUserByName** metodu ile elde ediliyor. FindUserByName metodu, içerisine bir tane string kabul eder. Bu istediği string, tahmin edileceği üzere kullanıcı adıdır. Daha önceki sayfalardan da hatırlanacağı üzere User.Identity.Name, o an oturum açmış olan kullanıcının adını döndürmektaydi. FindUserByName metodu MembershipUserCollection türünde bir koleksiyon döndürür. Bu senaryoda, bunların içinden bir tanesini alıp kullanmak gerekmektedir ve dolayısıyla **[]** operatörü ile kullanılacak olan user belirtilip ele alınıyor. MembershipUser sınıfının **ChangePassword** metodu gerekli parola

değiştirme işlemini yapar ve eğer parola doğru bir şekilde değiştirildiyse true, hata varsa false değerini döndürür. Bu metod'un ilk parametresi eski, ikinci parametresi ise yeni paroladır. Kodlarda if-else bloklarıyla parolanın doğru bir şekilde değiştirilip değiştirilmediği kullanıcıya bildiriliyor.

LoginStatus Kontrolü

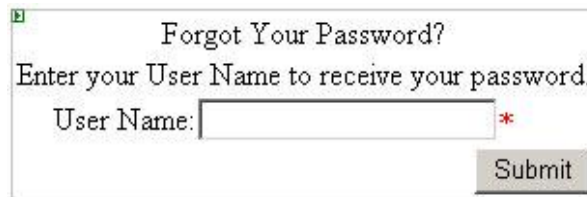
Kullanıcının Oturum durumuna göre farklı iki görüntüsü olan bu kontrol kullanıcının oturumu açıkken kapatmasını ve kullanıcı oturum açmamış durumda ise oturum açmasını sağlar. Kullanımı oldukça basit olan bu kontrol varsayılan olarak text tabanlıdır. Ama yazılım geliştiriciler isterlerse text yerine oturum kapalı iken oturum açılması için bir resim veya tam tersi durum için bir resim kullanarak kullanıcılara daha hoş görüntüler gösterebilirler. Eski zamanlarda yazılım geliştiriciler bu işlemleri bir takım tekniklerle panellerin görünürlüğü ile oynayarak yaparlardı. Ama artık bunların hiçbirine gerek kalmadı sadece bir LoginStatus kontrolü ile kullanıcının oturum durumuna göre Oturumu Kapat veya Oturum Aç linkleri kullanıcılara gösterilebilmektedir.

PasswordRecovery Kontrolü

Oturum açılması gereken bir sitede kullanıcılar genellikle parolalarını unuturlar ve hemen hemen her site bir şifre hatırlatma mekanizması kullanmaktadır. Bunlardan bazıları kullanıcının şifresini sıfırlayıp kullanıcıya o anda gösterirken, güvenliğe daha fazla önem verenler kullanıcının parolasını sıfırlayıp kullanıcının sistemde kayıtlı olan email adresine yeni şifresini göndermektedir. ASP.NET içerisinde yer alan PasswordRecovery kontrolünün de yaptığı işlem aslında tam olarak budur. Kullanıcının şifresini sıfırlar ve yeni şifreyi kullanıcının kayıtlı e-mail adresine gönderir. PasswordRecovery kontrolünün mail atabilmesi için SMTP ayarlarının uygulama ayarlama dosyasında (web.config) yapılmış olması gerekmektedir. SMTP ayarları aşağıdaki kodda görüldüğü gibi yapılabilmektedir.

```
<configuration>
.....
<system.net>
<mailSettings>
<smtp>
<network host="localhost" port="25"
defaultCredentials="true" />
</smtp>
</mailSettings>
</system.net>
.....
</configuration>
```

Bu mail ayarlarını yaptıktan sonra, herhangi bir sayfaya PasswordRecovery kontrolü sürükleyip bırakılarak kullanılabilir. Kontrol, sürüklenip bırakıldığında aşağıdaki gibi görünecektir.

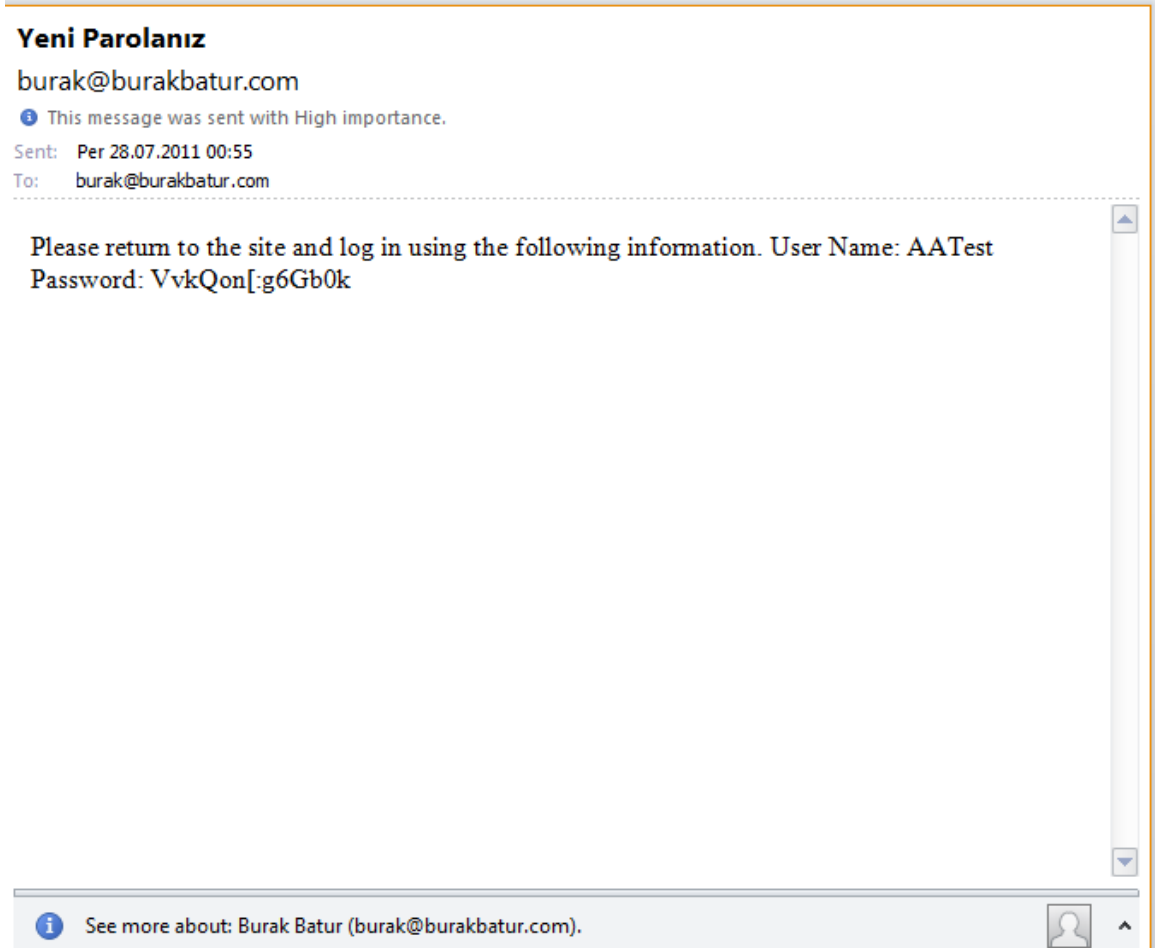


Sonrasında, tek yapılması gereken, kullanıcılara gönderilecek olan mailin ayarlarının yapılmasıdır. Kontrolün tüm ayarları yapıldıktan sonra kaynak kodları aşağıdaki gibi olmalıdır.

```
<asp:PasswordRecovery ID="PasswordRecovery1" runat="server">
  <MailDefinition From="burak@burakbatur.com"
    IsBodyHtml="True"
    Priority="High"
    Subject="Yeni Parolanız">
```

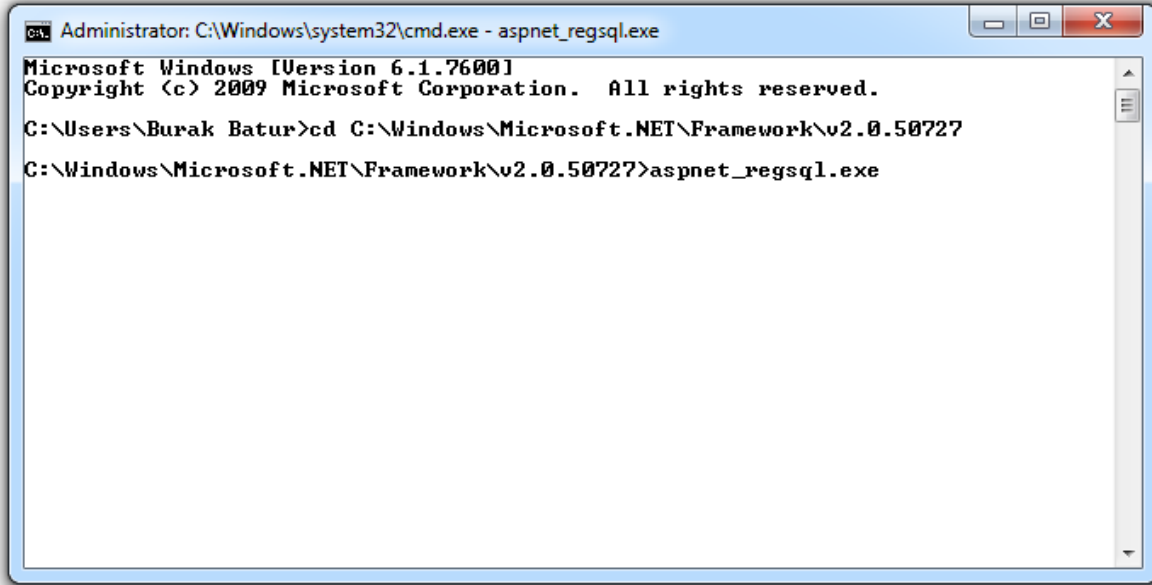

</MailDefinition>
</asp:PasswordRecovery>

Bu ayarlarda kullanıcılara gönderilecek mail'in burak@burakbatur.com adresinden gideceği ve konusunun Yeni Parolanız şeklinde olacağı belirtiliyor. Sayfa çalıştırılıp kontrol denendiğinde ilk olarak kullanıcı adını soracak. Geçerli bir kullanıcı adı belirtildikten sonra hatırlatma sorusu sorulacak ve geçerli cevap girildiğinde mail gönderilip bu durum kullanıcıya belirtilecek. Kullanıcılara gönderilen mail aşağıdaki gibi olmalıdır. Resim dikkatle incelendiğinde konu belirtilen şekilde geldi ve parola oldukça karmaşık bir şekilde üretilerek sıfırlandı.



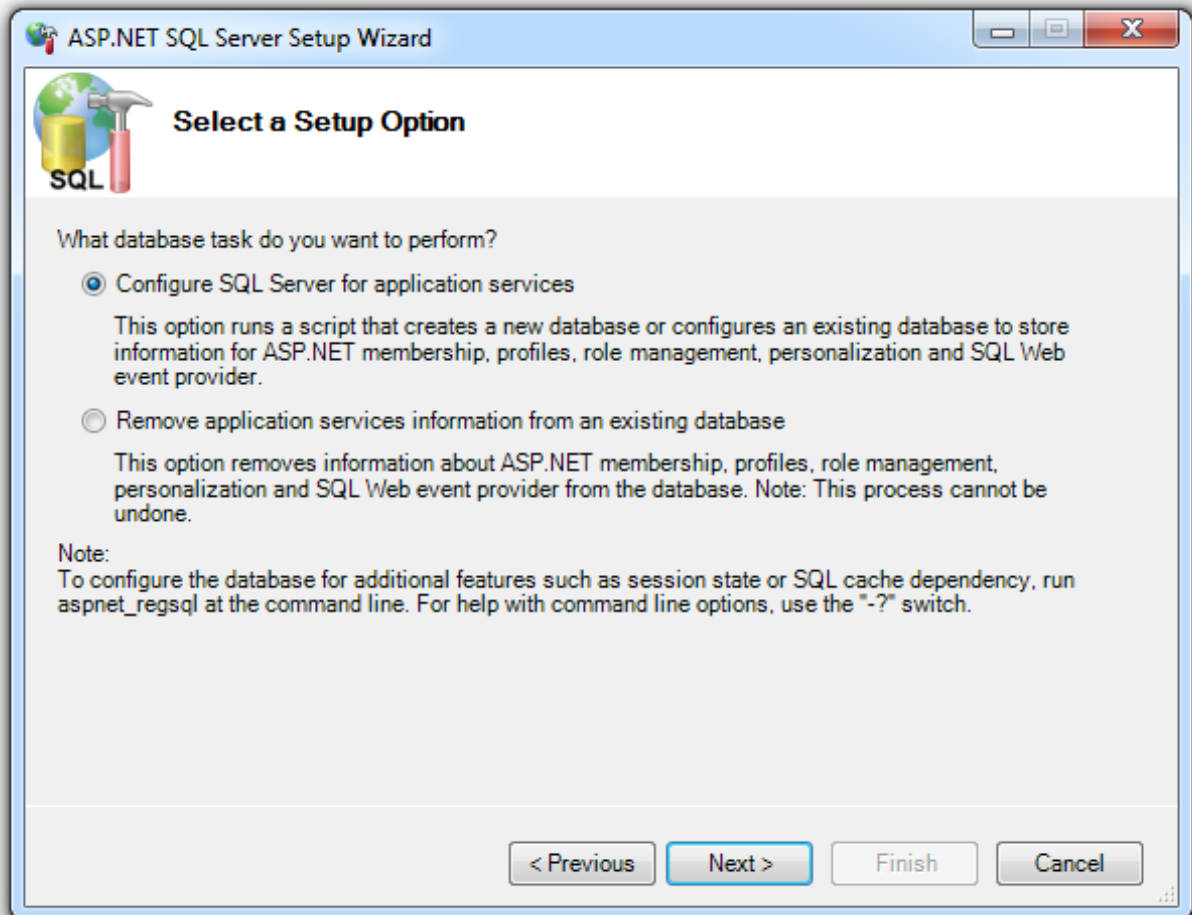
Üyelik ve Rol Bilgilerini İstenilen Bir Veritabanında Depolamak

Membership bilgileri varsayılan olarak **SqlExpress** üzerinde, sitede yeni oluşturulan bir veritabanında depolanmaktaydı. Çünkü; Membership sağlayıcısındaki ayarlar **LocalSqlServer**'i işaret ediyordu. Bu da Machine.config'de tanımlı olan SqlExpress'i işaret eden bağlantı cümlesi ile sağlanmıştı. Küçük uygulamalar için SqlExpress yeterli olacaktır ama daha büyük veya daha önceden çalışan ve yeni sisteme geçirilmeye çalışılan bir site için SqlExpress uygun bir çözüm değildir. Bu gibi durumlarda veriler mutlaka başka bir veritabanında depolanmalıdır. Bu işlem için ilk olarak verilerin depolanacak olduğu veritabanı üyelik ve rol bilgilerini depolayacak şekilde ayarlanmalıdır. Bu ayarlamayı yapmak için **Aspnet_RegSql.exe** uygulamasından faydalanılmalıdır. Aspnet_Regsql.exe uygulaması C:\WINDOWS\Microsoft.NET\Framework\vx.XXXX yolu altında bulunmaktadır. Uygulamayı çalıştırmak için aşağıdaki resimde gördüğün gibi Dos ekranında bu yola gidilir ve Aspnet_RegSql.exe yazılıp Enter'a basılır.

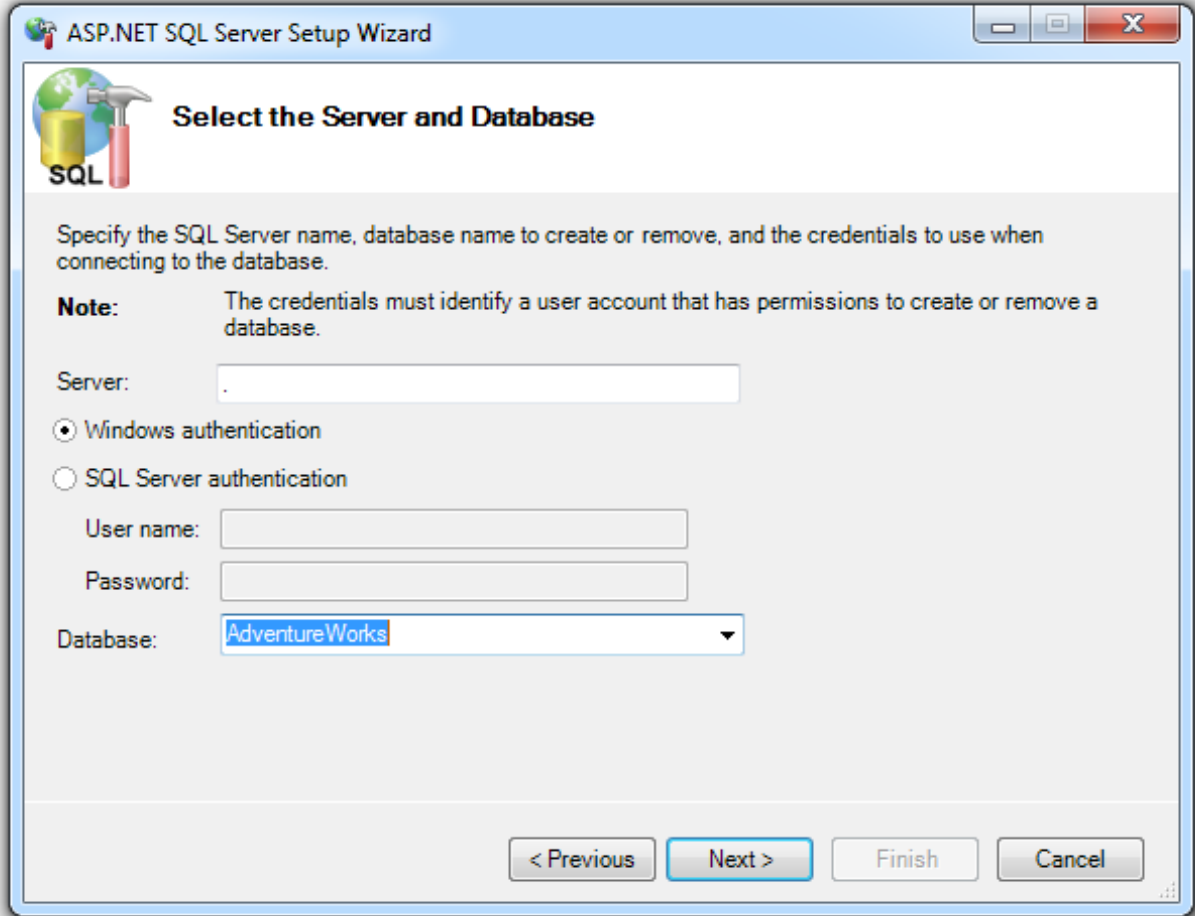


```
Administrator: C:\Windows\system32\cmd.exe - aspnet_regsql.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Burak Batur>cd C:\Windows\Microsoft.NET\Framework\v2.0.50727
C:\Windows\Microsoft.NET\Framework\v2.0.50727>aspnet_regsql.exe
```

Bu işlemten sonra bir sihirbaz ile karşılaşılacaktır. Next butonuna basıldıktan sonra iki seçenek ile karşılaşacaksınız. Bunlardan üstteki bir veritabanına üyelik yönetimi desteği verilmek istendiğini, diğeri ise veritabanından bu desteğin kaldırılmak istendiğini belirtir.

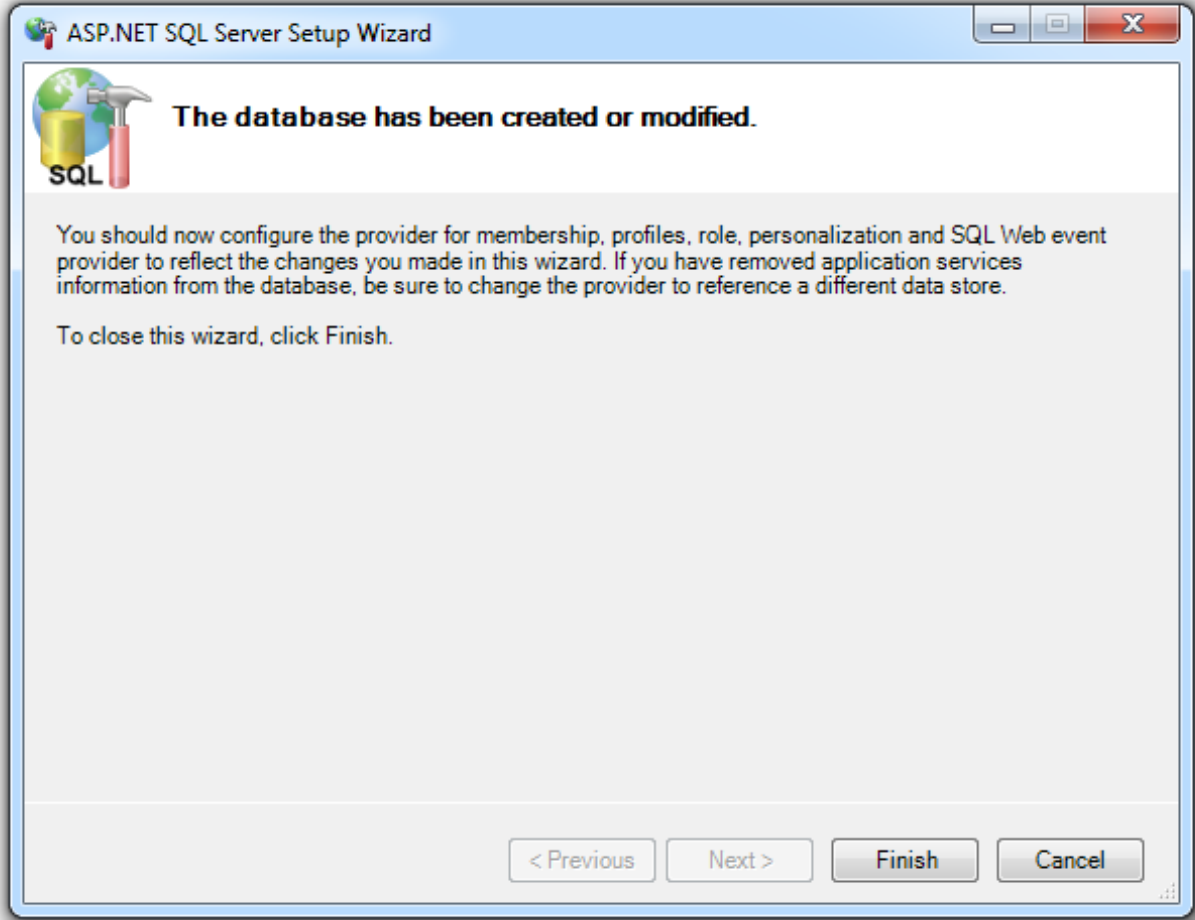


Configure SQL Server for application services seçeneğinin seçili olduğundan emin olunup Next tuşu ile devam edilir. Next butonuna basıldıktan sonra yeni karşılaşılan pencere uygulama servisleri için destek verilecek olan veritabanının seçilecek olduğu bölümdür. Bu alandan istenilen veritabanı seçilir ve Next tuşuna basılır.



The image shows a screenshot of the 'ASP.NET SQL Server Setup Wizard' window, specifically the 'Select the Server and Database' step. The window has a blue title bar with the text 'ASP.NET SQL Server Setup Wizard' and standard Windows window controls. Below the title bar is a header area with a SQL Server icon and the title 'Select the Server and Database'. The main content area contains instructions: 'Specify the SQL Server name, database name to create or remove, and the credentials to use when connecting to the database.' A note states: 'Note: The credentials must identify a user account that has permissions to create or remove a database.' The form includes a 'Server:' text box, two radio buttons for 'Windows authentication' (selected) and 'SQL Server authentication', and corresponding 'User name:' and 'Password:' text boxes. A 'Database:' dropdown menu is set to 'AdventureWorks'. At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Bu alandan sonra seçili olan veritabanı bilgileri gösterilip son bir onay için tekrar Next tuşuna basılır. Eğer herhangi bir problem yoksa belirtilen veritabanında gerekli tablo, stored procedure vb. gibi nesneler oluşturulur. Eğer herhangi bir hata oluşmadıysa veritabanının başarıyla oluşturulduğunu belirten aşağıdaki ekran görüntüsü ile karşılaşılr.



Bu ekran görüntüsünde bir şey daha söylenmektedir. Veritabanının başarıyla oluşturulduğu ve Membership ayarlarının yapılması gerektiği belirtilmektedir. Ancak şöyle bir yöntem tercih edilecektir. Membership sağlayıcısı LocalSqlServer adındaki bağlantı cümlesini kullanıyordu. Uygulamada bu bağlantı cümlesi için machine.config'den gelen ayarlar iptal edilip, aşağıda görüldüğü gibi yeniden oluşturulursa başka bir ayar değiştirmeden amaca ulaşılmış olunur.

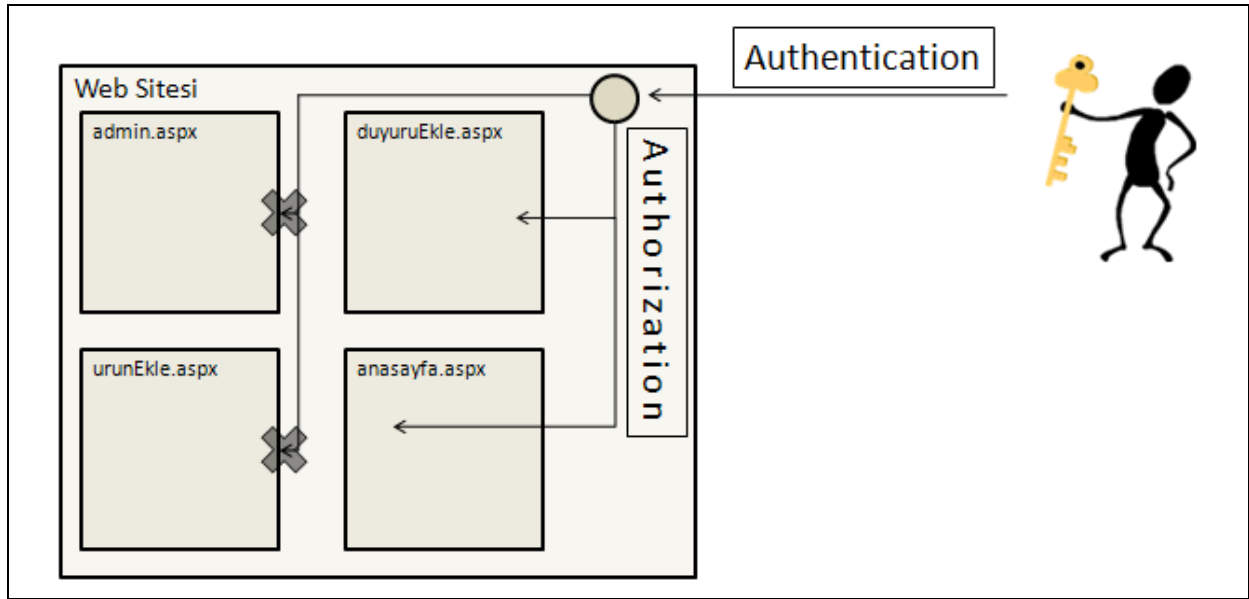
```
<configuration>
.....
<connectionStrings>
<remove name="LocalSqlServer" />

<add name="LocalSqlServer" connectionString="Data
Source=.;Initial Catalog=Adventureworks;Integrated
Security=True" providerName="System.Data.SqlClient" />

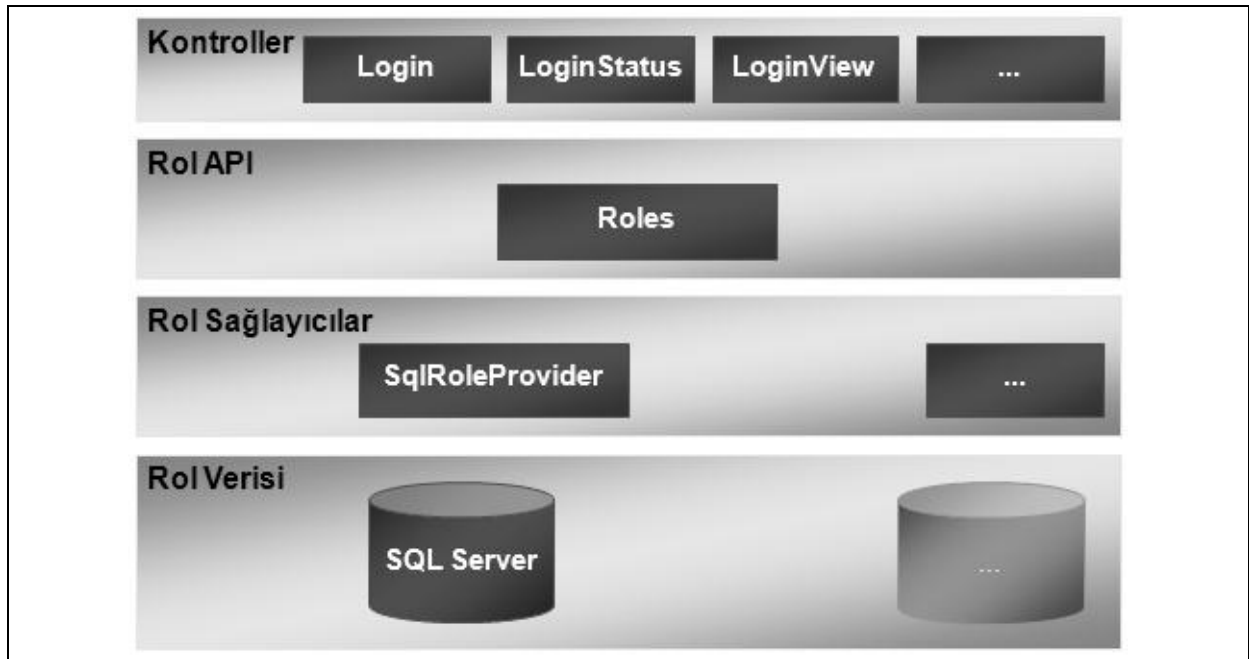
</connectionStrings>
.....
</configuration>
```

Rol Yönetimi

Önceki ekranlarda kullanıcıların sisteme giriş yapmaları ile ilgili konuları ele aldık ancak sisteme giriş yapan kullanıcıların sistemin içinde neler yapabileceği gibi işlemler hakkında konuşmadık. Hatırlayacağın üzere bir kullanıcının sisteme kabul edilmesi olayına Authentication, sistem içindeki davranışlarının kontrol edilmesine de Authorization denilmekteydi. Aşağıdaki resim incelendiğinde Authentication ve Authorization kavramları daha iyi anlaşılabilir.



Resimde gösterilen senaryoda, kullanıcı siteye girebiliyor ama her işlemi yapamıyor. Kullanıcının, site içerisinde yapabileceği işler oklarla belirlenmiş durumdadır. Kullanıcı, site içerisine giriş yapabildiği halde admin.aspx ve urunEkle.aspx sayfalarına giriş yapamamaktadır. ASP.NET'te bu işlem Rol yönetimi olarak adlandırılır.



Yukarıdaki resimde ASP.NET Rol yönetim şeması yer almaktadır. Şemada kontroller bölümünde her ne kadar bir kaç kontrol bulunsa da bu kontroller üyelik ile ilgilidir ve Rol yönetimi için sürüklenip bırakılabilecek hazır kontroller bulunmamaktadır. Bu kontroller daha önce de bahsedilen Membership kontrolleridir. Ama tabii ki Rol yönetimi kullanılan yerlerde de kullanılabilir. Bir altta Rol yönetimini gerçekleştirmek için gerekli sınıfların bulunduğu API yer almaktadır. Veri tabanı katmanına erişmek için sınıflar ve SQL Server arasında SqlRoleProvider yer almaktadır. SQL Server katmanında ise Rollerin tutulabileceği tablolar, view'lar ve veri giriş-çıkış işlemlerini gerçekleyecek olan yapılar yer almaktadır.

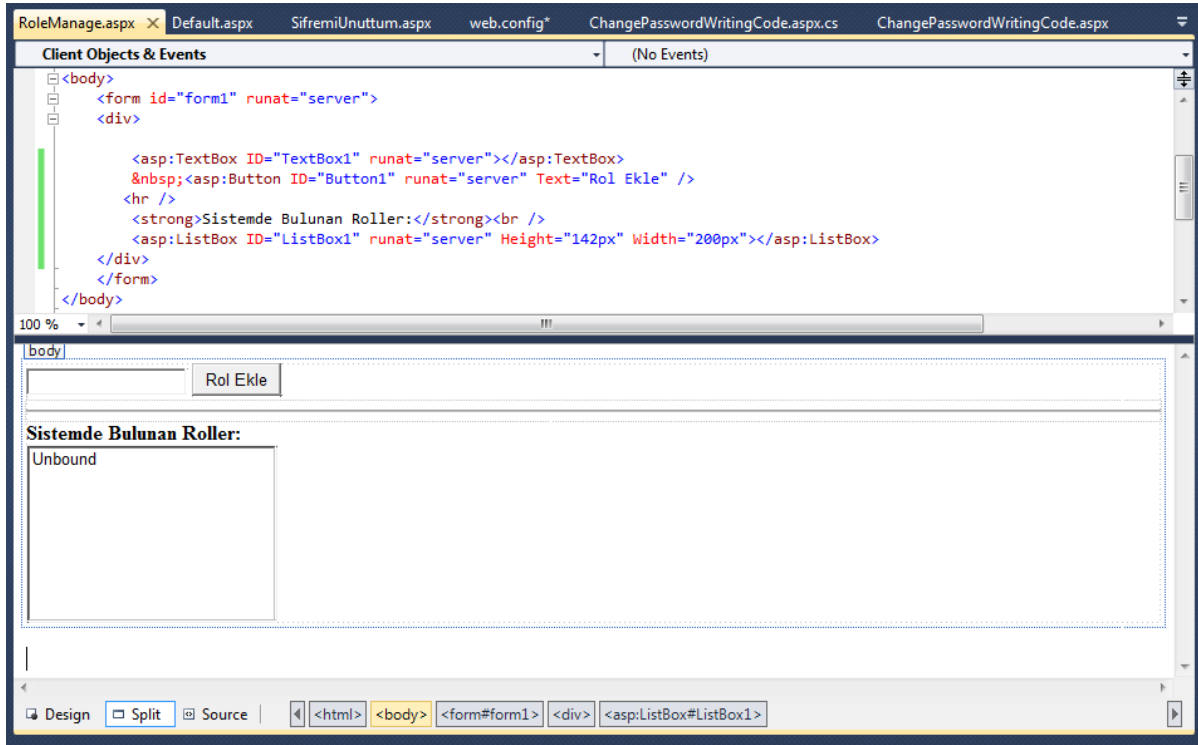
Sitede rol yönetim sistemini kullanmak için, ilk olarak sitede rollerin kullanımı aktif yapılmalıdır. Bu işlem için uygulama ayarlama dosyasında (web.config), aşağıdaki ayarları gerçekleştiriyoruz.

```
<system.web>
.....
<roleManager enabled="true" />
.....
</system.web>
```

Bu ayarlardan sonra web sitesi için rol yönetim servisi aktif hale gelecektir. Rol yönetim servisi için gerçekleştirilebilecek diğer ayarlar da aşağıdaki tabloda yer almaktadır.

Özellik	Açıklaması
enabled	Site için rol yönetim servisinin aktif olup olmayacağı belirtiliyor.
cacheRolesInCookie	Kullanıcının rollerinin istemcide bir cookie'de saklanıp saklanmayacağı belirtiliyor.
cookieName	İstemcide saklanacak olan cookie'nin adı belirtiliyor. Cookie'nin varsayılan adı .ASPXROLES olarak belirlenmiştir.
cookieTimeout	İstemciye atılan Cookie'nin dakika cinsinden zaman aşımını belirliyor.
cookieRequireSSL	Rol yönetimi bilgilerinin şifrlenerek gönderilmesini sağlıyor.
cookieSlidingExpiration	Bu özellik Cookie'nin her istekten sonra TimeOut özelliğinde belirtilen zaman kadar veya varsayılan olan 30 dakikada sona erdirilip erdirilmeyeceği belirtiliyor. Bu özellik varsayılan olarak True gelmektedir bu özellik False olarak ayarlandığında Cookie ilk istekten sonra belirtilen zamanda sona erdirilecektir.
createPersistentCookie	Cookie'nin zamanla sonlanıp sonlanmayacağı belirtiliyor. Varsayılan değer güvenlik nedenleri ile False'tur.
cookieProtection	Kullanıcının sistemine atılacak olan Cookie'nin korunma modu belirtiliyor. Burada mümkün olan toplam dört tane seçenek vardır. Bunlar: All: Cookie hem data doğrulama hem de şifreleme algoritmaları ile korunur. Varsayılan mod budur. None: Cookie'ye herhangi bir koruma algoritması kullanılmaz. Encryption: Cookie şifreleme algoritması ile korunur ama veri doğrulama aktif değildir. Validation: Veri doğrulama aktiftir ama şifreleme aktif değildir.
DefaultProvider	Rol yönetim servisi için varsayılan olarak kullanılacak olan sağlayıcı belirleniyor. Varsayılan değer AspNetSqlRoleProvider'dır.

Rol yönetimi için hazır kontroller olmadığından bahsetmiştik bu sebeple sistemde bulunacak rolleri tanımlamak için aşağıdaki sayfayı hazırlayalım.



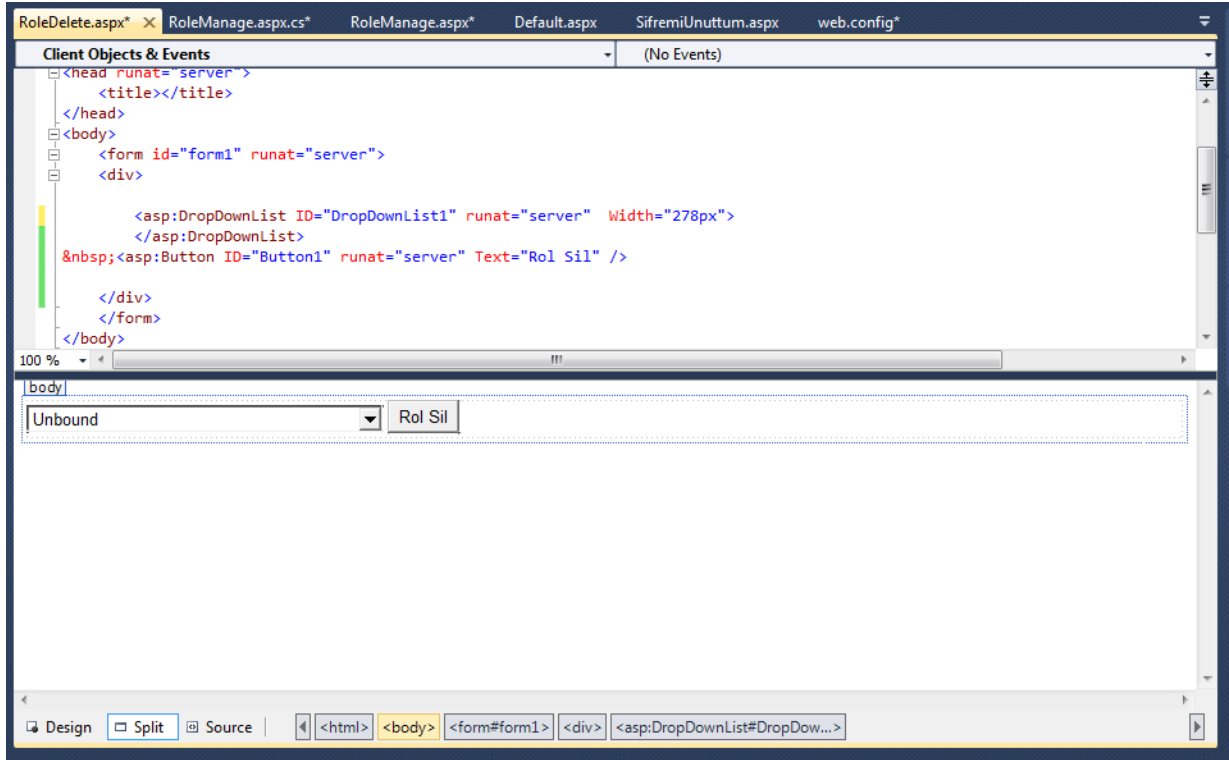
Yukarıda sayfa tasarımı görülen bu sayfa sisteme yeni rol eklemek için kullanılacaktır. TextBox'ın içerisinde belirtilen isimde bir rol, **Rol Ekle** butonuna tıklanıldığında sisteme eklenecektir. Hemen altta bulunan ListBox'ın içerisinde ise sistemde var olan roller görüntüleniyor olacaktır. Button'ın click olayına yazılacak olan kodlar aşağıdaki gibi olmalıdır.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Roles.CreateRole(TextBox1.Text); //Bu satır sisteme yeni rol ekleyecektir.
    ListBox1.DataSource = Roles.GetAllRoles(); //Bu satır ile sistemde bulunan
    roller döndürülerek ListBox'a bağlanacaktır.
    ListBox1.DataBind();
}
```

Yukarıdaki kodlar incelendiğinde, yeni bir Rol eklemek için **Roles** sınıfının **CreateRole** metodunun kullanıldığı görülecektir. Sistemde bulunan rolleri elde etmek için de, **Roles** sınıfının **GetAllRoles** metodu kullanılmaktadır. GetAllRoles metodu string türünden bir dizi döndürmektedir, dolayısı ile direkt DropDownList kontrolüne DataSource olarak gösterilebilmektedir. Yukarıdaki kodlar sisteme TextBox'ın Text özelliğinde belirtilen isimdeki rolü ekliyor ve ardından sayfada bulunan ListBox kontrolünde sistemde bulunan rolleri listeliyor. Aşağıdaki kodlar ise bu listeleme işlemini sayfanın Page_Load olayında gerçekleştiriliyor.

```
if (!Page.IsPostBack)
{
    ListBox1.DataSource = Roles.GetAllRoles();
    ListBox1.DataBind();
}
```

Bu işlemlerden sonra uygulamada kullanmak için yeni roller sisteme eklenebiliyor olacaktır. Eklene rollerin silinmesi de istenilebilecek bir durumdur. Bu durumu gerçeklemek için de bir sayfa tasarlayalım. Bu sayfaya bir DropDownList ve bir de Button kontrolü ekleyelim. Sayfada yapılacak olan işlem çok basittir. DropDownList'te seçili olan rol Button'a basılınca silinecek ve DropDownList'in içeriği yeniden doldurulacaktır. Bu işlemi gerçeklemek için tasarlanan sayfanın görünümü aşağıdaki gibidir.



Sayfaya veri getirmek için yazılacak olan kodlar aşağıda yer almaktadır.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownList1.DataSource = Roles.GetAllRoles();
        DropDownList1.DataBind();
    }
}
```

Button'a tıklanıldığı zaman işletilip belirtilen rolü silecek olan kodlar da aşağıdaki gibidir.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Roles.DeleteRole(DropDownList1.SelectedItem.ToString());
    DropDownList1.DataSource = Roles.GetAllRoles();
    DropDownList1.DataBind();
}
```

Görüldüğü üzere veri silme işlemleri **Roles** sınıfının **DeleteRole** metodu ile gerçekleştirilmektedir. DeleteRole metodunun iki tane aşırı yüklenmiş sürümü bulunmaktadır. Bunlardan ilki yukarıdaki örnekte kullanılan, içerisine string türünden bir değer ile silinecek olan rolün adını alan metot, diğeri ise silinecek olan rolün adı ile birlikte bool türünden bir değer alan bir metot. İkinci metodun ikinci parametresi şu amaçla kullanılmaktadır; Silinecek olan role kayıtlı üye olması durumunda, ortama hata mesajı fırlatılmasını sağlamak.

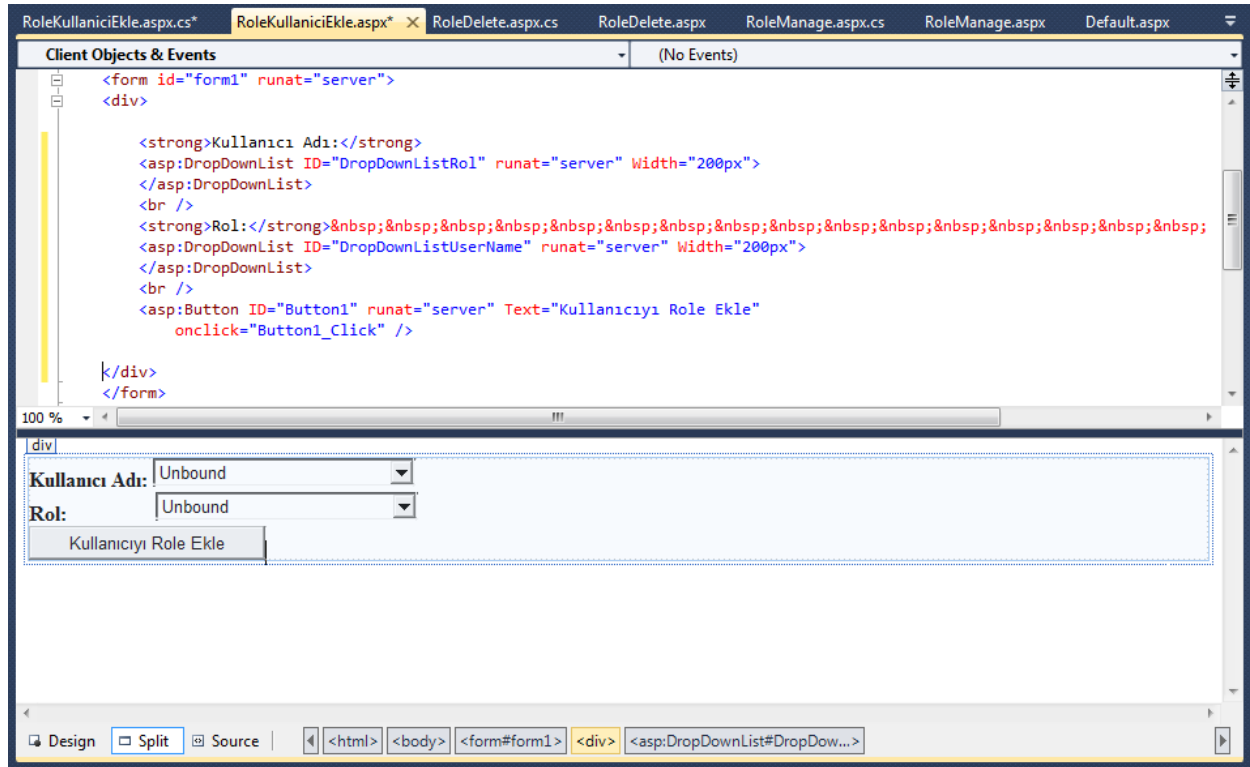
Şimdi projeyi çalıştırıp ilk olarak bir kaç tane rol ekleyelim. Ardından da az önce oluşturduğumuz sayfaya geçip rol silme tarafını test edelim. Herhangi bir hata varsa bu hataları giderip bir sonraki adımla ilerleyelim.

Kullanıcıları Bir Role Dahil Etmek

Yukarıdaki kodlar ile sisteme rol ekledikten sonra, sıra geldi bu role kullanıcı(ları) dahil etmeye. Roles sınıfında bu işlem için kullanılabilecek dört tane metod yer almaktadır. Bu metodlar açıklamaları ile birlikte aşağıdaki tabloda bulunmaktadır.

Metot	Açıklaması
AddUserToRole	Bir kullanıcıyı bir role dahil etmek için kullanılabilecek metod.
AddUserToRoles	Bir kullanıcıyı birden fazla role dahil etmek için kullanılabilecek metod.
AddUsersToRole	Birden fazla kullanıcıyı bir role dahil etmek için kullanılacak olan metod.
AddUsersToRoles	Birden fazla kullanıcıyı birden fazla role dahil etmek için kullanılacak olan metod.

Bu metodların tümü, kullanıcıları role dahil etmek için kullanılmaktadır. Birden fazla kullanıcı veya rol belirtilirken, metodlar parametre olarak string türünden bir dizi almaktadırlar. Bu durumu örneklemek için de yeni bir sayfa ekleyelim ve bu sayfa üzerinden bir kullanıcıyı bir role dahil edelim. Sayfada iki tane DropDownList kontrolü yer alsın. Bu kontrollerden birisi rolleri listelerken, diğeri User'ları listeliyor olsun. DropDownList'lerin altında yer alacak olan Button'da seçilen kullanıcıyı, seçilen role dahil etsin. Bahsedilen senaryoyu gerçeklemek için tasarlanan web sayfasının görünümü aşağıdaki gibi olmalıdır.



Button'a tıklanılınca çalıştırılacak olan kodlar yazılmadan önce DropDownList kontrollerini doldurmak gerekmektedir. Bu sebeple ilk olarak yeni eklenen sayfanın PageLoad olayına aşağıdaki kodlar eklenmelidir. Kodlar daha önceki örneklerden tanıdık geliyor değil mi!

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownListRol.DataSource = Roles.GetAllRoles();
        DropDownListRol.DataBind();
    }
}
```

```

        DropDownListUserName.DataSource = Membership.GetAllUsers();
        DropDownListUserName.DataBind();
    }
}

```



Membership sınıfının **GetAllUsers** metodu sistemde bulunan tüm kullanıcıları getirir.

Button'a tıklanıldığında işletilecek olan kod aşağıda görüntülenmektedir.

```

protected void Button1_Click(object sender, EventArgs e)
{
    Roles.AddUserToRole(DropDownListUserName.SelectedItem.ToString(),
        DropDownListRol.SelectedItem.ToString());
}

```

Rollere Dahil Edilen Kullanıcılar İle Çalışmak

Bir role dahil olan kullanıcıları elde etmek için yine **Role** sınıfı kullanılacaktır. Rollere dahil olan kullanıcıları elde etmek için, Role sınıfının **GetUsersInRole** ve **GetRolesForUser** metotları kullanılabilir. **GetUsersInRole** metodu parametre olarak rol adını kabul eder ve geriye bu role dahil olan kullanıcıların kullanıcı adlarını bir string dizisi halinde döndürür. **GetRolesForUser** metodu ise benzer şekilde içerisine parametre olarak kullanıcı adı alır ve belirtilen kullanıcı adının dahil olduğu rolleri bir string dizisi halinde döndürülür. **GetRolesForUser** metodunun iki adet aşırı yüklenmiş sürümü vardır. Bunlardan birisi parametre olarak kullanıcı adını kabul ederken, diğeri ise parametre almaz ve sisteme giriş yapmış olan mevcut kullanıcının dahil olduğu rolleri döndürür.

Aşağıdaki örnek incelendiğinde, bu iki metodun kullanımı daha iyi anlaşılabilir. Örneği gerçeklemek için projeye bir sayfa ekle ve sayfanın **PageLoad** olayına aşağıdaki kodları yazıp çalıştır.

```

protected void Page_Load(object sender, EventArgs e)
{
    string[] kullanicilar = Roles.GetUsersInRole("Admin");
    Response.Write("<b>Admin rolüne dahil olan kullanıcılar:</b> <br>");

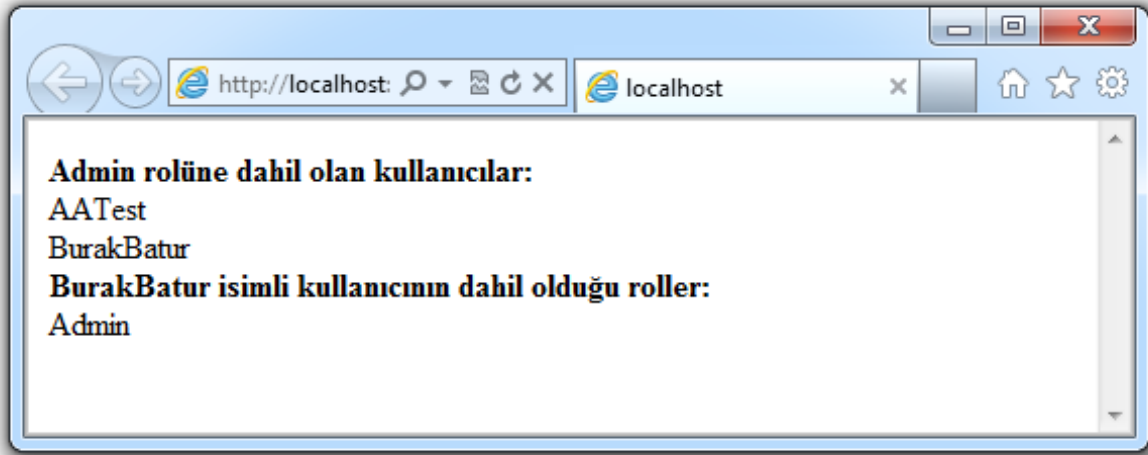
    foreach (string kullanici in kullanicilar)
    {
        Response.Write(kullanici + "<br>");
    }

    Response.Write("<b>BurakBatur isimli kullanıcının dahil olduğu roller:</b><br>");
    string[] roller = Roles.GetRolesForUser("BurakBatur");

    foreach (string rol in roller)
    {
        Response.Write(rol + "<br>");
    }
}

```

Kodlar incelendiğinde ilk olarak Admin isimli role dahil olan tüm kullanıcıların listelendiği ve ikinci olarak alttaki bölümde de BurakBatur isimli kullanıcının dahil olduğu tüm rollerin listelendiği görülmektedir. Kodlar çalıştırıldığında aşağıdaki sonuca ulaşılıyor olacaktır.



Bir kullanıcının role dahil olup olmadığını kontrol etmek

Roles sınıfının yazılım geliştiricilere sunduğu imkanlar sayesinde bir kullanıcının belirtilen role dahil olup olmadığı da Roles sınıfı kullanılarak belirlenebiliyor. Bu işlem sayesinde farklı rol gruplarından gelen farklı kullanıcılara göre kod tarafında işlem yaptırılabilir. Bir kullanıcının bir role dahil olup olmadığını belirten metod Roles sınıfı içerisinde yer alan **IsUserInRole** metodudur. Bu metodun toplam iki adet yapılandırıcı bulunmaktadır. Bunlardan ilki içerisine string türünden bir parametre ile rol adını alır ve o an oturum açmış olan kullanıcının belirtilen role dahil olup olmadığını bool türünden döndürür.

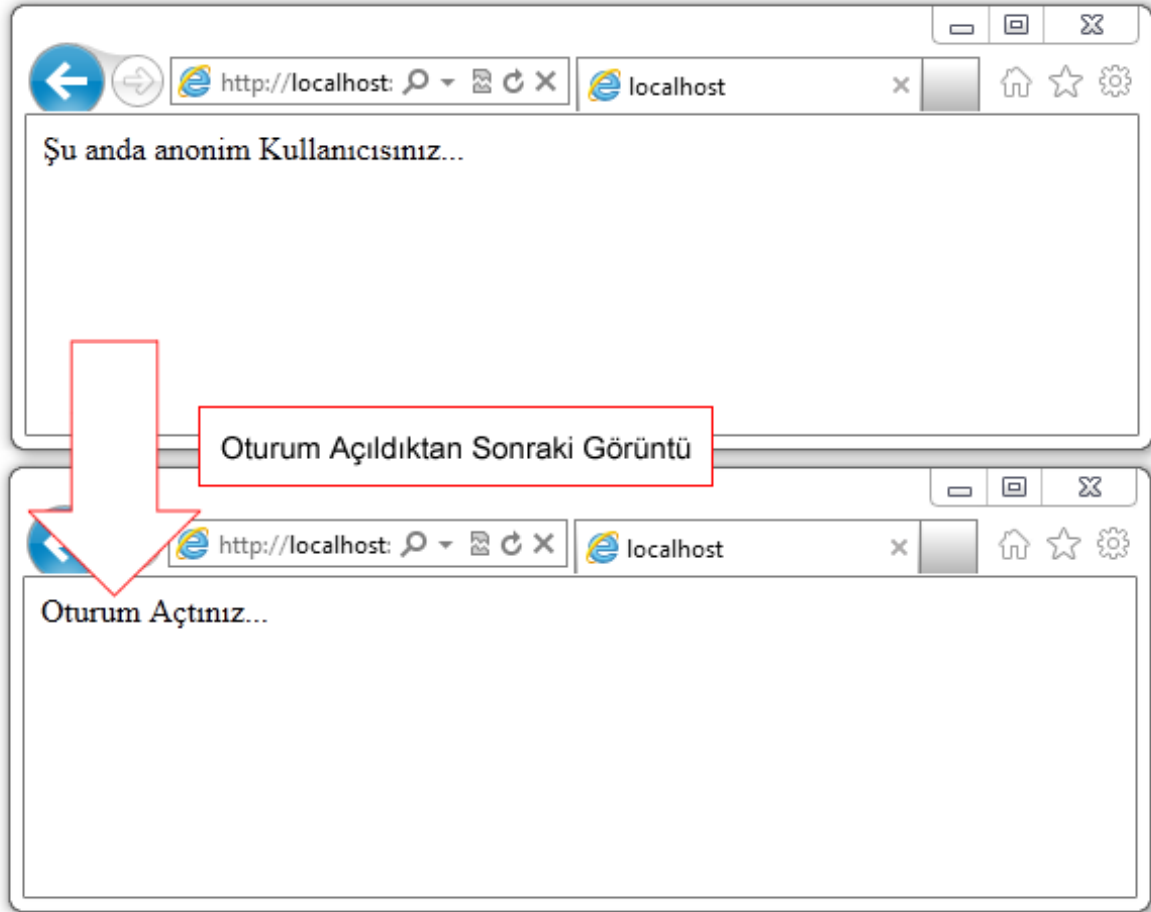
İkinci yapılandırıcı ise ilk parametre olarak string türünden kullanıcı adını alır, ikinci parametre olarak yine string türünden rol adını alır ve belirtilen kullanıcının belirtilen rolde olup olmadığını kontrol ederek bool türünden bir sonuç döndürür.

LoginView Kontrolü

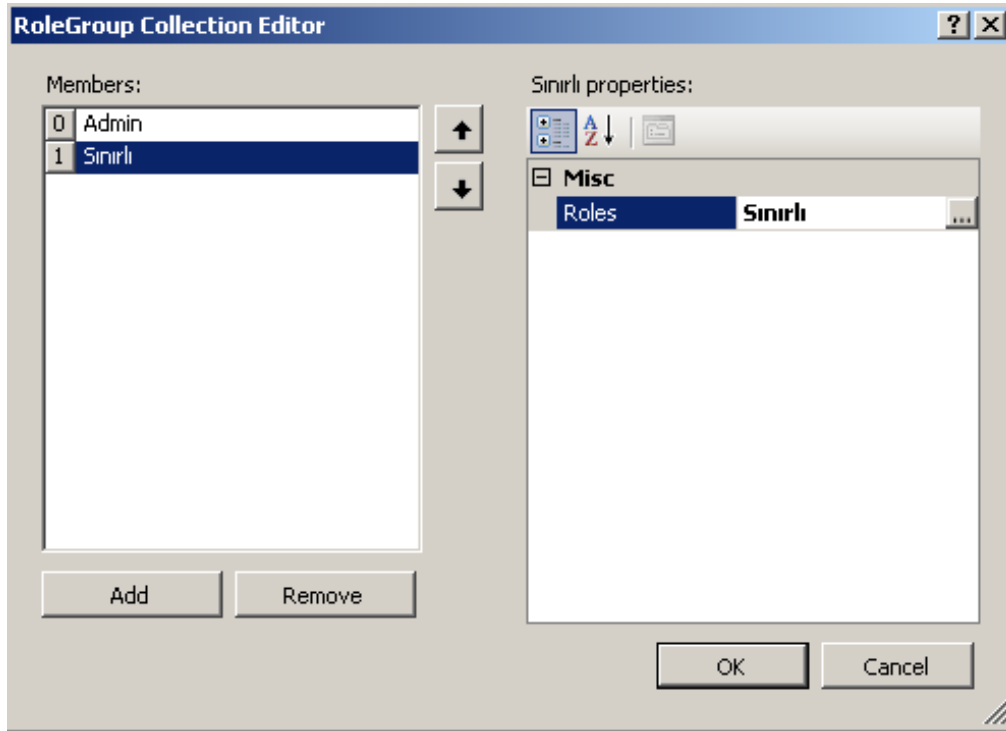
LoginView kontrolü Membership tarafında kullanıcılara, kullanıcının Oturum açmış olup olmamasına göre farklı görünümler sunulmasına olanak tanımaktadır. Ancak işin içine bir de roller girdiğinde bu kontrol her role göre farklı görünümler sunulmasına olanak tanıyabilmektedir. İlk olarak kontrolün özelliklerini daha sınırlı kullanarak sadece kullanıcının oturum durumunun gösteren bir örnekle ilerleyelim. Bu işlem için, sayfaya bir tane LoginView kontrolü sürüklenip bırakılır. Kontrolün Smart Tag'ine göz atıldığında, Views bölümünde listelenen görünüm modlarında, iki adet görünüm olduğu fark edilmelidir. Bunlardan biri Anonim kullanıcılara gösterilecek olan şablon ve diğeri de oturum açmış olan kullanıcılara gösterilecek olan şablondur. Varsayılan olarak Anonim kullanıcılara gösterilecek olan **AnonymousTemplate** alanına Kullanıcılara anonim bir kullanıcı olduğunu belirten bir mesaj ekleyelim. Daha sonra Smart Tag üzerinden **LoggedInTemplate** bölümüne geçilip bu alana da kullanıcının oturum açtığını belirten bir mesaj ekleyelim. Bu işlemlerden sonra sayfanın kaynak tarafına geçildiğinde LoginView kontrolünün kodları aşağıdaki gibi oluşturulmuş olacaktır.

```
<asp:LoginView ID="LoginView1" runat="server">
  <AnonymousTemplate>
    Şu anda anonim kullanıcısınız...
  </AnonymousTemplate>
  <LoggedInTemplate>
    Oturum Açtınız...
  </LoggedInTemplate>
</asp:LoginView>
```

Sayfa ilk çalıştırıldığında anonim kullanıcı olarak bu sayfanın çalıştırıldığı bilgisi verilecektir. Ancak, oturum açıldıktan sonra, LoginView kontrolünün bulunduğu sayfaya gelindiğinde, oturum açmış bir kullanıcı olduğu kullanıcıya bildiriliyor olacaktır.



LoginView kontrolünün farklı rol gruplarına göre de farklı görünümlere bürünebileceğinden söz etmiştik. Bu işlem için görüntülenmesi istenilen her rol grubu için bir tanımlama yapmak gerekiyor. Bu işlem için VisualStudio üzerinden LoginView kontrolünün özelliklerinden RoleGroups kullanılabilir. Bu alanı kullanmak için üzerinde üç nokta bulunan elips button tıklanılmalıdır. Bu işlemden sonra aşağıdaki resimdeki gibi görülen yeni bir pencere açılacak ve bu alanda da rol grupları LoginView kontrolüne bildirilebilecektir.



Bu pencere yardımı ile istenilen rol grupları belirlenip Ok tuşuna basıldıktan sonra LoginView kontrolünün Smart Tag'ine yeniden göz atıldığında belirtilen rol gruplarının da burada yer aldığı görülecektir. Tıpkı diğer şablonlar gibi bu alanlara da Visual Studio üzerinden içerik eklenebilir. Admin rol grubu için ilgili alana Oturum Açtınız Şu anda Admin yetkisine sahipsiniz, Sınırlı alanı için de Oturum açtınız şu anda Sınırlı yetkisine sahipsiniz yazılsın. Bu değişiklikler de yapıldıktan sonra LoginView kontrolünün kodları aşağıdaki hali alacaktır.

```
<asp:LoginView ID="LoginView1" runat="server">
  <AnonymousTemplate>
    Şu anda anonim kullanıcısınız...
  </AnonymousTemplate>

  <LoggedInTemplate>
    Oturum Açtınız...
  </LoggedInTemplate>

  <RoleGroups>

    <asp:RoleGroup Roles="Admin">
      <ContentTemplate>
        Oturum Açtınız: Şu anda Admin Yetkisine sahipsiniz...
      </ContentTemplate>
    </asp:RoleGroup>

    <asp:RoleGroup Roles="Sınırlı">
      <ContentTemplate>
        Oturum Açtınız: Şu anda Sınırlı yetkisine sahipsiniz...
      </ContentTemplate>
    </asp:RoleGroup>

  </RoleGroups>
</asp:LoginView>
```

Bu değişiklikten sonra LoginView kontrolü artık sadece kullanıcının oturum açıp açmadığına göre değil kullanıcının üyesi olduğu rol grubuna göre de farklı içerik sunabiliyor olacaktır.



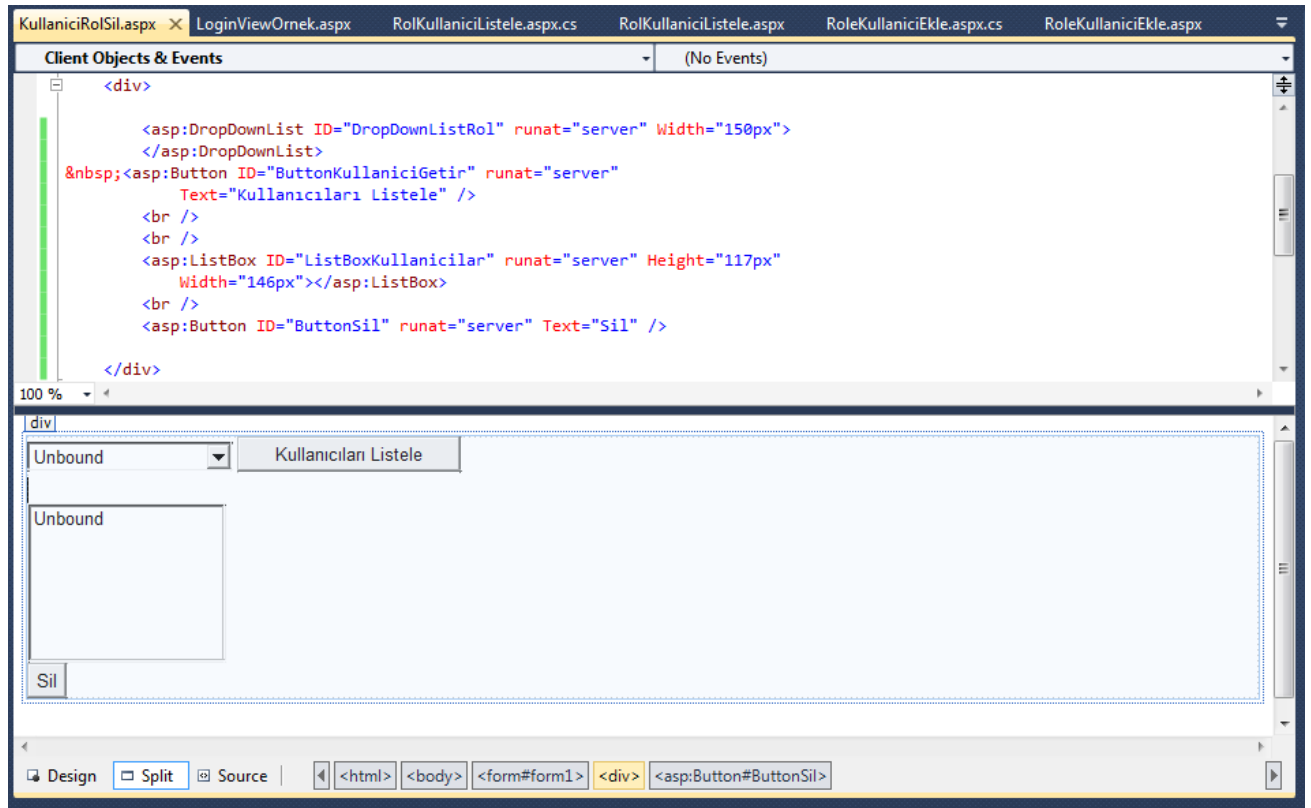
Yazılım geliştiriciler LoginView kontrolünün şablonlarının içine sadece text tabanlı veri eklemek ile sınırlı değillerdir. Eğer istenirse bu alanlara herhangi bir ASP.NET Server kontrolü eklenebileceği gibi herhangi bir içerik de eklenebilir. Gerçek hayatta bu kontrol farklı rol grubundan kişilere farklı menüler görüntülemek için ASP.NET Navigation kontrolleri ile ya da istenilen bir menü ile kullanılabilir.

Bir Rol'den Kullanıcı Silmek

Bir role kullanıcı eklenebildiği gibi tabii ki silinebilmektedir de. Bir rolden veya rollerden kullanıcı(lar) silmek için toplam dört adet metod mevcuttur. Aşağıdaki tabloda bu metodlar ve açıklamaları yer almaktadır.

Metot	Açıklaması
RemoveUserFromRole	Bir rolden bir kullanıcıyı silmek için kullanılacak olan metod.
RemoveUserFromRoles	Bir kullanıcıyı birden fazla rolden silmek için kullanılacak olan metod.
RemoveUsersFromRole	Birden fazla kullanıcıyı bir rolden silmek için kullanılacak olan metod.
RemoveUsersFromRoles	Birden fazla kullanıcıyı birden fazla rolden silmek için kullanılacak olan metod.

Bu metodlar incelendiğinde çoğul alanlar string türünden bir dizi isterken tekil alanlar ise string türünden bir değer kabul etmekte ve işlem yapmaktadır. Bir örnekle ilerleyip bu metodlardan bazılarını kullanalım. Projemize kullanıcıları bulundukları rolden silmek için kullanılacak olan bir sayfa ekleyelim. Sayfa ilk olarak rolleri listelemeli ardından da seçilen role dahil olan kullanıcıları listeliyor olmalı ve bir tane Sil Butonu yardımı ile Seçilen kullanıcıyı seçilen rolden silip formu yeniden yüklemeli. Bu işlem için sayfaya bir tane DropDownList, bir tane ListBox ve iki tane de Button kontrolü sürükleyip bırakalım. Kontroller, sayfada, aşağıdaki resimde görüldüğü gibi dizilmelidir.



Yapılması gereken işlemlerden ilki DropDownList kontrolünü sistemde bulunan tüm rollerde doldurmak olmalıdır. Bu işlem için sayfanın PageLoad olayına aşağıdaki kodlar eklenmelidir. Ancak bu kodlar sayfa her

PostBack olduğunda değil sayfa ilk defa çalıştığında çalışıyor olmalıdır bu yüzden kodlar if bloğu içerisinde yazılmaktadır.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownListRo1.DataSource = Roles.GetAllRoles();
        DropDownListRo1.DataBind();
    }
}
```

Bu kodlar önceki sayfalardan tanıdık geliyor olmalıdır. Roles sınıfının GetAllRoles metodu ile sistemde bulunan tüm roller string dizisi halinde elde edildi. Yapılacak olan ikinci işlem seçilen rol grubuna dahil olan kullanıcıları ListBox içerisinde listelemek olacaktır. Bu işlem içinde DropDownList kontrolünün yanında bulunan butonun Click olayına aşağıdaki kodlar eklenmelidir.

```
protected void ButtonKullaniciGetir_Click(object sender, EventArgs e)
{
    ListBoxKullanicilar.DataSource =
Roles.GetUsersInRole(DropDownListRo1.SelectedItem.ToString());
    ListBoxKullanicilar.DataBind();
}
```

Bu kodlar GetUserInRole metodu ile DropDownList'te seçili olan rolde bulunan tüm kullanıcıları ListBox kontrolünde listeleyecektir. Sıra geldi üçüncü ve son işleme. Üzerinde Sil yazan butona tıklanıldığında ise seçili olan rolden seçili olan kullanıcı silinip ListBox'ın içerisi boşaltılarak DropDownList kontrolün seçili değeri de ilk değer olarak belirlenecektir. Bu işler için yazılan kodlar da aşağıda görüntülenmektedir.

```
protected void ButtonSil_Click(object sender, EventArgs e)
{
    Roles.RemoveUserFromRole(ListBoxKullanicilar.SelectedItem.ToString(),
DropDownListRo1.SelectedItem.ToString());

    ListBoxKullanicilar.Items.Clear();
    DropDownListRo1.SelectedIndex = 0;
}
```

Kodlar incelendiğinde Roles sınıfının RemoveUserFromRole metodunun kullanıldığı görülecektir. Bu metot iki tane string türünde parametre alır. Bunlardan ilki kullanıcı adıdır ve ikincisi de rol adıdır. Bu metot belirtilen kullanıcıyı belirtilen rolden siler.

EĞİTİM :

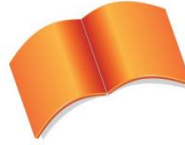
**GÜVENLİK VE ÜYELİK
YÖNETİMİ**

Bölüm :

Üyelik Yönetimi

Konu :


Web Site Admin Tool

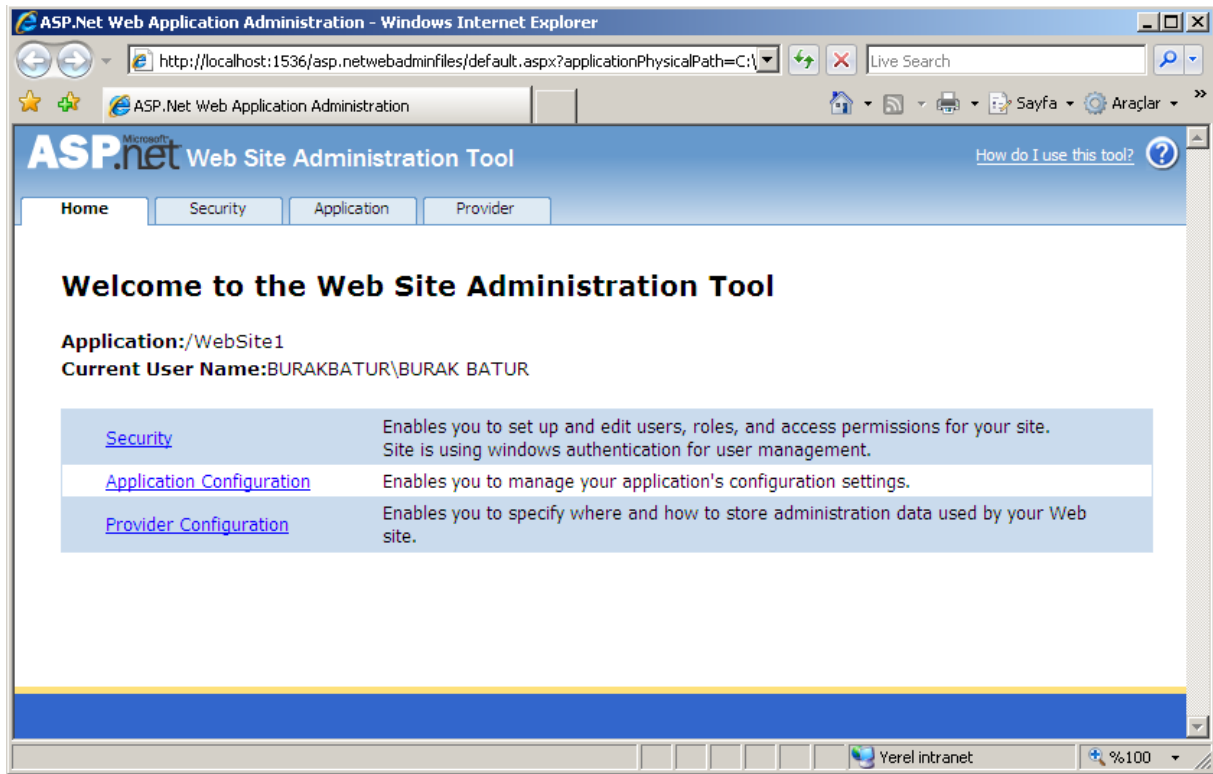


Microsoft Türkiye

Açık Akademi

ASP.NET Web Sitesi Yönetim Aracı (ASP.NET Web Site Administration Tool)

ASP.NET 2.0'la birlikte yazılım geliştiricilere Web Sitesi ayarlarının yapılabilmesi için **ASP.NET Web Site Administration Tool** adında bir araç sunulmuştur. Bu araç ile birlikte yazılım geliştiriciler; site için güvenlik ayarlarını (Security), uygulama ayarlarını (Application Configuration) ve uygulama verilerinin nerede saklanacağını (Provider Configuration) kolaylıkla belirleyebilmektedir. **ASP.NET Web Site Administration Tool** Visual Studio üzerinden çalıştırılıp kullanılabilir. Aracı kullanmak için Visual Studio'nun menülerinden **WebSite** altında yer alan **ASP.NET Configuration** seçilebileceği gibi, Solution Explorer'dan  simgesine de tıklanabilir. Araç açıldıktan sonra aşağıdaki resimde görülen görüntü ile karşılaşılacaktır.



ASP.NET Web Site Administration Tool, geliştirilen uygulama üzerinde ayarların yapılabilmesi için bir ASP.NET uygulamasıdır. Dolayısıyla, web tarayıcısı üzerinden çalışmaktadır. Araç açıldığında ASP.NET Development Server'da çalışacak ve bu yerel web sunucusu üzerinden hizmet veriyor olacaktır. Ana sayfa incelendiğinde, bu araç kullanılarak yapılabilecek olan ayarların tanımlanmasının yapılmış olduğu ve mevcut bölümlere yönlendirmelerin bulunduğu fark edilmektedir. Bu araç kullanılarak yapılabilecek olan işlemler şunlardır:

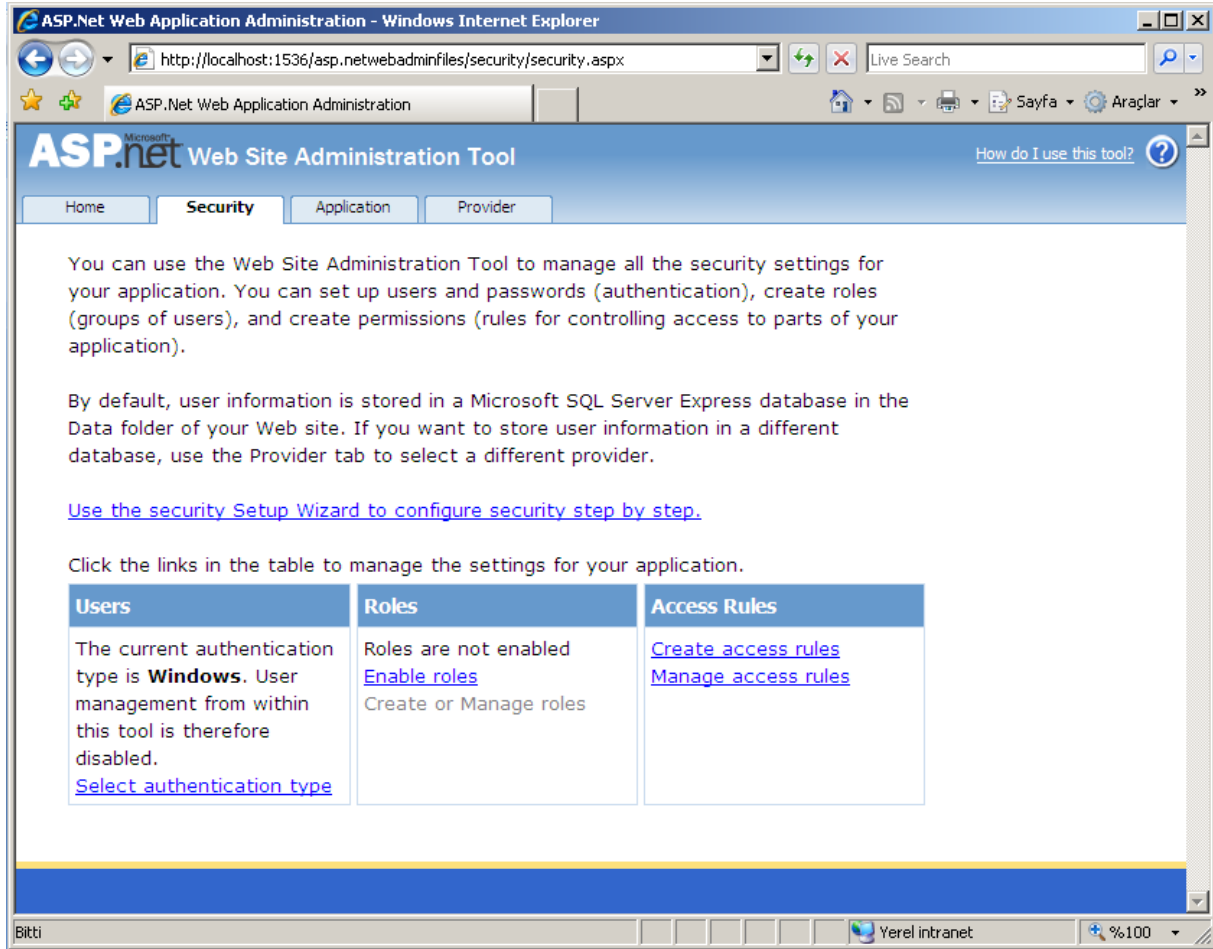
- **Security (Güvenlik):** Bu alandan sitede kullanılacak olan güvenlik tipi ayarlanabilmektedir ve ayrıca Form Tabanlı Güvenlik tercih edildiğinde kullanıcı yönetimi de yapılabiliyor olacaktır. Güvenlik bölümünde tahmin edileceği üzere rol ayarları da yapılabilmektedir.
- **Application Configuration (Uygulandırma Yapılandırması):** Bu alanda uygulamanın birkaç ayarı yapılandırılabilir. Bu alanda yapılan değişiklikler uygulama ayarlama dosyasına (web.config) yazılacaktır.
- **Provider Configuration (Sağlayıcı Yapılandırması):** Provider alanında Üyelik ve Rol yönetim servisleri gibi uygulama servislerinin kullanacak olduğu sağlayıcılar ayarlanabilmektedir.

Security (Güvenlik)

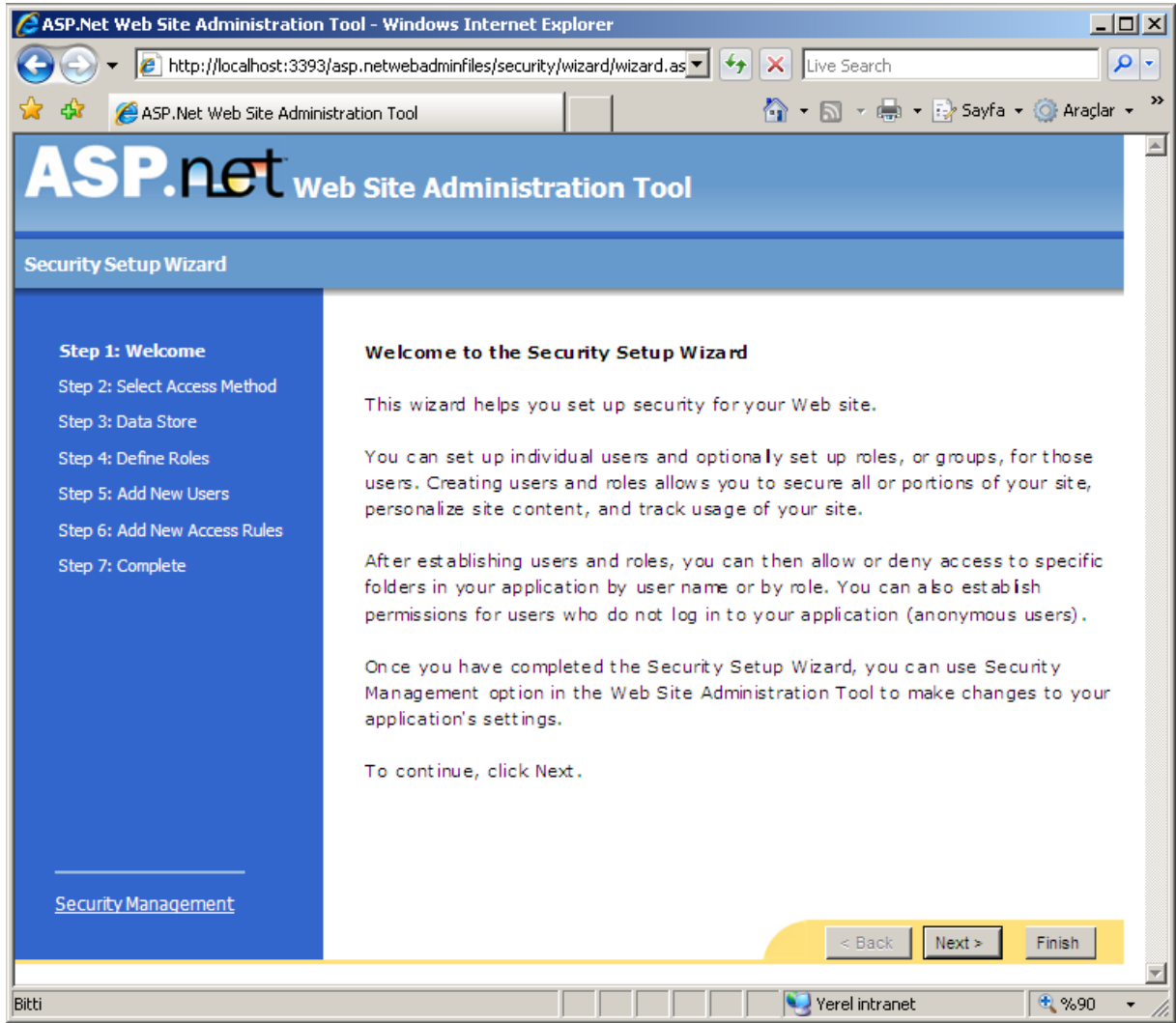
Security bölümü uygulamanın güvenlik ayarlarının yapılabileceği bölümdür. Bu bölüme geçildiğinde eğer daha önce oluşturulmadıysa APP_Data klasörünün içine uygulama servislerinde kullanılmak üzere **ASPNETDB.MDF** dosyası oluşturulacak ve bu dosyanın içerisinde de uygulamada kullanılmak üzere tablolar ve diğer nesneler oluşturulacaktır.



ASPNETDB.MDF dosyası uygulamanın geliştirilmiş olduğu makinede SQL Server Express kurulu ve çalışır durumda ise otomatik olarak oluşturulacaktır. Eğer SQL Server Express yoksa ya da servis çalışmıyorsa istenilen herhangi bir veritabanı hedef olarak gösterilebilir.



Security bölümünden ayarlar bir sihirbaz yolu ile yapılabileceği gibi, kendi özel bölümlerinden de yapılabilmektedir. Eğer **Use the security Setup Wizard to configure security step by step** linki tıklanırsa sitede kullanılacak olan güvenlik modunun seçiminden başlanarak bu kapsamdaki tüm ayarlar yapılandırılabilir olacaktır. Form tabanlı güvenlik seçildiğinde sihirbaz yardımıyla sisteme yeni kullanıcılar da eklenebilir. Sihirbazı adım adım anlatmak yerine her, bir ayar grubunu ayrı ayrı anlatmak daha uygun olacaktır.



Users (Kullanıcılar)

Bu alandan sistemde bulunan kullanıcıların yönetimi gerçekleştirilebilmektedir. ASP.NET projelerinde varsayılan güvenlik modu Windows Tabanlı Güvenlik olduğu için, araç ilk sefer açılıp bu alana gelindiğinde, kullanıcı yönetiminin Windows Tabanlı Güvenlik'te aktif olmadığını belirten bir uyarı ile karşılaşılacaktır. Ancak, uyarı ile birlikte güvenlik modunun da değiştirilebilmesini sağlayacak olan bir link de yazılım geliştiricileri bekliyor olacaktır. Bu linke tıklanıldığında, kullanıcıların uygulamaya nereden erişeceğini soran ve seçenek olarak da **From the Internet** ve **From a local network** şeklinde iki seçenek sunan bir sayfa ile karşılaşılacaktır. Bu sayfa yardımı ile sitede kullanılacak olan güvenlik tipi belirlenebilmektedir. From the Internet tahmin edileceği üzere Form Tabanlı Güvenlik, From a local network seçeneği ise Windows Tabanlı Güvenlik anlamına gelmektedir. Bu alandan From the Internet seçilerek kullanıcı yönetimi aktif hale getirilebilir.



Security alanından seçilen güvenlik tipi, uygulama ayarlama dosyasına (web.config) yazılacaktır.

Kullanıcı yönetimi aktif hale getirildikten sonra Users bölümünün görünümü değişecektir. Artık Users alanı kullanılarak sisteme kullanıcı eklenebilecek ve var olan kullanıcılar yönetilebilecektir.

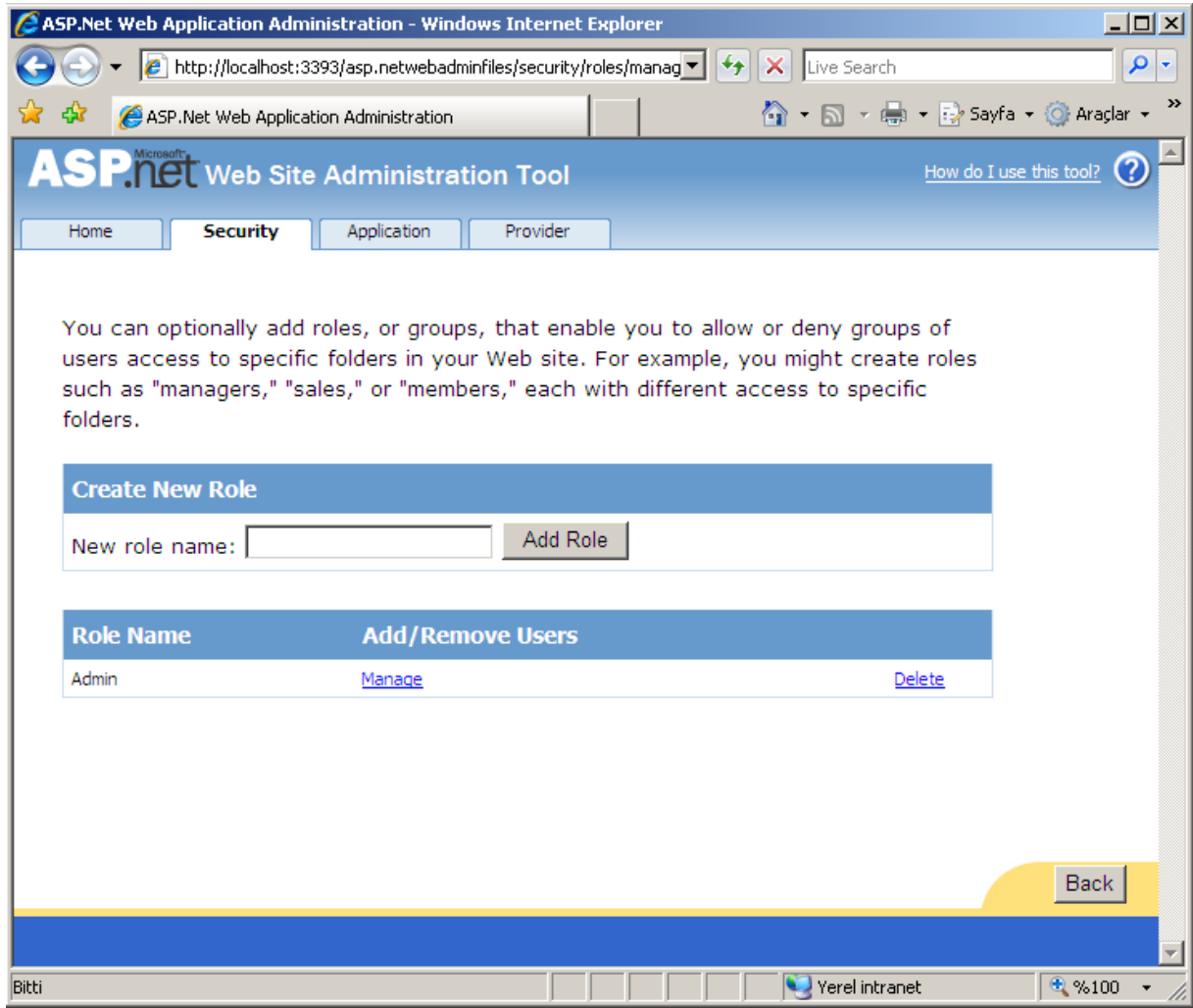


Users bölümünden eklenen kullanıcılar, Security alanına geçildiğinde oluşturulan ASPNETDB.MDF dosyasında SQL Server Express servisi kullanılarak depolanacaktır. Oluşturulan kullanıcılar Visual Studio ekranından ASPNETDB.MDF dosyası üzerine çift tıklanarak görüntülenebilir.

Roles (Roller)

Rol yönetimi varsayılan olarak aktif değildir ASP.NET Web Site Administration Tool kullanılarak rol yönetimi aktifleştirilebilir. Rol yönetimini aktifleştirmek için Roles bölümünden **Enable roles** kullanılmalıdır. Enable roles butonuna tıklandıktan sonra rol yönetimi tasarlanan site için aktif hale getirilmiş olacaktır. Bu işlemten sonra artık siteye yeni roller ve bu rollere kullanıcılar eklenebiliyor olacaktır. Siteye yeni rol eklemek veya sitede bulunan rolleri yönetmek için **Create or Manage roles** butonu kullanılacaktır. Bu butona tıklandıktan sonra yeni rol eklemek için bir sayfa ile karşılaşılacaktır. Bu sayfa yardımı ile sisteme yeni rol eklenebilir. Rol sisteme eklendikten sonra, rol eklenen bölümün hemen altında yer alan bölümde, eklenen rol görünür olacaktır.

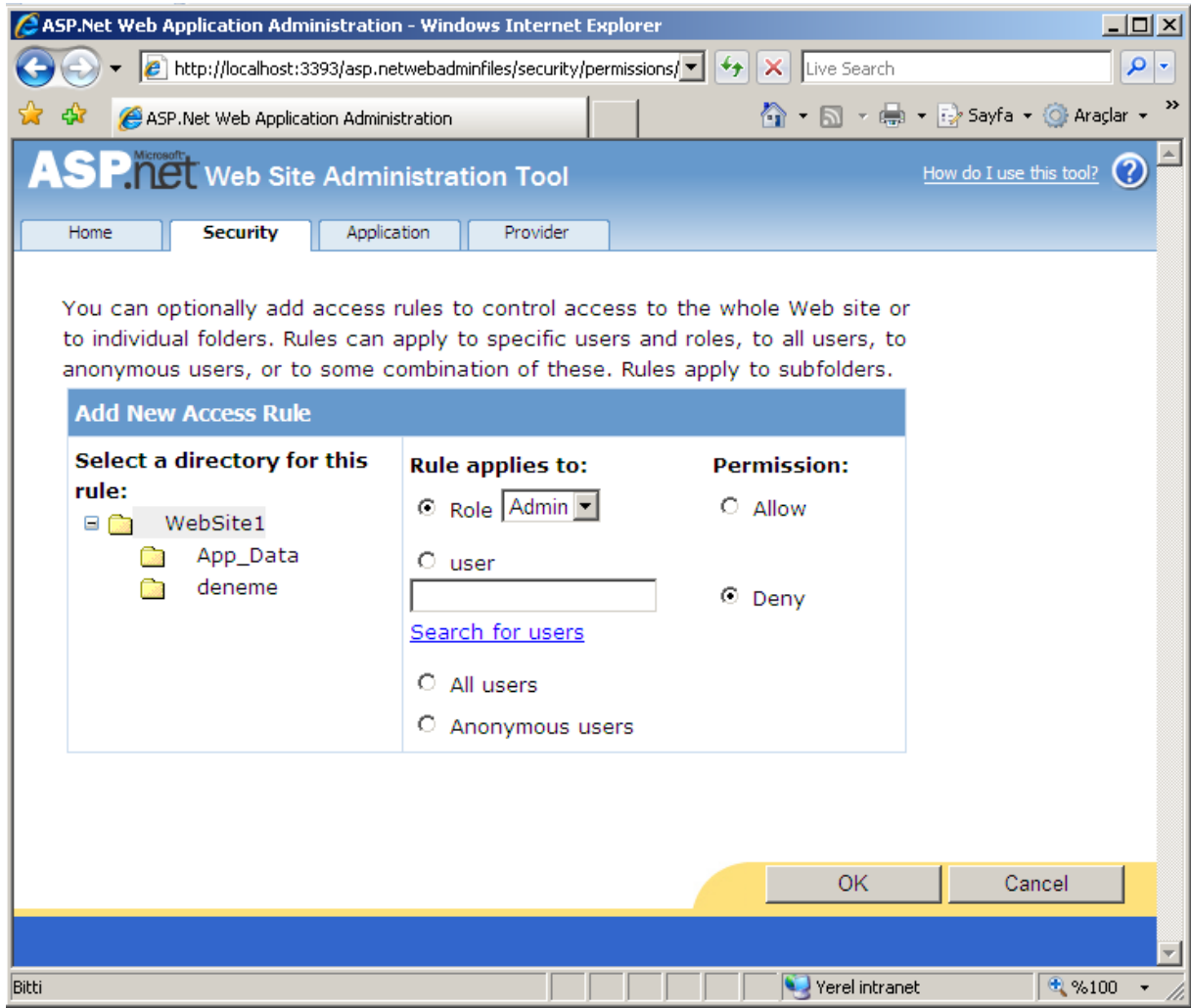
Listelenen rollerin yan tarafında, **Manage** ve **Delete** olarak görünen iki buton yer alacaktır. Bu iki buton yardımı ile herhangi bir rol'e kullanıcı dahil edilebilir veya Delete butonu ile rol sistemden silinebilir. Sistemde bulunan bir kullanıcı, seçili olan role dahil edilmek istenildiğinde, Manage butonuna tıklanır ve yönlendirilen sayfadan sistemdeki herhangi bir kullanıcı bulunarak seçili olan role dahil edilir.



Access Rules (Eriřim Kuralları)

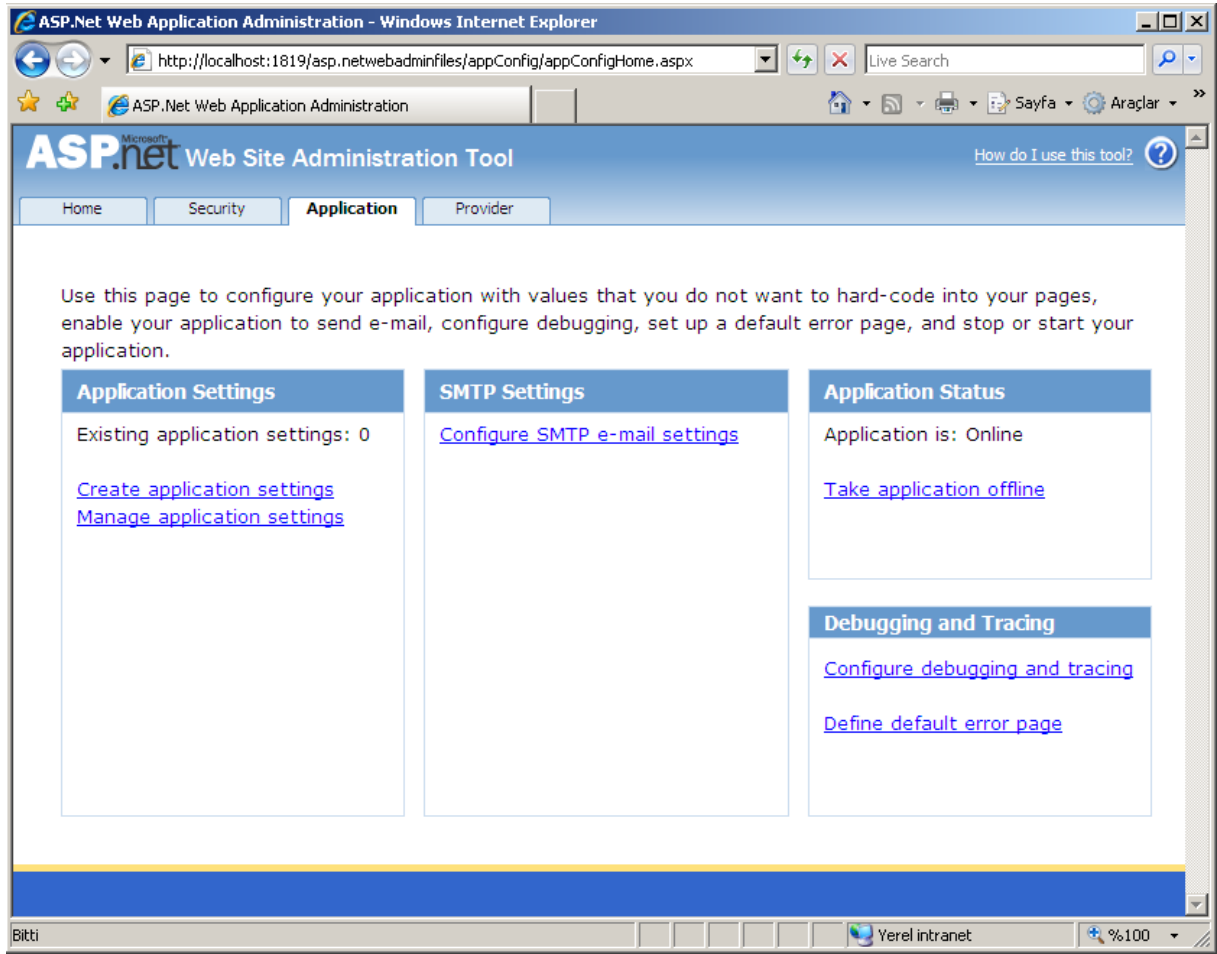
Bu alandan uygulamada bulunan klasörler için erişim kuralı verilebilmektedir. Yeni erişim kuralı belirlenebileceği gibi var olan bir erişim kuralı da güncellenebilmektedir. Bu alanda yapılan değişiklikler web.config dosyasına yazılacaktır.

Access Rules bölümü kullanılarak herhangi bir klasör için kullanıcı bazında bir takım ayarlamalar yapılabileceği gibi bir rol içinde ayarlama yapılabilmektedir. Bu alandan herhangi bir klasörün erişim kurallarının belirlenmesi için ilk olarak ilgili klasör seçilmelidir. Hemen ardından da erişim belirlemesi ayarı yapılacak olan kullanıcı veya rol grubu seçilip daha sonrada Permission bölümünden seçili olan kullanıcı veya rol grubuna izin mi verilecek yoksa seçili olan rol grubu veya kullanıcı yasaklanacak mı bunun belirlenmesi gerekiyor. Bu alandan özel bir kullanıcı veya grup seçilebileceği gibi, istenilirse tüm kullanıcılar veya oturum açmamış olan tüm kullanıcılar da erişim kuralı belirlemek için seçilebilmektedir.



Application Configuration (Uygulama Yapılandırması)

Application ayarları bölümünden uygulamanın genelini ilgilendiren bir takım ayarlar yapılabilmektedir. Bu bölüm kullanılarak Uygulama Ayarları (Application Settings), SMTP Ayarları (SMTP Settings), Uygulamanın Durumu (Application Status), Debug ve Trace Ayarları yapılabilmektedir. Bu bölümde yapılan ayarlar da tahmin edileceği üzere web.config dosyasına yazılacaktır.



Application Settings (Uygulama Ayarları)

Application Settings alanı kullanılarak uygulamada kullanılmak üzere yeni uygulama ayarları tanımlanabilir veya daha önceden tanımlanmış olan ayarlar buradan güncellenebilir. Bu alanda yapılan ayarlamalar web.config dosyasında <appsettings> düğümünde yer alıyor olacaktır.

SMTP Settings (SMTP Ayarları)

SMTP Settings ayarları altından site üzerinden mail göndermek için gerekli olan SMTP Server ayarları yapılabilmektedir. **Configure SMTP e-mail settings** bağlantısına tıklandıktan sonra SMTP ayarlarının yapılabilecek olduğu bölüm ile karşı karşıya kalınacaktır. Bu alan yardımı ile Mail gönderilirken kullanılacak olan sunucunun adı, mail gönderilirken kullanılacak olan port ve mail'in gönderen adresi gibi ayarlar ile birlikte mail gönderilirken kullanılacak olan sunucuya login olma ayarları da yapılabilmektedir. Sunucuya login olma işlemi üç farklı yol ile gerçekleştirilebilir. Bunlardan ilki login gerektirmeyen None modu, diğeri, sunucuda bulunan mail'in gönderilecek olduğu hesabın bilgileri ve sonucusu da Windows Tabanlı Güvenlik ile sunucuya login olma seçeneğidir.

Configure SMTP Settings

Server Name:

Server Port:

25

From:

Authentication:

☒ None

☐ Basic
Choose this option if your e-mail server requires you to explicitly pass a user name and password when sending an e-mail message.

Sender's user name:

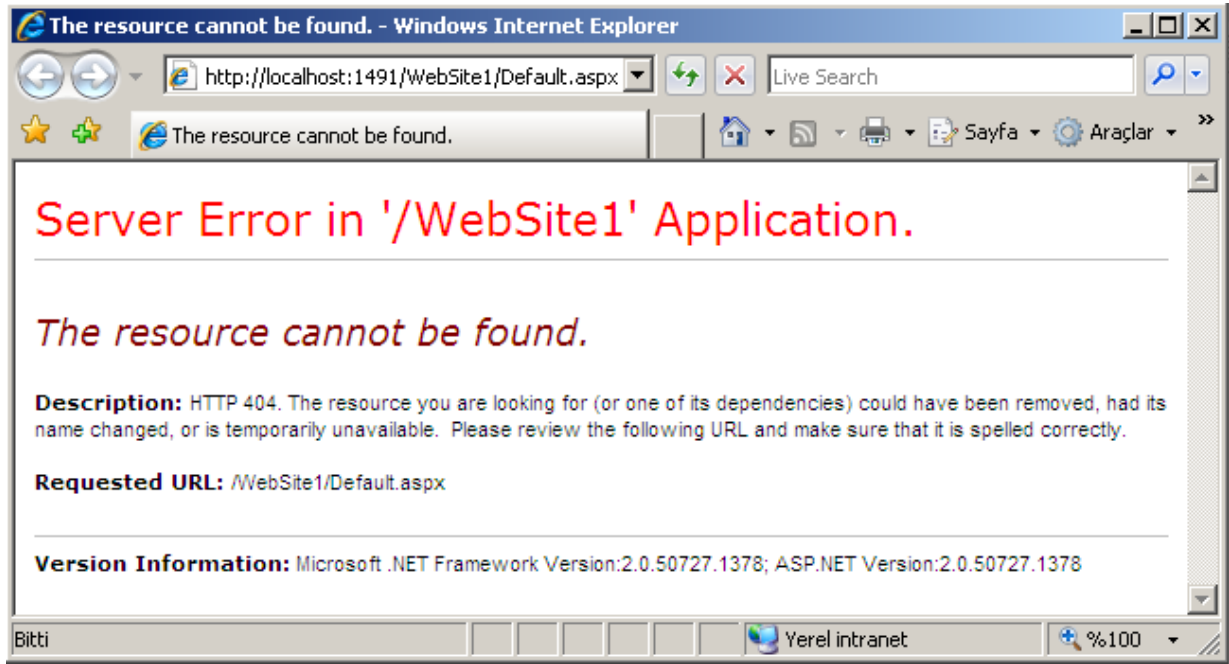
Sender's password:

☐ NTLM (Windows authentication)
Choose this option if your e-mail server is on a local area network and you connect to it using windows credentials.

Save

Application Status (Uygulama Durumu)

Application Status alanı ile uygulamanın durumu belirlenebilmektedir. Bu alandaki uygulamanın durumu sözcüğü ile söylenilmek istenilen şey, uygulamanın Online'mı yoksa Offline'mı olacağıdır. Yani bu alan kullanılarak, uygulama tek tık ile aktif ya da pasif yapılabilir. Uygulama pasif yapıldığında erişilmeye çalışıldığı zaman, aşağıdaki hata mesajı ile karşılaşılacaktır.



Debugging and Tracing (Hata Ayıklama ve İzleme)

Uygulamanın çalışması yakından incelenmek istendiğinde gerekli ayarlamaları yapmak için bu alan ziyaret edilmelidir. Bu alan kullanılarak uygulamada Debug ve Tracing aktif hale getirilip gerekli ayarları yapıldıktan sonra uygulama hata yakalamak için izlenebilir. Bu alan kullanılarak istenilirse hatalar için özel bir hata sayfası da belirlenip kullanıcılara daha hoş hata mesajları gösterilebilmektedir.

Provider Configuration (Sağlayıcı Yapılandırması)

Bu bölüm yardımı ile sitede kullanılacak olan uygulama servisleri için sağlayıcı (Provider) seçimi yapılabilmektedir. Bu bölüm kullanılarak tüm ASP.NET servisleri için tek bir sağlayıcı seçilebileceği gibi her servis için ayrı ayrı servis de seçilebilmektedir. Arayüz yardımıyla seçilen servislerin test edilebilmesi de mümkün olduğu için, her servis ayarlamasından sonra uygulamayı çalıştırıp test etmeye gerek kalmadan direkt arayüz üzerinden test yapılabilmekte ve dolayısıyla zamandan tasarruf edilebilmektedir.