ASP.NET - Validators

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

BaseValidator Class

The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.

Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
   runat="server" ControlToValidate ="ddlcandidate"
   ErrorMessage="Please choose a candidate"
   InitialValue="Please choose a candidate">

</asp:RequiredFieldValidator>
```

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.

It has three specific properties:

Properties	Description

Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
   ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
   MinimumValue="6" Type="Integer">

</asp:RangeValidator>
```

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control.

It has the following specific properties:

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
   ErrorMessage="CompareValidator">


</asp:CompareValidator>
```

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The following table summarizes the commonly used syntax constructs for regular expressions:

Character Escapes	Description
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Metacharacters	Description

.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Quantifiers could be added to specify number of times a character could appear.

Quantifier	**Description**
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.

{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
   ValidationExpression="string" ValidationGroup="string">


</asp:RegularExpressionValidator>
```

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">


</asp:CustomValidator>
```

ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

The following two mutually inclusive properties list out the error message:

- **ShowSummary** : shows the error messages in specified format.
- **ShowMessageBox** : shows the error messages in a separate window.

The syntax for the control is as given:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
   DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

Validation Groups

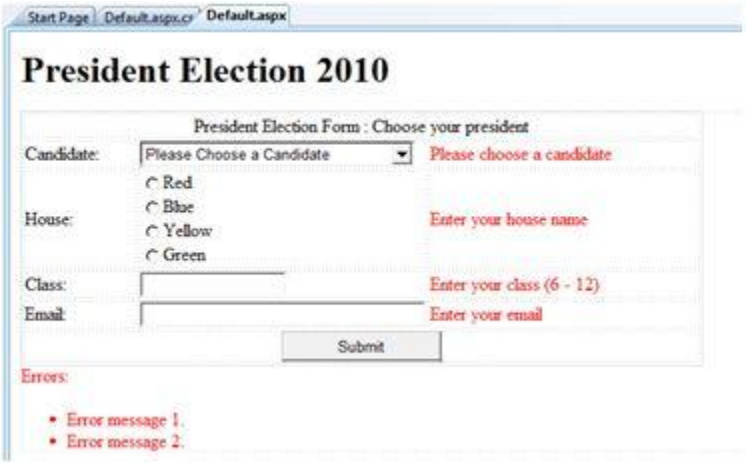
Complex pages have different groups of information provided in different panels. In such situation, a need might arise for performing validation separately for separate group. This kind of situation is handled using validation groups.

To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup* property.

Example

The following example describes a form to be filled up by all the students of a school, divided into four houses, for electing the school president. Here, we use the validation controls to validate the user input.

This is the form in design view:



The content file code is as given:

```
<form id="form1" runat="server">

   <table style="width: 66%;">

      <tr>
         <td class="style1" colspan="3" align="center">
         <asp:Label ID="lblmsg"
            Text="President Election Form : Choose your president"
            runat="server" />
         </td>
      </tr>

      <tr>
         <td class="style3">
            Candidate:
         </td>

         <td class="style2">
            <asp:DropDownList ID="ddlcandidate" runat="server"  style="width:239px">
               <asp:ListItem>Please Choose a Candidate</asp:ListItem>
               <asp:ListItem>M H Kabir</asp:ListItem>
               <asp:ListItem>Steve Taylor</asp:ListItem>
               <asp:ListItem>John Abraham</asp:ListItem>
               <asp:ListItem>Venus Williams</asp:ListItem>
            </asp:DropDownList>
         </td>

         <td>
            <asp:RequiredFieldValidator ID="rfvcandidate"
               runat="server" ControlToValidate ="ddlcandidate"
               ErrorMessage="Please choose a candidate"
               InitialValue="Please choose a candidate">
            </asp:RequiredFieldValidator>
```

```
        </td>
    </tr>

    <tr>
        <td class="style3">
            House:
        </td>

        <td class="style2">
            <asp:RadioButtonList ID="rblhouse" runat="server" RepeatLayout="Flow">
                <asp:ListItem>Red</asp:ListItem>
                <asp:ListItem>Blue</asp:ListItem>
                <asp:ListItem>Yellow</asp:ListItem>
                <asp:ListItem>Green</asp:ListItem>
            </asp:RadioButtonList>
        </td>

        <td>
            <asp:RequiredFieldValidator ID="rfvhouse" runat="server"
                ControlToValidate="rblhouse" ErrorMessage="Enter your house name" >
            </asp:RequiredFieldValidator>
            <br />
        </td>
    </tr>

    <tr>
        <td class="style3">
            Class:
        </td>

        <td class="style2">
            <asp:TextBox ID="txtclass" runat="server"></asp:TextBox>
        </td>
```

```
    <td>
        <asp:RangeValidator ID="rvclass"
            runat="server" ControlToValidate="txtclass"
            ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
            MinimumValue="6" Type="Integer">
        </asp:RangeValidator>
    </td>
</tr>

<tr>
    <td class="style3">
        Email:
    </td>

    <td class="style2">
        <asp:TextBox ID="txtemail" runat="server" style="width:250px">
        </asp:TextBox>
    </td>

    <td>
        <asp:RegularExpressionValidator ID="remail" runat="server"
            ControlToValidate="txtemail" ErrorMessage="Enter your email"
            ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
        </asp:RegularExpressionValidator>
    </td>
</tr>

<tr>
    <td class="style3" align="center" colspan="3">
        <asp:Button ID="btnsubmit" runat="server" onclick="btnsubmit_Click"
            style="text-align: center" Text="Submit" style="width:140px" />
    </td>
```

```
      </tr>
   </table>
   <asp:ValidationSummary ID="ValidationSummary1" runat="server"
      DisplayMode ="BulletList" ShowSummary ="true" HeaderText="Errors:" />
</form>
```

The code behind the submit button:

```
protected void btnsubmit_Click(object sender, EventArgs e)
{
   if (Page.IsValid)
   {
      lblmsg.Text = "Thank You";
   }
   else
   {
      lblmsg.Text = "Fill up all the fields";
   }
}
```