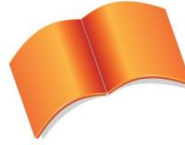


EĞİTİM :

**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

**Bölüm :
Doğrulama Kontrolleri**

**Konu :
Doğrulama Kontrolleri Ne İşe
Yarar?**



Microsoft Türkiye
Açık Akademi

Kullanıcıdan veri alınmasını gerektiren durumlarda, kullanıcıdan alınan verilerin doğruluğunu denetlemek için doğrulama kontrolleri kullanılabilir. Doğrulama kontrolleri, istemcide çalışan script tabanlı kontrollerdir. Herhangi bir hata olduğunda, sayfanın sunucuya gitmesini engelleyerek, istemcide bir hata mesajı göstererek durumu kullanıcıya bildirebilirler.

EĞİTİM :

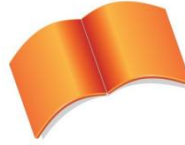
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

Doğrulama Kontrolleri

Konu :

Required Field Validator



Microsoft Türkiye

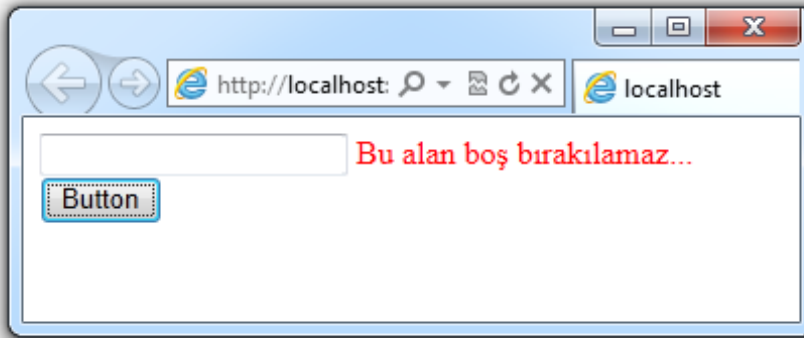
Açık Akademi

Kullanıcıdan veri istenildiği durumda, kullanıcının belirtilen alanı boş geçmeden mutlaka doldurmasını sağlayan doğrulama kontrolüdür. Bu kontrol, belirtilen alana veri girilmediği zaman hata üreterek sayfanın sunucuya gitmesini engeller ve belirtilen hata mesajını kullanıcıya gösterir. RequiredFieldValidator ile kontrol edilecek olan sunucu kontrolü, **ControlToValidate** özelliği ile belirlenmektedir.

ControlToValidate özelliğine, kontrol edilecek olan sunucu kontrolünün ID'si belirtilerek; kullanıcı tarafından belirtilen sunucu kontrolüne mutlaka veri girilmesi sağlanmış olur. Belirtilen kontrolün boş geçilmesi durumunda, kullanıcıya gösterilecek olan hata mesajı da **ErrorMessage** özelliği ile ayarlanmaktadır. RequiredFieldValidator kontrolünün kullanımını örneklemek için Visual Studio ortamında yeni bir tane sayfa oluşturup; sayfanın içerisine bir TextBox, bir tane RequiredFieldValidator ve bir tane de Button kontrolü eklenip, kontrollerin özelliklerini ekranda görüldüğü gibi ayarlayalım.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>  
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"  
ControlToValidate="TextBox1" ErrorMessage="Bu alan boş bırakılamaz...">  
</asp:RequiredFieldValidator>  
<br />  
<asp:Button ID="Button1" runat="server" Text="Button" />
```

Tasarlanan sayfa çalıştırılıp TextBox'a veri girilmeden Button kontrolüne tıklandığında; RequiredFieldValidator kontrolü sayfanın sunucuya gönderilip işlem yapılmasına izin vermeyecek ve sayfanın görünümü aşağıdaki resimde görüldüğü gibi olacaktır.



EĞİTİM :

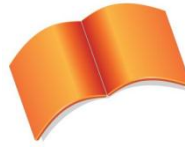
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

Doğrulama Kontrolleri

Konu :

RangeValidator



Microsoft Türkiye

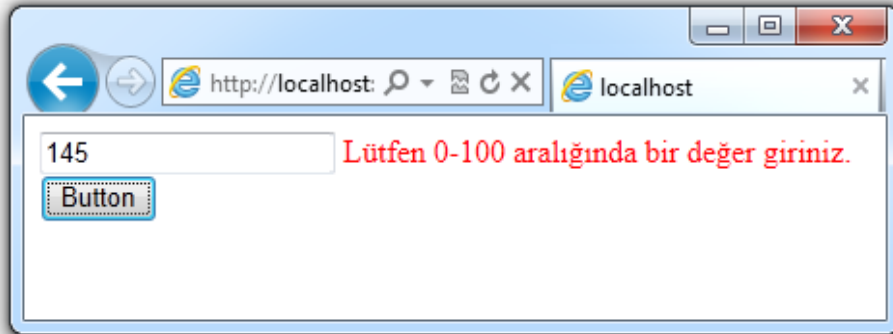
Açık Akademi

Kullanıcının girmiş olduğu değerin belirtilen aralıkta olması isteniyorsa, RangeValidator kontrolü kullanılabilir. Kontrol, belirtilen alandaki değerin istenilen aralıkta olmasını sağlamaktadır. Kontrolü kullanmak için tahmin edeceğin üzere bir tane minimum değer ve bir tane de maksimum değer belirtilmelidir.

Bu değerler, isimlerinden de anlaşılacağı üzere **MaximumValue** ve **MinimumValue** özellikleri ile belirtilmektedir. Karşılaştırılacak olan değerin, hangi veri tipinde olacağı da kontrole bildirilmelidir. Kontrol String, Integer, Double, Date ve Currency veri tiplerinden biri ile karşılaştırma yapabilmektedir. Kontrolün hangi veri tipine göre karşılaştırma yapacağını belirtileceği özellik ise **Type** özelliğidir.

Kontrolün kullanım alanını örneklemek için bir öğrencinin 1. vize notlarının sisteme girildiği bir web uygulaması tasarlandığını düşünelim. Vize notları 0 ile 100 arasında olan değerlerdir ve kullanıcının bu aralık dışında bir değer girmemesi gerekmektedir. Bu düşünce ışığında tasarlanan web sayfasının kodları aşağıda yer almaktadır. Kodlar incelendiğinde MinimumValue özelliğinin "0", MaximumValue özelliğinin "100" ve Type özelliğinin ise "Double" olarak ayarlandığı görülecektir.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="TextBox1"
ErrorMessage="Lütfen 0-100 aralığında bir değer giriniz."
MaximumValue="100"
MinimumValue="0"
Type="Integer">
</asp:RangeValidator>
<br />
<asp:Button ID="Button1" runat="server" Text="Button" />
```



EĞİTİM :

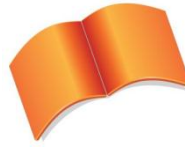
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

Doğrulama Kontrolleri

Konu :

RegularExpressionValidator



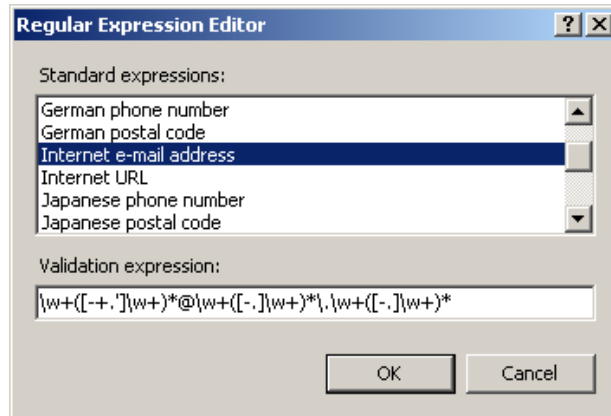
Microsoft Türkiye

Açık Akademi

RegularExpressionValidator

Kullanıcı tarafından girilen değerlerin belli bir söz dizimine uygun olup olmadığını kontrol eden kontroldür. Girilen değer, kontrolün **ValidationExpression** özelliğinde belirtilen söz dizimi yapısına uygun değil ise kontrol hata üretir ve sayfanın sunucuya taşınması engellenmiş olur. Kontrolün kullanım alanı oldukça geniştir, örneğin kullanıcıdan e-mail adresi, web sitesi bilgileri, telefon numaraları, vb. gibi bilgiler alınırken bu kontrol kullanılabilir. Visual Studio, yazılım geliştiricilere bu kontrol ile birlikte kullanılacak hazır kurallar sunmaktadır ancak Visual Studio ile sunulan dilbilgisi kuralları çok fazla değildir. Yazılım geliştiriciler **RegularExpression** denilen bu düzenli ifadeleri, ihtiyaçlarına göre kendileri yazıp RegularExpressionValidator kontrolü ile birlikte kullanabilirler.

Web sitesine gelen ziyaretçilerden e-mail adreslerinin alındığı bir senaryo düşünüldüğünde RegularExpressionValidator kontrolünün kullanımı oldukça mantıklı olacaktır. Bu senaryoyu gerçeklemek için bir sayfaya, bir TextBox, bir tane RegularExpressionValidator ve bir tane de Button kontrolü ekleyelim. RegularExpressionValidator kontrolünün ControlToValidate özelliği TextBox1 ve ErrorMessage özelliği de kullanıcıya geçerli e-mail adresi girmesi gerektiğini belirtecek şekilde uygun olarak ayarlayalım. ValidationExpression özelliğini ayarlamak için ise Visual Studio'nun nimetlerinden faydalanarak kontrolün özellikler penceresinden **ValidationExpression** özelliğinin yanındaki elips buton ile açılan diyalogdan en uygun olan ifadeyi seçelim. Bu alanda seçilen ifade düşünülen senaryoya uygunluk açısından **"Internet e-mail address"** olmalıdır.



Gerekli ayarlamalar yapıldıktan sonra sayfanın kodları aşağıdaki gibi olmaktadır. Sayfa çalıştırıldığında TextBox'taki değer de @ işareti yoksa ya da girilen değer standart bir e-mail adresinin söz dizimine uygun değilse hata üretilecektir.



RegularExpressionValidator kontrolü girilen değerlerin belirtilen söz dizimine uygun olup olmadığını kontrol eder. Örnek senaryoda kullanılan e-mail adresi ifadesinde gerçekte girilen değerdeki gibi bir e-mail adresinin kayıtlı olup olmadığı kontrolü yapılmaz, sadece söz dizimi kontrolü yapılarak girilen değerlerin en azından bir e-mail adresi olabilecek bir değer olması sağlanmış olur.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ErrorMessage="Lütfen geçerli bir e-mail adresi giriniz."
ControlToValidate="TextBox1"
ValidationExpression="\w+([-+.' ]\w+)*@\w+([-.\ ]\w+)*\.\w+([-.\ ]\w+)*">
</asp:RegularExpressionValidator>
<br />
<asp:Button ID="Button1" runat="server" Text="Button" />
```


EĞİTİM :

**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

**Bölüm :
Doğrulama Kontrolleri**

**Konu :
CompareValidator**



CompareValidator kontrolü adından da anlaşılacağı üzere kullanıcının girmiş olduğu değerlerin başka bir değerle karşılaştırılmasını yapan kontroldür. Kontrol ile kullanıcının girmiş olduğu değer, başka bir alandaki değer ya da belirtilen sabit bir değer ile karşılaştırılabilmektedir. Girilen değerın karşılaştırılacağı alan ControlToValidate özelliği ile kontrole belirtilmektedir. Belirtilen alana girilen değer; eğer başka bir alanın içindeki değer ile karşılaştırılacaksa **ControlToCompare** özelliği ile bu alan belirtilir. Eğer, sabit bir değer ile karşılaştırılacaksa **ValueToCompare** özelliği ile bu değer ayarlanır.

```
<asp:TextBox ID="TbParola" runat="server" TextMode="Password"></asp:TextBox>  
<br />  
Parola Doğrula:  
<asp:TextBox ID="TbParolaDogrula" runat="server" TextMode="Password"></asp:TextBox>  
<asp:CompareValidator ID="Comparevalidator1" runat="server"  
ControlToCompare="TbParola"  
ControlToValidate="TbParolaDogrula"  
ErrorMessage="Parola Alanları Uyuşmuyor!">  
</asp:CompareValidator>  
<br />  
<asp:Button ID="Button1" runat="server" Text="Button" />
```

EĞİTİM :

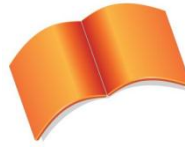
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

Doğrulama Kontrolleri

Konu :

CustomValidator



Microsoft Türkiye

Açık Akademi

CustomValidator

Kullanıcı tarafından girilen deęerleri kontrol etmek için var olan doęrulama kontrollerinin ihtiyaca cevap vermedięi durumlarda, yazılım geliřtiricilerin kendi yazdıkları istemci taraflı doęrulama fonksiyonların kullanımı için kullanılan bir kontroldür.

EĞİTİM :

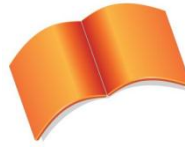
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

Doğrulama Kontrolleri

Konu :

Validation Summary



Microsoft Türkiye

Açık Akademi

ValidationSummary

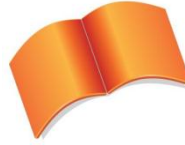
Doğrulama kontrollerinin üretmiş olduğu hata mesajları tek bir yerde toplu halde gösterilmek isteniyorsa ValidationSummary kontrolü kullanılabilir. Kontrol, sayfaya sürükleyip bırakılarak başka ayar yapmadan direkt olarak kullanılabilir. Bir hata oluştuğu anda, hata üreten doğrulama kontrolünün hata mesajı ValidationSummary içerisinde görüntüleniyor olacaktır. Ancak, herhangi bir ayarlama yapılmadı ise hata mesajı hem ValidationSummary hem de doğrulama kontrolünün bulunduğu alanda görüntülenecektir. Bu durum kullanıcı deneyimi açısından pek de hoş bir durum değildir. Aynı hata mesajını iki yerde görüntülemek iyi bir tasarım yapmak adına pek de olumlu bir davranış değildir. Daha iyi bir tasarım için doğrulama kontrollerinin **Text** özelliği kullanılarak hata oluşması durumunda hata oluşan yerde bu Text bilgisinin, hata mesajının da ValidationSummary içinde görüntülenmesi sağlanabilir. Text özelliğine * işareti gibi tek karakterli bir değer atanarak sadece hatanın o alanda oluştuğunu gösteriyor olmak, oldukça hoş görünümlü ve kullanıcıları memnun edecek bir tasarım modelidir.

EĞİTİM :

**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

**Bölüm :
Doğrulama Kontrolleri**

**Konu :
Doğrulama Kontrollerinin Bir
Arada Kullanımı**



Microsoft Türkiye

Açık Akademi

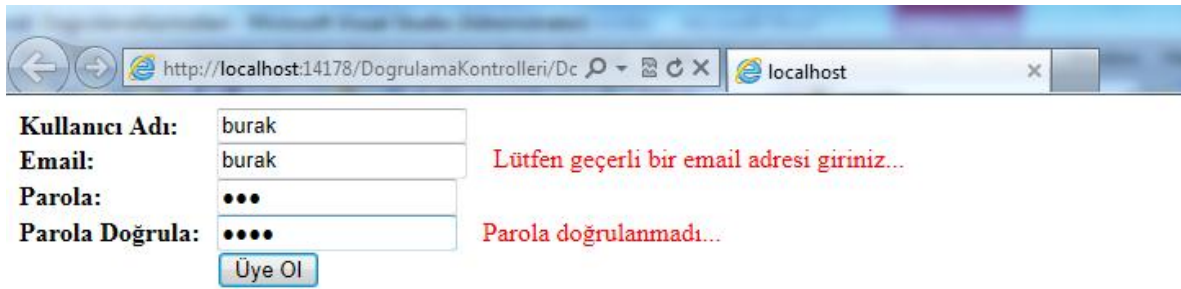
Doğrulama Kontrollerinin Bir Arada Kullanımı

Projelerde doğrulama kontrolleri genellikle bir arada kullanılır, hatta birden fazla doğrulama kontrolünün aynı alanı kontrol etmesi gibi durumlarda genellikle söz konusudur. RegularExpressionValidator kontrolü belirtilen alanın bir formata uygun olup olmadığını kontrol ederken alana veri girilip girilmediği kontrolünü yapmaz. Alanın boş olup olmadığının kontrolü ise RequiredFieldValidator kontrolü ile sağlanmalıdır. Doğrulama kontrollerinin bir arada kullanımını örneklemek için kullanıcıdan bilgi istenilen bir formun tasarlandığı bir senaryo düşünelim ve bu kapsamda aşağıdaki resimde görülen form tasarımını yapalım.

Form tasarlandıktan sonra sıra geldi bu forma doğrulama kontrollerinin eklenmesine; İlk olarak formdaki tüm alanların doldurulmasını sağlamak adına her TextBox'ı kontrol etmek için toplam dört tane RequiredFieldValidator kontrolüne ihtiyaç vardır. Her TextBox'ın yanına bir tane RequiredFieldValidator ekleyip gerekli ayarlar yapıldıktan sonra sayfanın tasarımı aşağıdaki gibi olacaktır.

RequiredFieldValidator kontrolleri eklendikten sonra sayfa çalıştırıldığında boş geçilen bir alan olduğunda sayfanın sunucuya gönderilmediği görülecektir. Düşünülen senaryo gereği birkaç tane daha doğrulama kontrollerine ihtiyaç vardır. Bunlardan ilki kullanıcının geçerli bir e-mail adresi girmesini sağlayacak olan RegularExperssionValidator kontrolüdür. RegularExpressionValidator kontrolü e-mail alanının yanında bulunan RequiredFieldValidator kontrolünün hemen yanına eklendikten sonra gerekli ayarlamalar yapıp sayfa çalıştırıldığında e-mail alanının boş geçilmemesi ve geçerli bir e-mail adresi girilmesi sağlanmış oldu. Ancak, sayfa çalıştırılıp hatalı bir e-mail girildiğinde hata mesajının yeri aşağıdaki resimde görüldüğü gibi olmaktadır.

Resimden de görüldüğü gibi e-mail adresinin hata mesajı kontrolün bulunduğu yerde görüldü. Bir önceki adımda eklenen RequiredFieldValidator hata mesajı üretmese bile görüldüğü gibi bu kontrol eklendiği yerde durmaktadır. Hata oluşmadığı durumlarda kontrollerin sayfa içinde yer alması istenmiyorsa hata mesajlarının dinamik olarak oluşturulup görüntülenmesi sağlanabilir. Bu işlem için tüm doğrulama kontrollerinde bulunan **Display** özelliği kullanılabilir. Display özelliği Static, Dynamic ve None olmak üzere üç değer alabilmektedir. Static özelliği kontrolün sayfada eklendiği şekilde kalacağını, Dynamic özelliği kontrolün hata oluştuğunda sayfaya dinamik olarak ekleneceğini ve None özelliği de hata mesajının görüntülenmeyeceğini belirtmektedir. Tasarlanan sayfa biraz daha geliştirilip ilk olarak bir tane CompareValidator eklensin ve Parola alanları birbiri ile karşılaştırılsın hemen ardından da tüm doğrulama kontrollerinin Display özelliği Dynamic olarak ayarlınsın. Bu durumdan sonra sayfada geçerli olmayan bir durum oluştuğunda hata mesajlarının görünümü aşağıdaki gibi olacaktır.



The screenshot shows a web browser window with the address bar displaying "http://localhost:14178/DogrulamaKontrolleri/Dc". The page contains a login form with the following fields and labels:

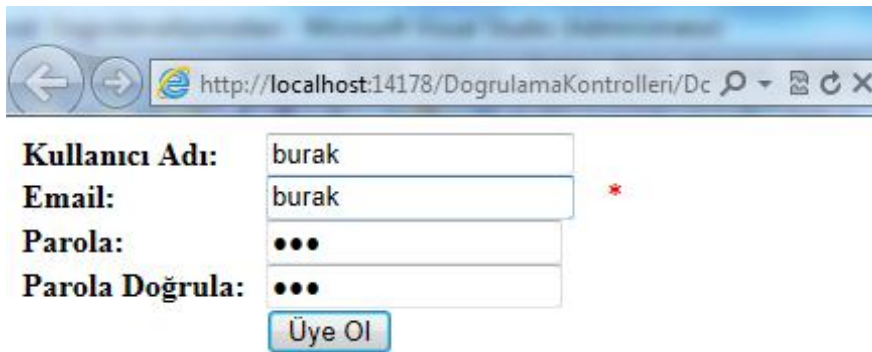
- Kullanıcı Adı:** burak
- Email:** burak
- Parola:** ●●●
- Parola Doğrula:** ●●●

Below the fields is a button labeled "Üye Ol". To the right of the form, there are two red error messages:

- "Lütfen geçerli bir email adresi giriniz..." (Please enter a valid email address...)
- "Parola doğrulanmadı..." (Password not verified...)

Hata mesajları bu şekilde değil de, grup halinde ValidationSummary kontrolü içerisinde alınmak istendiğinde yapılacak işlem oldukça basittir. Sayfaya bir tane ValidationSummary kontrolü eklenip diğer doğrulama kontrollerinin de Text özelliği * işareti olarak atandığında, hata alındığında aşağıdaki görünüm ile karşılaşıyor olacaktır. Gerekli ayarlar yapıldıktan sonra projeyi çalıştırmadan önce bir doğrulama kontrollerinin bir özelliğini daha ayarlamak mantıklı olacaktır. Doğrulama kontrollerinin hata oluştuğu anda imlecin hatanın oluştuğu alana odaklanması için kullanılabilecek **SetFocusOnError** adında bir özelliği vardır. Bu özellik varsayılan olarak False olarak gelmektedir ve True olarak ayarlandığında hata oluştuğu anda hatanın oluşmasına sebep olan alana imleci odaklar.

Proje çalıştırılıp sayfaya göz atıldığında hata oluşması durumunda hata mesajları en altta grup halinde görüntülenmektedir. Bu görünümün nasıl olacağı da ValidationSummary özelliğinin **DisplayMode** özelliğinden ayarlanabilmektedir. E-mail alanına geçersiz bir veri girildiğinde SetFocusOnError özelliği True olarak ayarlandığı için imleç, E-mail alanına odaklandırılarak kullanıcının hatalı alanı hızlıca düzeltebilmesi sağlanmaktadır.



The screenshot shows the same web browser window as before, but now the error messages are grouped at the bottom of the page. The form fields are the same:

- Kullanıcı Adı:** burak
- Email:** burak
- Parola:** ●●●
- Parola Doğrula:** ●●●

The button "Üye Ol" is still present. A single red error message is displayed at the bottom:

- "Lütfen geçerli bir email adresi giriniz..." (Please enter a valid email address...)

- Lütfen geçerli bir email adresi giriniz...

Gerçekleştirilen bu işlemlerden sonra sayfa oldukça güvenli gibi görünse de tarayıcıda script çalıştırılmadığı zaman bu önlemler işe yaramamaktadır. Daha önce de bahsedildiği gibi, doğrulama kontrolleri script tabanlı kontrollerdir ve çalışması için tarayıcının script çalıştırabiliyor olması gerekmektedir. Script desteği olmayan tarayıcılarda sayfa doğrulanmayacaktır. Ancak bu durum kullanıcıya belirtilemeyecektir. Bu durumun ele alınabilmesi için kod tarafında işlem yapmadan önce bu kontrol yapılmalıdır. Bu senaryoda Button kontrolüne tıklanıldığında bir işlem yapılması beklenmektedir. Bu durumda, Button kontrolünde aşağıdaki kodlar ile sayfanın doğrulama kontrollerinden geçip geçmediği anlaşılabilmektedir.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        Response.Write("Kodlar çalışır...");
    }
}
```

Yukarıdaki kodlar incelendiğinde sayfanın doğrulanıp doğrulanmadığını belirten Page sınıfının **IsValid** adında bir özelliği olduğu dikkatli gözlerden kaçmayacaktır. Bu özellik, sayfa geçerli ise True, geçerli değilse False değerini döndürecektir.

EĞİTİM :

**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

**Bölüm :
Doğrulama Kontrolleri**

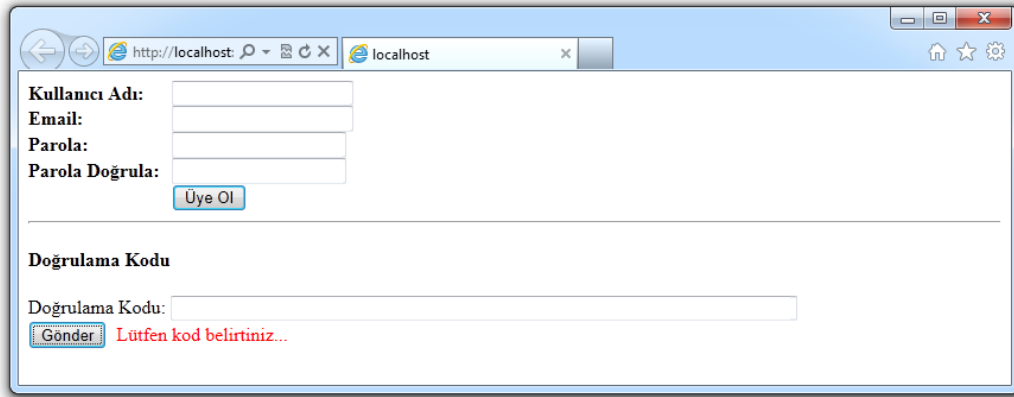
**Konu :
ValidationGroup Özelliği**



ValidationGroup Özelliği

ASP.NET ile birlikte yazılım geliştiricilere doğrulama grupları oluşturmak için ValidationGroup adına bir özellik sunuldu. Bu özellik yokken aynı sayfada farklı doğrulama grupları oluşturmak problem oluştuyordu. ValidationGroup özelliği ile aynı sayfa içerisinde farklı doğrulama grupları kurularak bu gruplardaki doğrulama kontrollerinden diğer grubun etkilenmesinin önüne geçilmiş oldu.

Bu durumu örneklemek için az önceki senaryoya devam ederek tasarlanan sayfaya bir tane de Doğrulama Kodu girilebilecek bir alan ekleyelim. Kullanıcılar bu sayfaya giriş yaptıklarında sisteme kayıt olmak için form doldurabilecekler ya da kayıt olduktan sonra kendilerine gönderilen doğrulama kodunu bu sayfada belirtilen alana girerek hesaplarını aktif hale getirebileceklerdir. Şu andaki durumda, bu işlem için sayfada herhangi bir işlem yapmak istenildiğinde, her iki alanda da herhangi bir doğrulama kontrolünden geçilmediği anda iki Buttonda çalışmıyor olacaktır. Bahsedilen problemi çözmek için yapılacak işlem çok basittir. Kayıt ekleme bölümünde bulunan tüm kontroller ve doğrulama kontrolleri seçilip ValidationGroup özelliği "Kayıt", diğer bölümdeki kontroller de seçilip ValidationGroup özelliği "Dogrulama" olarak ayarlandığında sayfada iki tane doğrulama grubu tanımlanmış olacak ve iki taraftaki doğrulama kontrolleri birbirinden etkilenmeyerek sayfa istenilen şekilde çalışmaya devam edecektir. Sayfada gerekli işlemler yapıp sayfa çalıştırıldığında aşağıdaki ekran görüntüsü ile karşılaşılacaktır.



The screenshot shows a web browser window with the address bar displaying "http://localhost:". The page contains two main sections. The first section, titled "Kullanıcı Adı:", includes input fields for "Email:", "Parola:", and "Parola Doğrula:", followed by a "Uye Ol" button. The second section, titled "Doğrulama Kodu", includes an input field for "Doğrulama Kodu:" and a "Gönder" button. Below the "Gönder" button, there is a red error message that reads "Lütfen kod belirtiniz...".

EĞİTİM :

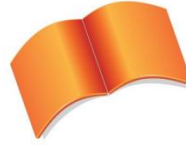
**DOĞRULAMA
KONTROLLERİ VE
KULLANICI TANIMLI
KONTROLLER**

Bölüm :

**Kullanıcı Tanımlı
Kontroller (User Controls)**

Konu :

**Kullanıcı Tanımlı Kontroller Ne
İşe Yarar?**



Microsoft Türkiye

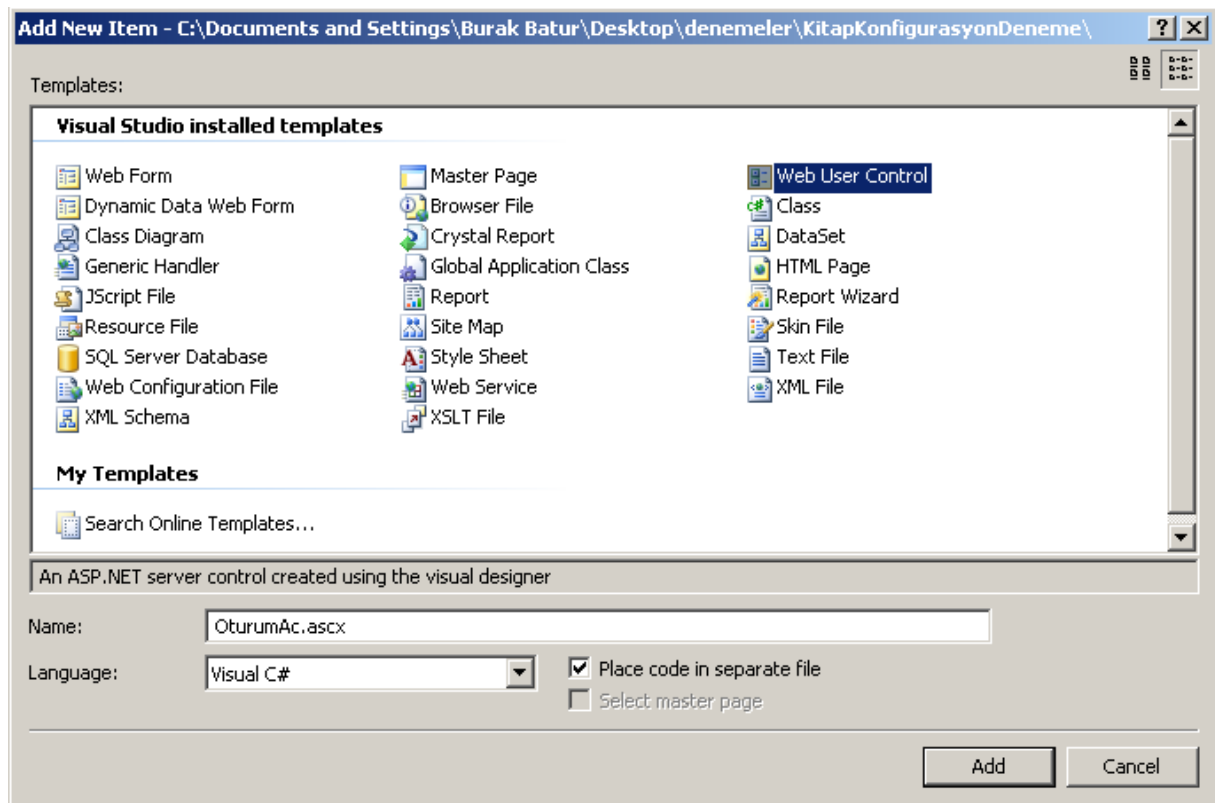
Açık Akademi

Kullanıcı Kontroller (User Controls) Ne İşe Yarar?

Kullanıcı Kontroller, bir içeriğin birden fazla sayfada kullanılmasına olanak tanıyan yapılardır. Kullanıcı Kontroller, ASP'deki include sayfaları ile benzerlik göstermektedirler. Herhangi bir ASP.NET Web formuna eklenip, eklendiği alana içerik sağlarlar ve böylece içeriğin tek bir yerden yönetilmesi sağlanmış olur.

Kullanıcı kontroller, tasarım ve programlama olarak ASP.NET Web Formları ile benzerlik göstermektedirler. Ancak, kullanıcı kontrollerinde sunucu tarafında çalışan bir form yer almaz. Bu nedenle de kullanıcı kontrolleri tek başına çalışamazlar. Çalışmak için mutlaka sunucu tarafında çalışan bir Web Form'un içine eklenmelidirler.

ASP.NET 1.0 ve 1.1.'de kullanım alanı oldukça geniş olan Kullanıcı Kontroller'in kullanımı, ASP.NET 2.0 ve sonrasında azalmıştır. Eski versiyonlarda tasarımın sabit kalması için sayfalarda menü ve resim gibi kontrolleri barındırmak amacıyla kullanılan Kullanıcı Kontroller, ASP.NET 2.0 ile gelen MasterPage yapılarının tasarım alanında devrim niteliğindeki yenilikleri ve kullanım kolaylığı nedeni ile artık tasarım amaçlı olarak çok fazla kullanılmamaktadır. Projeye yeni bir UserControl eklemek için proje üzerinde sağ tıklayıp, Add New Item seçeneği ile açılan dialog penceresinden **Web User Control** seçeneği seçilmelidir.



Yukarıdaki seçenekten Web User Control seçeneği seçildiğinde dosya uzantısının **ascx** olduğu fark edilmiş olmalıdır. UserControl'lerin dosya uzantısı ascx'tir. Yukarıdaki resimde dikkat çeken bir diğer nokta da *"Place code in separate file"* seçeneğinin sunuluyor olduğudur. Daha önce de bahsedildiği gibi UserControl'lerde tıpkı normal ASP.NET sayfaları gibi programlanabilmektedir ve kod tarafı farklı bir dosyada tutulabilmektedir. Bu dosyanın uzantısıda kullanılan programlama diline göre değişiklik göstermektedir, C# için bu uzantı **ascx.cs**'dir. Eklenen kontrolün kod tarafına göz atıldığında ise, bu tarafında normal sayfalar ile benzerlik gösterdiği dikkatli gözlerden kaçmayacaktır. Klasik ASP.NET sayfaları Page direktifi ile başlarken UserControllerler Control direktifi ile başlar ve kontrole özgü ayarların bir kısmı bu alandan gerçekleştirilebilir. Yeni bir sayfa eklendiğinde Control direktifi aşağıdaki gibi olacaktır. Arka planda kullanılacak olan programlama dili **Language** özelliği ile belirtiliyor.

Varsayılan, PageLoad gibi olayların otomatik olarak tetiklenmesini sağlamak amacı ile **AutoEventWireup** özelliği true olarak atanmış durumda. Arka plandaki kodları tutacak olan dosya **CodeFile** özelliği ile ve son olarak da oluşturulan UserControl'ün sınıf adı **Inherits** isimli özellik ile belirtiliyor.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="OturumAc.ascx.cs" Inherits="OturumAc" %>
```

Projeye yeni eklenen UserControl'e dikkat edildiğinde, tasarım yaparken de normal ASP.NET sayfası ile benzerlik gösterdiği fark edilecektir. Dolayısıyla normal sayfada nasıl tasarım yapılıyorsa UserControl üzerinde de aynı tasarım gerçekleştirilebilir. Durumu örneklemek için üzerinde iki tane TextBox ve bir tane de Button kontrolü bulunan bir UserControl tasarlanacaktır ve güzel görünüm için de kontroller tablo içerisinde konumlandırılıyor olacaktır. Aşağıda gerçekleştirilen tasarımın ekran görüntüsü yer almaktadır.

UserControl'ün tasarımı bittikten sonra oturum açma durumunu simule etmek için, Oturum Aç butonuna çift tıklayıp Click olayını ele alalım. Ardından da, oluşturulan metod içerisine aşağıdaki kodları yazalım. Bu UserControl'ün amacı birkaç farklı yerde kullanılıp eklendiği sayfa içerisinden direkt oturum açılmasını sağlamak olacaktır.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text == "Burak" && TextBox2.Text == "1234")
        Response.Write("Oturum Açıldı");
    else
        Response.Write("Hatalı kullanıcı adı veya parola...");
}
```

UserControl hazırlandıktan sonra sıra geldi bir sayfa içerisinde kullanmaya. UserControl üzerindeyken uygulama direkt çalıştırılırsa ekrana UserControl'ün yerine açılış sayfasının geldiği görülecektir. Bu durum, son derece doğaldır çünkü daha önce de belirtildiği gibi UserController tek başlarına çalışamazlar mutlaka sunucu tarafında çalışan bir form içerisine eklenmelidirler.

UserControl'ü bir sayfa içerisine ekleyip kullanmak için Visual Studio ortamında direkt sayfa içerisine sürüklenip bırakılabilir. Sürüklenip bırakıldığı yerde direkt içeriği görüntüleniyor olacaktır. Böylece, sayfayı çalıştırmadan sayfanın tasarımı kolay bir şekilde gerçekleştirilebilir. UserControl sayfaya sürüklenip bırakıldıktan sonra, sayfa içerisinde kullanılabilmesi için Page direktifinden hemen sonra register direktifi ile sayfaya kaydedilmektedir. Register direktifi ile sayfa içerisinde kullanılan UserControl'ün adresi ve sayfa içerisindeki etiket adı tanımlamaları yapılmaktadır

```
<%@ Register src="OturumAc.ascx" tagname="OturumAc" tagprefix="uc1" %>
```

Eklenen UserControl HTML tarafında aşağıdaki gibi görüntüleniyor olacaktır. Görüleceği üzere yukarıda tanımlandığı şekilde kullanılmıştır ve bu tanımlamaların tamamı Visual Studio tarafından gerçekleştirilmektedir.

```
<uc1:OturumAc ID="OturumAc1" runat="server" />
```

Dizayn tarafında ise, direkt tasarlanan görünüm ile karşılaşılacaktır. Visual Studio ile geliştirme yapılıyorsa UserControl'ün bir adet de SmartTag'ı olduğu dikkat çekecektir. SmartTag üzerinden UserControl'ün içeriği yenilebileceği gibi, içeriği düzenlemek için UserControl'ün tasarlandığı sayfaya da geçilebilir. Aşağıdaki resimde UserControl'ün bir sayfaya eklendikten sonraki sayfa içerisindeki görünümü yer almaktadır. Sayfa çalıştırıldığında, iki içerik birleştirilip sanki tek sayfadaymış gibi sorunsuz bir şekilde çalışmaya devam edecektir.

