# Bab 12

# Validasi pada ASP.NET

## 12.1 Validator

Sebuah validator adalah suatu pemeriksaan terhadap aksi yang dilakukan oleh pemakai (biasanya berupa input text), kemudian diproses sesuai dengan kebutuhan yang dapat diterifema oleh server. Memvalidasi input pengguna adalah skenario umum dalam aplikasi berbasis Web. Suatu validasi dapat dilakukan pada sisi bowser dan sisi server. Pada bab ini val idasi pada sisi server, sedangkan pada sisi browser, hal ini dapat memanfaatkan suatu javascript

Pada ASP.NET, suatu validasi server controls adalah serangkaian kontrol yang membantu memvalidasi data ketika pengguna masuk ke kontrol yang disediakan pada ASP.NET. Sebuah validator menentukan apakah form dapat diproses berdasarkan aturan yang telah ditetapkan dalam pengendalian validasi server.

Salah satu elemen yang paling umum dari halaman web adalah bentuk di mana pengguna dapat mengirimkan data input ke server dan menerima kembali *respon* dari server. Sebuah validasi adalah pengujian untuk menentukan apakah pengguna memasukkan sesuatu ke *field* sesuai dengan yang harapkan. Aturan validasi menentukan sesuatu yang dimasukkan, kemudian juga

memeriksa untuk melihat apakah apa yang dimasukkan, misalnya berupa angka atau karakter dan jika berada dalam format yang benar.

Sebuah validasi dapat dilakukan setelah user memasukkan data ke dalam bentuk Web, kemudian klik tombol Kirim, dan mengirimkan data formulir ke server sebagai permintaan, setelah kejadian tsb kemudian dilakukan validasi sisi-server terhadap data yang dikirimkan oleh browser. Jika data tidak benar atau tidak valid, *response* dikirmkan ke browser sebagai nilai yang tidak valid.

Pada validasi sisi server, kerugian tentang validasi server-side adalah memerlukan perjalanan bolak-balik dari browser ke server. Hal ini membutuhkan banyak sumber daya dan membuat proses menjadi lebih lambat bagi pengguna.

Jenis lain validasi seperti yang disebutkan sebelumnya, yaitu validasi sisi browser, sebelum pengguna mengklik tombol Kirim (*submit*), sebuah bahasa scripting yang merupakan bagian dari kode halaman HTML dijalankan untuk memeriksa validitas data sebelum dikirim ke server, inilah yang disebut sebagai validasi sisi klien.

Suatu validasi form pada sisi browser dengan cara menaruh beberapa kode program tipe *client-side* seperti JavaScript atau VBScript di bagian atas halaman ASP yang memeriksa apakah informasi dalam suatu *field* adalah benar. Metode ini menangani masalah lebih efisien karena tidak perlu melakukan perjalanan ke server. Contoh dibawah ini, menunjukkan contoh penggunaan client-side JavaScript untuk melakukan validasi form.

```
<script language="javascript">
<!-- Function CheckForm(form){
  for(var intCtr = 0; intCtr <= (form.elements.length - 5);
       ++intCtr){
       var temp = form.elements[intCtr];
       if(temp.type == "text" && temp.value == ""){
              alert("Please Enter All Information!");
              temp.focus();
              return false;}  }
  return true;}
//-->
</script>
```

Kode program dalam bahasa javascript diatas, memeriksa semua isian form yang bertipe text, jika data isian tsb kosong, maka program akan menampilkan pesan `alert("Please Enter All Information!");` . Kemudian fokus isian (kursor) diatur kembali ke form isian yang kosng dengan perintah : `temp.focus();`

Pengembangan validasi diatas, diadopsi oleh ASP.NET, sehingga ketika validasi dibuat atas kontrol-kontrol yang ada, maka ASP.NET akan menentukan secara otomatis, apakah memakai validasi sisi klien ataukah validasi sisi server. Dengan demikian suatu validasi server kontrol pada ASP.NET yang juga menghasilkan script sisi klien. Enam kontrol validasi sisi server tersedia untuk ASP.NET: `RequiredFieldValidator, CompareValidator, RangeValidator, RegularExpressionValidator, CustomValidatorValidationSummary`

Selain enam kontrol validasi diatas, validasi dapat dibuat dengan menyesuaikan untuk kebutuhan yang inginkan. Kemudian, jika ada kesalahan dalam data formulir, pengendalian validasi server ini memungkinkan untuk menyesuaikan tampilan informasi kesalahan pada browser.

Kontrol validasi server ditempatkan pada halaman Web sebagaimana jenis lainnya. Setelah pengguna mengirimkan formulir, informasi formulir pengguna dikirim ke kontrol validasi yang tepat, di mana data tsb akan dievaluasi. Pada tabel 12.1 dijelaskan setiap dari enam kontrol validasi server.

**Tabel 12.1 Validasi Server Control.**

| Validasi Server Control | Deskripsi |
|---|---|
| RequiredFieldValidator | Memastikan bahwa pengguna tidak melewatkan isian. Dengan kata lain, semua field yang memakai validasi ini, harus di-isi |
| CompareValidator | Memungkinkan untuk perbandingan antara input pengguna dan item lain dengan menggunakan operator perbandingan (sama, lebih besar dari, kurang dari, dan sebagainya) |

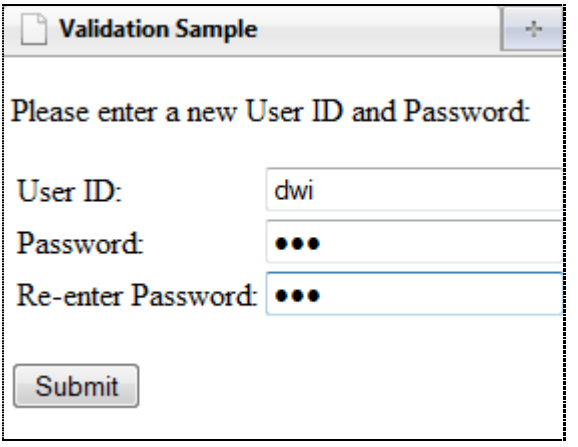
| | |
|---|---|
| RangeValidator | Cek input pengguna berdasarkan nilai range pada angka atau karakter |
| RegularExpressionValidator | Cek bahwa entri pengguna cocok dengan pola yang didefinisikan dengan ekspresi reguler. Biasanya dipakai untuk memeriksa alamat e-mail dan nomor telepon |
| CustomValidator | Cek bahwa entri pengguna dengan menggunakan logika validasi. |
| ValidationSummary | Menampilkan semua pesan error dari validator di satu tempat khusus pada halaman |

### 12.1.1 Validator RequiredFieldValidator

Validasi ini adalah memastikan bahwa pengguna tidak melewatkan isian. Dengan kata lain, semua field yang memakai validasi ini, harus di-isi. Contoh pemakain validasi, dapat diperlihatkan pada inputan untuk pemakaian halaman login, pada kode HTML biasa halaman login dibuat seperti kode dibawah ini:

```
<html><head><title>Validation Sample</title></head>
<body>
<form id="Form1" runat="server">
<p>Please enter a new User ID and Password:</p>
<table><tr><td>User ID:</td>
<td><input type="text" runat="server" id="txtName"/></td></tr>
<tr><td>Password:</td>
<td><input type="password" runat="server" id="txtPWord" /></td>
 </tr><tr>
<td>Re-enter Password:</td>
<td><input type="password" runat="server" id="txtRePWord"
/></td>
</tr></table><br>
<input type="submit" runat="server" id="cmdSubmit"
value="Submit" />
</form></body></html>
```

Jika kode HTML tsb dijalankan, maka visualisai hal itu terlihat pada gb 12.1:




**Gambar 12.1 Halaman Login**

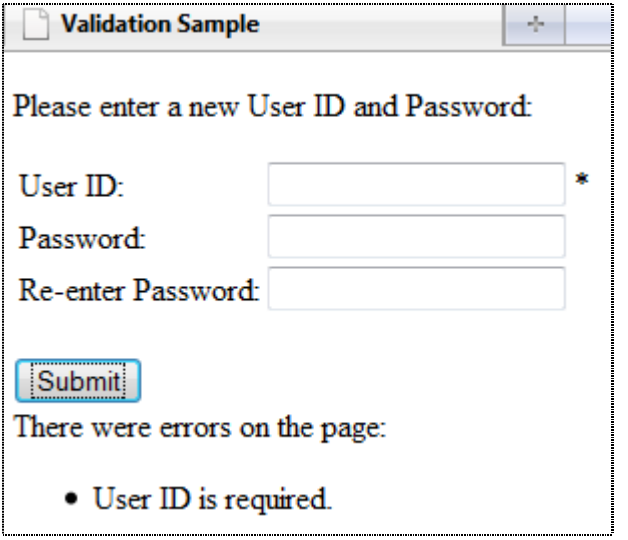
Kode-kode HTML yang dituliskan sebelumnya, dapat ditambakah kode ASP.NET untuk menampilkan tanda asterik (*) sebagai field isian yang wajib di –isi (tidak boleh kosong).

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat=server
      ControlToValidate=txtName
      ErrorMessage="User ID is required."> *
 </asp:RequiredFieldValidator>
```

Kode ASP.NET dapat pula ditambahkan sebagai *header* pada sebuah pesan kesalahan dengan tambahan kode:

```
<asp:ValidationSummary ID="ValidationSummary1" runat=server
HeaderText="There were errors on the page:" />
```

Hasil dari penambahan kode-kode ASP.NET merujuk pada pesan kesalahan ketika salah satu field isian yang tidak di isi, seperti terlihat pada gambar 12.2




**Gambar 12.2 Halaman Login dengan pesan kesalahan**

### 12.1.2 Validator CompareValidator

Validasi ini adalah memungkinkan untuk perbandingan antara input pengguna dan item lain dengan menggunakan operator perbandingan (sama, lebih besar dari, kurang dari, dan sebagainya) Pada contoh gambar 12.2, sebuah validasi dapat diterapkan terhadap field password dan re-enter password, dimana kedua isi field tsb harus sama. Pemakaian `CompareValidator` dapat diterapkan pada kasus ini, seperti ditunjukkan pada kode program dibawah ini:

```
<asp:CompareValidator ID="CompareValidator1" runat=server
   ControlToValidate=txtRePWord
   ControlToCompare=txtPWord
   ErrorMessage="Passwords do not match." />
```

Secara default, CompareValidator melakukan perbandingan string yang cocok sederhana. Jika diperlukan, ia bisa melakukan

perbandingan lebih kompleks yang melibatkan tanggal dan angka. Contoh validasi yang biasanya dipakai sebagai cek nilai terhadap tipe data tertentu, misalnya sebuah nilai integer, kode program dapat dilihat dibawah ini:

```
Age: <asp:TextBox id="TextBox1" runat="server" MaxLength="3">
</asp:TextBox> 
<asp:CompareValidator id="CompareValidator1" runat="server"
  ErrorMessage="You must enter a number"
  ControlToValidate="TextBox1" Type="Integer"
  Operator="DataTypeCheck"></asp:CompareValidator>
```

Dalam contoh ini, pengguna harus memasukkan sebuah integer dalam kotak teks, jika tidak, control CompareValidator akan menampilkan pesan kesalahan pada halaman. Dengan memberikan kontrol server CompareValidator bertipe Integer, isian *field* umur tsb, dipastikan hanyalahn nilai yang bertipe integer saja yang dapat dimasukkan dalam kotak teks sesuai dengan jenis NET Framework. data. Selain dengan tipe data, pilihan lain untuk membandingkan nilai terhadap konstanta tertentu.

```
Age: <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
   
<asp:CompareValidator id="CompareValidator1" runat="server"
  Operator="GreaterThan" ValueToCompare="18"
  ControlToValidate="TextBox1"
  ErrorMessage="You must be older than 18"
  Type="Integer">
</asp:CompareValidator>
```

Pada kasus ini, beberapa pemeriksaan dilakukan terhadap nilai apapun yang diketik oleh pengguna dalam kotak teks. Yang pertama didasarkan pada properti control CompareValidator.

Tipe Properti memiliki nilai Integer, sehingga setiap nilai yang ditempatkan dalam kotak teks harus sesuai dengan jenis integer yang ada pada Framework .NET. data. Jika pengguna memasukkan string ke dalam kotak teks, pengiriman formulir tidak valid.

Properti berikutnya adalah properti Operator. Properti ini memiliki nilai GreaterThan, yang berarti bahwa untuk pengiriman formulir menjadi valid, jika nilai yang dimasukkan dalam kotak teks harus lebih besar dari nilai yang ditetapkan ke properti ValueToCompare. Properti Operator dapat berisi salah satu dari nilai berikut: `Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, DataTypeCheck`

### 12.1.3 Validator RangeValidator

Kontrol server RangeValidator mirip dengan kontrol server CompareValidator, tetapi kontrol server RangeValidator membandingkan apa yang dimasukkan ke dalam kolom formulir dengan dua nilai dan memastikan bahwa apa yang dimasukkan oleh pengguna adalah diantara dua nilai yang ditentukan.

Sebagai contoh, sebuah kotak teks untuk isian umur pengguna, dengan nilai lebih besar dari atau kurang dari sebuah konstanta tertentu. Umur biasanya dimasukkan antara kisaran tertentu angka. Contoh kode program ini seperti dibawah ini:

```
Age: <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
   
<asp:RangeValidator id="RangeValidator1" runat="server"
  ControlToValidate="TextBox1" Type="Integer"
  ErrorMessage="You must be between 30 and 40"
  MaximumValue="40" MinimumValue="30"></asp:RangeValidator>
```

Dalam hal ini, pengguna harus memasukkan nilai antara 30 dan 40 dalam kotak teks. Jika beberapa nomor yang dimasukkan berada di luar kisaran ini, server kontrol RangeValidator akan membngkitkan pesan kesalahan dan menganggap pengiriman formulir tidak valid.

Jenis Properti `RangeValidator` memungkinkan untuk membuat perbandingan terhadap banyak tipe data yang berbeda di

dalam Framework. NET, seperti String, Integer, Double, Tanggal, dan Mata Uang.

Contoh berikut ini menggunakan tipe data String dalam kontrol server RangeValidator untuk memastikan bahwa nilai berada dalam kisaran tertentu karakter.

```
Last name:
<asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
   
<asp:RangeValidator id="RangeValidator1" runat="server"
  ControlToValidate="TextBox1"
  ErrorMessage="Your last name needs to be between M and P"
  MaximumValue="Q" MinimumValue="M"></asp:RangeValidator>
```

Pada contoh diatas, nilai sedang diperiksa terhadap rentang yang ditentukan dengan menggunakan properti MaximumValue dan MinimumValue. Perhatikan, dalam contoh ini, bahwa properti type tidak ditentukan. Dalam kasus ini, tidak perlu ditetapkan karena nilai default dari properti adalah Type String. Jika tidak ditentukan, maka dianggap memiliki nilai String.

### 12.1.4 Validator RegularExpressionValidator

Kontrol server RegularExpressionValidator adalah kontrol validasi yang memungkinkan untuk memeriksa input pengguna berdasarkan pola yang didefinisikan dengan ekspresi reguler. Pemakaian kontrol ini biasanya untuk memeriksa apakah pengguna telah memasukkan alamat e-mail yang valid atau nomor telepon. Di masa lalu, jenis validasi ini menerapkan cukup banyak kode JavaScript, namun dengan memakai Kontrol RegularExpressionValidator dalam ASP.NET kode kode javascript dapat dihemat dalam waktu pengkodean.

Suatu validasi `RegularExpressionValidator` yang merujuk pada karakter dan jumlah karakter, dapat dimasukkan pada isian field nama, seperti contoh kode dibawah ini:

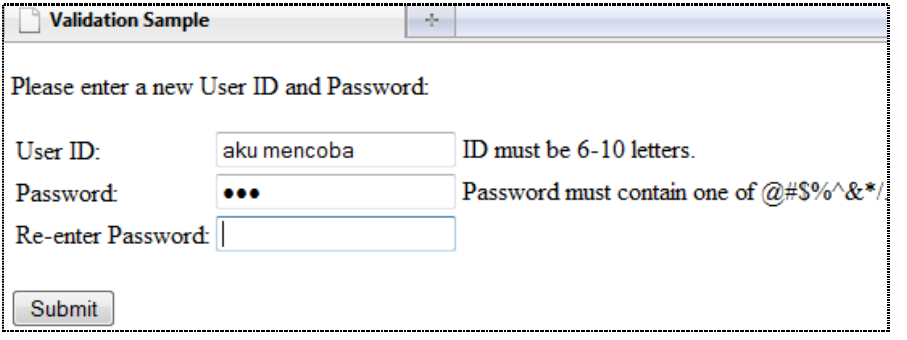
```
<asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat=server
```

```
ControlToValidate="txtName"
ErrorMessage="ID must be 6-10 letters."
ValidationExpression="[a-zA-Z]{6,10}" />
```

Karakter tertentu dapat divalidasi dengan memakai `RegularExpressionValidator` seperti pada pengisian field passwotrd dibawah ini:

```
<asp:RegularExpressionValidator
ID="RegularExpressionValidator2" runat=server display=dynamic
 ControlToValidate="txtPWord"
 ErrorMessage="Password must contain one of @#$%^&*/."
 ValidationExpression=".*[@#$%^&*/].*" />
<asp:RegularExpressionValidator
ID="RegularExpressionValidator3" runat=server display=dynamic
  ControlToValidate="txtPWord"
  ErrorMessage="Password must be 4-12 nonblank characters."
  ValidationExpression="[^\s]{4,12}" />
```

Pemakaian validasi diatas, dapat divisualisikan pada gambar 12.3 dibawah ini:



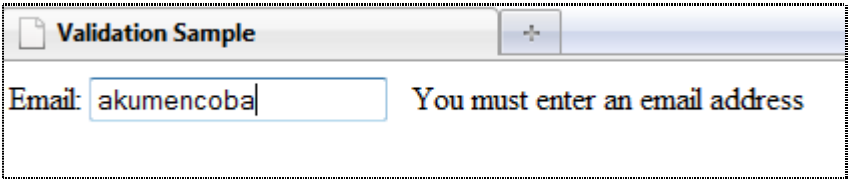

**Gambar 12.3 Validasi Halaman Login**

Contoh yang lain untuk memvalidasi input berupa masukan alamat email dengan memakai Kontrol server RegularExpressionValidator, berikut ini contoh programnya:

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

```
<asp:RegularExpressionValidator
  id="RegularExpressionValidator1" runat="server"
  ControlToValidate="TextBox1"
  ErrorMessage="You must enter an email address"
  ValidationExpression="\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*">
</asp:RegularExpressionValidator>
```

Dalam contoh ini, isian alamat e-mail akan diperiksa sebagai property regular expression di ValidationExpression. Gambar 12.4 menunjukkan pesan kesalahan bahwa jika pengguna memasukkan alamat e-mail yang salah dalam kotak teks.




**Gambar 12.4 Validasi Email**

Selain pesan kesalahan yang diperlihatkan gb 12.4 berupa teks, pesan tsb dapat pula diganti berupa sebuah gambar, berikut kode programnya: `ErrorMessage='<img src="error.gif">'`

### 12.1.5 Validator CustomValidator

Kontrol server CustomValidator memungkinkan untuk mengembangkan validasi sendiri apakah tervalidasi pada sisi server atau validasi sisi-klien. Sebagai misalnya, validasi mungkin ingin membandingkan input pengguna terhadap nilai dalam database, atau untuk menentukan apakah masukan sesuai dengan beberapa validasi aritmatika yang dicari (misalnya, jika nomor tersebut genap atau ganjil). Hal ini, dapat dilakukan dengan menggunakan jenis kontrol validasi.

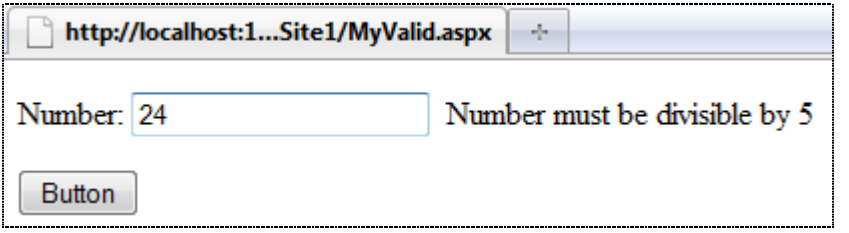
## Client-side Validation

Validasi pada sisi client-server dapat membuat sendiri fungsi JavaScript yang menyediakan validasi tingkat yang lebih tinggi kemampuan validasi. Contoh pemakain ini diperlihatkan pada kode program dibawah ini:

```
<%@ Page Language="C#" %>
<script runat="server">
    void Button1_Click(Object sender, EventArgs e) {
       Label1.Text = "VALID NUMBER!";}

</script>
<html><head>
    <script language="JavaScript">
        function validateNumber(oSrc, args) {
            args.IsValid = (args.Value % 5 == 0);
        }
    </script>
</head>
<body>
<form id="Form1" runat="server">
<p>Number:
<asp:TextBox id="TextBox1"
  runat="server"></asp:TextBox> 
<asp:CustomValidator id="CustomValidator1"
  runat="server" ControlToValidate="TextBox1"
  ErrorMessage="Number must be divisible by 5"
  ClientValidationFunction="validateNumber">
 </asp:CustomValidator></p>
 <p><asp:Button id="Button1" onclick="Button1_Click"
         runat="server" Text="Button"></asp:Button></p>
 <p><asp:Label id="Label1" runat="server"></asp:Label></p>
 </form></body></html>
```

Program diatas memeriksa masukan pengguna dengan sebuan `CustomValidator` dimana sisi client diberikan sebuah fungsi yaitu `ClientValidationFunction="validateNumber"`. Sebuah fungsi dalam bahasa *javascript* akan memeriksa masukan pengguna dengan fungsi arimatika modulus. Ketika pengguna memasukkan nilai yang tidak habis dibagi 5, maka pesan kesalahan akan dibangkitan sebagai contoh pada gb 12.5




**Gambar 12.5 Validasi Nilai**

Properti args.IsValid dapat bernilai *true* atau *false*. Properti args.Value adalah nilai dari user yang diambil dari kontrol server CustomValidator terkait, dalam hal ini adalah TextBox1. Menggunakan properti ClientValidationFunction memungkinkan untuk menggabungkan fungsi-fungsi JavaScript ke ASP.NET sehingga mereka berperilaku seperti kontrol validasi lainnya.

## Server-side Validation

Selain memakai validasi pada sisi client, cara lain melakukan validasi pada formulir Web dengan menggunakan kontrol server CustomValidator adalah dengan menggunakan validasi pada sisi server-side. Contoh pemakaian validasi pada sisi server diperlihatkan pada kode program dibawah ini:

```
<%@ Page Language="C#" %>
<script runat="server">
void Button1_Click(Object sender, EventArgs e)
{
 if (Page.IsValid) {Label1.Text = "VALID ENTRY!";}
}

void ValidateNumber(object source,
                    ServerValidateEventArgs args)
{
try {
       int num = int.Parse(args.Value);
       args.IsValid = ((num%5) == 0);
} catch(Exception ex){args.IsValid = false;}
}
```

```
</script>
<html><head></head>
<body>
<form id="Form1" runat="server">
<p>Number:
<asp:TextBox id="TextBox1"
 runat="server"></asp:TextBox> 
<asp:CustomValidator id="CustomValidator1"
 runat="server" ControlToValidate="TextBox1"
 ErrorMessage="Number must be divide 5"
 OnServerValidate="ValidateNumber"></asp:CustomValidator></p>
 <p><asp:Button id="Button1" onclick="Button1_Click"
        runat="server" Text="Button"></asp:Button></p>
 <p><asp:Label id="Label1" runat="server"></asp:Label></p>
    </form>
</body>
</html>
```

Validasi server-side harus menggunakan properti OnServerValidate bukan properti ClientValidationFunction, karena properti ClientValidationFunction digunakan dengan kontrol server CustomValidator ketika bekerja dengan validasi client-side.

Pemakaian kontrol CustomValidator untuk validasi pada sisi klien, harus mempertimbangkan kembali untuk mengevaluasi input pengguna menggunakan fungsi validasi server-side, agar aplikasi ASP.NET lebih aman.

Contoh pemakaian yang lain, selain memeriksa input pada textbox, control `CustomValidator` dapat diterapkan pada checkbox, berikut program:

```
<%@ Page Language="C#" %>
<script runat="server">
void CustomValidator1_ServerValidate(Object source,
       ServerValidateEventArgs args)
{      args.IsValid = (CheckBox1.Checked == true); }

void Button1_Click(Object sender, EventArgs e)
{
 if (Page.IsValid) {Label1.Text = "Thank "; }
 else {Label1.Text = "";}
}
</script>
<html><head></head><body>
```

```
<form id="Form1" runat="server">
<p>Check checkbox if you want to donate $10 </p>
<p><asp:CheckBox id="CheckBox1" runat="server"
    Text="Donate $10"></asp:CheckBox> 
   <asp:CustomValidator id="CustomValidator1"
    runat="server" ErrorMessage="Please donate $10"
    OnServerValidate="CustomValidator1_ServerValidate">
   </asp:CustomValidator></p>
<p><asp:Button id="Button1" onclick="Button1_Click"
    runat="server" Text="Submit"></asp:Button></p>
<p><asp:Label id="Label1" runat="server"></asp:Label></p>
</form></body></html>
```

Contoh diatas memaksa pengguna untuk melakukan centang pada pilihan checkbox.

### 12.1.6 Validator ValidationSummary

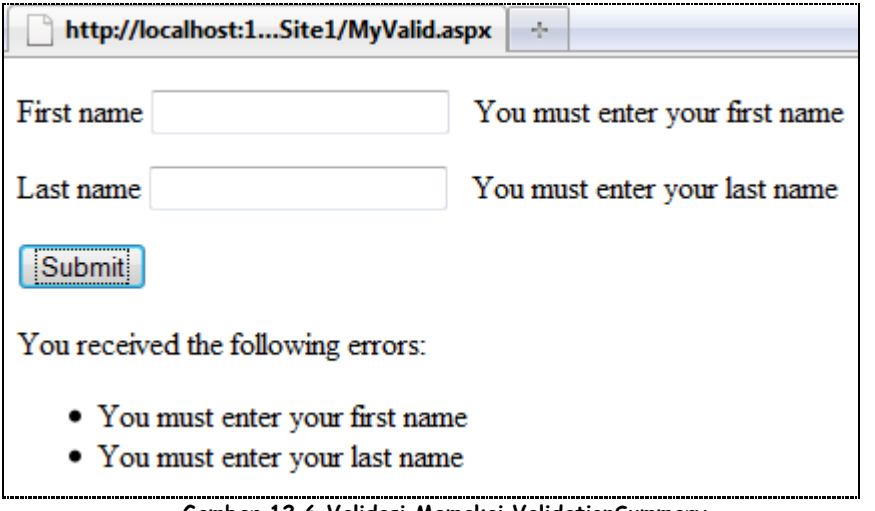
Kontrol server ValidationSummary bekerja dengan semua kontrol validasi server pada halaman. Isi pesan kesalahan dapat ditampilkan dalam daftar, bullet, atau paragraf. Contoh pemakaian validasi ini diperlihatkan pada program dibawah ini:

```
<%@ Page Language="C#" %>
<script runat="server">
void Button1_Click(Object sender, EventArgs e)
{
 if (Page.IsValid) {Label1.Text = "VALID ENTRY!";}
}
</script>
<html><head></head>
<body>
<form id="Form1" runat="server">
<p>First name
<asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
     
<asp:RequiredFieldValidator id="RequiredFieldValidator1"
       runat="server"
ErrorMessage="You must enter your first name"
ControlToValidate="TextBox1">
</asp:RequiredFieldValidator></p>
<p>Last name
<asp:TextBox id="TextBox2" runat="server"></asp:TextBox>
```

```
     
 <asp:RequiredFieldValidator id="RequiredFieldValidator2"
       runat="server"
ErrorMessage="You must enter your last name"
ControlToValidate="TextBox2">
</asp:RequiredFieldValidator></p>
<p><asp:Button id="Button1"
onclick="Button1_Click" runat="server"
       Text="Submit"></asp:Button></p>
<p><asp:ValidationSummary id="ValidationSummary1"
       runat="server"
       HeaderText="You received the following errors:">
   </asp:ValidationSummary></p>
<p><asp:Label id="Label1" runat="server"></asp:Label></p>

    </form>
</body>
</html>
```

Ketika program tsb dijalankan, kemudian ditemukan terjadi kesalahan pada isian pengguna, maka kumpulan pesan kesalahan ditampilkan seperti yang ditunjukkan pada gb 12.6 dibawah ini:




**Gambar 12.6 Validasi Memakai ValidationSummary**

Umumnya pesan kesalahan ditampilkan berupa *bullet* , yang merupakan default, server kontrol ValidationSummary menunjukkan kesalahan dalam daftar bullet pada halaman menggunakan teks merah. Pilihan yang lain adalah `BulletList, List, SingleParagraph`

Untuk mengubah bagaimana pesan kesalahan ditampilkan, dengan cara mengubah nilai dari properti DisplayMode dari kontrol ValidationSummary. Pilihan lain untuk menampilkan kesalahan ini adalah dengan memanfaatkan textbox seperti dibawah ini:

```
<asp:ValidationSummary id="ValidationSummary1" runat="server"
        ShowMessageBox="True"
        ShowSummary="False"></asp:ValidationSummary>
```

Fitur ini dapat menunjukkan kesalahan halaman ASP.NET dalam kotak pop-up dialog, ataupun memiliki pilihan untuk menunjukkan ringkasan dalam browser dan kotak dialog bersama-sama, atau hanya di kotak dialog.

Properti yang mengendalikan apakah pesan muncul dalam browser adalah properti ShowSummary. Untuk mematikan tampilan kesalahan validasi di browser, set nilai dari properti ShowSummary menjadi False. Untuk menampilkan kesalahan validasi dalam kotak dialog, setel kontrol ValidationSummary property ShowMessageBox ke True

## 12.2 UserControl

Selain menggunakan kontrol server Web di halaman Web ASP.NET, kontrol dapat pula dibuat sendiri, kemudian kontrol dapat digunakan kembali menggunakan teknik yang sama seperti membuat halaman Web ASP.NET. Sebuah kontrol pengguna adalah jenis kontrol komposit yang bekerja seperti sebuah kontrol ASP.NET Web, kemudian kontrol tsb dapat di-embed dalam halaman ASP.NET Web, di mana mereka bertindak sebagai unit.
Langkah-langkah untuk membuat user control adalah

1. Ubah nama kontrol sehingga nama file berkstensi ascx.
2. Hapus kode elemen html, body, dan form.
3. Ubah direktif Page @ ke direktif Control @.
4. Hapus semua attribute pada direktif @ Control kecuali Language, AutoEventWireup (jika ada), CodeFile, dan Inherits.
5. Sertakan atribut className di direktif Control @. Hal ini memungkinkan kontrol pengguna akan mengikuti tuntunan pengetikan terhadap kelas yang dimaksud.

Pemakaian user control diperlihatkan sebagai kelompok kontrol yang dapat digunakan seperti kendali tunggal pada contoh dibawah ini:


**Gambar 12.7 Kelompok kontrol dengan satu kendali**

Kode program terhadap kontrol dropdownlist ditunjukkan pada program dibawah ini:

```
<%@ Control Inherits="MoneyFieldBase"
Src="MoneyField.ascx.cs" %>
<asp:TextBox ID="amount" Runat="server" />
<asp:DropDownList ID="currency" AutoPostBack="true"
 OnSelectedIndexChanged="Select" Runat="server">
 <asp:ListItem Text="Euro" Value="1.0" Selected="true" />
 <asp:ListItem Text="Dollars" Value="0.88" />
 <asp:ListItem Text="Francs" Value="1.47" />
 <asp:ListItem Text="Pounds" Value="0.62" />
</asp:DropDownList>
```

Penanganan kontrol tsb pada sisi server sesuai dengan kode program diatas, yaitu sebuah kelas `MoneyFieldBase` yang berada pada file `MoneyField.ascx.cs`

```
using System;  using System.Web.UI;
using System.Web.UI.WebControls;
public class MoneyFieldBase : UserControl {
```

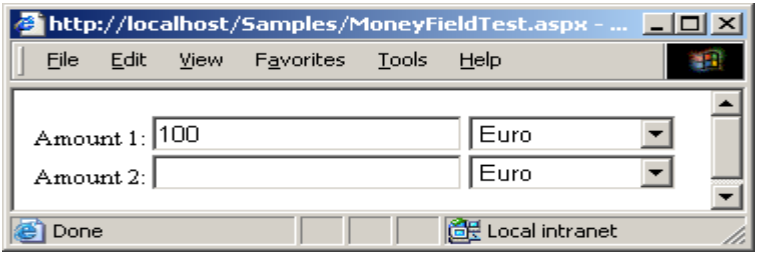
```
protected TextBox amount;
protected DropDownList currency;
public string Text {
 get { return amount.Text; }
 set { amount.Text = value; }}
private double OldRate {
get { return
ViewState["rate"] == null ? 1 : (double)ViewState["rate"]; }
set { ViewState["rate"] = value; }}

public void Select (object sender, EventArgs arg) {
try {
double val = Convert.ToDouble(amount.Text);
double newRate =
       Convert.ToDouble(currency.SelectedItem.Value);
double newVal = val / OldRate * newRate;
amount.Text = newVal.ToString("f2");
OldRate = newRate;} catch (Exception) {amount.Text = "0";}}}
```

Beberapa instan user control diatas dapat digunakan pada halaman yang sama, contoh berikut ini menunjukkan pemakaian dua user control diatas:

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="my" TagName="MoneyField"
    Src="MoneyField.ascx" %>
<html><body>
<form Runat="server">
Amount 1: <my:MoneyField ID="field1" Text="100"
             Runat="server" /><br>
Amount 2: <my:MoneyField ID="field2" Runat="server" />
</form></body></html>>
```

Hasil dari program yang memakai dua user control diatas, diperlihatkan pada gb 12.8 dibawah ini:




**Gambar 12.8 Dua user control pada satu halaman**