GEÇERLİLİK KONTROLLERİ
VALİDATİON CONTROLS

Öğr. Gör. Emine TUNÇEL
Kırklareli Üniversitesi Pınarhisar Meslek Yüksekokulu

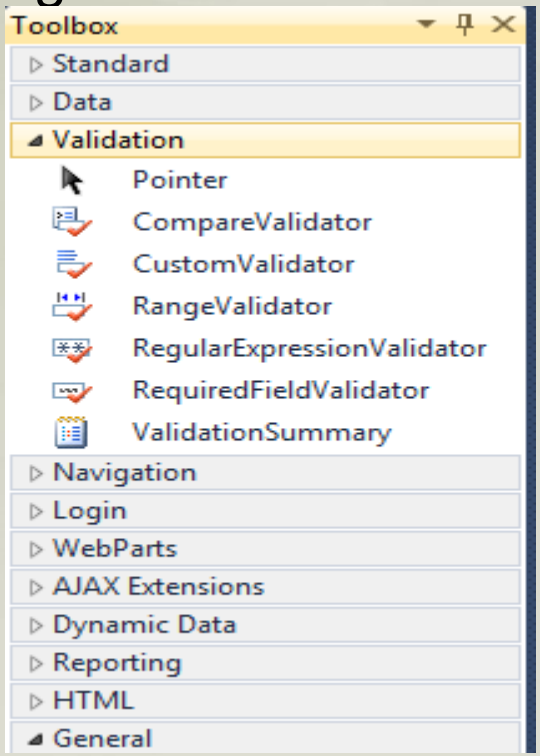
Giriş

- Kullanıcı bilgilerinin istenilen kriterlere uygun olup olmadığını denetlemek amacıyla kullanılan kontrollerdir
- Örneğin kullanıcının bir alanı boş bırakmasını engellemek, girilecek şifre için karakter sayısını belirlemek gibi..
- ASP.NET de 5 adet geçerlilik kontrolü

mevcuttur:

1) RangeValidator
2) RequiredFieldValidator
3) CompareValidator
4) RequiredExpressionValidator
5) CustomValidator

- ValidationSummary ise bir geçerlilik denetim kontrolü değil, bu kontrollerden gelen mesajların verildiği yerdir.



Geçerlilik Kontrolleri

1. RangeValidator Kontrolü

- Kullanıcının belirtilen aralıklarda değer girmesini sağlar.

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="TextBox1"
    ErrorMessage="Belirtilen aralıklarda bir değer girilmedi"
    MaximumValue="100"
    MinimumValue="50"
    Type="Integer">
</asp:RangeValidator>
```

- **ControlToValidate** : Hangi kontrole girilen değerlerin denetleneceğini belirten özelliktir.
- **ErrorMessage:** Belirtilen aralıklarda değer girilmediğinde gösterilecek hata mesajını belirtir.
- **MaximumValue** : ControlToValidate özelliği ile belirtilen kontrole girilebilecek en yüksek değerin belirtildiği özelliktir
- **MinimumValue** : ControlToValidate özelliği ile belirtilen kontrole girilebilecek en küçük değerin belirtildiği özelliktir
- **Type:** ControlToValidate özelliği ile belirtilen kontrole hangi tür verilen girilebileceğini belirtir.

Geçerlilik Kontrolleri

1. RangeValidator Kontrolü

- Kullanıcının belirtilen aralıklarda değer girmesini sağlar.

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="TextBox1"
    ErrorMessage="Belirtilen aralıklarda bir değer girilmedi"
    MaximumValue="100"
    MinimumValue="50"
    Type="Integer">
</asp:RangeValidator>
```

- **ControlToValidate** : Hangi kontrole girilen değerlerin denetleneceğini belirten özelliktir.
- **ErrorMessage:** Belirtilen aralıklarda değer girilmediğinde gösterilecek hata mesajını belirtir.
- **MaximumValue** : ControlToValidate özelliği ile belirtilen kontrole girilecek değer aralığının üst sınırını belirtir.
- **MinimumValue** : ControlToValidate özelliği ile belirtilen kontrole girilecek değer aralığının alt sınırını belirtir.
- **Type:** ControlToValidate özelliği ile belirtilen kontrole hangi tür verilen girilebileceğini belirtir.

Geçerlilik Kontrolleri

1. RangeValidator Kontrolü

- Denetleme işlemi 2 farklı şekilde yapılabilir:
  - **Client taraflı javascript ile yapılan kontrol.** Bu yöntemde bizim sayfaya herhangi bir Javascript kodu eklememize gerek yoktur. Geçerlilik kontrollerinden biri sayfaya eklendiğinde ASP.NET otomatik olarak bu javascript kodlarını ekler. Böylece sayfa sunucuya gitmeden geçerlilik denetimi yapılır. Eğer geçerli değilse sayfa sunucuya post edilmez.
  - **Sunucu taraflı yapılan kontrol.** Bu yöntemde sayfa sunucuya gönderilir denetimi sunucu yapar. Özellikle browser'lardaki güvenlik amaçlı JavaScript kullanımının iptal edildiği durumlarda kullanışlıdır. Bunun için geçerlilik kontrolü bildirimimize **EnableClientScript="False"** gibi bir ifade eklemeliyiz. Sunucu taraflı denetimde sayfanın geçerliliğini **if(Page.IsValid)** ifadesiyle kontrol edebiliriz. Eğer sayfa geçerli değilse yani tanımladığımız kıstaslar içinde değerler girilmediyse istenilen işlemler yapılmaz.

Geçerlilik Kontrolleri

2. RequiredFieldValidator Kontrolü

- Değer girilmesi zorunlu olan alanları kontrol eden denetim kontrolüdür.

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
     ControlToValidate="TextBox1"
     ErrorMessage="Bu alan boş geçilemez">
</asp:RequiredFieldValidator>
```

- **ControlToValidate** : kontrol edilecek olan alanın (web kontrolünün) id si burada belirtilir.
- **ErrorMessage:** Giriş yapılmamışsa hata mesajının yazılacağı alandır

Geçerlilik Kontrolleri

2. RequiredFieldValidator Kontrolü

- RequiredFieldValidator yalnızca boş alan olup olmadığını kontrol etmez. Aynı zamanda önceden tanımlanan değerler içinde geçerlilik denetimi yapar.
- Özellikle liste web kontrollerinin kullanımında bu özellikten faydalanılır.
- Bunun için InitialValue bildirimi yapmalıyız
- Örneğin açılır listeden bir seçim yapılmadığında hata oluşmasını sağlayalım.

Geçerlilik Kontrolleri

2. RequiredFieldValidator Kontrolü

- Öncelikle kontrol edilecek elemanı yani DropDownList kontrolünü sayfaya ekleyip aşağıdaki gibi düzenleyelim

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem Value="bos">Bir Kategori Seç</asp:ListItem>
    <asp:ListItem>Rock</asp:ListItem>
    <asp:ListItem>Pop</asp:ListItem>
    <asp:ListItem>Classic</asp:ListItem>
</asp:DropDownList>
```

- Daha sonra RequiredFieldValidator geçerlilik kontrolünü ekleyelim:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
    ControlToValidate="DropDownList1"
    ErrorMessage="Bir Kategori Seçmelisiniz."
    InitialValue="bos">
</asp:RequiredFieldValidator>
```

- Burada InitialValue="bos" bildirimi ile kontrol edilen alandan geri dönecek olan değerin "bos" olması durumunda geçersiz girdi olduğu belirtilir

Geçerlilik Kontrolleri

3. CompareValidator Kontrolü

- CompareValidator, temel olarak değerleri karşılaştıran bir kontroldür.
- 3 tip geçerlilik denetimi yapar.
- Bunlardan ilki girilen değer tipinin tanımlanan değer tipine eşit olup olmadığının kontrolünü yapar. Örneğin bir metin alanına tarih tipli bir değer girilmesi gerekiyorsa CompareValidator kontrolünün tipini Date olarak atayarak tarih formatındaki girdileri geçerli kılabiliriz.
- Diğer bir kullanım şekli sabit bir değer tanımlayarak girdinin bununla karşılaştırılmasıdır. Örneğin bir açık arttırma sitesinde girilen en son değeri sabit kabul ederek bundan yüksek değerin geçerli kabul edilmesi gibi.
- Son olarak form elemanlarının birbirine göre karşılaştırılmasının yapılmasıdır. Örneğin şifre tekrar şifre alanlarının aynı olması gibi.

Geçerlilik Kontrolleri

3. CompareValidator Kontrolü

- CompareValidator ile tip kontrolü


```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToValidate="TextBox1"
    ErrorMessage="Geçersiz tarih tipi"
    Operator="DataTypeCheck"
    Type="Date">
</asp:CompareValidator>
```

- Örneğimizde denetimin Operator="DataTypeCheck" ile veri tipi şeklinde yapılacağını, bu tipin de Type="Date" ile tarih olacağını belirtiyoruz
- Diğer bir ifadeyle metin kutusuna girilen değerin tarih formatında olup olmadığını kontrol ediyoruz

Geçerlilik Kontrolleri

3. CompareValidator Kontrolü

- CompareValidator ile sabit değer kontrolü


```
<asp:CompareValidator ID="CompareValidator2" runat="server"
    ControlToValidate="TextBox3"
    ErrorMessage="Girilen sayı 100 ve 100den küçük olmalıdır"
    Operator="LessThanEqual"
    Type="Integer"
    ValueToCompare="100">
</asp:CompareValidator>
```

- Bu örneğimizde de metin kutusuna girilebilecek değerlerin 100 ve daha küçük olmasını sağladık.
- Burda farklı olarak ValueToCompare metodu ile kıyaslama yapılacak değeri bildirim kısmında belirttik

Geçerlilik Kontrolleri

3. CompareValidator Kontrolü

- CompareValidator ile form elemanlarını kıyaslama


```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToCompare="TextBox1"
    ControlToValidate="TextBox2"
    ErrorMessage="Şifreler uyuşmuyor"
    Type="Integer">
</asp:CompareValidator>
```

- Diğer örneklerimizden farklı olarak bur da kullanıcının değer girdiği iki kutucuk var. İnteger olarak tutulan bu değerlerin eşit olmasını istiyoruz. Bunun için

  ControlToCompare="TextBox1"

  ControlToValidate="TextBox2"

- bildirimini kullandık

Geçerlilik Kontrolleri

4. RegularExpressionValidator Kontrolü

- RegularExpression'lar String'ler için format tanımlamaya yarayan özel ifadelerdir.
- Örneğin bir e-posta formatını özel ifadelerle belirleyip girişin o formatta olup olmadığını bu sayede kontrol edebiliriz.


```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
     ControlToValidate="TextBox1"
     ErrorMessage="Geçersiz E-Posta"
     ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp:RegularExpressionValidator>
```

- Örneğimizde **RegularExpression** tanımı **ValidationExpression** özelliği ile belirtild
- RegularExpression ile ilgili daha fazla bilgiyi http://regexlib.com/default.aspx adresinden alabilirsiniz. Bu adreste değişik kalıplar için **Regular Expression** tanımları bulabilirsiniz.

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Sunucu kontrolleri üzerinde denetim yapmak için bizim tarafımızdan tanımlanmış, kurallarını bizim belirlediğimiz bir denetim kontrolüdür
- Örneğin bir kullanıcıdan alınacak şifrenin en az 8 karakter olması ve içinde en az bir rakamın bulunması şeklinde bir kural tanımlanıp, girilen değerlerin uygun olup olmadığı denetlenebilir.
- CustomValidator, ClientSide ve ServerSide olmak üzere 2 farklı şekilde yapılabilir.

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Server Side

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
     ControlToValidate="TextBox1"
     ErrorMessage="Kullanıcı adınız 2 karakterden fazla olmalıdır"
     onservervalidate="denetim">
</asp:CustomValidator>
```

- **onservervalidate** olayında çalışacak denetim adlı metod :

```
protected void denetim(object source, ServerValidateEventArgs args)
{
    if (args.Value.Length > 2)
        args.IsValid = true;
    else
        args.IsValid = false;
}
```

- Bu metod ServerValidateEventArgs isminde özel bir argüman döndürür. Tanımladığımız koşulun gerçekleşmesi durumunda bu argümanın IsValid özelliğine true değeri atanır. Aksi takdirde false değeri ile denetim sağlanmamış olur ve ErrorMessage ile belirtilen mesaj ekrana yazılır

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Server Side

```
<asp:CustomValidator ID="CustomValidator2" runat="server"
ErrorMessage="Şifreniz en az 8 karakter olmalı ve en az bir numerik değer bulunmalıdır"
onservervalidate="parola_denetim"
ControlToValidate="TextBox2">
</asp:CustomValidator>
```

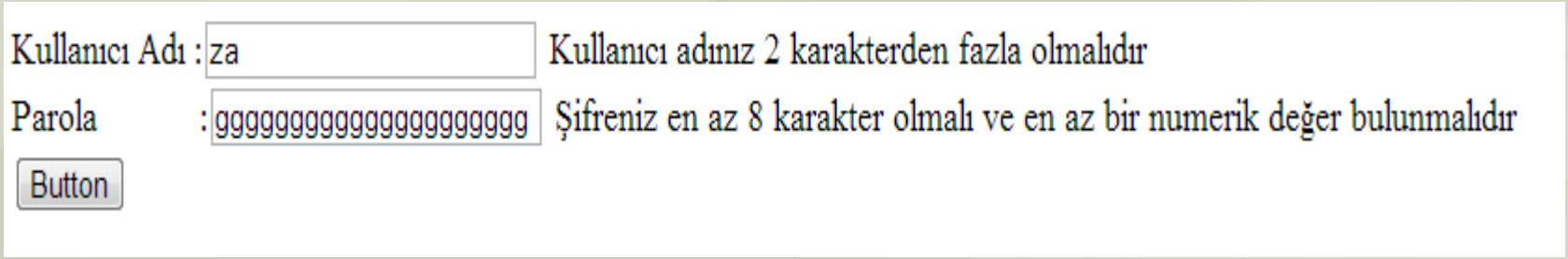
- **onservervalidate** olayında çalışacak parola_denetim adlı metot :

```
protected void parola_denetim(object source, ServerValidateEventArgs args)
{
    args.IsValid=false;//Başlangıçta koşulun gerçeklenmediğini kabul ediyoruz
    if (args.Value.Length < 8)
    {//Kontrol edilen metin alanına girilen değer 8 den küçük ise
        return;//programdan çık
    }
    foreach (char c in args.Value)
    {//metin alanına girilen değeri harf harf kontrol ediyoruz
        if (c >= '0' && c <= '9')
        {//eğer 0-9 arasında bir karakter varsa
            args.IsValid = true;//koşul sağlandı, hata mesajı görüntülenmeyecek
            break;
        }
    }
}
```

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Server Side



Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Örnek

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head><body>
    <form id="form1" runat="server">
    <div>
        Aşağıdaki soruya 10
        saniye içinde cevap vermelisiniz...<br />
        İlk Ağ Tarayıcısı Yazılımını kim geliştirmiştir?<br />
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:CustomValidator ID="CustomValidator1" runat="server"
            ControlToValidate="TextBox1"
            ErrorMessage="Verilen süreyi geçtiniz"
            onservervalidate="sure_denetimi"></asp:CustomValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Gönder" />
        <asp:Button ID="Button2" runat="server" Text="Tekrar süre iste"
            onclick="Button2_Click" />
    </div></form></body></html>
```

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Örnek

```
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
            Session["baslama_zamani"] = DateTime.Now;
    }
    protected void sure_denetimi(object source, ServerValidateEventArgs args)
    {
        DateTime baslama_zamani = (DateTime)Session["baslama_zamani"];
        if (baslama_zamani.AddSeconds(10) > DateTime.Now)
            args.IsValid = true;
        else
            args.IsValid = false;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        Session["baslama_zamani"] = DateTime.Now;
    }
</script>
```

Geçerlilik Kontrolleri

5. CustomValidator Kontrolü

- Custom Validator – Örnek
- Sayfamızın işleyişi şu şekilde:
- Sayfa ilk yüklendiğinde karşımıza bir soru çıkacak ve metin kutusuna 10 sn içinde bir cevap girilmesi beklenecek. Bu süre içinde cevap girilmemişse metin girişi geçersiz kabul edilecek
- Bunu sağlamak için Session değişkenlerinden faydalanıyoruz. Session'a şimdiki zamanı atayıp daha sonra Butona tıklandığında 10 sn eklenmiş session zamanımız ile karşılaştırıyoruz. Duruma göre IsValid'e true veya false değeri veriyoruz. Diğer bir ifadeyle geçerli veya geçersiz kılıyoruz
- Bu metodu da CustomValidator'de onservervalidate="sure_denetimi" bağlıyoruz.

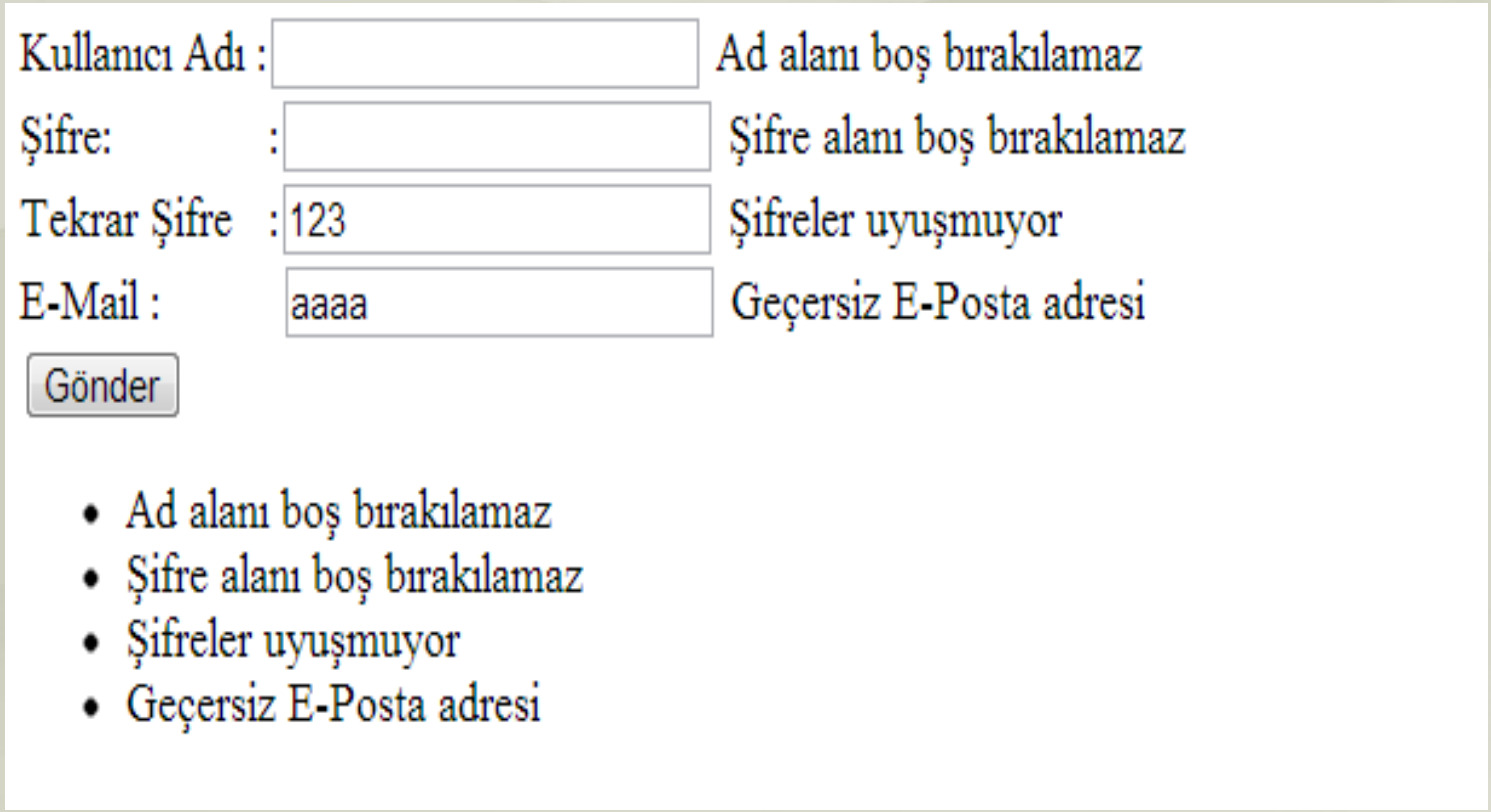
Geçerlilik Kontrolleri

ValidationSummary

- Bu kontrol aslında bir geçerlilik denetimi yapmaz.
- Diğer geçerlilik kontrollerinin bir özetini gösterir.
- Pek çok alandan oluşan uzun bir formunuz olduğunu düşünün.
- Kullanıcı formun sonuna ya da başına geldiğinde o zamana kadar eksik ya da geçersiz girdi hata mesajlarını tek tek form elemanlarının yanında aramak yerine hepsini derli toplu bir yerde görmesi oldukça kullanışlı bir özelliktir.
- İşte **ValidationSummary** bunu sağlıyor

Geçerlilik Kontrolleri

ValidationSummary



- KAYNAK
- C# ile ASP.NET
- Zafer Demirkol