Unit-3
ASP.NET Validation Controls & State Management

3.1 What is Validation?

In ASP.NET, validation refers to the process of ensuring that user input meets certain criteria before it is processed by the server. This is essential for maintaining data integrity and security in web applications. ASP.NET provides various validation controls and techniques to accomplish this task.

Client-Side Validation: Validation performed on the client side using JavaScript before the form is submitted to the server.

Server-Side Validation: Validation performed on the server side after the form data is submitted. This is essential for ensuring that no malicious or incorrect data bypasses client-side validation.

Some of the reasons why validation controls are important in ASP.NET are:

- **Preventing invalid data from being submitted:** 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.
- **Improving user experience:** When users enter invalid data, they may receive error messages or other feedback that helps them corrects their mistakes. This can help prevent frustration and improve the overall user experience.
- **Enhancing security:** Validation controls can help prevent malicious code or other attacks that could harm the web application or its users.
- **Meeting business requirements:** 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.

Validation Controls in ASP.NET: ASP.NET provides several types of validation controls that can be used to validate user input in web forms. Some of the common validation controls are:

1. RequiredFieldValidator Control
2. CompareValidator Control
3. RangeValidator Control
4. RegularExpressionValidator Control
5. CustomValidator Control
6. ValidationSummary

The below table describes the controls and their use.

Validation Control	Description
RequiredFieldValidator	This control ensures that a field is not left empty or blank. It can be used for textboxes, dropdown lists, checkboxes, and other input controls.
CompareValidator	This control compares the value of one input control to another. It can validate passwords, confirm email addresses, and other scenarios where two values must match.
RangeValidator	This control checks if a value falls within a specific range. For example, it can be used to validate a user's age, income, or date of birth.
RegularExpressionValidator	This control checks if a value matches a specified regular expression pattern. For example, it can validate email addresses, phone numbers, zip codes, and other input types.
CustomValidator	This control allows developers to define their validation logic. It usually depends on the business rules.
ValidationSummary	This control displays a report of all validation errors that occurred on a Web page.

1. RequiredFieldValidatorControl:
In ASP.NET, the RequiredFieldValidator control ensures that a user has entered a value into an input control on a web form. If the user does not enter a value in a required field, there will be an error message.
Syntax:<asp:RequiredFieldValidatorID="reqName"ControlToValidate="txtName"runat="server"ErrorMessage="*"></asp:RequiredFieldValidator>

Property	**Description**
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ErrorMessage	It is used to set error message that display when validation fails.
ControlToValidate	It takes ID of control to validate.

2. CompareValidatorControl :
CompareValidator control is used to compare the value of an input control against the value of another input control on the basis of the specified operator. CompareValidator control compares inputted value with another input control value.
Syntax:<asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="Compare Validator"></asp:CompareValidator>

Properties	**Description**
ErrorMessage	Message to display in a ValidationSummary when validation fails.
Text	Text to display when validation fails.
ConrolToCompare	Set ID of input control to compare with.
ControlToValidate	Set ID of input control to validate.
Operator	Set comparison operator. (Equal, NotEqual, LessThan, GreaterThan etc.)
Type	It is used to define Data type of values for comparison.

3. The RangeValidationControl :

This validator evaluates the value of an input control to check that the value lies between specified ranges.It allows us to check whether the user input is between a specified upper and lower boundary. This range can be numbers, strings and dates.

Syntax:<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="uesrInput " ErrorMessage="Enter value in specified range" ForeColor="Red" MaximumValue="val" MinimumValue="val"Type=" Datatype"**></asp:RangeValidator>**

Properties	Description
ErrorMessage	Message to display in a ValidationSummary when validation fails.
Text	Text to display when validation fails.
MaximumValue	It is used to set upper boundary of the range.
MinimumValue	It is used to set lower boundary of the range.
Type	It is used to define Data type of values for comparison.

4. The Regular Expression Validations:

This validator is used to validate the value of an input control against the pattern defined by a regular expression. It allows us to check and validate predictable sequences of characters like: e-mail address, telephone number etc.The **ValidationExpression** property is used to specify the regular expression, this expression is used to validate input control.

Syntax:<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"ControlToValidate="username" ErrorMessage="Please enter valid username" ForeColor="Red"ValidationExpression=”RegExp"></asp:RegularExpressionValidator>

Properties	Description
ErrorMessage	Message to display in a ValidationSummary when validation fails.
Text	Text to display when validation fails.
ValidationExpression	It is used to set regular expressions to determine validity.

5. The Validation Summary:

This validator is used to display list of all validation errors in the web form. It allows us to summarize the error messages at a single location.

Syntax:<asp:ValidationSummary ID="ValidationSummary1" runat="server" />

Properties	Description
ShowMessageBox	It displays a message box on error in up-level browsers.
ShowSummary	It is used to show summary text on the form page.
DisplayMode	Specifies display mode of validation summary.

6. The Customvalidationcontrol :

The CustomValidator control in ASP.NET provides a way to perform customized validation. It allows you to define your own validation logic for a specific scenario that may not be covered by the built-in validation controls. To use the CustomValidator control, you can create a validation control that checks the entered value against your specific criteria.

Syntax :<asp:CustomValidatorrunat="server" id="cusCustomTxt" controltovalidate="txtCustomValidator"onservervalidate="cusCustomTxt_ServerValidate"errormessage="enter number" / >

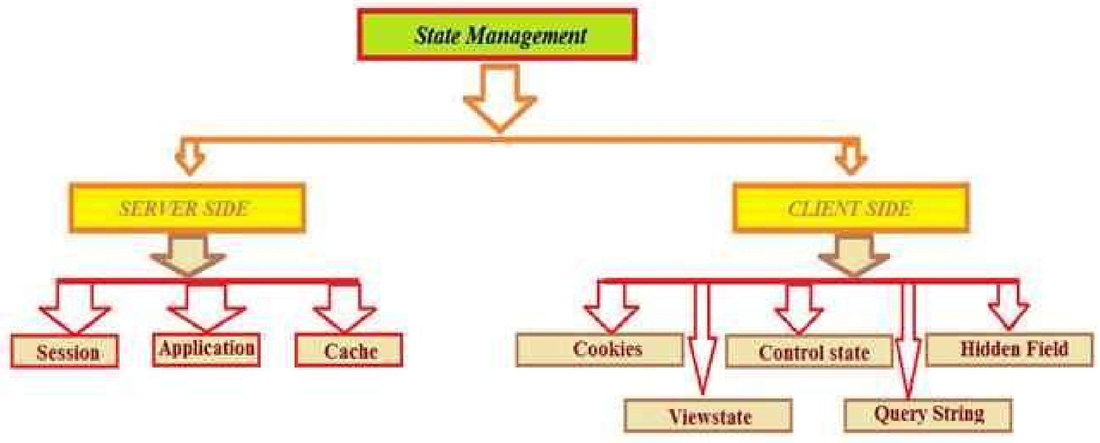
Properties	Description
ErrorMessage	Message to display in a ValidationSummary when validation fails.
Text	Text to display when validation fails.
ControlToValidate	Set ID of input control to validate.

3.2 State Management in Asp.NET:

State Management is a process by which state and page information is maintained over multiple requests for same or different pages. Response is sent back to client based on his request.

Requirement of State Management:

- Maintain user-specific information
- Enable cross-page communication
- Enhance scalability
- Performance



A. Client-Side State Management:

Whenever we use Client-Side State Management, the state related information will directly get stored on the client-side browser.

QueryString :A query string is a set of parameters included in an HTTP request that enables the passing of information from one page to another. It consists of a series of key-value pairs appended to a specific URL. The query string is specified by the values following the ? (question mark) in the URL, with each parameter separated by an ampersand (&) symbol. The purpose of the query string is to provide additional data to the receiving page, allowing it to process and respond accordingly.

ViewState :ViewState is a important client side state management technique. ViewState is used to store user data on page at the time of post back of web page.ViewState does not hold the controls, it holds the values of controls. It does not restore the value to control after page post back. ViewState can hold the value on single web page, if we go to other page using response.redirect then ViewState will be null.

Cookies: ASP.NET Cookie is a small bit of text that is used to store user-specific information. This information can be read by the web application whenever user visits the site. When a user requests for a web page, web server sends not just a page, but also a cookie containing the date and time. Browser stores separate cookie for each different sites user visited.

Note: The Cookie is limited to small size and can be used to store only 4 KB (4096 Bytes) text.
Type of Cookies
Persist Cookie - A cookie that doesn't have expired time is called a Persist Cookie
Non-Persist Cookie - A cookie which has expired time is called a Non-Persist Cookie

Hidden Field: HiddenField, as name implies, is hidden. This is non visual control in ASP.NET where you can save the value. This is one of the types of client-side state management tools. It stores the value between the roundtrip. Anyone can see HiddenField details.HiddenFields are not encrypted or protected. However, from a security point of view, this is not suggested. So, don’t store any important or confidential data like password and credit card details with this control.

Control State: Control state should be used only for small amounts of critical data that are essential for the control. Some examples of when to use control state include:

- Storing the current page number of a pager control.
- Storing the current selection state of a control.
- Storing the current value of a text box control that is used to enter a search query.

B. Server-Side State Management:

Server-Side State Management is different from Client-Side State Management but the operations and working is somewhat the same in functionality. In Server-Side State Management all the information is stored in the user memory.

Session: Sessions are used in ASP to handle the user interactions with the web page or the web application. In ASP.NET session objects are an important component of any application or a modern website. Basic information like username, password, shopping cart information, and the location can be stored in these sessions.

Modes of Session:
InProc Mode : The InProc Session mode is the default Session Mode. Using this Session Mode, the Session Mode is stored in the application worker process.

StateServer Mode /Out-Proc Mode: This is also the Out-Proc Session mode. It allows to storing the session data in any other servers.

SQL Server: In this mode the session data is stored inside the SQL Server database.

Custom: You can create your own session state provider.

Cache: Caching is a technique of storing frequently used data/information in memory, so that, when the same data/information is needed next time, it could be directly retrieved from the memory instead of being generated by the application.

Application State: ASP.Net Application state is a server side state management technique.Application state is a global storage mechanism that used to stored data on the server and shared for all users, means data stored in Application state is common for all user. Data from Application state can be accessible anywhere in the application.