Validation Controls

UNIT - 2

Validation

- Validation server control is used to validate the data of an input control.
- Validation controls validate the user input data to ensure that useless, unauthenticated or contradictory data will not get stored.
- If the data does not pass validation, it will display an error message to the user.
- 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.

Types of Validation

- There are two ways we can perform validation:
  - **Client side validation**
  - **Server side validation**

Client Side Validation

- Client side validation is works 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.

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.

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.

Types of Validation Control in ASP,NET

Control	Description
RequiredFieldValidator	Enables you to require a user to enter a value in a form field Compulsorily.
RangeValidator	Enables you to check whether a value falls between a certain minimum and maximum value.
CompareValidator	Enables you to compare a value against another value or perform a data type check.
RegularExpressionValidator	Enables you to compare a value against a regular expression.
CustomValidator	Enables you to perform custom validation.
ValidationSummary	Enables you to display a summary of all validation errors in a page.

RequiredFieldValidator

- RequiredFieldValidator control enables you to require a user to enter a value into a form field before submitting the form.
- You must set two important properties when using the RequiredFieldValdiator control:

  ControlToValidate—The ID of the form field being validated.

  Text—The error message displayed when validation fails.
- By default, the RequiredFieldValidator checks for a nonempty string (spaces don't count).
- If you enter anything into the form field associated with the RequiredFieldValidator, then the RequiredFieldValidator does not display its validation error message.
- You can use the RequiredFieldValidator control's InitialValue property to specify a default value other than an empty string.

RangeValidator

- The RangeValidator control enables you to check whether the value of a form field falls between a certain minimum and maximum value.
- You must set five properties when using this control:
  - ControlToValidate—The ID of the form field being validated.
  - Text—The error message displayed when validation fails.
  - MinimumValue—The minimum value of the validation range.
  - MaximumValue—The maximum value of the validation range.
  - Type—The type of comparison to perform. Possible values are String, Integer, Double, Date, and Currency.
- Validation message is also displayed if you enter a value that is not a number. If the value entered into the form field cannot be converted into the data type represented by the RangeValidator control's Type property, then the error message is displayed.

RangeValidator

- If you don't enter any value into the age field and submit the form, no error message is displayed.
- If you want to require a user to enter a value, you must associate a RequiredFieldValidator with the form field.
- Don't forget to set the Type property when using the RangeValidator control.
- By default, the Type property has the value String, and the RangeValidator performs a string comparison to determine whether a values falls between the minimum and maximum value.

CompareValidator

- The CompareValidator control enables you to perform three different types of validation tasks.
- You can use the CompareValidator to perform a data type check.
- In other words, you can use the control to determine whether a user has entered the proper type of value into a form field, such as a date in a birth date field.
- You also can use the CompareValidator to compare the value entered into a form field against a fixed value.
- you can use the CompareValidator to compare the value of one form field against another.
- For example, you use the CompareValidator to check whether the value entered into the meeting start date is less than the value entered into the meeting end date.

CompareValidator

- The CompareValidator has six important properties:
  - ControlToValidate—The ID of the form field being validated.
  - Text—The error message displayed when validation fails.
  - Type—The type of value being compared. Possible values are String, Integer, Double, Date, and Currency
  - Operator—The type of comparison to perform. Possible values are DataTypeCheck, Equal, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and NotEqual.
  - ValueToCompare—The fixed value against which to compare.
  - ControlToCompare—The ID of a control against which to compare.

RegularExpressionValidator

- The RegularExpressionValidator control enables you to compare the value of a form field against a regular expression. You can use a regular expression to represent string patterns such as email addresses, Social Security numbers, phone numbers, dates, currency amounts, and product codes.

Character	Meaning for Character in Expression
\w	Any word character (Letter, Number or Underscore)
\d	Any digit
\D	Any character that is not Digit
\s	Any white space character like Tab or Space
\S	Any non-white space character

RegularExpressionValidator

- Just like the other validation controls, the RegularExpressionValidator doesn't validate a form field unless the form field contains a value. To make a form field required, you must associate a RequiredFieldValidator control with the form field.

Character	Meaning for Character in Expression
\w {10}	10 word character (Letter, Number or Underscore)
\w {5,10}	Word character between 5 to 10
\d {5,10)	Any digits ranging from 5 to 10
[A-D]	Any 1 alphabet between A to D
[A-Z]{5,15}	Any alphabet between A to D but the no. of alphabets can be between 5 to 15.
\d{2}-\d{7,10}	Ex. 99-999999 (2 digits then hyphen and after that digits between 7 to 10

CustomValidator

- If none of the other validation controls perform the type of validation that you need, you can always use the CustomValidator control.
- You can associate a custom validation function with the CustomValidator control.
- The CustomValidator control has three important properties:
  - ControlToValidate : The ID of the form field being validated.
  - Text : The error message displayed when validation fails.
  - ClientValidationFunction : The name of a client-side function used to perform client-side validation.
- The CustomValidator also supports one event:
  - ServerValidate : This event is raised when the CustomValidator performs validation.

CustomValidator

- You associate your custom validation function with the CustomValidator control by handling the ServerValidate event.
- The second parameter passed to the ServerValidate event handler is an instance of the ServerValidateEventArgs class. This class has two properties:
  - **Value :** Represents the value of the form field being validated.
  - **IsValid :** Represents whether validation fails or succeeds.
  - **ValidateEmptyText :** Represents whether validation is performed when the form field being validated does not contain a value.

ValidationSummary

- The ValidationSummary control enables you to display a list of all the validation errors in a page in one location.
- This control is particularly useful when working with large forms. If a user enters the wrong value for a form field located toward the end of the page, then the user might never see the error message.
- If you use the ValidationSummary control, however, you can always display a list of errors at the top of the form.
- You might have noticed that each of the validation controls includes an ErrorMessage property. We have not been using the ErrorMessage property to represent the validation error message. Instead, we have used the Text property.

ValidationSummary

- The distinction between the ErrorMessage and Text property is that any message that you assign to the ErrorMessage property appears in the ValidationSummary control, and any message that you assign to the Text property appears in the body of the page.
- The ValidationSummary control supports the following properties:
  - **DisplayMode :** Enables you to specify how the error messages are formatted. Possible values are BulletList, List, and SingleParagraph.
  - **HeaderText :** Enables you to display header text above the validation summary.
  - **ShowMessageBox :** Enables you to display a popup alert box.
  - **ShowSummary :** Enables you to hide the validation summary in the page.
- If you set the ShowMessageBox property to the value true and the ShowSummary propertyto the value False, then you can display the validation summary only within a popup alert box.