# UNIT-III Validation Controls

## *<u>Introduction To Validation</u>-

When you create an ASP.Net form than before submitting the form data on the server it's necessary ***to ensure that the user has provided valid data to avoid any erroneous data to be inserted into the database***.

**<u>Definition</u>**- ***Ensuring that data entered by the user into the form is proper and valid as per our business requirements is called Validation***.

OR

***Validation is process to accept only correct information as input from user.***

## <u>Advantages of Validation-</u>

- **<u>Preventing invalid data from being submitted</u>**:

  Validation controls can check for required fields, format and type of data, and other criteria that must be met before the data can be submitted to the server. This helps ensure that the data is accurate and complete and that the application can process it correctly.

- **<u>Improving user experience:</u>**

  When users enter invalid data, they may receive error messages or other feedback that helps them correct their mistakes. This can help prevent frustration and improve the overall user experience.

- **<u>Enhancing security</u>**:

  Validation controls can help prevent malicious code or other attacks that could harm the web application or its users. For example, validation controls can check for potentially dangerous input such as SQL injection or cross-site scripting (XSS) attacks.

- **<u>Meeting business requirements</u>**:

  Many businesses have specific rules and requirements for data entry, such as a minimum or a maximum number of characters or certain formatting conventions. Validation controls can help ensure that these requirements are met**.**

## Types of Validation-

- Client-Side Validation
- Server-Side Validation

1) **Client-side validation**

***Validation done in the browser (i.e. at Client side) before sending the form data to the server using JavaScript, jQuery and VBScript is called client-side validation.***

2) **Server-Side Validation**

***Validation is done at the server level after sending the form data to the server but before entering the data into the database is called server-side validation.***

- Server-Side Validation can be done using Validator Server Controls.

## Validation Controls-

***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**

***IMP Note- Make Following Changes in Web.Config file while using VS 2015 & Next Version***

```
<appSettings>
 <add key ="ValidationSettings:UnobtrusiveValidationMode"
value="None"/>


</appSettings>
```

## 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 message in Validation Summary Control. |
| **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 Example of the control is as given:**

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1"
Display="Dynamic" ErrorMessage="Roll Number is Mandatory"
ForeColor="Red"> Roll Number is Mandatory
</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 Example of the control is as given:**

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="TextBox5" Display="Dynamic"
ErrorMessage="Age Must Be in between 18-25" ForeColor="Red"
MaximumValue="25" MinimumValue="18" Type="Integer"> Age Must
Be in between 18-25</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 **Example** of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox7" ControlToValidate="TextBox6"
Display="Dynamic" ErrorMessage="Both Password Must Be Same"
ForeColor="Red"> Both Password Must Be Same
</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"
   ValidationExpression="\d{10}" Text=”Mobile Number Must be
correct…” >
 </asp:RegularExpressionValidator>
```

## CustomValidator

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

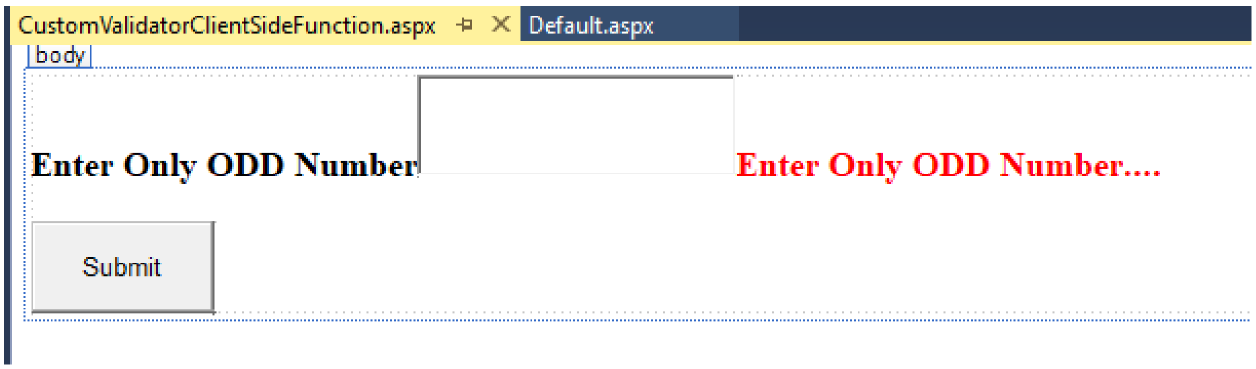
The client side validation is accomplished through the **ClientValidationFunction** property. ***The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.***

***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="CheckOdd" ErrorMessage="CustomValidator">
  </asp:CustomValidator>
```

### Example Client Side Validation Function-



Enter Only ODD Number

Enter Only ODD Number....

### Source code-

```
<head runat="server">
    <script lang="javascript">
        function CheckOdd(sender,args)
        {
            var num =
document.getElementById("TextBox1").value;
            if (num % 2 != 0)
                args.IsValid = true;
            else
```

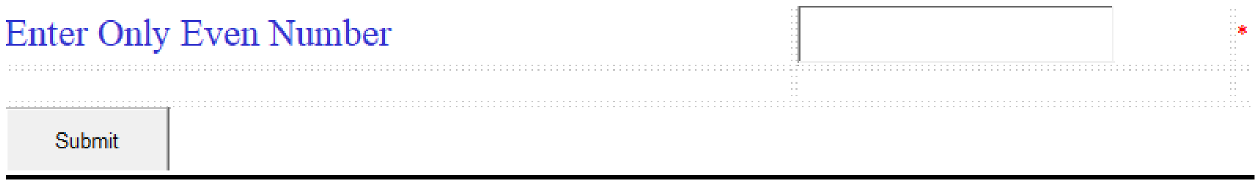
```
                    args.IsValid = false;
            }
        </script>
```

```
  <form id="form1" runat="server">
    <h3>Enter Only ODD Number<asp:TextBox ID="TextBox1"
runat="server" Height="46px" Width="156px">
</asp:TextBox>

<asp:CustomValidator ID="CustomValidator1" runat="server"
ClientValidationFunction="CheckOdd"
ControlToValidate="TextBox1"
Display="Dynamic" ErrorMessage="CustomValidator"
ForeColor="Red">
Enter Only ODD Number....</asp:CustomValidator>

<asp:Button ID="Button1" runat="server" Text="Submit"
Height="47px" Width="94px" />
  </form>
```

## 2.Example Server side Validation Event-



```
<asp:CustomValidator ID="CustomValidator1" runat="server"
ControlToValidate="TextBox8" Display="Dynamic"
ErrorMessage="Number Must Be EVEN Only..." ForeColor="Red"
OnServerValidate="CustomValidator1_ServerValidate">*</asp:Cu
stomValidator>
```

**Page Behind-**

```
protected void CustomValidator1_ServerValidate(object
source, ServerValidateEventArgs args)
    {
        int num = Convert.ToInt32(args.Value);
        if (num % 2 == 0)
            args.IsValid = true;
```

```
                    else
                args.IsValid = false;

    }
```

## ValidationSummary Control

***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:" />
```

Example

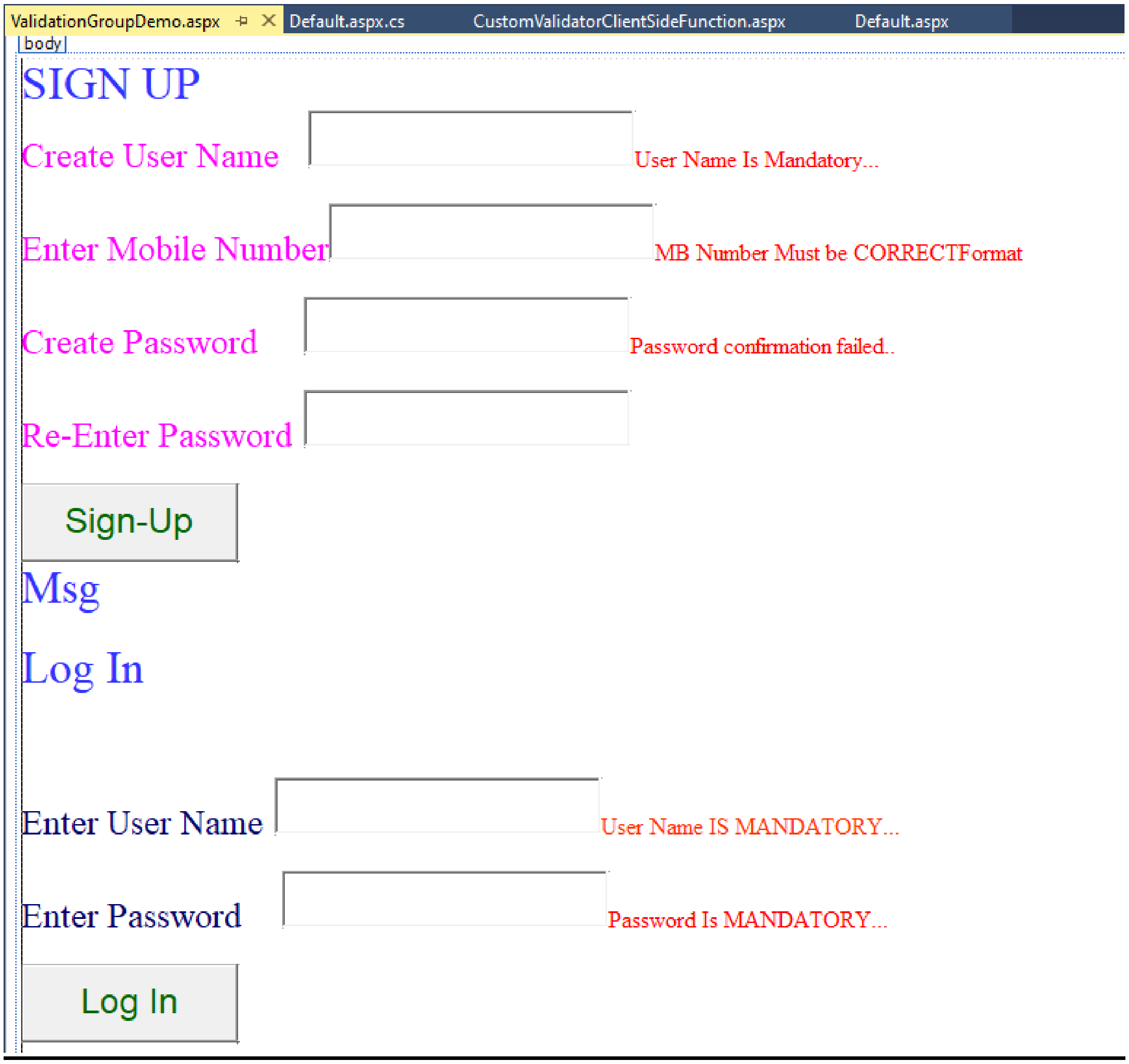
```
<asp:ValidationSummary ID="ValidationSummary1"runat="server"
 Font-Size="X-Large" ForeColor="Red" HeaderText="ALL ERRORS"
 Height="229px" ShowMessageBox="True" Width="1448px" />
```

## 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-**



SIGN UP

Create User Name  User Name Is Mandatory...

Enter Mobile Number  MB Number Must be CORRECTFormat

Create Password  Password confirmation failed..

Re-Enter Password

Sign-Up

Msg

Log In

Enter User Name  User Name IS MANDATORY...

Enter Password  Password Is MANDATORY...

Log In

**Source Code-**

```
<asp:Label ID="Label1" runat="server" Font-Size="XX-Large"
ForeColor="#3333FF" Text="SIGN UP"></asp:Label>


<asp:Label ID="Label2" runat="server" Font-Size="X-Large"
ForeColor="Fuchsia" Text="Create User Name"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server" Height="33px"
Width="216px"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1"
Display="Dynamic"
ErrorMessage="RequiredFieldValidator" ForeColor="Red"
ValidationGroup="signup">User Name Is
Mandatory...</asp:RequiredFieldValidator>


<asp:Label ID="Label5" runat="server" Font-Size="X-Large"
ForeColor="Fuchsia" Text="Enter Mobile Number"></asp:Label>

<asp:TextBox ID="TextBox4" runat="server" Height="33px"
Width="216px"></asp:TextBox>

<asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat="server"
ControlToValidate="TextBox4" Display="Dynamic"
ErrorMessage="RegularExpressionValidator" ForeColor="Red"
ValidationExpression="\d{10}" ValidationGroup="signup">MB
Number Must be CORRECTFormat
</asp:RegularExpressionValidator>


<asp:Label ID="Label3" runat="server" Font-Size="X-Large"
ForeColor="Fuchsia" Text="Create Password"></asp:Label>

<asp:TextBox ID="TextBox2" runat="server" Height="33px"
Width="216px"></asp:TextBox>

<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox3" ControlToValidate="TextBox2"
Display="Dynamic" ErrorMessage="CompareValidator"
ForeColor="Red" ValidationGroup="signup">Password
confirmation failed..</asp:CompareValidator>


<asp:Label ID="Label4" runat="server" Font-Size="X-Large"
ForeColor="Fuchsia" Text="Re-Enter Password"></asp:Label>
```

```
<asp:TextBox ID="TextBox3" runat="server" Height="33px"
Width="216px"></asp:TextBox>


<asp:Button ID="Button1" runat="server" Font-Size="X-Large"
ForeColor="#006600"
Height="54px" Text="Sign-Up" ValidationGroup="signup"
Width="149px" OnClick="Button1_Click" />

<asp:Label ID="Label9" runat="server" Font-Size="XX-Large"
ForeColor="#3333FF" Text="Msg"></asp:Label>

<asp:Label ID="Label6" runat="server" Font-Size="XX-Large"
ForeColor="#3333FF" Text="Log In"></asp:Label>


<asp:Label ID="Label7" runat="server" Font-Size="X-Large"
ForeColor="#000066" Text="Enter User Name"></asp:Label>

<asp:TextBox ID="TextBox5" runat="server" Height="33px"
Width="216px"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server" ControlToValidate="TextBox5"
Display="Dynamic" ErrorMessage="RequiredFieldValidator"
ForeColor="#FF3300" ValidationGroup="login">User Name IS
MANDATORY...</asp:RequiredFieldValidator>


<asp:Label ID="Label8" runat="server" Font-Size="X-Large"
ForeColor="#000066" Text="Enter Password"></asp:Label>

<asp:TextBox ID="TextBox6" runat="server" Height="33px"
Width="216px"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3"
runat="server" ControlToValidate="TextBox6"
```

```
ErrorMessage="RequiredFieldValidator" ForeColor="Red"
ValidationGroup="login">Password Is
MANDATORY...</asp:RequiredFieldValidator>


<asp:Button ID="Button2" runat="server" Font-Size="X-Large"
ForeColor="#006600" Height="54px" Text="Log In"
ValidationGroup="login" Width="149px"
OnClick="Button2_Click" />

<asp:Label ID="Label10" runat="server" Font-Size="XX-Large"
ForeColor="#3333FF" Text="Msg"></asp:Label>
</form>
```

Page Behind Code-

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Label9.Text = "Signed Up Successfully....";
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        Label10.Text = "logged in Successfully....";
    }
```