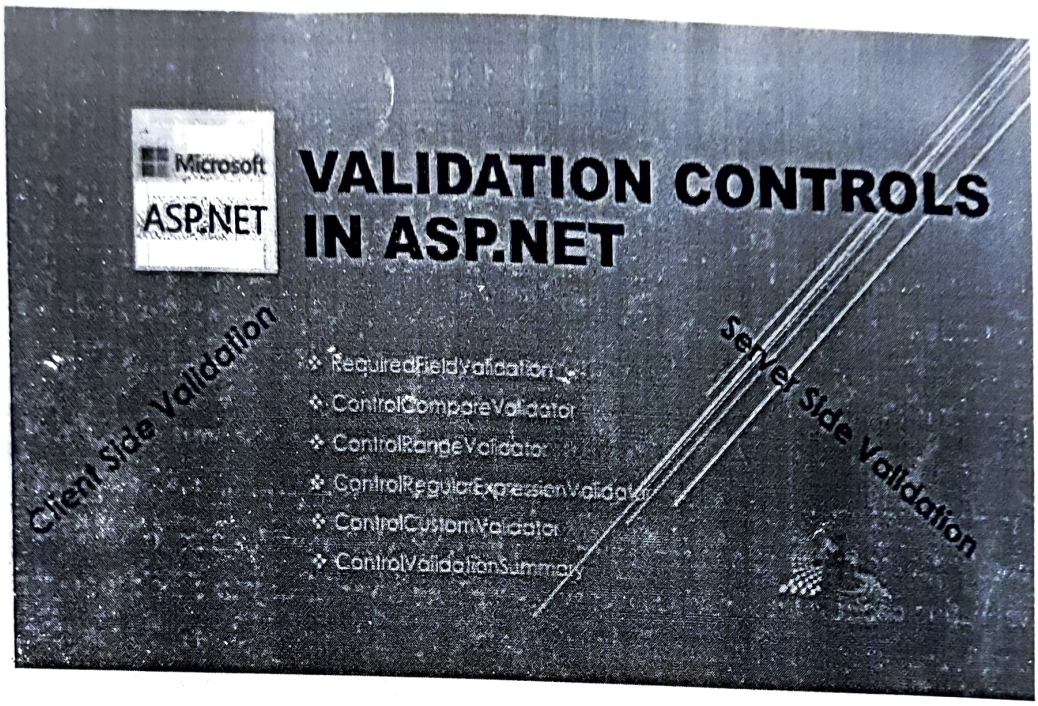
Validation Controls In ASP.NET

Why we use validation controls?

Validation is important part of any web application. User's input must always be validated before sending across different layers of the application.



Validation controls are used to,

- Implement presentation logic.
- To validate user input data.
- Data format, data type and data range is used for validation.

Validation is of two types

1. Client Side
2. Serve Side

Client side validation is good but we have to be dependent on browser and scripting language support.

Client side validation is considered convenient for users as they get instant feedback. The main advantage is that it prevents a page from being postback to the server until the client validation is executed successfully.

For developer point of view serve side is preferable because it will not fail, it is not dependent on browser and scripting language.

You can use ASP.NET validation, which will ensure client, and server validation. It work on both end; first it will work on client validation and than on server validation. At any cost server validation will work always whether client validation is executed or not. So you have a safety of validation check.

For client script .NET used JavaScript. WebUIValidation.js file is used for client validation by .NET

CHAPTER 15

Introduction to Validation Controls

15.1 VALIDATION

Validation means to verify that the value entered in web form is valid. There are always validation conditions which must be followed before submitting the form data. Commonly used validations are:

- Field should not be empty.
- Field should contain values in a specified range.
- Two fields should contain same values.
- Field should contain value in a specified format.

ASP.NET provides different types of validation controls which can be used to validate the data of an input control. If the data does not pass validation, it will display an error message to the user.

The Web Forms framework includes a set of validation server controls that provide an easy to use but powerful way to check input forms for errors and display messages to the user in case an error occurs.

Validation controls are added to a Web Forms like any other server controls. There are specific controls for specific types of validation such as range checking or pattern matching or checking that a user does not skip a field without entering anything.

We can attach more than one validation control to an input control. For example we may need both that an entry is required and that it must contain a specific range of values.

Validation controls work with a limited HTML and Web server controls. For each control, a specific property contains the value to be validated. The following table lists the input controls that may be validated.

Control	Validation Property
HtmlInputText	Value
HtmlTextArea	Value
HtmlSelect	Value
HtmlInputFile	Value
TextBox	Text
ListBox	SelectedItem.Value
DropDownList	SelectedItem.Value
RadioButtonList	SelectedItem.Value
FileUpload	FileName

Validation controls make page validation much easier and reduce the amount of code that the developer must write to perform page validation. All validation controls in the .NET Framework are derived from the BaseValidator class. This class serves as the base abstract class for the validation Controls. Validation controls always perform validation checking on the server. Validation controls also have complete client -side implementation that allows browsers that support DHTML to perform validation on the client. Client side validation enhances the validation scheme by checking user input as the user enters data. This allows errors to be detected on the client before the form is submitted, preventing the round trip necessary for server-side validation. In addition, more than one validator may be used on a page to validate different aspects.

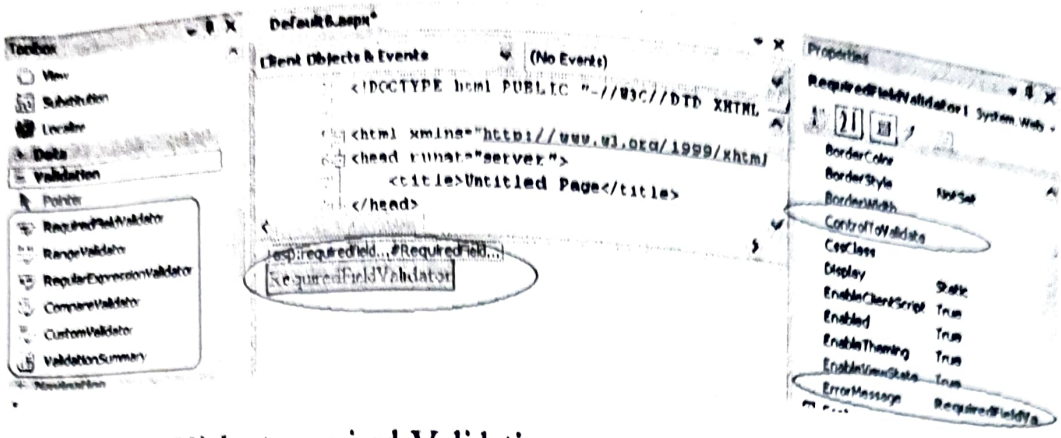
If web browser supports JavaScript, validation is performed client side. The best thing about ASP.NET validators is that "validation is only performed server side if necessary". To check the effect, we can add **enableclientscript="false"** to the RequiredFieldValidator. Now browser will post back to the server, but the result will be the same.

Various validation controls provided by ASP.NET are:

- **RequiredFieldValidator Control**
- **CompareValidator Control**
- **RangeValidator Control**
- **RegularExpressionValidator Control**
- **CustomValidator Control**
- **ValidationSummary Control**

15.2 STEPS TO INSERT A VALIDATION CONTROL.

- Open the ASP.NET page in design view.
- Go to Toolbox and Select the Required Validation Control



- Double Click at required Validation control under Validation Section of Toolbox..
- We can set properties of Validation control from Properties window.

15.3 REQUIREDFIELDVALIDATOR

The **RequiredFieldValidator** is actually very simple, and yet very useful. We can use it to make sure that

the user has entered something in a TextBox control. This validator ensures that the user does not skip an entry. With this control, the validation fails if the input value does not change from its initial value. By default, the initial value is an empty string (“”).Leading and trailing spaces of the input value are removed before validation.

The InitialValue property does not set the default value for the input control. It indicates the value that we do not want the user to enter in the input control.

Most important properties of RequiredFieldValidator are ControltoValidate, ErrorMessage and Text. ControltoValidate property is used to specify the ID of the control on which validation is applied. ErrorMessage property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set. Text property specifies the message to display when validation fails.

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

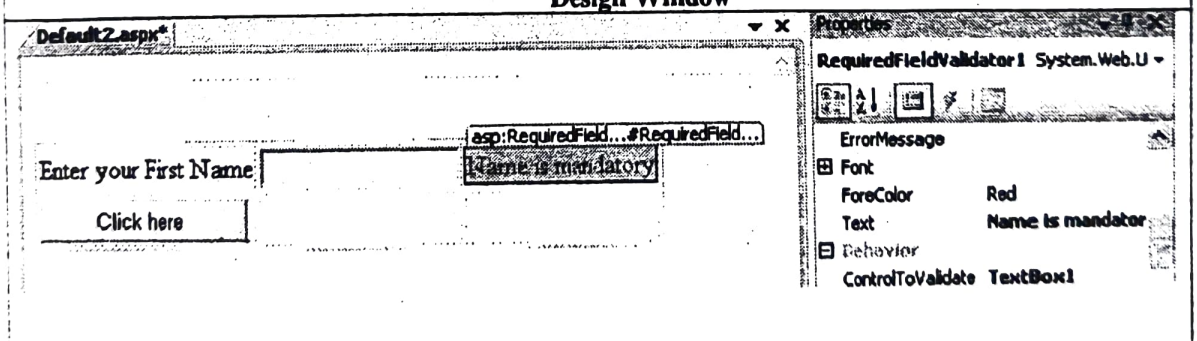
15.3.1 Properties of RequiredFieldValidator

15.3.1 Properties of RequiredFieldValidator

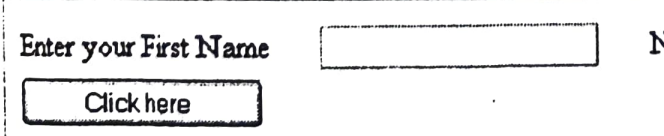
Property	Description
BackColor	This property specifies the background color of the RequiredFieldValidator control
ControlToValidate	This property specifies the id of the control which has to be validated.
Display	This property specifies the display behavior for the validation control. It can have following values: • None (the control is not displayed. It is used to show the error message only in the ValidationSummary control) • Static (the control displays an error message if validation fails). Space is reserved on the page for the message even if the input passes validation. • Dynamic (the control displays an error message if validation fails). Space is not reserved on the page for the message if the input passes validation
EnableClientScript	This property can have two values True or False. This property specifies whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specifies whether the validation control is enabled or not.
ErrorMessage	This property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set.
ForeColor	This property is used to specify the foreground color of the control.
id	This property is used to specify a unique id for the control.
InitialValue	Specifies the starting value of the input control. Default value is ""
IsValid	This property can have two values True or False. This property indicates whether the control specified by ControlToValidate is valid or not.
runat	This property specifies that the control is a server control. It must be set to "server"
Text	This property specifies the message to display when validation fails.

Program 15.1 Program to demonstrate the use of Required Field Validator Control

Design Window



Output



Validation Property

easier and reduce the amount of
e validation. All validation controls
eValidator class. This class serves
ntrols. Validation controls always
lation controls also have complete
s that support DHTML to perform
nhances the validation scheme by
allows errors to be detected on the
e round trip necessary for server-
idator may be used on a page to

s performed client side. The best
on is only performed server side if
ableclientscript="false" to the
back to the server, but the result

are:

In this program, a TextBox with ID TextBox1, RequiredFieldValidator with ID RequiredFieldValidator1, Button with ID Button1 have been taken. RequiredFieldValidator has been applied to TextBox1 by using ControlToValidate property. On Clicking Button,. If we don't fill any value in the TextBox1, "Name is Mandatory" will be displayed at RequiredFieldValidator Control.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <table class="style1">
            <tr>
                <td>Enter your First Name</td>
                <td><asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox></td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
                        ControlToValidate="TextBox1">Name is
mandatory</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr><td>
                    <asp:Button ID="Button1" runat="server"
                        style="height: 26px" Text="Click here" Width="131px"
/>
                </td>
                </tr>
        </table>
        </div>
    </form>
</body>
</html>
```

15.4 COMPAREVALIDATOR CONTROL

The CompareValidator control is used to compare the value of one input control to the value of another input control or to a fixed value. This validator compares the value of control with a constant value or a property value of another control using a comparison operator like less than (<),greater than(>), equal to (==) etc. If the input control is empty, the validation will succeed. We should use the RequiredFieldValidator control to make the field required.

Most important properties of RequiredFieldValidator are ControltoValidate, ControltoCompare, ErrorMessage and Text. ControltoValidate property is used to specify the ID of the control on which validation is applied. This property specifies the name of the control whose value will be compared with the value of the control

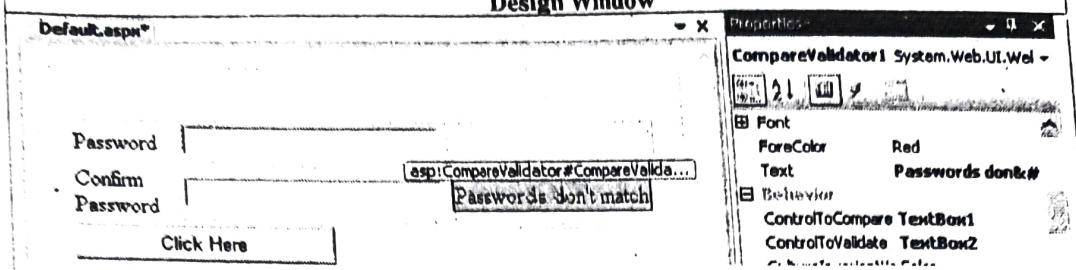
specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set. Text property specifies the message to display when validation fails.

15.4.1 Properties of CompareValidator

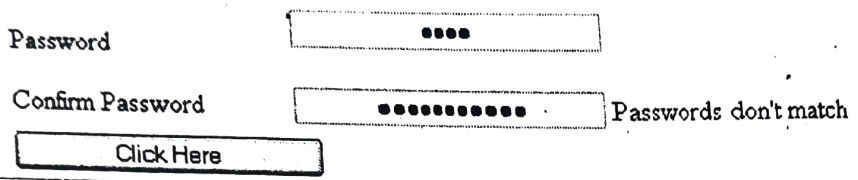
Property	Description
BackColor	This property specifies the background color of the CompareValidator control
ControlToCompare	This property specifies the name of the control whose value will be compared.
ControlToValidate	This property specifies the id of the control to be validated.
Display	This property specifies the display behavior for the validation control. It can have following values: • None (the control is not displayed. Used to show the error message only in the ValidationSummary control) • Static (the control displays an error message if validation fails. Space is reserved on the page for the message even if the input passes validation. • Dynamic (the control displays an error message if validation fails. Space is not reserved on the page for the message if the input passes validation
EnableClientScript	This property can have two values True or False. This property specifies whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specifies whether the validation control is enabled or not.
ErrorMessage	This property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set.
ForeColor	This property is used to specify the foreground color of the control.
id	This property is used to specify a unique id for the validation control.
IsValid	This property is used to check whether the control to be validated is a valid control or not.
Operator	The type of comparison to perform. The operators are: • Equal • GreaterThan • GreaterThanEqual • LessThan • LessThanEqual • NotEqual • DataTypeCheck
runat	This property specifies that the control is a server control. It must be set to "server".
Text	This property specifies the message to display when validation fails.
Type	This property specifies the data type of the values to compare. The types are: • Currency • Date • Double • Integer • String
ValueToCompare	This property specifies a constant value to compare with.

Program 15.2 Program to demonstrate the use of CompareValidator Control

Design Window



Output



In this program, two TextBoxes with ID TextBox1, TextBox2, CompareValidator with ID CompareValidator1, Button with ID Button1 have been taken. CompareValidator has been applied to TextBox1 by using ControlToValidate property, ControlToCompare property has been set for TextBox2. On Clicking Button. If we don't fill same value both the TextBoxes, "Passwords don't match" will be displayed at CompareValidator Control.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

</head>
<body>
    <form id="form1" runat="server">
    <div>

        <b>
        <br class="style6" />
        </b>
        <table align="center" class="style1">
            <tr>
                <td>
                    Password                    <td>
                    <asp:TextBox ID="TextBox1" runat="server"
                        style="text-align: center; margin-left: 0px"
Width="181px"
                        TextMode="Password" ></asp:TextBox>
 </td>
                <td>
                     </td>
            </tr>
            <tr>
                <td>
                    Confirm
                    Password</td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server"
```

```
width="181px"                    style="text-align: center; margin-left: 0px"

                                 TextMode="Password"></asp:TextBox>
                    </td>
                    <td>
runat="server"          <asp:CompareValidator ID="CompareValidator1"

                                 ControlToCompare="TextBox1"
ControlToValidate="TextBox2"
                                 ErrorMessage="Password mismatch">Passwords don&#39;t
match</asp:CompareValidator>
                    </td>
                </tr>
                <tr>
                    <td colspan="3">
                        <asp:Button ID="Button1" runat="server"
onclick="Button1_Click1"
                                 Text="Click Here" Width="181px" />
                    </td>
                </tr>
            </table>

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

15.5 RANGEVALIDATOR CONTROL

The **RangeValidator** control is used to check that the user enters an input value that falls between two values. It is possible to check ranges within numbers, dates, and characters. The validation will not fail if the input control is empty. We should use the RequiredFieldValidator control to make the field required.

The validation will not fail if the input value cannot be converted to the data type specified. We should use the CompareValidator control with its Operator property set to ValidationCompareOperator's DataTypeCheck property to verify the data type of the input value.

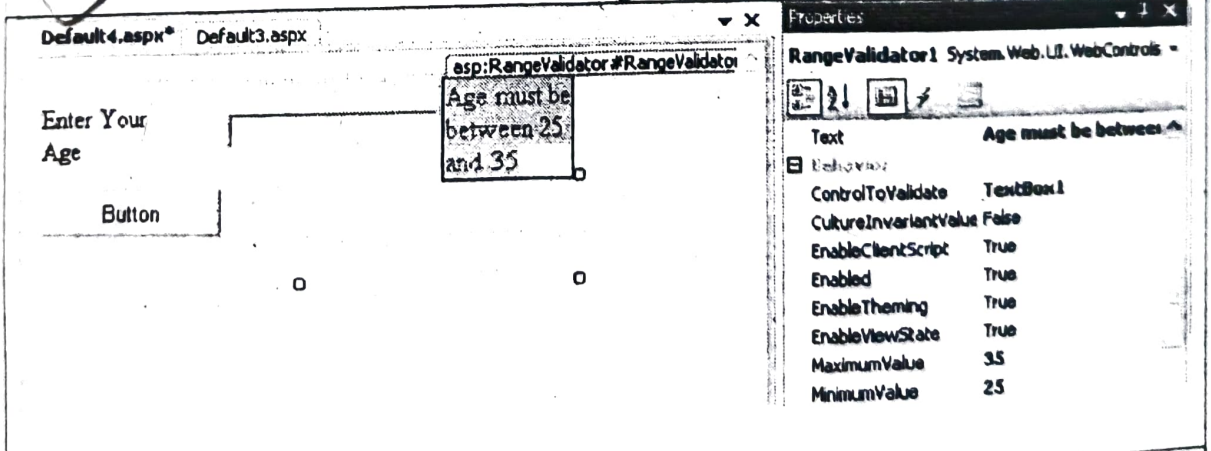
Most important properties of RangeValidator are ControltoValidate, MinimumValue, MaximumValue, ErrorMessage and Text. ControltoValidate property is used to specify the ID of the control on which validation is applied. This property specifies the name of the control whose value will be compared with range of values specified by MinimumValue and MaximumValue properties. ErrorMessage property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set. Text property specifies the message to display when validation fails.

15.5.1 Properties of RangeValidator

Property	Description
BackColor	This property specifies the background color of the RangeValidator control.
ControlToValidate	This property specifies the id of the control to be validated.
Display	This property specifies the display behavior for the validation control. It can have following values: • None (the control is not displayed. Used to show the error message only in the ValidationSummary control) • Static (the control displays an error message if validation fails. Space is reserved on the page for the message even if the input passes validation. • Dynamic (the control displays an error message if validation fails. Space is not reserved on the page for the message if the input passes validation
EnableClientScript	This property can have two values True or False. This property specifies whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specifies whether the validation control is enabled or not.
ErrorMessage	This property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set.
ForeColor	This property is used to specify the forcground color of the control.
id	This property is used to specify a unique id for the validation control.
IsValid	This property is used to check whether the control to be validated is a valid control or not.
MaximumValue	This property specifies the maximum value of the input control
MinimumValue	This property specifies the minimum value of the input control
runat	This property specifies that the control is a server control. Must be set to "server"
Type	This property specifies the data type of the value to check. The types are: • Currency • Date • Double • Integer • String
Text	This property specifies the message to display when validation fails.

Program 15.3 Program to demonstrate the use of RangeValidationControl

Design Window



Output

Enter Your Age 10 Age must be between 25 and 35

Button

In this program, a TextBox with ID TextBox1, RangeValidator with ID RangeValidator1 and a Button with ID Button1 have been taken. RangeValidator has been applied to TextBox1 by using ControlToValidate property, On Clicking Button. If we don't fill values between 25 and 35 in Textbox, "Age must be between 25 and 35" will be displayed at RangeValidator Control. Range has been specified by using MaximumValue and MinimumValue properties.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default4.aspx.cs"
Inherits="Default4" %>

<!DOCTYPE html PUBLIC " //W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    </head>
<body>
    <form id="form1" runat="server">
    <div>

        <table>
            <tr>
                <td>
                    Enter Your Age</td>
                <td>
                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                </td>
                <td>
                    <asp:RangeValidator ID="RangeValidator1" runat="server"
                        ControlToValidate="TextBox1" ErrorMessage="Your age
must be between 25 and 35"
                        MaximumValue="35" MinimumValue="25">Age must be
between 25 and 35</asp:RangeValidator>
                    <br />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Button ID="Button1" runat="server" Height="28px"
Text="Button"
                        Width="114px" />
                </td>
                <td>
                     </td>
                <td>
                     </td>
            </tr>
            <tr>
                <td>
                    <br />
                </td>
            </tr>
        </table>
    </div>
    </form>
</body>
</html>
```

15.6 REGULAREXPRESSIONVALIDATOR CONTROL

The RegularExpressionValidator control is used to ensure that an input value matches a specified pattern. Both server- and client-side validation are performed unless the browser does not support client-side validation or the EnableClientScript property is set to false. The validation will not fail if the input control is empty. Use the RequiredFieldValidator control to make the field required.

Most important properties of RegularExpressionValidator are ControltoValidate, ValidationExpression, ErrorMessage and Text. ControltoValidate property is used to specify the ID of the control on which validation is applied. ValidationExpression specifies the expression used to validate input control. Expression may be predefined or user defined. This expression specifies the syntax of value to be entered. ErrorMessage property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set. Text property specifies the message to display when validation fails.

15.6.1 Properties of Regular Expression Validator

Property	Description
BackColor	This property specifies the background color of the RegularExpressionValidator control.
ControlToValidate	The id of the control to validate
Display	This property specifies the display behavior for the validation control. It can have following values: • None (the control is not displayed. Used to show the error message only in the ValidationSummary control) • Static (the control displays an error message if validation fails. Space is reserved on the page for the message even if the input passes validation. Dynamic (the control displays an error message if validation fails. Space is not reserved on the page for the message if the input passes validation
EnableClientScript	This property can have two values True or False. This property specifies whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specifies whether the validation control is enabled or not.
ErrorMessage	This property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set.
ForeColor	This property is used to specify the foreground color of the control.
id	This property is used to specify a unique id for the validation control.
IsValid	This property is used to check whether the control to be validated is a valid control or not.
runat	This property specifies that the control is a server control. Must be set to "server"
Text	This property specifies the message to display when validation fails.
ValidationExpression	This property specifies the expression used to validate input control. ValidationExpression specifies the expression used to validate input control. Expression may be predefined or user defined. This expression specifies the syntax of value to be entered. The expression validation syntax is different on the client than on the server. JScript is used on the client. On the server, the language we have specified is used

ValidationExpresion can contain various literals as well as metacharacters. Literals represent specific defined characters whereas metacharacters represent some specific type of expression. Various metacharacters are as follows:

- . A dot represents any character other than new line.
- \s represents any whitespace character (such as a space or tab).

- \d represents any digit.

  Example: Regular Expression 333\s\d\d\d would match 333 333 and 333 945 but not 334 333 or 3334 945.

- * sign represents Zero or more occurrences of the previous character or subexpression.

  Example, 7*8 matches 7778 or just 8.

- + sign represents repeated character.

  Example: 5+7 means "one or more occurrences of the character 5, followed by a single 7." The number 57 would match, as would 555557.

- ( ) Group of parenthesis is used to group a sub expression.

  Example: (52)+7 would match any string that started with a sequence of 52. Matches would include 527, 52527, 5252527, and so on.

- [ ] Square brackets are used to specify range of characters from which only one character can be used.

  Example: [a-f] would match any single character from a to f (lowercase only).

  Example: [a-f]\w+ing would match any word that starts with a letter from a to f, contains one or more "word" characters (letters), and ends with "ing". Possible matches include acting and developing.

  Example: .+@.+ would match any e-mail address by verifying that it contains the @ symbol. The dot is a metacharacter used to indicate any character except a newline. However, some invalid e-mail addresses would still be allowed, including those that contain spaces and those that don't include a dot (.).

- {N} Braces are used to specify number of characters that may be entered.

  Example: {4} would check that 4 characters are entered.

- {m,n} The previous character (or subexpression) can occur from m to n times.

  Example A{1,3}matches A, AA, or AAA.

- [^] Matches a character that isn't in the given range.

  Example, [^A-B] matches any character except A and B.

- \S Any nonwhitespace character.
- \D Any character that isn't a digit.
- \w Any "word" character (letter, number, or underscore).

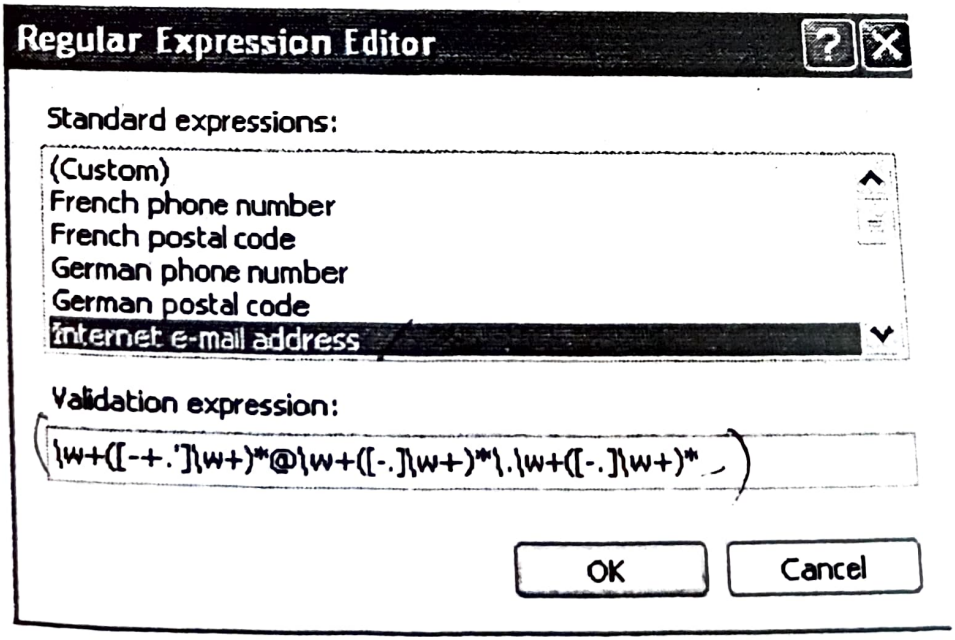
- \W Any character that isn't a "word" character (letter, number, or underscore).

15.6.2 Commonly Used Regular Expressions

Content	Regular Expression	Description
E-mail address*	\S+@\S+\.\S+	Check for an at (@) sign and dot (.) and allow nonwhitespace characters only.
Password	\w+	Any sequence of one or more word characters (letter, space, or underscore).
Specific-length password	\w{4,10}	A password that must be at least four characters long but no longer than ten characters.
Advanced password	[a-zA-Z]\w{3,9}	As with the specific-length password, this regular expression will allow four to ten total characters. The twist is that the first character must fall in the range of a–z or A–Z (that is to say, it must start with a non accented ordinary letter).
Another advanced password	[a-zA-Z]\w*\d+\w*	This password starts with a letter character, followed by zero or more word characters, one or more digits, and then zero or more word characters. In short, it forces a password to contain one or more numbers somewhere inside it. We could use a similar pattern to require two numbers or any other special character.
Limited-length field	\S{4,10}	Like the password example, this allows four to ten characters, but it allows special characters (asterisks, ampersands, and so on).
US Social Security number	\d{3}-\d{2}-\d{4}	A sequence of three, two, and then four digits, with each group separated by a dash. We could use a similar pattern when requiring a phone number.

15.6.3 Steps to apply ValidationExpression property to Regular ExpressionValidator Control

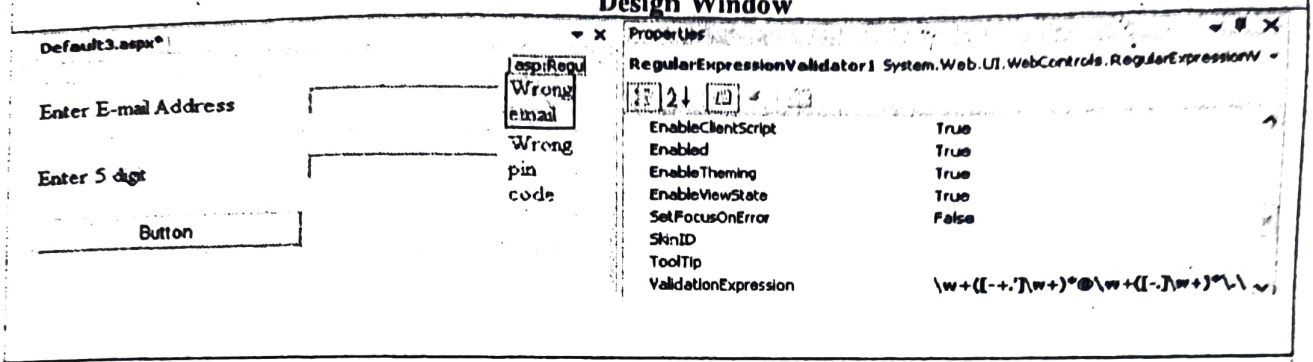
- Open ASP.NET page in design view.
- Place RegularExpressionValidator control in ASP.NET page.
- Click at … button in ValidationExpression property from Properties window.
- RegularExpressoinEditor Dialog box will open as:



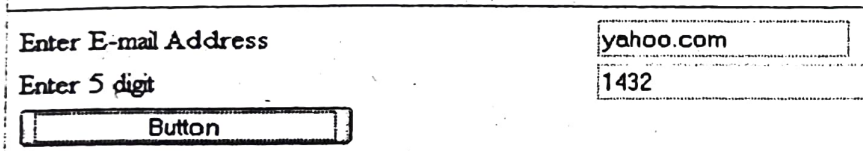
- We can choose any of the readymade expressions or we can create our own expression using Custom value from this dialog box.

Program 15.4 Program to demonstrate the use of RegularExpressionValidation Control

Design Window



Output



In this program, two TextBoxes with IDs TextBox1, TextBox2, RegularExpressionValidators with IDs RegularExpressionValidator1 and RegularExpressionValidator2 and a Button with ID Button1 have been taken. RegularExpressionValidator1 has been configured for valid E-mail address and RegularExpressionValidator2 has been configured for entering 5 digit number. RegularExpressionValidator1 has been applied to TextBox1 by using ControlToValidate property and RegularExpressionValidator2 has been applied to TextBox2, On Clicking Button. If we don't fill proper values in TextBoxes, "Wrong email" message will be displayed at RegularExpressionValidator1 and "Wrong pin code" message will be displayed at RegularExpressionValidator2.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default3.aspx.cs"
Inherits="Default3" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .style1
        {
            width: 54%;
            height: 73px;
        }
        </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <table
            class="style1">
            <tr>
                <td>
                    Enter E-mail Address</td>
```

```
                <td>
                    <asp:TextBox ID="TextBox1" runat="server" Width="128px"
></asp:TextBox>
                </td>
                <td>
                    <asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat="server"
                        ControlToValidate="TextBox1" ErrorMessage="Wrong
email entered"
                        ValidationExpression="\w+([-+.']\w+)*@\w+([-
.]\w+)*\.\w+([-.]\w+)*">Wrong
                        email</asp:RegularExpressionValidator>
                </td>
            </tr>
            <tr>
                <td>
                    Enter 5 digit
                </td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                </td>
                <td>
                    <asp:RegularExpressionValidator
ID="RegularExpressionValidator2" runat="server"
                        ControlToValidate="TextBox2" ErrorMessage="Wrong pin
code"
                        ValidationExpression="\d{5}">Wrong pin
code</asp:RegularExpressionValidator>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Button ID="Button1" runat="server" Text="Button"
Width="176px">
                </td>
                <td>
                     </td>
                <td>
                     </td>
            </tr>
        </table>

    </div>
    </form>


</body>
</html>
```

15.7 CUSTOMVALIDATOR CONTROL

If no other validator can help us, we can use **CustomValidator**. It doesn't come with a predefined way of working. We need to write the code for validation purpose. This is very powerful validator as we can create any type of validation as per our requirement. A common use of CustomValidator is when we need to make a database lookup to see if the value is valid. The CustomValidator control allows us to write a method to handle the validation of the value entered.

Most important properties of RegularExpressionValidator are ControltoValidate, OnServerValidate , ErrorMessage and Text. ControltoValidate property is used to

specify the ID of the control on which validation is applied. OnServerValidate specifies the name of user defined function which will be used to validate input. ErrorMessage property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also be displayed in the validation control if the Text property is not set. Text property specifies the message to display when validation fails.

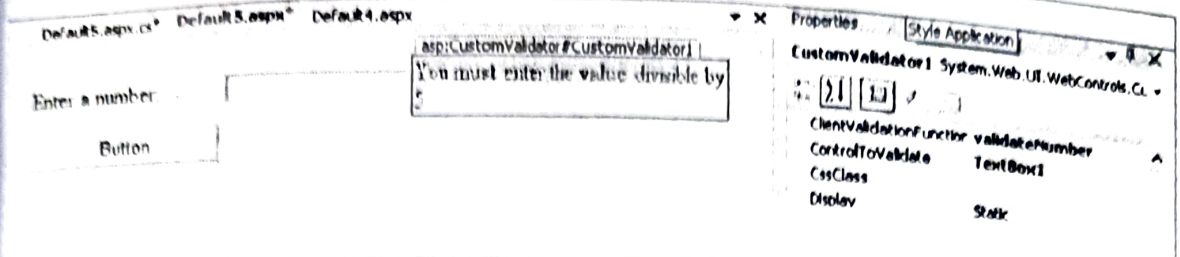
15.7.1 Properties of CustomValidator

Property	Description
BackColor	This property specifies the background color of the CustomValidator control
ClientValidationFunction	This property specifies the name of the client-side validation script function to be executed. The script must be in a language that the browser supports, such as VBscript or Javascript With VB, the function must be in the form: **Sub FunctionName (source, arguments)** With C#, the function must be in the form: **FunctionName (source, arguments)**
ControlToValidate	This property specifies the id of the control to validate
Display	This property specifies the display behavior for the validation control. I can have following values: • None (the control is not displayed. Used to show the error message only in the ValidationSummary control) • Static (the control displays an error message if validation fails. Space is reserved on the page for the message even if the input passes validation. Dynamic (the control displays an error message if validation fails. Spa is not reserved on the page for the message if the input passes validatic
EnableClientScript	This property can have two values True or False. This property specifi whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specif whether the validation control is enabled or not.
ErrorMessage	This property is used to specify the text to be displayed in the ValidationSummary control when validation fails. This text will also displayed in the validation control if the Text property is not set.
ForeColor	This property is used to specify the foreground color of the control.
id	This property is used to specify a unique id for the validation control
IsValid	This property is used to check whether the control to be validated is valid control or not.
OnServerValidate	This property specifies the name of the server-side validation script function to be executed. It specifies the name of user defined functic which will be used to validate input.
runat	This property specifies that the control is a server control. Must be to "server".
Text	This property specifies the message to display when validation fail

Program 15.5 Program to demonstrate the use of CustomValidation Control

Note: In this program, a javascript based user defined function has been created in head section of program. It is used to validate the custom validator control.

Design Window



Output

In this program, a TextBox with ID TextBox1, CustomValidator with ID CustomValidator1 and a Button with ID Button1 have been taken. CustomValidator 1 has been applied to TextBox1 by using ControlToValidate property. CustomValidator1 has been associated with a user defined function named "ValidateNumber" which will check that number should be divisible by 5 by using ClientValidationFunction property. On Clicking Button. If we don't fill a value which is not a multiple of 5 in TextBox, "You must enter the value divisible by 5" message will be displayed at CustomValidator1.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default5.aspx.cs"
Inherits="Default5" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .style1
        {
            width: 61%;
            height: 54px;
        }
        </style>
    <script language = "JavaScript" type = "text/javascript">
                function validateNumber(value, arg)
                {
                      arg.IsValid = (arg.Value%5 == 0);
                }
            </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <table class="style1">
            <tr>
                <td>
                    Enter a number:</td>
                <td style="margin-left: 40px">
                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                </td>
                <td>
                    <asp:CustomValidator ID="CustomValidator1" runat="server"
                        ControlToValidate="TextBox1"
```

```
                    ErrorMessage="You must enter the  value divisible by
5"
ClientValidationFunction="validateNumber"></asp:CustomValidator>
                </td>
            </tr>
            <tr>
                <td style="margin-left: 40px">
                    <asp:Button ID="Button1" runat="server" Text="Button"
                        Width="140px" />
                </td>
                <td>
                     </td>
                <td>
                     </td>
            </tr>
        </table>
        </div>
    </form>
    </body>
</html>
```

15.8 VALIDATIONSUMMARY CONTROL

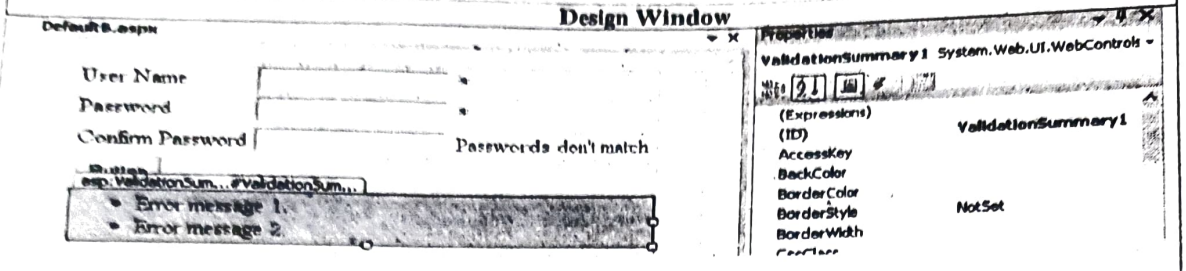
The ValidationSummary control is used to display a summary of all validation errors occurred in a Web page. The error message displayed in this control is specified by the ErrorMessage property of each validation control. If the ErrorMessage property of the validation control is not set, no error message is displayed for that validation control.

15.8.1 Properties of ValidationSummary

Property	Description
DisplayMode	This property specifies the display behavior for the validation control. It can have following values: • BulletList • List • SingleParagraph
EnableClientScript	This property can have two values True or False. This property specifies whether client-side validation is enabled or not.
Enabled	This property can have two values True or False. This property specifies whether the validation control is enabled or not.
ForeColor	This property specifies the fore color of the control
HeaderText	This property specifies a header in the ValidationSummary control
id	This property specifies the unique id for the control
runat	This property specifies that the control is a server control. It must be set to "server"
ShowMessageBox	This property specifies whether the summary should be displayed in a message box or not. Values could be True or False.
ShowSummary	This property specifies whether the ValidationSummary control should be displayed or hidden. Values could be True or False.

Program 15.6 Program to demonstrate the use of Question Bank

Design Window



Output

User Name *

Password

Confirm Password Passwords don't match

Button

- User Name is empty
- Passwords don't match

In this program, three TextBoxes with ID TextBox1, TextBox2, TextBox3 along with RequiredFieldValidators applied on TextBox1 and TextBox2 and CompareValidator applied on TextBox3 to compare values of TextBox2 and TextBox3. ValidationSummary Control with ID ValidationSummary1 and a Button with ID Button1 have been taken. On Clicking Button. If we don't fill user name, password or if password and confirm password don't match, ValidationSummary Control will display errormessages set for All other validators.

ASP.NET Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default6.aspx.cs"
Inherits="Default6" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
    <style type="text/css">
        .style1
        {
            width: 400px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <table align="center" class="style1">
            <tr>
                <td>
                    User Name</td>
                <td>
                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
```

```
                    ControlToValidate="TextBox1" ErrorMessage="User Name
is empty">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td>
                    Password</td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server"
TextMode="Password"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server"
                    ControlToValidate="TextBox2" ErrorMessage="Password
is empty">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td>
                    Confirm Password</td>
                <td>
                    <asp:TextBox ID="TextBox3" runat="server"
TextMode="Password"></asp:TextBox>
                </td>
                <td style="margin-left: 80px">
                    <asp:CompareValidator ID="CompareValidator1"
runat="server"
                    ControlToCompare="TextBox2"
ControlToValidate="TextBox3"
                    ErrorMessage="Passwords don't
match"></asp:CompareValidator>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Button ID="Button1" runat="server" Text="Button" />
                </td>
                <td>
                     </td>
                <td style="margin-left: 80px">
                     </td>
            </tr>
            <tr>
                <td colspan="3">
                    <asp:ValidationSummary ID="ValidationSummary1"
runat="server" />
                </td>
            </tr>
        </table>

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

Question Bank

Q1. What do you mean by validation?

Q2. What are various validation controls provided by ASP.NET.

Q3. Explain RequiredFieldValidator control giving a suitable example.

Q4. Explain CompareValidator control giving a suitable example.

Q5. Explain RangeValidator control giving a suitable example.

Q6. Explain RegularExpressionValidator control giving a suitable example.

Q7. Explain CustomValidator control giving a suitable example.

Q8. Explain ValidationSummary control giving a suitable example.