

<app settings>

<add key="Validation Settings: Unobtrusive

Validation Mode" value="none"/>

</app settings>

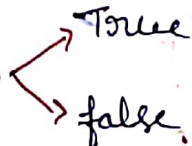
### 3. Compare Validator:-

This control is used to compare the data of one control with the data of other control or with fixed value. If user entered data in control to validate matches with the data of control to compare or with value to compare property. Then compare validator will return true otherwise compare validator will return false.

### \* Properties With Compare Validator:-

1. Control to Validate

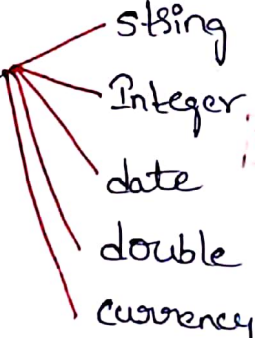
2. Error message

3. Set focus on Error 

4. Text

5. Validation Group

6. Control to compare

7. Type 

8. Value to compare

## Control to Compare:-

→ Used to set or get the id of the Control whose data is compared with the data of Control to Validate.

→ If the data present in Control to Validate matches with the data of Control to Compare. Then Compare Validate. Will return true otherwise Compare Validate will return false.

## Type:-

→ Used to set or get the required data type Value with which the data is to be compared.

## Value to compare:-

→ Used to set or get the fixed value with which data in Control to Validate is to be compared.

## \* Example to implement Compare Validator:-

Enter <sup>First</sup> Name

Enter Last Name

Enter username

Enter password

Retype password

Select Gender • Male • female

Select Qualification

Select Skill 

CH

C++

Java

Python

password donot Match  
password is mandatory.

→ Create a New Webpage design the Webpage Set the following properties

(i) Required field Validator :-

→ Control to Validate : txt password

→ Error Message : password is Mandatory

(ii) Compare Validator :-

→ Control to Validate : txt password

→ Control to Compare : txt Retype password

→ Error message : passwords do not Match.

→ Write the following code :-

```
Protected void btnInsert_Click(object sender, EventArgs)
{
    lblDisplay.Text = "User Created";
}
```

**Range Validator :-**

→ This Control is used to Implemented Validation to check the data is within the range or not. If user Entered data is within the scope of the set value range validation will return true to the Webpage. Otherwise range validation will return false to the Webpage.

**\* Properties With Range Validator :-**

1. Control to Validate

2. Error Message

3. Set focus on Error



4. Text

5. Validation Group

6. Type

- string
- Integer
- date
- double
- currency

7. Maximum Value

8. Minimum Value

**Maximum Value :-**

used to set or get the highest value upto  
where the range validation is required to check

**Minimum Value :-**

used to check the set or get the least value  
that is to be checked from using range validation

**\* Example With Range Validation :-**

Enter first Name

Enter last Name

Enter user Name

Enter password

Retype password

Select Gender ☐ ☐

Enter Age

Range  
Range Validation

Select Qualification

Insert

## \* Properties to set for Range Validator:-

1. Control to Validate : txtAge
2. Minimum Value : 15
3. Maximum Value : 50
4. Type : Integer
5. Error's message : Age Value should be from 15 to 50 only

## \* Example With Validation Group Property:-

Enter User Name  User Name is Mandatory  
Enter password  Pass Word is Mandatory

Enter Hash Ticket Number

## \* Properties to set Required field Validation 1 (RFV1)

1. Control to Validate : txtUserName
2. Error's message : User Name is Mandatory

## RFV2:-

1. Control to Validate : txtpassword
2. Error's Message : password is Mandatory

RFV3:-

1. Control to Validate : txtNum
2. Error Message : Hall ticket No. is Mandatory.

Code

\* Write the following code for .cs file.

```
Protected void btnlogin_Click(object sender, EventArgs e)
{
    lbl display.Text = "Login successful"
}
```

```
Protected void btnsubmit_Click ( object sender, EventArgs e )
{
    lbl Result.Text = "Result is pass."
}
```

→ In the above example if a user entered login-Name password, click on login button, Request will not be submitted to server rather it is asked to enter Hall ticket Number, similarly if any user's enters hall ticket Number and clicks on submit button request will not be submitted to the server. rather it will be prompted to ask to enter user's Name and password.

→ To overcome this problem then we use Validation Group property.

→ from the controls txtName, txtpassword, btnlogin, Rfv, & Rfv<sub>2</sub> set the Validation Group Property as validate login, similarly for the controls,



txtHTNum, btnSubmit & RefV2 set the Validation Group property as validate HTNum, run the Webpage & check.

### 5. Custom Validation:-

This Control is used to Implement the Validations if Existing Validation controls do not need the user requirements.

#### \* Properties With Custom Validation:-

1. Control to Validate
2. Error Message
3. Set focus on Error
4. Text
5. Validation Group
6. Client Validation Function

### 6. Client Validation Function:-

This property is used to set the javascript function Name using Which We Will Implement the Custom Validation.

#### \* Example With Custom Validation:-

→ In the above diagram Enter password beside you will take CV1 (Custom Validator 1)

## \* Properties to Set Custom Validator

1. Control to Validate : txtpassword
2. Error Message : Password should be minimum 6 characters.

→ Go to source Write the following code for JavaScript function.

```
</style>  
<script type="text/javascript">  
    Function validate password (sender, args)  
    {  
        if (args.value.length < 6)  
            args.IsValid = false;  
        else  
            args.IsValid = true;  
    }  
</script>
```



Enter First Name

Enter Last Name

Enter User Name

Enter Password

Retype Password

Select Gender ☐ Male ☐ Female

Enter Age

Select Qualification

C  
C++  
C#  
ASP.Net

Select Skill

Insert

Passwords Do not Match Password is Mandatory

asp:CustomValidator#CustomValidator1

Password Sholud be Minimum 6 Characters

Age Value Should be From 15 to 50 Only

Custom validator 1

Implementation of password

```
15  
16 </style>  
17 <script type="text/javascript">  
18     function ValidatePassword(sender, args)  
19     {  
20         if (args.Value.length < 6)  
21             args.IsValid = false;  
22         else  
23             args.IsValid = true;  
24     }  
25 </script>  
26 </head>  
27  
28 <body>  
29 <form id="form1" runat="server">
```



## \* Properties to set Custom Validator 1.

1. Control to Validate : txtpassword
2. Error Message : Password should be minimum 6 characters

→ Go to source Write the following code for

Javascript function.

```
</style>
<script type="text/javascript">
    Function validate password (sender, args)
    {
        if (args.value.length < 6)
            args.IsValid = false;
        else
            args.IsValid = true;
    }
</script>
```

## \* Example With Causes Validation Property:-

Enter Username

Enter password

## \* Properties to set Causes Validation property:-

1. Rfvy ~~user name~~ is ~~man~~ Control to Validate : txtName
2. Error message : username is Mandatory



1. **Run Control to Validate :** txt-password

2. **Error Message :** password is mandatory

→ link button1 property is postbackURL :

~/forgot password.aspx

→ Run the application to check when user clicks on forgot password link then it will also be asked to enter username and password which should not be the case. To overcome this we use CausesValidation property.

→ Select linkbutton1 set CausesValidation property to false. Now run the webpage and check when user clicks on forgot password link it will not prompt to enter username & password, rather it will redirect to the requested page.

### Validation Summary Control:

This control is used to display the error messages of all validation controls of the web page that return false.

### Example With Validation Summary Control

Enter username

Enter password

Login

Password

lbl display