Visual Training

Of

Microsoft Visual Studio2015

Web based Applications using ASP.Net with MS Visual C-Sharp

Visual Training

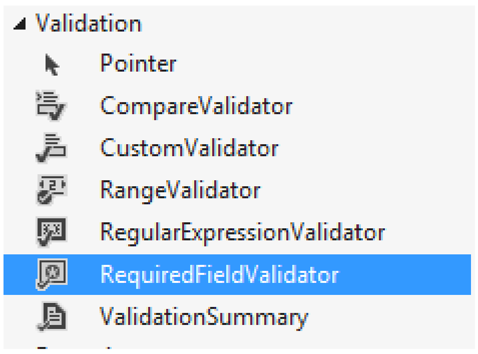
Of

Microsoft Visual Studio2015

Web based Applications using ASP.Net with MS Visual C-Sharp

Chapter No.5

Validating User Data



DAY5

5.1 Web Form Validation

Whenever we have an application that expects user input, then it becomes important to ensure the validity of the data input by the user. We might have scenarios when some data is mandatory for the user to enter. There are scenarios when the user data has to be in some particular format example email ID. There could be scenarios when we want the data to be in some range example date input.

So for all the above mentioned scenarios, if we take the user input without validation, then chances are that we will end up having wrong data with us (perhaps in database). If it is a bad day for us then possibly our application might also end up behaving in an unexpected manner and even crash on us (like if we try to convert a non numeric `string` to `int`). Worst case scenario, the user will use the input field to perform SQL injection and cause serious damage to our database. So it is always a good idea to have validation in place whenever we are taking input from the user.

5.2 Types of Validation

There are two ways we can perform validation:

- 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 is a good idea to have client side validation as the user gets to know what needs to be changed immediately, i.e., no trips to servers are made. So from the users' point of view, it gives him 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. From decades, developers have been using JavaScript for client side validation. It is always a good idea to have knowledge of JavaScript as it gives us full control over 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.

JavaScript provides full control to the developer on how client side validation should happen but developers will have to write the validation code themselves. 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. These controls will also use JavaSscript underneath to perform validations. We will see these controls in a moment.

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. So having server side validation provides more security and ensures that no invalid data gets processed by the application.

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. It sure takes some resources of server to validate the already validated data (on client side) but it ensures security and that is always good.*

Ever since the first dynamic website was created, form validation has been an important subject. Getting user input through the system requires much attention, and with previous serverside scripting languages, there were no standard way of dealing with form validation. For that reason, clientside scripting has been used in many cases, but as you probably know, clientside scripting is not bulletproof - it can easily be bypassed or simply turned off. With ASP.NET, webdevelopers were finally given a good, standard set of tools for validating forms. The validation controls of ASP.NET is based on both clientside and serverside technology, which brings the best from both worlds, to make sure that validating a form is as easy as ever.

5.3 ASP.Net Validation Controls

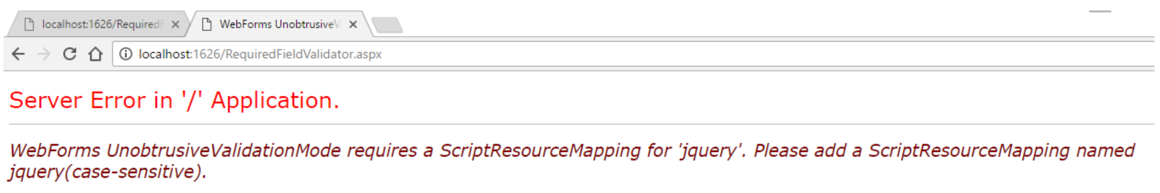
In ASP.NET there are several different validation controls for different purposes. Here, we will show examples of each of them. They all work in different ways, but essentially they do the same - they make sure that a form field has been properly filled by the user.

ASP.NET provides the following validation controls:

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

Type of validation	**Control to use**	**Description**
Required entry	RequiredFieldValidator	Ensures that the user does not skip an entry
Comparison to a value	CompareValidator	Compares a user's entry against a constant value, against the value of another control (using a comparison operator such as less than, equal, or greater than), or for a specific data type.
Range checking	RangeValidator	Checks that a user's entry is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates. For details
Pattern matching	RegularExpressionValidator	Checks that the entry matches a pattern defined by a regular expression. This type of validation enables you to check for predictable sequences of characters, such as those in e-mail addresses, telephone numbers, postal codes, and so on.
User-defined	CustomValidator	Checks the user's entry using validation logic that you write yourself. This type of validation enables you to check for values derived at run time. For details,

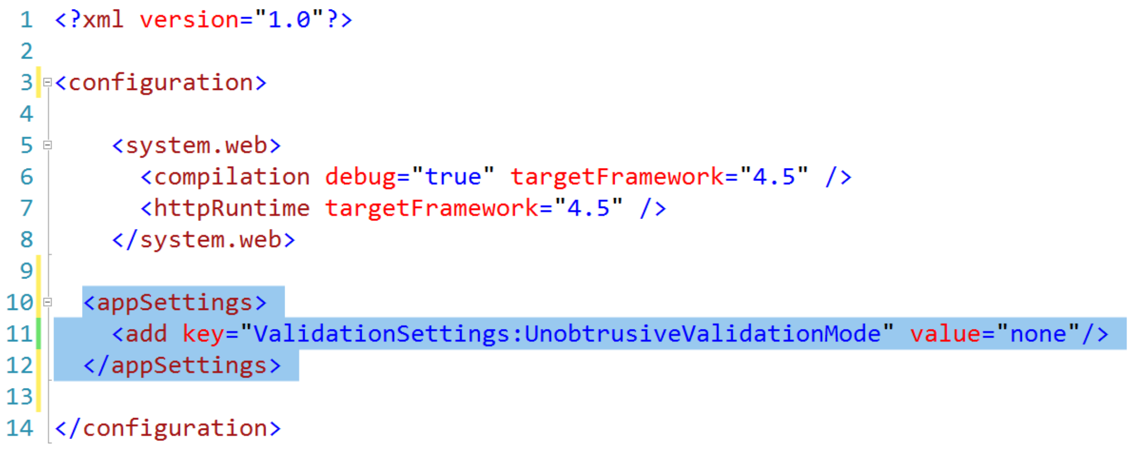
While using these validation controls, sometime the following error is return at run time,



Server Error in '/' Application.

WebForms UnobtrusiveValidationMode requires a ScriptResourceMapping for 'jquery'. Please add a ScriptResourceMapping named jquery(case-sensitive).

To overcome this error, do the following settings in the **web.config** file, as the text is highlighted in the next figure



```
<?xml version="1.0"?>

<configuration>

    <system.web>
      <compilation debug="true" targetFramework="4.5" />
      <httpRuntime targetFramework="4.5" />
    </system.web>

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

</configuration>
```

All the validation controls share some common properties while there are some properties which are specific to each of these validation controls

Property Name	**Value**
ControlToValidate	Specifies the Control name to be validated
SetFocusOnError	Specifies, Will focus be in the control when validation is failed
Display	Specifies will the error message display when validation is failed dynamically or not
ErrorMessage	Specifies the Message which will be displayed when validation is failed

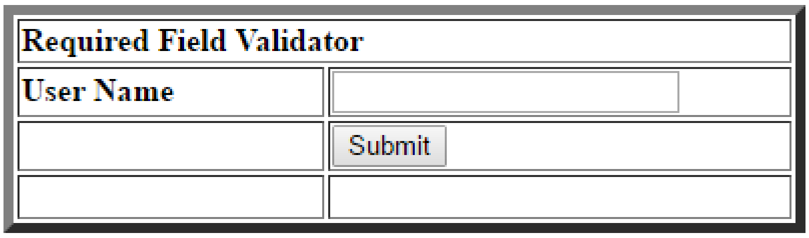
- **RequiredFieldValidator Control**

The **RequiredFieldValidator** is actually very simple, and yet very useful. You can use it to make sure that the user has entered something in a TextBox control.

For example,

- Add a RequiredFieldValidator on an ASP.Net page.
- Add a TextBox to validate, rename the TextBox to txtUserName
- Add a button to submit the form.

The designed Web form is shown here,

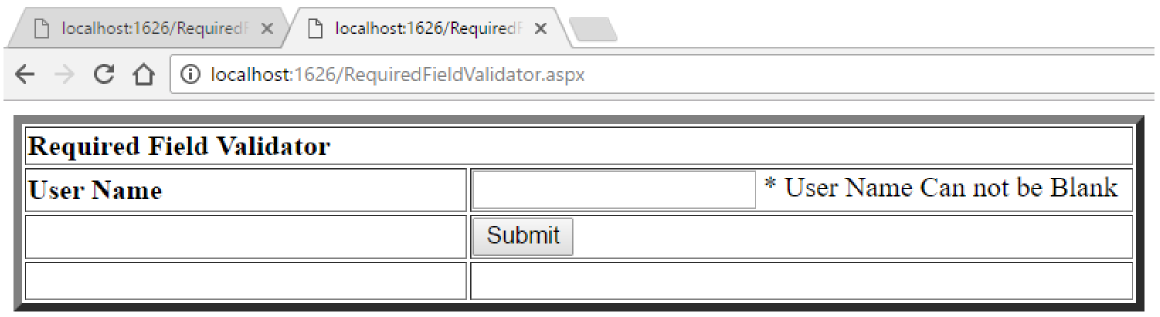


Required Field Validator	
User Name	
	Submit

Now, Drag and drop the RequiredFieldValidator from the Validation Tab in ToolBox and set the properties as shown below

Property Name	**Value**
ControlToValidate	txtUserName
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	*User Name Can not be blank

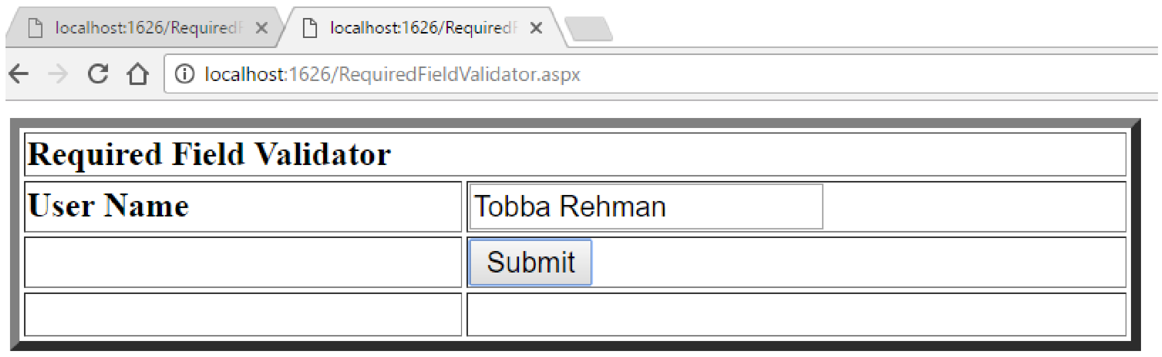
Run the web site, and press the Submit button while the leaving the textbox empty, you will see that validation check is performed showing the error messages,



localhost:1626/RequiredFieldValidator.aspx

Required Field Validator	
User Name	* User Name Can not be Blank
	Submit

Add the User Name in the textbox and you will see that error message will be disappear.



localhost:1626/RequiredFieldValidator.aspx

Required Field Validator	
User Name	Tobba Rehman
	Submit

- **RangeValidator Control**

The **RangeValidator** does exactly what the name implies; it makes sure that the user input is within a specified range. You can use it to validate both numbers, strings and dates, which can make it useful in a bunch of cases. Since we validated numbers the last time, we will try with a date this time.

The date format might seem a bit weird to you if you're not from Europe, where we use dd-mm-yy. You can just change it if it doesn't fit the date format on the machine you're working on. Now, try running the website, and enter a date in our new TextBox. It's only valid if the date is within 2006, and a cool side effect is that the date is also checked for validity.

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

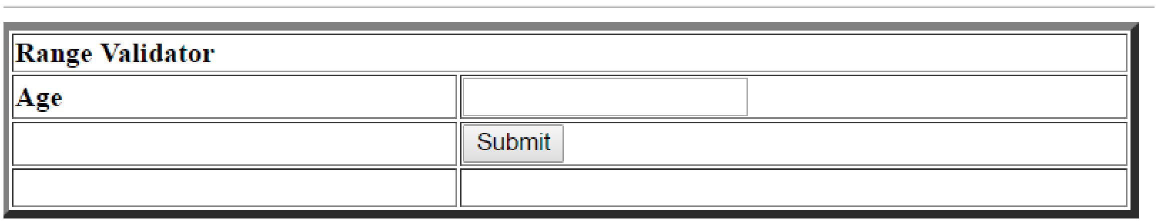
In addition to the four common properties, developer has to specify the following three specific properties for RangeValidator

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.

For example,

- Add a RangeValidator on an ASP.Net page.
- Add a TextBox to validate, rename the TextBox to txtAge
- Add a button to submit the form.

The designed Web form is shown here,

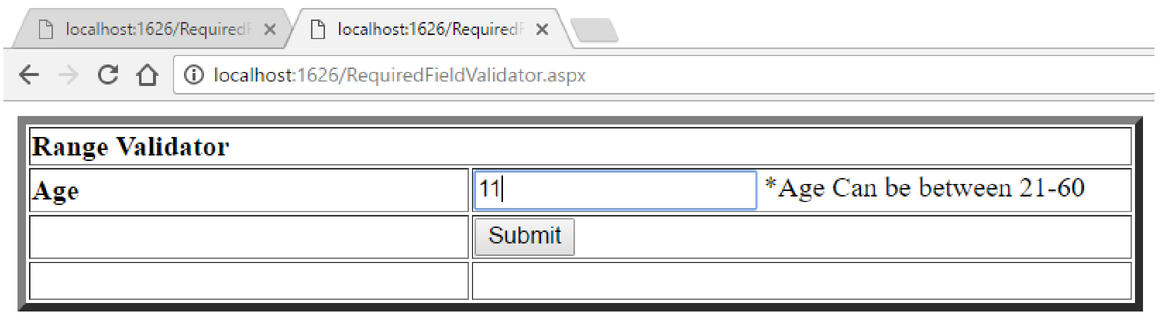


Range Validator	
Age	
	Submit

Now, Drag and drop the RangeValidator from the Validation Tab in ToolBox and set the properties as shown below

Property Name	**Value**
ControlToValidate	txtUserAge
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	*Age Can be between 21-60
Type	Integer
MaximumValue	60
MinimumValue	21

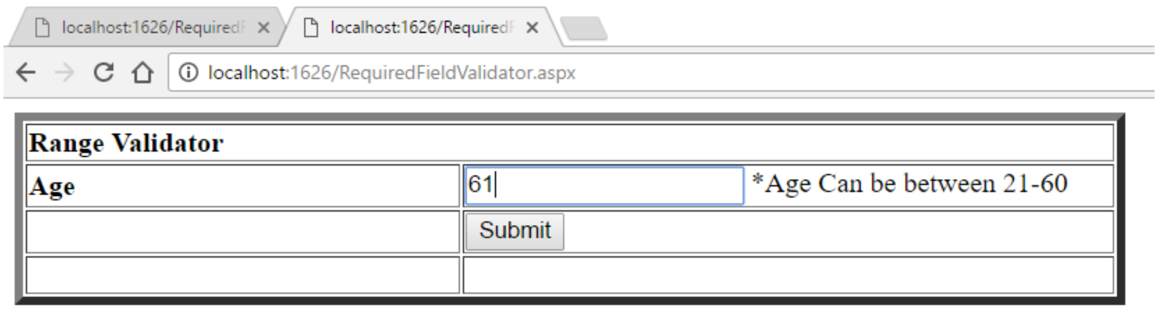
Run the web site, and press the Submit button while entering the Age value less than MinimumValue, you will see that validation check is performed showing the error messages,



localhost:1626/RequiredFieldValidator.aspx

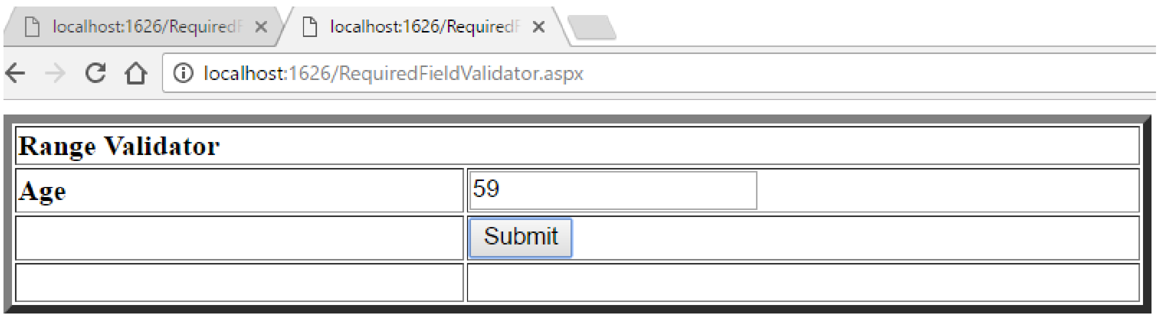
Range Validator	
Age	11 *Age Can be between 21-60
	Submit

Similarly, add value greater than the MaximumValue,



Range Validator	
Age	61 *Age Can be between 21-60
	Submit

Try, a number between 21 to 60 will hide the error message as shown



Range Validator	
Age	59
	Submit

Similarly, you can perform range validation on other data types as well such as string, date etc

- **CompareValidator Control**

The **CompareValidator** might not be the most commonly used validator of the bunch, but it's still useful in some cases. It can compare two values, for instance the values of two controls. In the next example, I will show you a small example of how it can be used.

As you see, we only use one validator to validate the two fields. It might seem a bit overwhelming, but it's actually quite simple. Like with the RequiredFieldValidator, we use the ControlToValidate attribute to specify which control to validate. In addition to that, we specify a control to compare. The operator attribute specifies which method to use when comparing. In this case, we use the Less Than operator, because we wish for the first control to have the smallest value. We set the type to integer, because we want to compare integers. Dates, strings and other value types can be compared as well.

If you switch the numbers, you will see another cool thing about the ASP.NET validators and clientside scripting: Validation is also performed upon exiting the fields. If you don't want this, you can turn off clientside scripting with the EnableClientscript attribute. You can also compare a field to a static value. To do this, remove the ControlToCompare attribute, and add the ValueToCompare attribute.

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

In addition to the four common properties, developer has to specify the following three specific properties for CompareValidator:

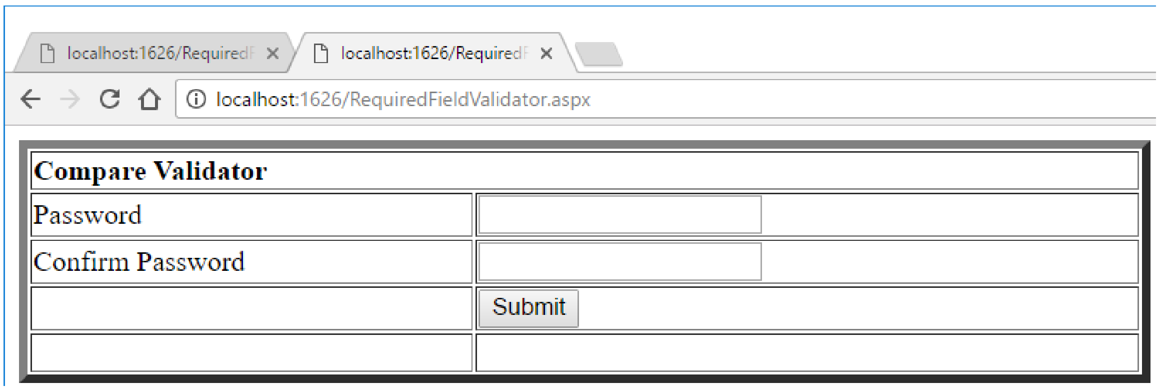
Properties	**Description**
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan,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

	DataTypeCheck.

For example,

- Add a CompareValidator on an ASP.Net page.
- Add two TextBoxes to validate, rename the TextBox to txtPassword, txtRetypePassword
- Add a button to submit the form.

The designed Web form is shown here,

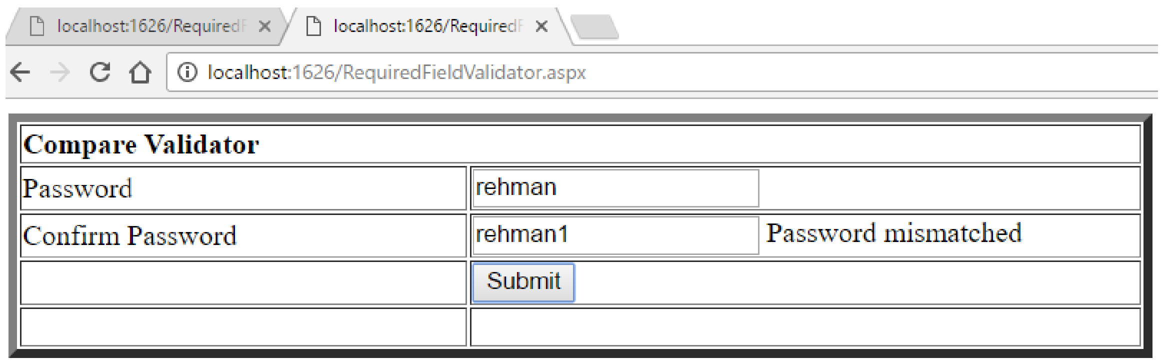


Compare Validator	
Password	
Confirm Password	
	Submit

Now, Drag and drop the CompareValidator from the Validation Tab in ToolBox and set the properties as shown below

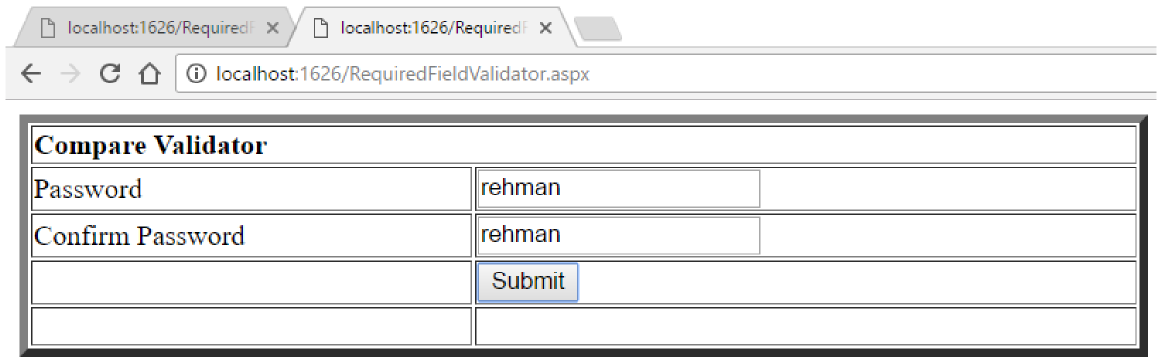
Property Name	**Value**
ControlToValidate	txtRetypePassword
ControlToCompareWith	txtPassword
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	*Password mismatched
Type	String
Operator	Equals

Run the web site, and press the Submit button while entering the Confirm Password value not matching with password value, you will see that validation check is performed showing the error messages,



Compare Validator		
Password	rehman	
Confirm Password	rehman1	Password mismatched

Similarly, Enter same password in both text boxes



Compare Validator	
Password	rehman
Confirm Password	rehman

Similarly, you can perform Compare validation on other data types as well such as integer, date etc with different values in Operator property of the CompareValidator.

- **RegularExpressionValidator**

The **RegularExpressionValidator** is a bit more difficult to write about, because it requires knowledge of Regular Expressions to really use it. However, RegularExpressionValidator is one of the most useful validators, because it can be used to check the validity of any kind of string. You can find many pre-made expressions out there, which can be very useful to you.

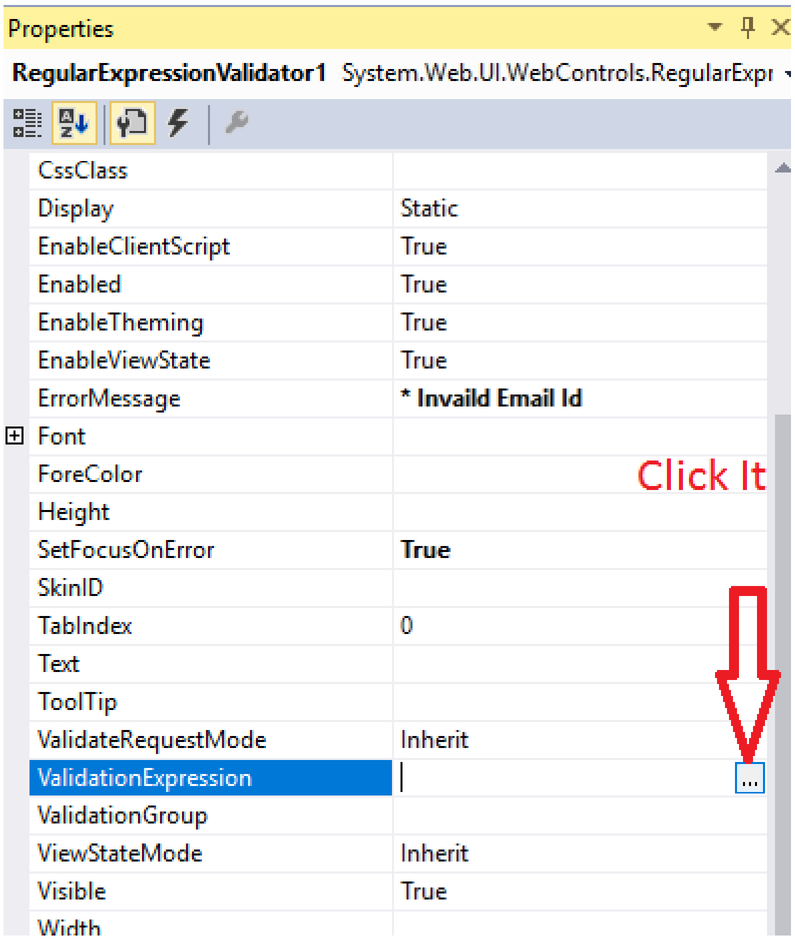
We can use the RegularExpressionValidator for almost everything, for instance validating an e-mail or an URL.

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.

In addition to the four common properties, developer has to specify the following three specific properties for CompareValidator:

Properties	**Description**
RegularExpression	Specifies the RegularExpression to be validated. It can an email id, internet url or any other user specified Regular Expression

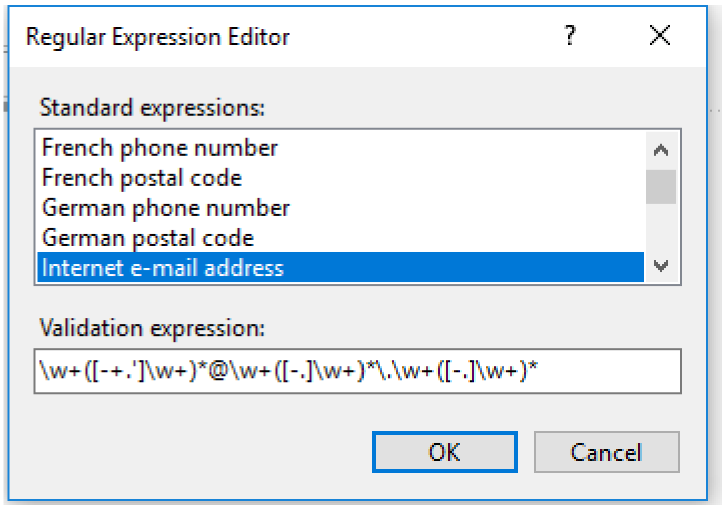
The ValidationExpression Property is shown here


Properties

RegularExpressionValidator1 System.Web.UI.WebControls.RegularExpr

CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	*** Invaild Email Id**
Font	
ForeColor	
Height	
SetFocusOnError	**True**
SkinID	
TabIndex	0
Text	
ToolTip	
ValidateRequestMode	Inherit
ValidationExpression	
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

When you click on the button in the Validaitonexpression Property, you will see all the options available for this property as shown


Regular Expression Editor

Standard expressions:

French phone number
French postal code
German phone number
German postal code
Internet e-mail address

Validation expression:

```
\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
```

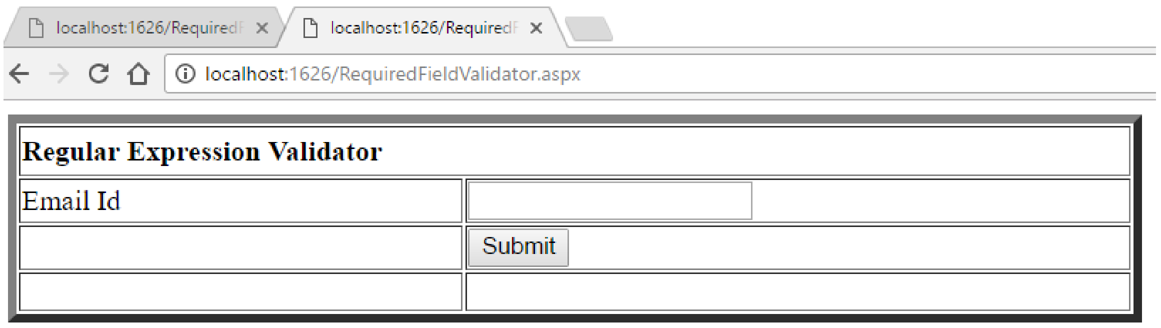
There is an option with (Custom) , where developers can design their own Regular Expressions

Note: Some details on different Regular Expression is designing is given in the next page

For example,

- Add a RegularExpressionValidator on an ASP.Net page.
- Add a TextBoxes to validate, rename the TextBox to txtEmailId
- Add a button to submit the form.

The designed Web form is shown here,

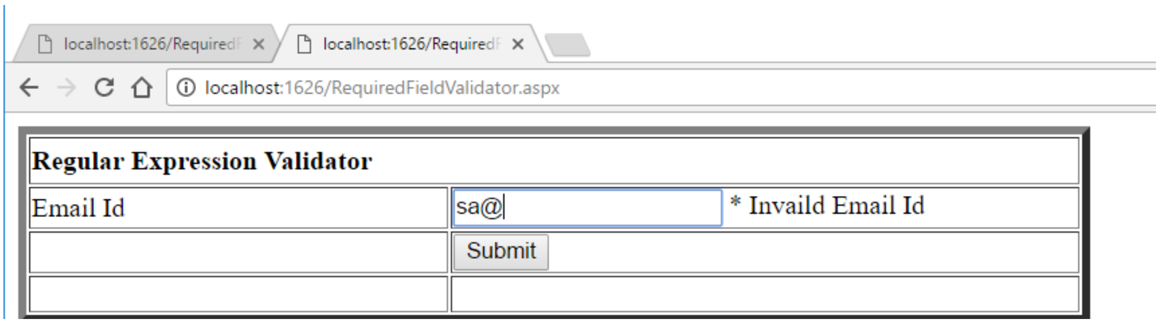


Regular Expression Validator	
Email Id	
	Submit

Now, Drag and drop the RegularExpressionValidator from the Validation Tab in ToolBox and set the properties as shown below

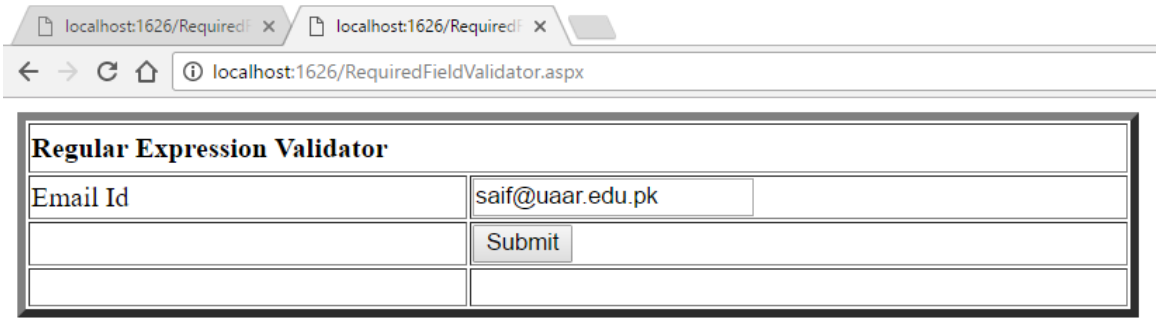
Property Name	Value
ControlToValidate	txtEmailId
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	* Invaild Email Id
ValidationExpression	Inter E-mail address (Select it from the open Regular Expression Editor)

Run the web site, and press the Submit button while entering the invalid email id, you will see that validation check is performed showing the error messages,



Regular Expression Validator	
Email Id	sa@ * Invaild Email Id
	Submit

Similarly, enter the correct email id



Regular Expression Validator	
Email Id	saif@uaar.edu.pk
	Submit

Similarly, you can design your own Regular Expression and check the RegularExpressionValidator for your custom design Regular Expression as well.

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.

- **CustomValidator**

If none of the other validators can help you, the **CustomValidator** usually can. A common way of using the CustomValidator is when you need to make a database lookup to see if the value is valid. The control allows you to validate both clientside and serverside, where the serverside approach is probably the most powerful. Of course, doing serverside validation requires a postback to validate, but in most cases, that's not really a problem.

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.

- **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.

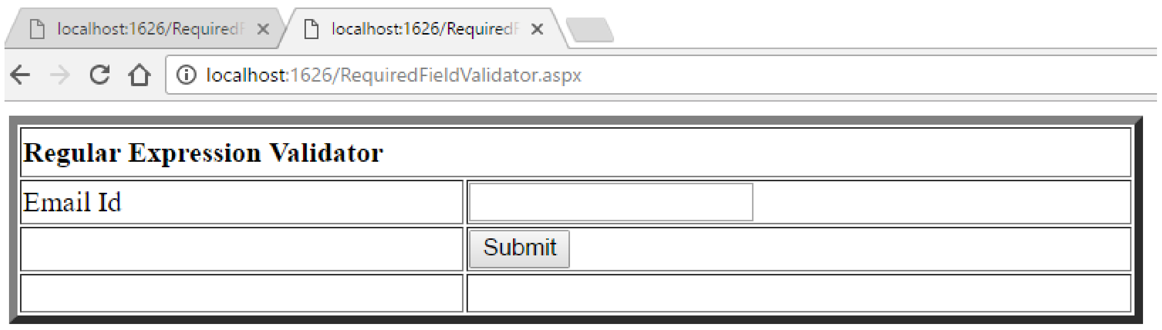
The following two mutually inclusive properties list out the error message:

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

For example,

- Add a RequiredFieldValidatot, RegularExpressionValidator and a RangeValidator on an ASP.Net page.
- Add three TextBoxes to validate the data.
- Rename TextBox1 to txtUserName , TextBox2 to txtEmailId and TextBox3 to txtAge
- Bind txtName to RequiredFieldValidatot, bind txtEmailId to RegularExpressionValidator and txtAge to RangeValidator (Specify the properties of each control as we did in previous pages)
- Add a button to submit the form.

The designed Web form is shown here,



Regular Expression Validator	
Email Id	
	Submit

Specify the properties of RequiredFieldValidator as shown in the next table,

Property Name	**Value**
ControlToValidate	txtUserName
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	*User Name Can not be blank

Specify the properties of RegularExpressionValidator as shown in the next table,

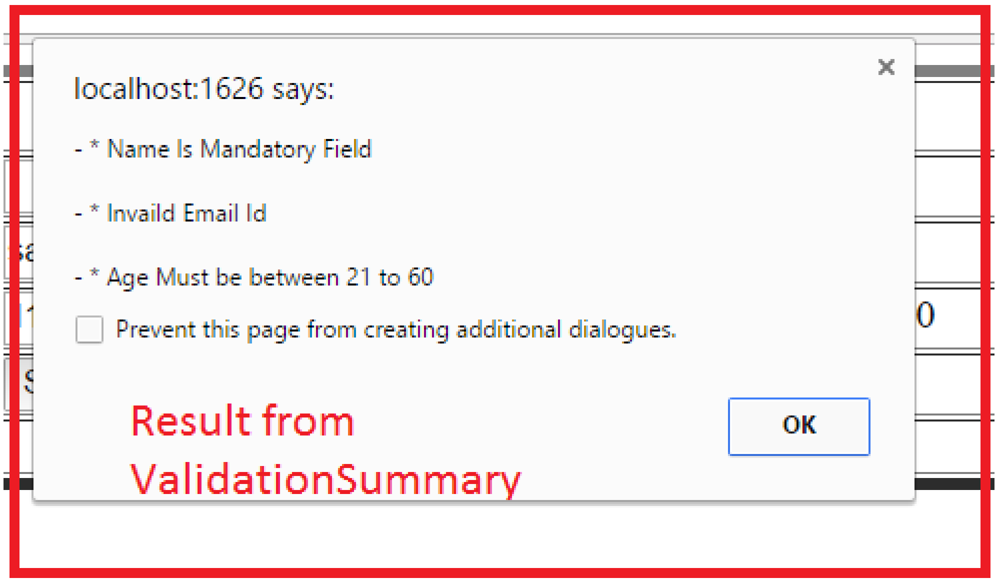
Property Name	**Value**
ControlToValidate	txtEmailId
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	* Invaild Email Id
ValidationExpression	Inter E-mail address (Select it from the open Regular Expression Editor)

Specify the properties of RangeValidator as shown in the next table,

Property Name	**Value**
ControlToValidate	txtAge
SetFocusOnError	Yes
Display	Dynamic
ErrorMessage	*Age Can be between 21-60
Type	Integer
MaximumValue	60

MinimumValue	21

Run the web site, and press the Submit button while leaving the txtName blank, entering the invalid email id in txtEmailId and entering 68 in txtAge you will see that validation check is performed showing the error messages in MessageBox



localhost:1626 says:

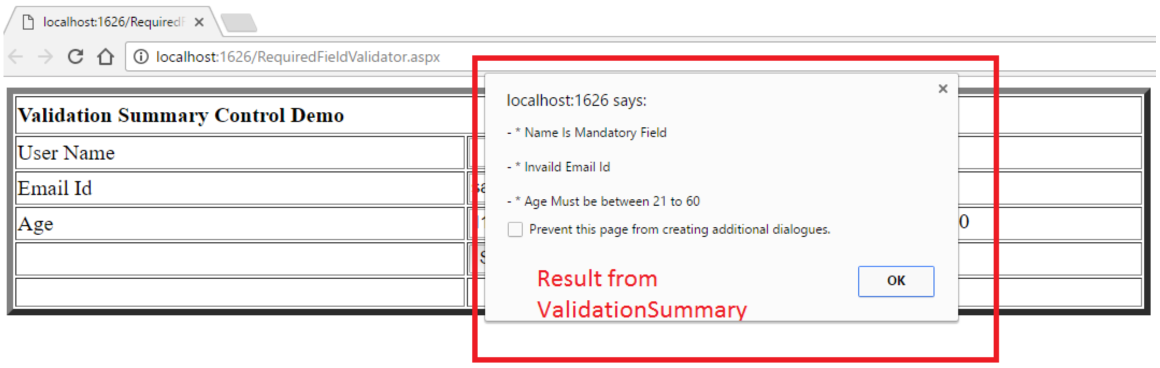
- * Name Is Mandatory Field

- * Invaild Email Id

- * Age Must be between 21 to 60

Prevent this page from creating additional dialogues.

OR



localhost:1626/RequiredFieldValidator.aspx

Validation Summary Control Demo
User Name
Email Id
Age

localhost:1626 says:

- * Name Is Mandatory Field

- * Invaild Email Id

- * Age Must be between 21 to 60

Prevent this page from creating additional dialogues.

- **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.The validators all share a couple of attributes, which can be very useful in some cases.

Saif Ur Rehman Saifi
Assistant Professor, UIIT
PMAS Arid Agriculture University, Rawalpindi
If you found this tutorial helpful or you want to give us any valuable suggestions contact at
www.ASPUIIT.blogspot.com