

# Validation Control in ASP.NET



# Validation

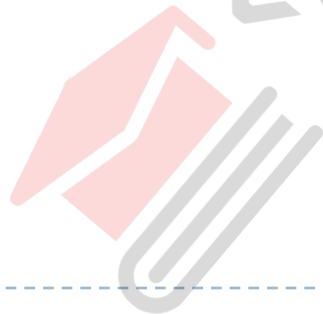
---

## **Validation:- Valid data**

- ▶ Check User input
- ▶ Against a set of rules

## **Validation Types:-**

- ▶ Client side validation
- ▶ Server side validation



# Client Side Validation

---

- ▶ Client side validation is something that will happen on users' browser.
- ▶ The validation will occur before the data gets posted back to server.
- ▶ it gives fast response and from the developers' point of view, it saves valuable resources of server.
- ▶ JavaScript is most widely used to perform client side validation.
- ▶ Now Microsoft is also embracing jQuery in its current versions so perhaps JavaScript and/or JQuery should be the right thing to use for client side validation.
- ▶ ASP.NET also provides some validation controls to perform client side validation which could help the developers in putting client side validation in place without writing a lot of code.

# Server Side Validation

---

- ▶ Server side validation occurs at server.
- ▶ The benefit of having server side validation is that if the user somehow bypasses the client side validation (accidentally or deliberately), then we can catch the problem on the server side.
- ▶ Server side validation is done by writing our custom logic for validating all the input.
- ▶ ASP.NET also provides us some controls which will facilitate the server side validation and provides a framework for the developers to do the same.
- ▶ **NOTE:** Web developer may choose to go with any one type of validation but usually it is a good idea to have client side validation and same validation on server side too.

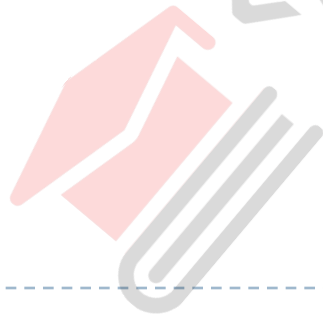
- 
- ▶ The validation Control classes are present in `System.Web.UI.WebControls` namespace
  - ▶ They are used to validate user input. if user data does not pass validation then error message is displayed
  - ▶ By default page validation is done when any type of buttons like `Button`, `ImageButton`, `LinkButton` is processed
  - ▶ Every button has `CausesValidation` property, it can be set to true or false. Its default value is true.
  - ▶ ASP.NET provides 6 built in validation controls. Each control performs a specific type of validation
-

# Validation Controls in ASP.NET

---

The validation controls provided by ASP.NET are:

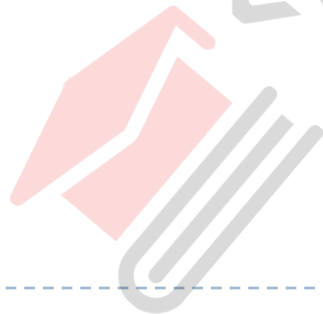
- ▶ **RequiredFieldValidator**
- ▶ **CompareValidator**
- ▶ **RangeValidator**
- ▶ **RegularExpressionValidator**
- ▶ **CustomValidator**
- ▶ **ValidationSummary**



---

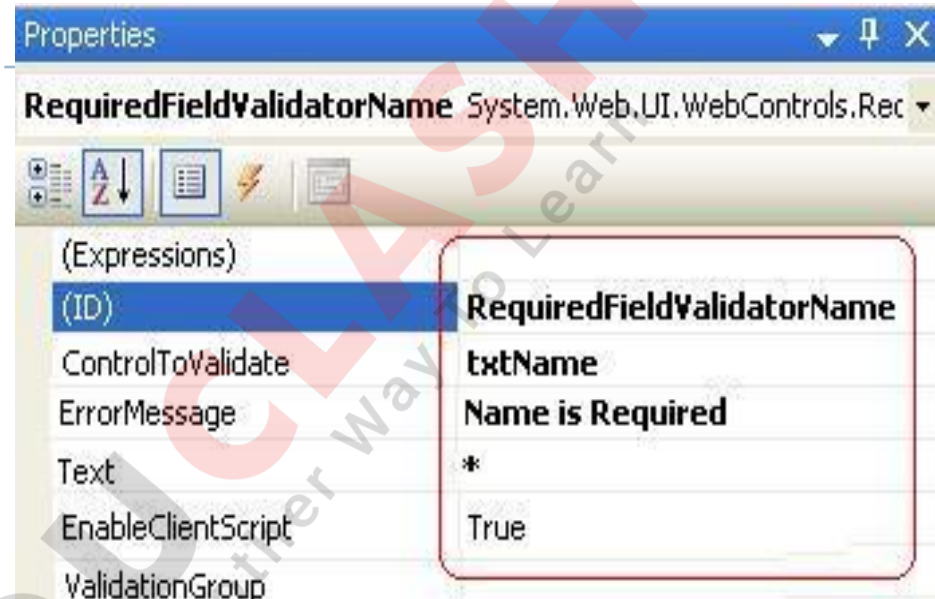
some properties common to all these controls.

- ▶ **ControlToValidate:** ID of the control which should be validated by this control.
- ▶ **ErrorMessage:** This message will be displayed to the user when validation fails. This should always be something which user can read and act upon.
- ▶ **Text:** This string will be visible where the validation control is placed. Usually it is a good idea to set it to "\*".



# RequiredFieldValidator

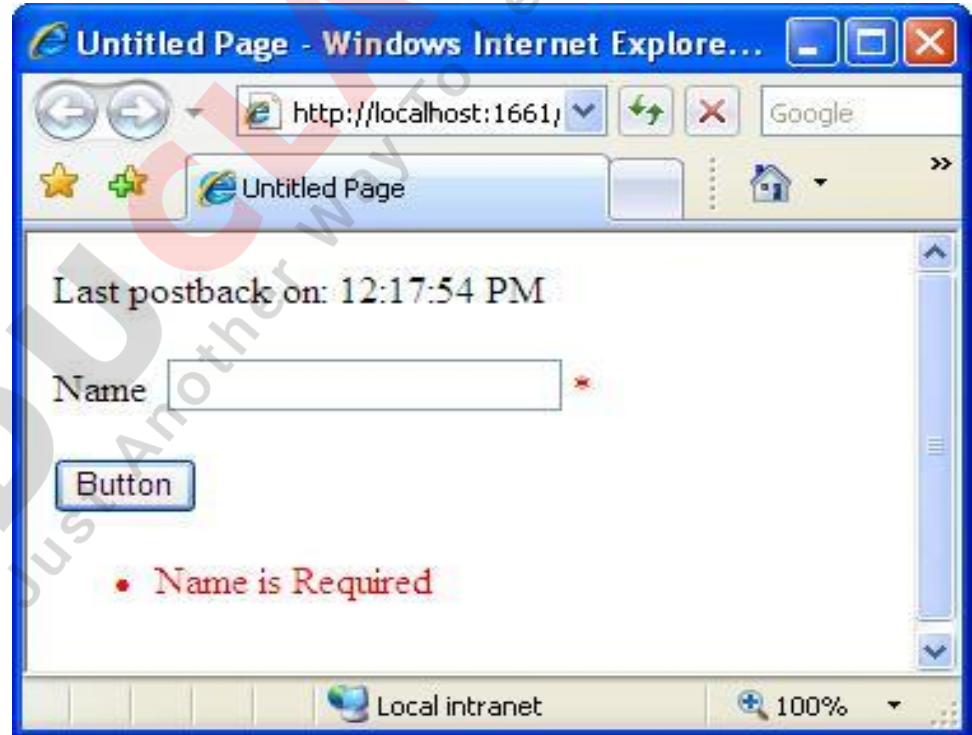
- ▶ This validation control will be used when we are mandating the user input for any particular field.
- ▶ Let's say we have a simple form with name field and we don't want this to be empty.
- ▶ so what we can do is add **aRequiredFieldValidator** to the page, set the **ControlToValidate** to the ID of the name input field, set the error message property.



```
<asp:RequiredFieldValidator ID="RequiredFieldValidator" runat="server" Text="*" ErrorMessage="Name is required" ControlToValidate="TextBox1"></asp:RequiredFieldValidator>
```



- ▶ When we try to do a postback without entering name, then the postback will not happen and the "\*" will be displayed in place to our validation control.
- ▶ The ValidationSummary we already added will display the full error message.
- ▶ If we need to change this to validation in server side, then we will have to set the EnableClientScript property of the validation control to false.



- ▶ Validation groups allow you to organize validation controls on a page as a set. Each validation group can perform validation independently from other validation groups on the page.

```
<asp:TextBox ID="TextBox1" Runat="server"
  ValidationGroup="Second"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" Runat="server"
  ValidationGroup="First"
  ErrorMessage="TextBox1 should not be blank" ControlToValidate="TextBox1">
</asp:RequiredFieldValidator>
<asp:Button ID="Submit1" Runat="server" ValidationGroup="First" Text="Submit 1"
 />
```

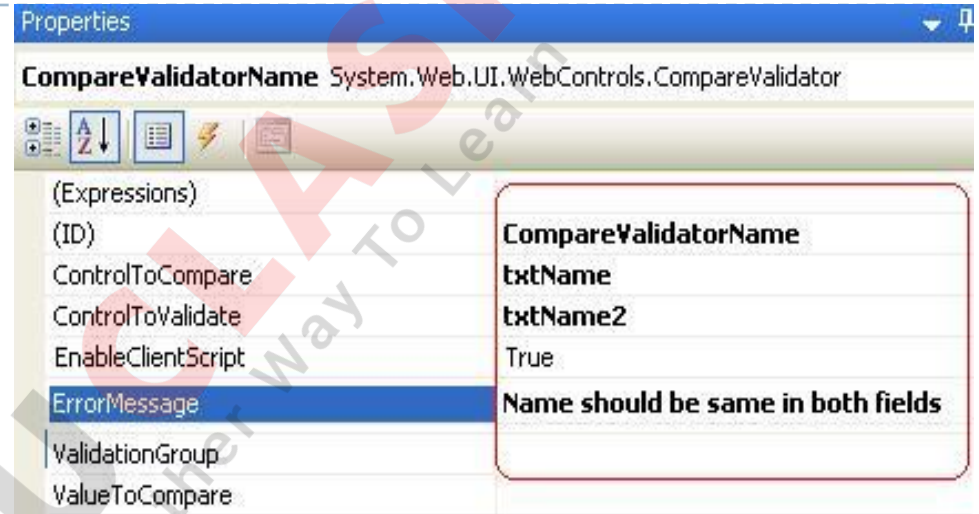
```
<asp:TextBox ID="TextBox3" Runat="server"
  ValidationGroup="Second"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" Runat="server"
  ErrorMessage="TextBox3 should not be blank"
  ControlToValidate="TextBox3" ValidationGroup="Second">
</asp:RequiredFieldValidator>
```

```
<asp:Button ID="Submit2" Runat="server" ValidationGroup="Second" Text="Submit
2" />
```

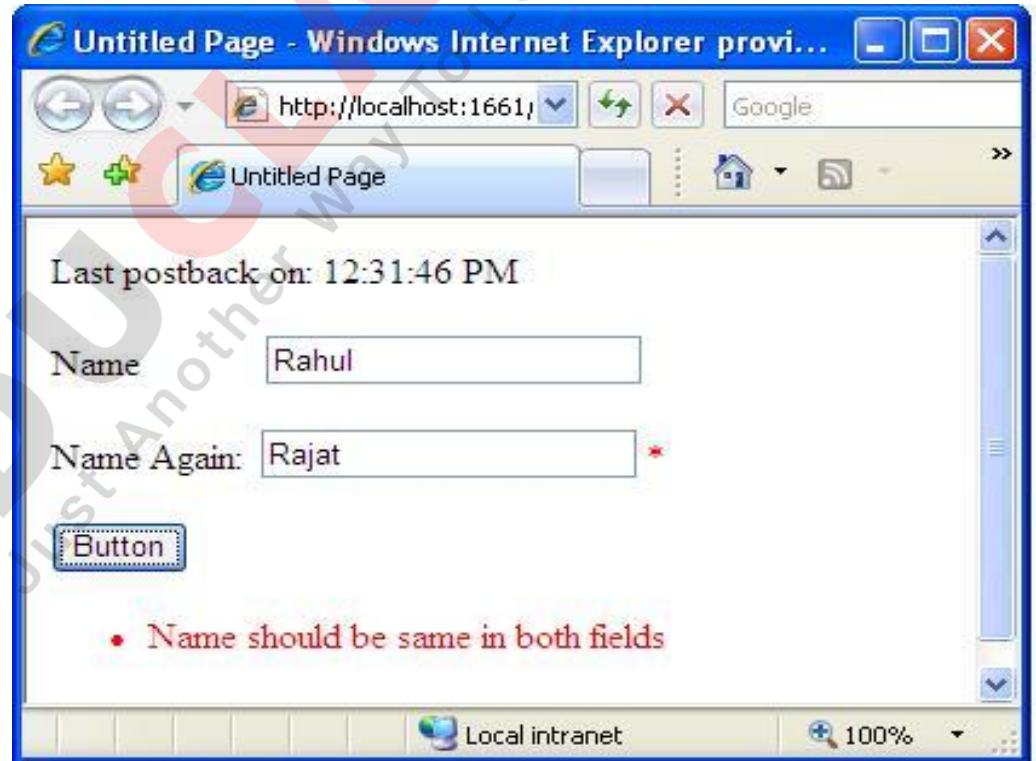
# CompareValidator

- ▶ This control will compare the value of its **ControlToValidate** with **ControlToCompare**.
- ▶ It uses the comparison operators to do the same..
- ▶ Now suppose you want the same name to be entered twice (it would be password) then we can have a **CompareValidator** in place with the following properties:
- ▶ In case we want this control to use some predefined value instead of value in some control, then we can use **ValueToCompare** property.



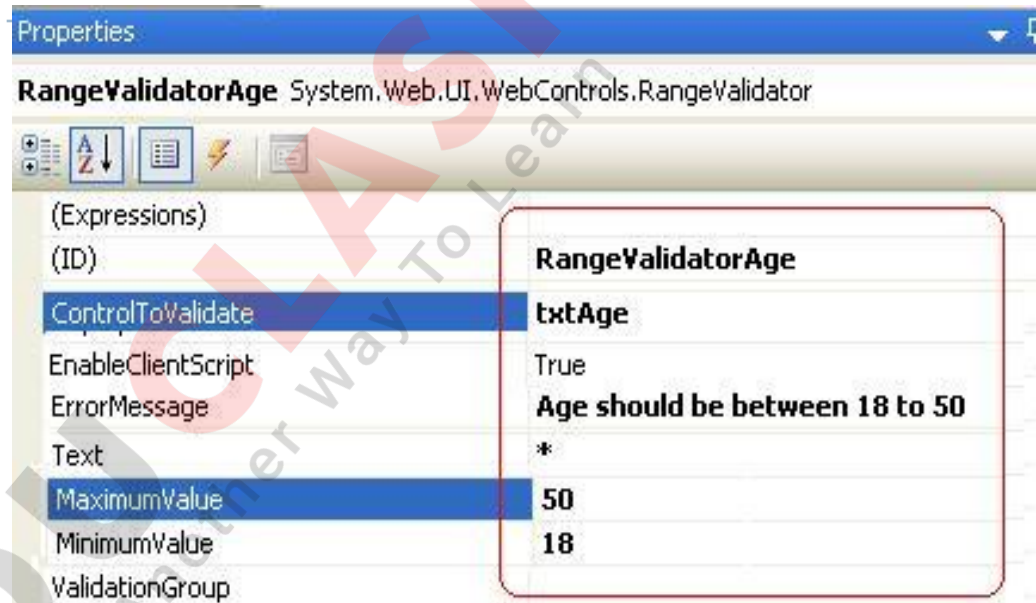
```
<asp:CompareValidator ID="CompareValidator1"
runat="server" ErrorMessage="Both Names must
match"
ControlToCompare="txtName1"
ControlToValidate="txtName2"></asp:CompareVal
idator>
```

- ▶ If we look at the timestamp we added on the page, we can verify that the validation is happening on client side and no postback is happening to the server.
- ▶ If we need to change this to do this validation in server side, then we will have to set the `EnableClientScript` property of the validation control to false.



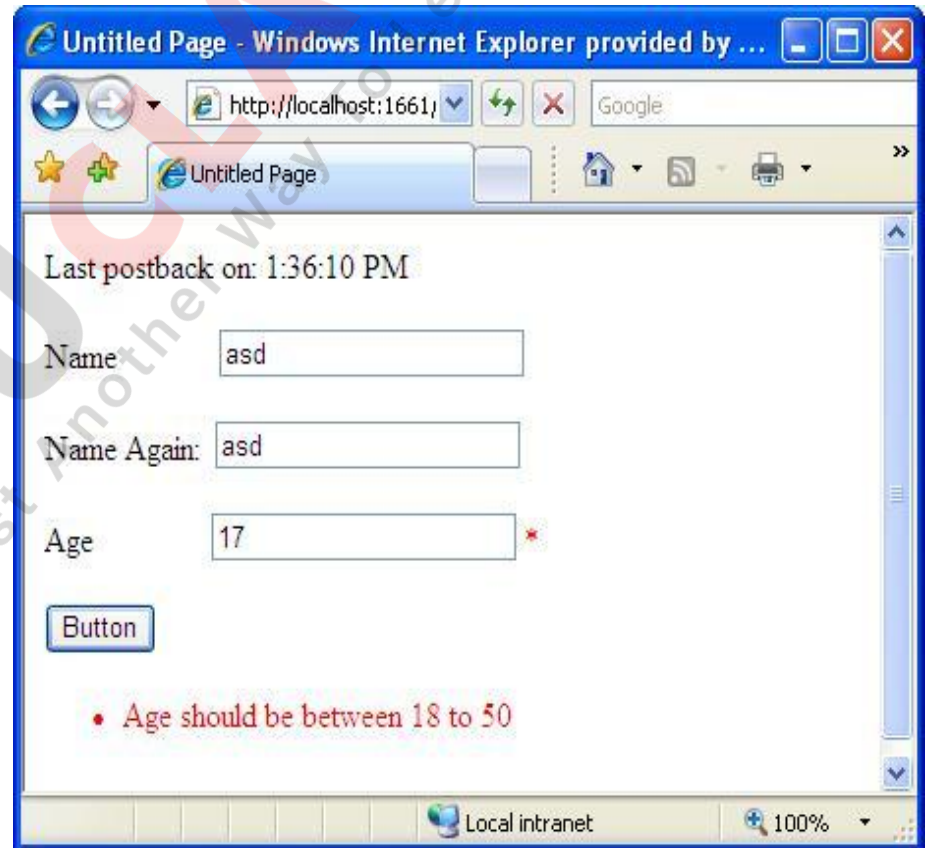
# RangeValidator

- ▶ In scenarios where we want to ensure that the value entered by the user is in some predefined range, we can use this control.
- ▶ Let us try to add this control on our page and use this to validate the age of the user.
- ▶ We are saying the valid age is between 18 to 50.



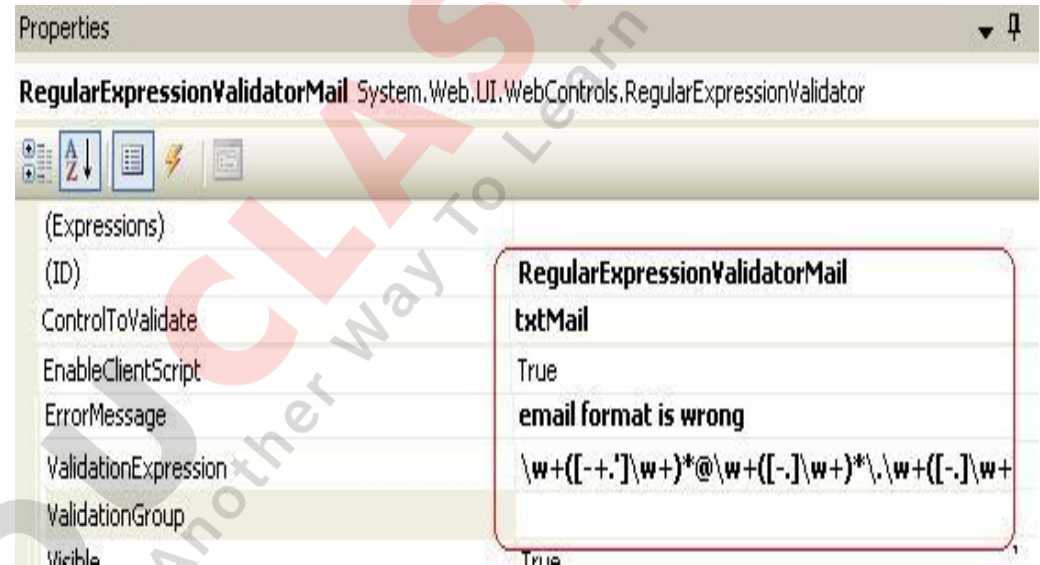
```
<asp:RangeValidator ID="RangeValidator" runat="server"
ErrorMessage="Age should be between 18 to 50"
ControlToValidate="txtAge" MaximumValue="50"
MinimumValue="18" Type="Integer">
</asp:RangeValidator>
```

- ▶ If we look at the timestamp we added on the page, we can verify that the validation is happening on the client side and no postback is happening to the server.
- ▶ If we need to change this to do this validation in server side, then we will have to set the `EnableClientScript` property of the validation control to false.

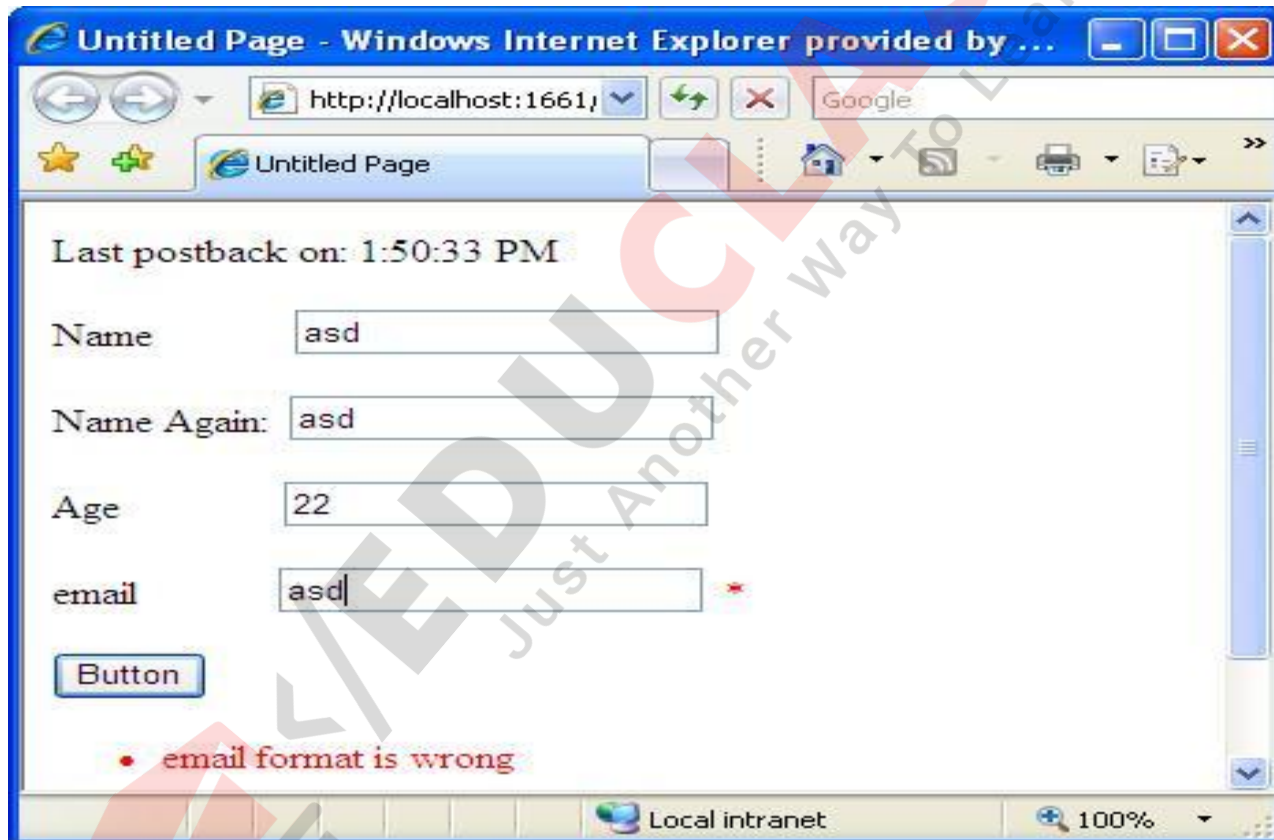


# Regular Expression Validator

- ▶ RegularExpressionValidator comes in handy when we want input data to be in some specific format.
- ▶ We will be using theRegularExpressionValidator for validating the format of email id.



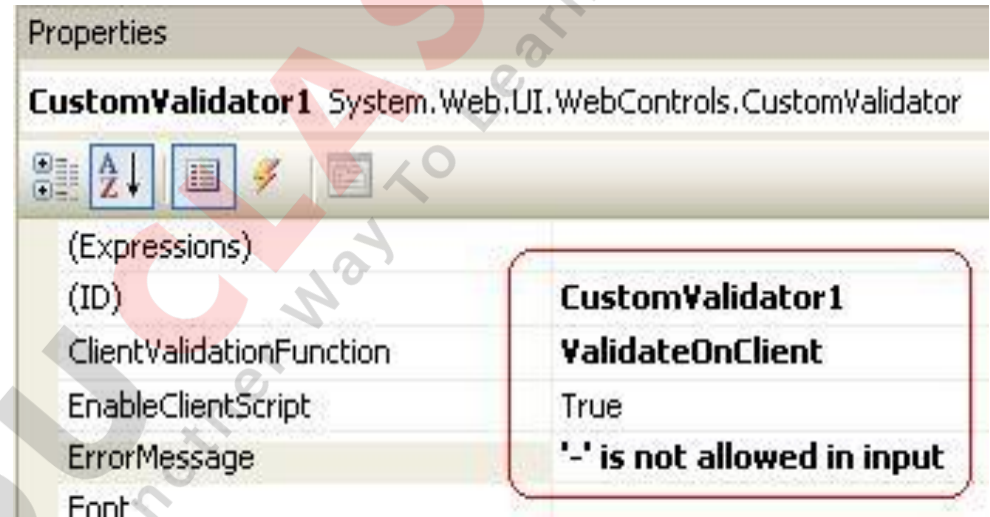
```
<asp:RegularExpressionValidator ID="RegularExpressionValidatorMail" runat="server" ErrorMessage="Email format is wrong" ControlToValidate="txtMail" ValidationExpression="\w+([-+.'\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*">
```





# CustomValidator

- ▶ If with all these validation controls provided by ASP.NET, we still find ourselves a scenario where we need customized validation behavior, we can use the CustomValidator Control.
- ▶ What we will do is we will check for '-' character in user input and reject the input if '-' is present in any field



```
<asp:CustomValidator ID="CustomValidator1"
runat="server" OnServerValidate="UserCustomValidate"ControlToValidate="TextBox1"
ErrorMessage="- is not allowed in input" SetFocusOnError="True"></asp:CustomValidator>
```

- ▶ Now let us look at the client side JavaScript we had to write to provide our custom validation behavior:

and when we run the page:

```
function ValidateOnClient(source,  
arguments)
```

```
{
```

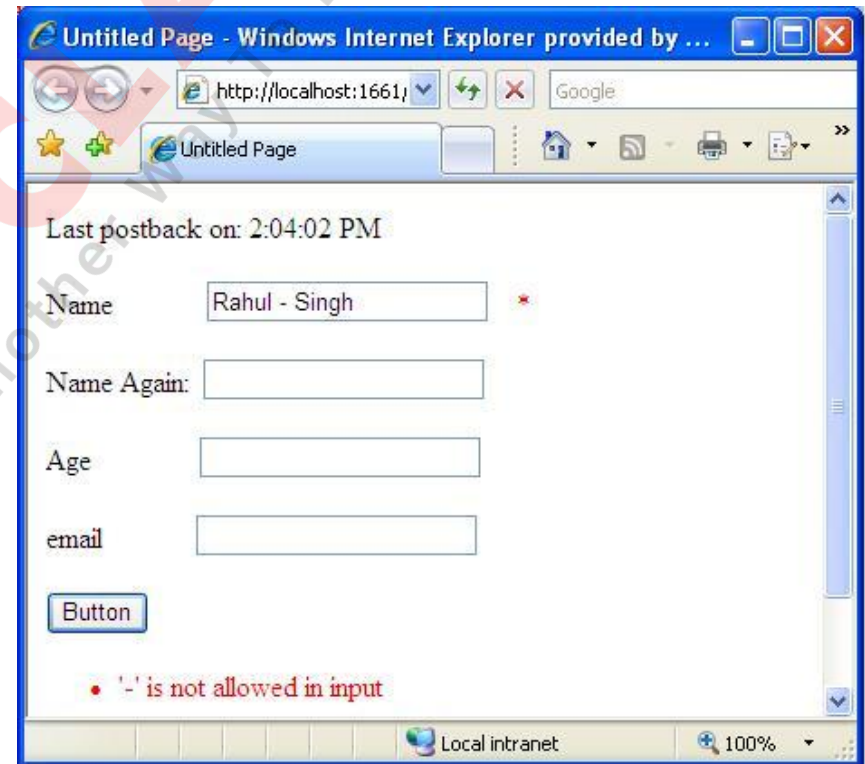
```
if( arguments.Value.search('-') != -  
1)
```

```
{
```

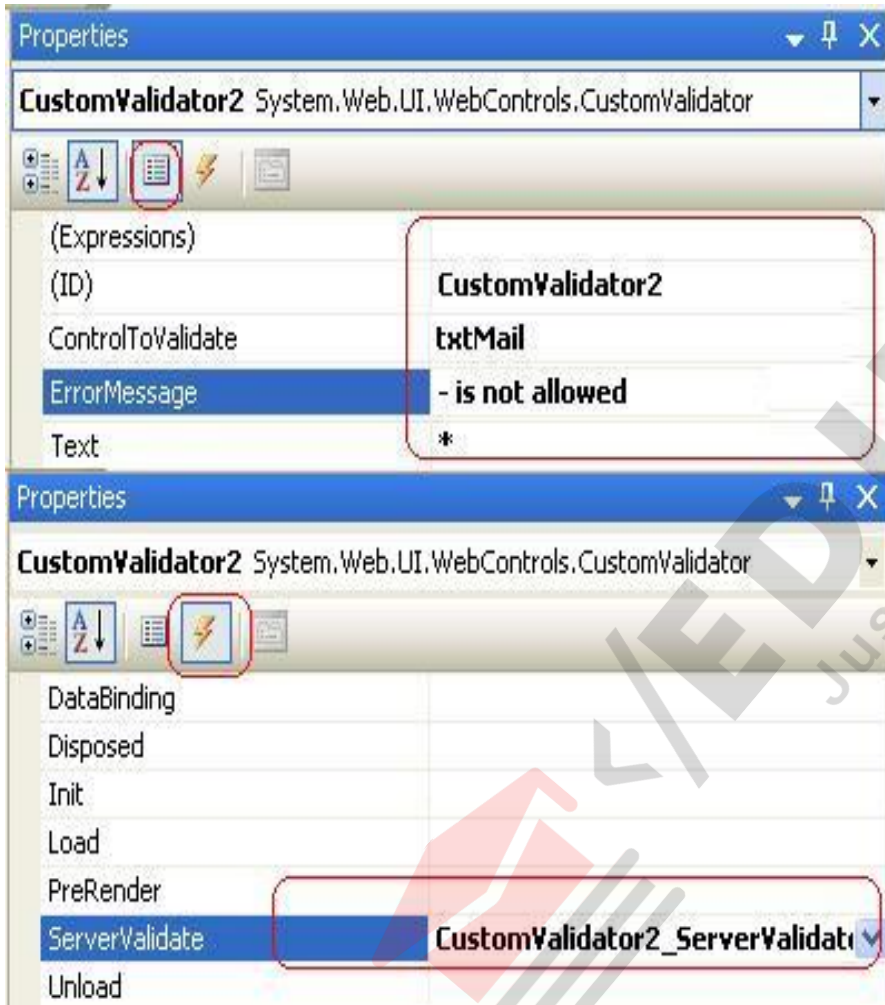
```
arguments.IsValid = false;
```

```
}
```

```
}
```



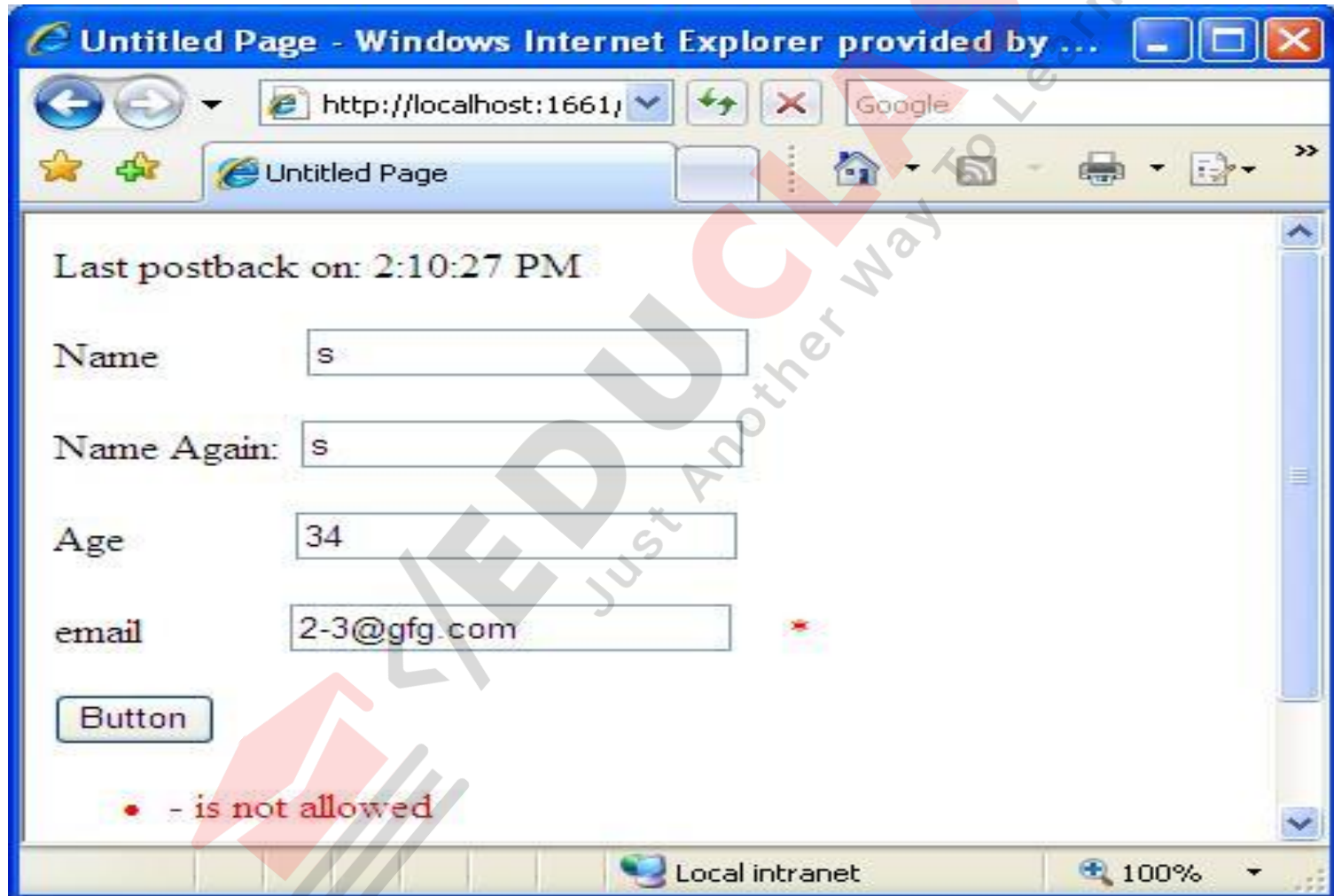
Now let us do the same by having the validation done at server side. We will do that for the email id field.



protected void  
CustomValidator2\_ServerValidate  
(object source,  
ServerValidateEventArgs args)

```
{  
    string value = args.Value;  
    if (value.Contains("-"))  
    {  
        args.IsValid = false;  
    }  
}
```

When we enter invalid data in email id field, the validation error will be displayed after postback.



# ValidationSummary

---

- ▶ The ValidationSummary control is reporting control, which is used by the other validation controls on a page.
- ▶ You can use this validation control to consolidate errors reporting for all the validation errors that occur on a page instead of leaving this up to each and every individual validation control.
- ▶ The validation summary control will collect all the error messages of all the non-valid controls and put them in a tidy list.
- ▶ `<asp:ValidationSummary ID="ValidationSummary1" runat="server" style="top: 390px; left: 44px; position: absolute; height: 38px; width: 625px" />`
- ▶ Both ErrorMessage and Text properties are used to display error messages.

Enter your name:



Enter Password:



Confirm Password:



Enter your Age:

- Confirm Password mandatory & should match password
- Name is required
- Password mandatory