1.1 Concepts of ASP .NET

What is ASP?

- ASP is a server side scripting language
- Files Containing HTML and Scripting code(VBScript, Jscript)
- Access Via HTTP Requests
- Scripting Code is interpreted on server side
- Simple procedural programming language
    - Access to COM components
    - Activex data object (ADO)
    - File system object
    - Custom components
- Script based : no compiling, just edit, save, run
    - VBScript, Jscript
- Support for multiple scripting languages
- ASP has been very popular

What is ASP.NET?

- ASP .NET is Not ASP.
- ASP stands for Active Server Pages.
- ASP .NET is the next generation of ASP, but it's not an upgraded version of ASP.
- Like ASP, ASP.NET is a server-side technology.
- ASP.NET is an entirely new technology for server-side scripting. It was written from the ground up and is not backword compatible with classic ASP.
- ASP.NET is the major part of the Microsoft's .NET Framework.
- ASP.NET is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server.
- ASP . NET provides services to allow the creation, deployment, and execution of Web

Applications and Web Services.

- ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful web applications.
- ASP.NET Microsoft Technology.
- ASP .NET is a program that runs inside IIS.
- IIS (Internet Information Services) is Microsoft's Internet server.
- IIS comes as a free component with Windows servers.
- IIS is also a part of Windows 2000 and XP Professional.
- Web Applications are built using Web Forms.
- ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications.
- ASP.NET defines an application as the sum of all files, pages, handlers, modules and executable code that can be invoked or run in the scope of a given virtual directory (and its subdirectories) on a Web application server.
- For example, an "order" application might be published the "/order" virtual directory on a Web server computer. For IIS the virtual directory can be set up in the Internet Services Manager; it contains all subdirectories, unless the subdirectories are virtual directories themselves.
- Each ASP.NET Framework application on a Web server is executed within a unique .NET Framework application domain, which guarantees class isolation (no versioning or naming conflicts), security sandboxing (preventing access to certain machine or network resources), and static variable isolation.
- ASP.NET maintains a pool of HttpApplication instances over the course of a Web application's lifetime
- ASP NET automatically assigns one of these instances to process each incoming HTTP request that is received by the application. The particular HttpApplication instance assigned is responsible for managing the entire lifetime of the request and is reused only after the request has been completed. This means that user code within the HttpApplication does not need to be reentrant.

WHAT CAN I DO WITH ASP.NET?

- Easily and quickly create simple Web applications.
- Generate dynamic Web content
- Client-side scripting for validation
- Access COM components to extend functionality

WHAT IS AN ASP.NET FILE?

- An ASP.NET file is just the same as an HTML file
- An ASP.NET file can contain HTML, XML, and scripts
- Scripts in an ASP.NET file are executed on the server
- An ASP.NET file has the file extension ".aspx"

HOW DOES ASP.NET WORK?

- When a browser requests an HTML file, the server returns the file.
- When a browser requests an ASP.NET file, IIS passes the request to the ASP.NET engine on the server.
- The ASP.NET engine reads the file, line by line, and executes the scripts in the file.
- Finally, the ASP.NET file is returned to the browser as plain HTML.

Advantages of ASP .NET

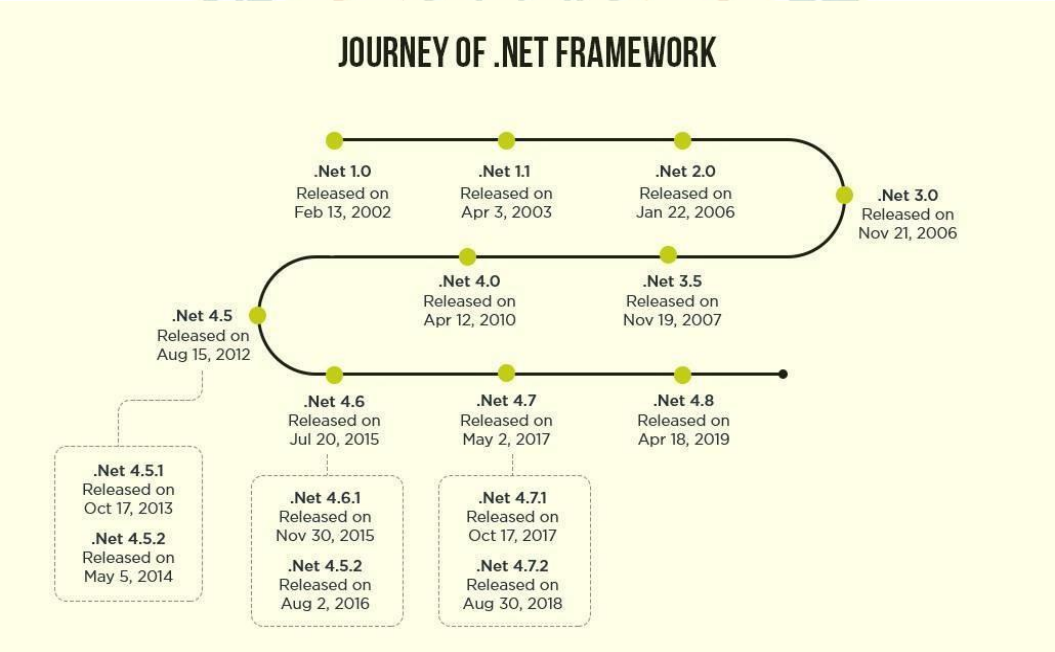
1. Separation of code from HTML
   - To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic.
   - This makes it much easier for teams of programmers and designers to collaborate efficiently.
2. Support for compiled languages
   - Developer can use VB.NET and access features such as strong typing and object oriented programming.
   - Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code ASP.NET pages are precompiled to byte-code and Just In Time (IT) compiled when first equested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.
3. Use services provided by the .NET Framework
   - The .NET Framework provides class libraries that can be used by your application.
   - Some of the key classes help you with input/output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.
4. Graphical Development Environment
   - Visual Studio .NET provides a very rich development environment for Web developers. You can drag and drop controls and set properties.
5. State management
   - To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management.
   - State information can, for example, be kept in memory or stored in a database.
6. Update files while the server is running
   - Components of your application can be updated while the server is online and clients are connected.
7. XML-Based Configuration Files
   - Configuration settings in ASP.NET are stored in XML files that you can easily read and edit.
   - You can also easily copy these to another server, along with the other files that comprise your application.

8. Security
   - With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

1.2.Net framework 2.0

- The **.NET Framework** is a software development platform that was introduced by Microsoft in the late 1990 under the NGWS. On 13 February 2002, Microsoft launched the first version of the .NET Framework, referred to as the **.NET Framework 1.0**.
- It is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc. It is also used to create a form based, console-based, mobile and web-based application or services that are available in Microsoft environment.
- Furthermore, the .NET framework is a pure object oriented, that similar to the Java language. But it is not a platform independent as the Java. So, its application runs only to the windows platform.
- The main objective of this framework is to develop an application that can run on the windows platform. The current version of the .Net framework is 4.8.
- The .NET Framework is not only a language, but it is also a software and language neutral platform.
- The .Net Framework supports more than 60 programming languages such as C#, F#, VB.NET, J#, VC++, JScript.NET, APL, COBOL, Perl, Oberon, ML,Pascal, Eiffel, Smalltalk, Python, Cobra, ADA, etc.

.Net Framework Timeline



Features of .Net Framework

- Following are the features that make .NET a wonderful platform for developing modern applications.

1) Object Oriented Programming System

- In C and C++ programs uses header files like "stdio.h", "iostream.h" etc.
- Visual Studio .NET contains hundreds of classes and namespaces that provides variety of functionality in applications.
- For e.g. to work with database we need to use **System.Data** namespace.

2) In-built Memory Management

- Visual Studio .NET provides a fully object oriented environment.
- Visual Studio .NET supports OOPs concepts.

3) Rich set of Classes

- Developers always worry for system resources like memory.
- Visual Studio .NET supports handling memory on its own.
- The garbage collector takes responsibility for freeing up unused objects at regular intervals.

4) Multi-Language and Multi-Device support

- Visual Studio .NET supports multiple languages.
- This means that if your friend has skills in C++, and you have skills in VB you both can work in same application. You can write logic in C# and they can write in VB .NET.
- The beauty of multi language support lies in the fact that even though the syntax of each language is different, the basic environment of developing the software is same.
- Visual Studio .NET supports Multi-Device.
- We can create mobile or PDA etc. device supported software.

5) Faster and easy development of web applications

- ASP .NET is useful for developing dynamic and database related web applications.
- ASP .NET contains rich and faster development controls for web applications.

6) XML support

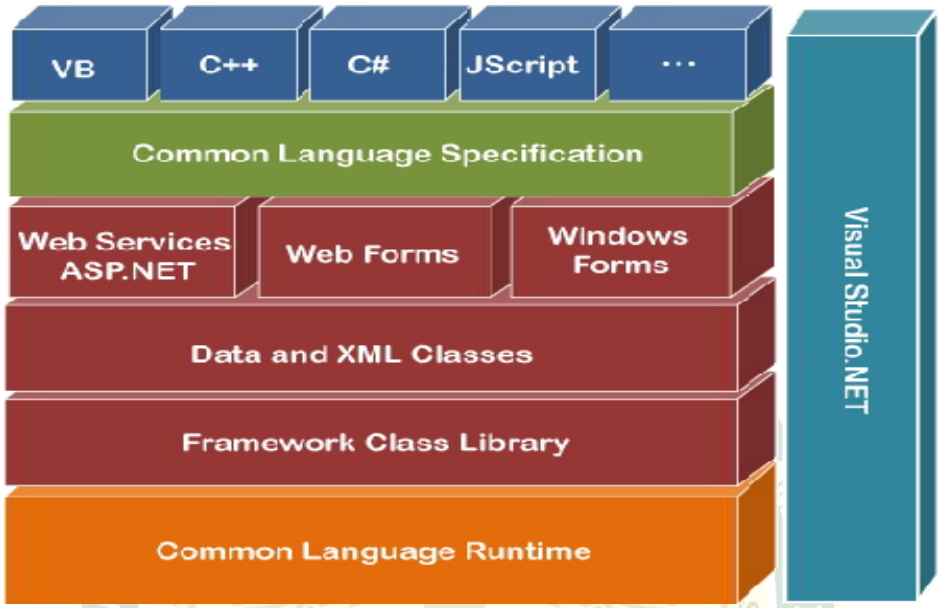
- Visual Studio .NET supports for writing, manipulating and transforming XML documents.

7) Ease of development and configuration

- Deploying windows applications especially that used COM components were always been a tedious task. Since .NET does not require any registration as such, much of the deployment is simplified.

.Net Framework Architecture

- **.Net Framework Architecture** is a programming model for the .Net platform that provides an execution environment and integration with various programming languages for simple development and deployment of various Windows and desktop applications.
- It consists of class libraries and reusable components.



Components of .NET Framework

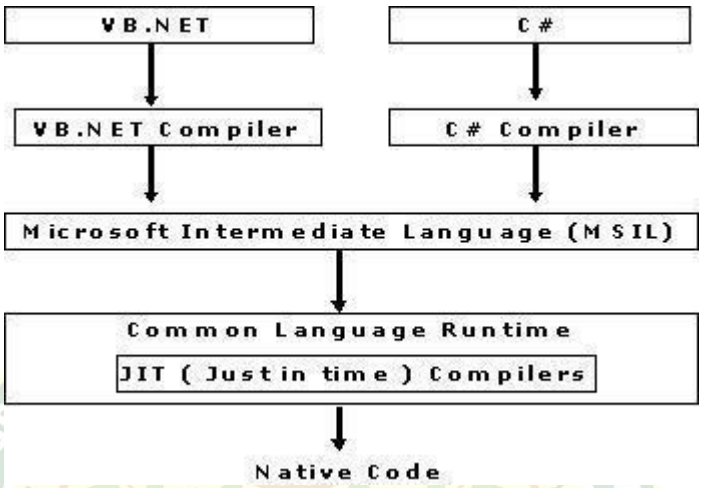
- There are following components of .NET Framework:

1. CLR (Common Language Runtime)
2. CTS (Common Type System)
3. BCL (Base Class Library)
4. CLS (Common Language Specification)
5. .NET Assemblies
6. XML Web Services
7. Window Services

1. CLR (Common Language Runtime)

- It is the execution engine for .NET framework applications.
- It is a run-time environment which executes the code written in any .NET programming language.
- It is backbone of .NET architecture.
- It works as a layer between Operating Systems and the applications written in.Net languages.

- The main function of Common Language Runtime (CLR) is to convert the Managed Code into native code and then execute the Program.
- The Common Language Runtime (CLR) 's Just In Time (JIT) compiler converts Microsoft Intermediate Language (MSIL) which is also known as the CIL(Common Intermediate Language) or IL(Intermediate Language) to native code on demand at application run time.



Other functions:-

- Memory Management
- Verification of type safety
- Access to metadata
- Code access Security
- Exception Handling
- Garbage collection

2. CTS (Common Type System)

- CTS allow programs written in different programming languages to easily share information.
- A class written in C# should be equivalent to a class written in VB.NET.
- Languages must agree on the meanings of these concepts before they can integrate with one another.
- CLS forms a subset of CTS. This implies that all the rules that apply to CTS also apply to CLS.
- It defines rules that a programming language must follow to ensure that objects written in different programming languages can interact with each other.
- CTS provide cross language integration.

- The common type system supports two general categories of types:

I. Value types
II. Reference types

I. Value types
- Stores directly data on stack.
- In build data type.
- **Example**: dim a as Integer

II. Reference types
- Stores a reference to the value's memory address, and are allocated on the heap.
- **Example**: dim obj as new Oledbconnection

3. BCL (Base Class Library)

- It is also known as framework class library (FCL).
- .NET Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
- It contains thousands of classes that support the following functions.
  - Base and user-defined data types
  - Support for exceptions handling
  - input/output and stream operations
  - Communications with the underlying system
  - Access to data
  - Ability to create Windows-based GUI applications
  - Ability to create web-client and server applications
  - Support for creating web services

4. CLS (Common Language Specification)

- It is a subset of CTS.
- It defines a set of rules and restrictions that every language must follow which runs under the. NET framework.
- CLS includes basic language features needed by almost all the application.
- It serves as a guide for library writers and compiler writers.

5. .NET Assemblies
- A .NET assembly is the main building block of the .NET Framework.
- It is a small unit of code that contains a logical compiled code in the Common Language infrastructure (CLI), which is used for deployment, security and versioning.
- It defines in two parts (process) DLL and library (exe) assemblies.
- When the .NET program is compiled, it generates a metadata with Microsoft Intermediate Language, which is stored in a file called Assembly.

6. XML Web Services

- Web-Based applications come under this category.
- ASP.Net is a framework for web and it provides the awesome integration of HTML, CSS and JavaScript which makes it useful to develop the web applications, websites and web services.

- Web services were added in .Net Framework 2.0 and considered as a part of ASP.NET web applications.

7. Window Services

- Form Based applications are considered under this category
- In simple terms, we can say client-based applications which read and writes the file system comes under this category.

1.3 Compile code

- In order to server the request given by the user, ASP.NET must first compile the code.
  - Benefits
  - Stability
  - Performance
  - Security
  - Interoperability

1.3.1 Code Behind and Inline Coding

Inline Code (In-Page Code)

- <script>

- This model is the closest to traditional ASP.
- All the code and HTML is stored in a single.aspx file.

Advantages of Inline code:

Advantages of the single-file page model include the following:

- In pages where there is not very much code, the convenience of keeping the code and markup in the same file.
- Pages written using the single-file model are slightly easier to deploy or to another programmer because there is only one file.
- Single-file page is easier to rename.
- Number of file in the project is less.

Code-behind:

- This model separates each ASP.NET web page into two files: an .aspx markup file with the HTML and control tags, and a .cs or. vb code file with the source code for the page.
- .aspx file and .aspx.vb file

Advantages of the code-behind model include the following:

- Code-behind pages offer a clean separation of the markup and code.
- It is practical to have a designer working on the markup while a programmer writes code.
- Code can be reused for multiple pages.

1.4 The Common Language Runtime

1.5 Event Driven Programming

- There are two programming approaches: Linear and Event Driven
- Any identifiable occurrence that has significance for system hardware or software is called the event.
- Event is the action or the occurrence that is detected by the program.
- User-generated events include keystrokes and mouse clicks, among a wide variety of other possibilities.
- The programming approach, which response to the event or handle the event is called Event Driven Programming.
- These events may be generated by user action or by system. Windows operating system is the Event Driven Program (EDP).

Asp.net event model

- Traditional ASP uses the Linear Model.
- It means that the code on the page is executed from top to bottom, whenever the page is loaded.
- But ASP.NET uses Event Driven Model.
- User can add controls on the page and according to the need we can decide which event it will respond. Here each event handler is the separate method.
    - When the page is executed for the first time, it creates the page and control objects, along with the initialization, code is executed and the page is rendered.
    - After the page is served to the user, any event can be triggered due to the action performed by the user.
    - If the Button click() event generates the postback of the form. ASP.NET responds the page by re-creating it.
    - Then it raises the appropriate event which has triggered the postback.
    - At last the modified page is rendered to HTML and served to the client.

1.6Server Controls (TextBox, Button, CheckBox, Image Map, Label, LinkButton, RadioButton)

- Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools. Using these tools, the users can enter data, make selections and indicate their preferences.

Button Controls

ASP.NET provides three types of button control:

- **Button** : It displays text within a rectangular area.
- **Link Button** : It displays text that looks like a hyperlink.
- **Image Button** : It displays an image.

- When a user clicks a button, two events are raised: Click and Command.
  Basic syntax of button control:

```
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Click" / >
```

Common properties of the button control:

Property	Description
Text	The text displayed on the button. This is for button and link button controls only.
ImageUrl	For image button control only. The image to be displayed for the button.
AlternateText	For image button control only. The text to be displayed if the browser cannot display the image.
CausesValidation	Determines whether page validation occurs when a user clicks the button. The default is true.
CommandName	A string value that is passed to the command event when a user clicks the button.
CommandArgument	A string value that is passed to the command event when a user clicks the button.
PostBackUrl	The URL of the page that is requested when the user clicks the button.

Text Boxes and Labels

- Text box controls are typically used to accept input from the user. A text box control can accept one or more lines of text depending upon the settings of the TextMode attribute.
- Label controls provide an easy way to display text which can be changed from one execution of a page to the next. If you want to display text that does not change, you use the literal text.

Basic syntax of text control:

```
<asp:TextBox ID="txtstate" runat="server" ></asp:TextBox>
```

Common Properties of the Text Box and Labels:

Property	Description
TextMode	Specifies the type of text box. SingleLine creates a standard text box, MultiLIne creates a text box that accepts more than one line of text and the Password causes the characters that are entered to be masked. The default is SingleLine.

Text	The text content of the text box.
MaxLength	The maximum number of characters that can be entered into the text box.
Wrap	It determines whether or not text wraps automatically for multi-line text box; default is true.
ReadOnly	Determines whether the user can change the text in the box; default is false, i.e., the user can not change the text.
Columns	The width of the text box in characters. The actual width is determined based on the font that is used for the text entry.
Rows	The height of a multi-line text box in lines. The default value is 0, means a single line text box.

Check Boxes and Radio Buttons:

- A check box displays a single option that the user can either check or uncheck and radio buttons present a group of options from which the user can select just one option.
- To create a group of radio buttons, you specify the same name for the GroupName attribute of each radio button in the group. If more than one group is required in a single form, then specify a different group name for each group.
- If you want check box or radio button to be selected when the form is initially displayed, set its Checked attribute to true. If the Checked attribute is set to true for multiple radio buttons in a group, then only the last one is considered as true.

Basic syntax of check box:

```
<asp:CheckBox ID= "chkoption" runat= "Server"> </asp:CheckBox>
```

Basic syntax of radio button:

```
<asp:RadioButton ID= "rdboption" runat= "Server"> </asp: RadioButton>
```

Common properties of check boxes and radio buttons:

Property	Description
Text	The text displayed next to the check box or radio button.
Checked	Specifies whether it is selected or not, default is false.

GroupName	Name of the group the control belongs to.

1.7 Post Back

- Basically Post back is an action performed by a interactive Webpage.
- When it goes to the server for a non-client Operation and Server again posts it back to the client.
- Each of the asp.net pages will be a separate entity with ability to process its own posted data. That is, the values of form are posted to the same page after processing the data.
- IsPostback is a read only property with each Asp.Net page (System.Web.UI.Page) class.
- This is false when first time the page is loaded and is true when the page is submitted and processed.
- This enables users to write the code depending on if the PostBack is true or false (with the use of property Page.IsPostBack).
- It is the property of the page.
- A PostBack is an action taken by an interactive webpage, when the entire page and its contents are sent to the server for processing some information and then, the server posts the same page back to the browser.
- If once the page is posted back IsPostBack property becomes true.

Example :

Page_load

```
Listbox1_Items.Add("ABC")        ABC
Listbox1_Items.Add("PQR")        PQR
Listbox1_Items.Add("XYZ")        XYZ
```

Then if we refresh the page or submit the page, the page goes to server and comes back, so this listbox will again get filled.

So to avoid this use IsPostBack

```
If not IsPostBack then
    Listbox1_Items.Add("ABC")    ABC
    Listbox1_Items.Add("PQR")    PQR
    Listbox1_Items.Add("XYZ")    XYZ
End If                           ABC
                                 PQR
                                 XYZ
```

1.8Data Binding

- Associate Data objects with web Controls
- Single value
- <%# expression %>
- Expression can be protected, public or internal
  - Value of a property
  - Member variable

- Return value of a function
- Static value

- E.g.

```
<asp:Lable ID="Lable2" runat="server" Text="<%# stnam %></asp:Lable>

Public stnam As String
Page_load()
    Stnam = txtname.text
    Me.databind()
```

- **$Expression**
  - <asp:Literal ID="Literal1" runat="server" Text="<%$AppSettings: Test %>"></asp:Literal>
  - Website =>ASP.NET configuration => Application => Create application setting
  - Evaluated by expression buider($), DataBind() is used for single value.
  - At page render or page load
  - If not DataBind() then not evaluated (single), no DataBind() with ($). With each request expression builder evaluate $ Expression.
  - <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString %> SelectCommand="SELECT * FROM [Student]"></asp: SqlDataSource >

1.8.1 Grid View

- The GridView control is fully featured, multicolumn, data-bound grid
- To customize the layout of individual columns in the GridView, you can set the column type to "templated" and modify the column's templates
- The GridView control can render without templates, which makes this control ideal for reporting scenarios
- GridView also supports selection, editing, deletion, paging, and sorting by column and button columns.
- The GridView Web server control must be bound to a data source via DataSource property or it will not be rendered on the page.
- Typical data sources for the GridView are DataSet and data readers

1.8.2 List Box

- The ListBox control is used to create a list control that allows single or multiple item selection.

- Its ASP.NET tag is <asp:ListBox> and HTML tag is <Select/>

Properties
Rows, SelectionMode

1.8.3 Data Binding Events

- The data source controls help the data-bound controls implement functionalities such as, sorting, paging, and editing data collections.
- The BaseDataBoundControl is an abstract class, which is inherited by two more abstract classes:

- DataBoundControl
- HierarchicalDataBoundControl

➢ The abstract class DataBoundControl is again inherited by two more abstract classes:
- ListControl
- CompositeDataBoundControl

➢ The controls capable of simple data binding are derived from the ListControl abstract class and these controls are:
- BulletedList
- CheckBoxList
- DropDownList
- ListBox
- RadioButtonList

1.8.4 Repeater

➢ The Repeater control is a templated, data-bound list.
➢ The Repeater control is "lookless;" that is, it does not have any built-in layout or styles.
➢ Therefore, you must explicitly declare all HTML layout, formatting, and style tags in the control's templates.

1.8.5 Form View

➢ It is a data-bound user interface control that renders a single record at a time from its associated data source.
➢ It also provides paging buttons to navigate between records.
➢ The templates available are Item Template, HeaderTemplate, FooterTemplate, EmptyData Template, PagerTemplate, EditItemTemplate, InsertItemTemplate.
➢ Like DataList Control it does not provide the pre-defined layout for display and / or updating the data.
➢ Instead, When the page is rendered the FormView is also renders and the content of the tables are displayed.

1.9 Validation Controls, Login Control

➢ ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

➢ **ASP.NET provides the following validation controls:**
- ➢ RequiredFieldValidator
- ➢ RangeValidator
- ➢ CompareValidator
- ➢ RegularExpressionValidator
- ➢ CustomValidator
- ➢ ValidationSummary

BaseValidator Class

- The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

- The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate" runat="server"
  ControlToValidate ="ddlcandidate"    ErrorMessage="Please choose a candidate"
InitialValue="Please choose a candidate"></asp:RequiredFieldValidator>
```

RangeValidator Control

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

It has three specific properties:

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.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
  ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
  MinimumValue="6" Type="Integer">
</asp:RangeValidator>
```

CompareValidator Control

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

It has the following specific properties:

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.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

RegularExpressionValidator

- 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.

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.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
  ValidationExpression="string" ValidationGroup="string">
</asp:RegularExpressionValidator>
```

CustomValidator

- 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.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
  ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
</asp:CustomValidator>
```

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.

The syntax for the control is as given:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
   DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

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.

Example

- The following example describes a form to be filled up by all the students of a school, divided into four houses, for electing the school president. Here, we use the validation controls to validate the user input.