

Unit 1: Introduction to Asp.Net

Web programming is the process of creating dynamic content for websites and web applications using various technologies. To get started, it is important to understand the basic terminology that forms the foundation of how the web works. These concepts help developers communicate clearly, write effective code, and build functional, user-friendly websites.

- Some Basic terminologies are listed below.

Website	A collection of related web pages accessed via a browser
Web Page	A single document on the web, typically written in HTML
HTML (HyperText Markup Language)	The structure/language of web pages
CSS (Cascading Style Sheets)	Used to style HTML elements (colors, layout, - fonts)
JavaScript	Programming language for web interactivity
Browser	Software to access and view websites (e.g., Chrome, Firefox)
Client-Server Model	The structure where the browser (client) requests data from a server
Web Server	A computer or software that delivers web pages to clients
HTTP/HTTPS	Protocols used to transfer web data
URL (Uniform Resource Locator)	The address of a web page
Front-end	Part of the website the user interacts with (HTML, CSS, JS)
Back-end	Server-side logic, databases, and application code (e.g., ASP.NET, PHP)
Database	Stores data used by web apps (e.g., SQL Server, MySQL)

1.1 Concepts of Asp.Net

ASP.NET (Active Server Pages .NET) is a web development framework developed by Microsoft. It is used to build dynamic websites, web applications, and web services using the .NET platform. It is widely used for building robust and scalable web applications, ranging from small websites to large enterprise-level systems. It offers a rich set of features, security enhancements, and integration with other Microsoft technologies, making it a popular choice among developers.

Features of ASP.NET include:

- **Server-Side Programming:** ASP.NET enables developers to write server-side code using programming languages such as C# (C Sharp) or Visual Basic .NET (VB.NET). This code runs on the web server and generates dynamic HTML pages that are sent to the client's web browser.
- **Common Language Runtime (CLR):** ASP.NET applications are executed within the Common Language Runtime, a component of the .NET Framework. CLR provides memory management, security, and other essential services for running the ASP.NET code.
- **Web Forms:** ASP.NET Web Forms is a feature of the framework that allows developers to build web applications using a visual drag-and-drop interface. It provides a set of controls, such as buttons, textboxes, and data grids, which can be easily placed on a web page and associated with server-side code.
- **Model-View-Controller (MVC):** ASP.NET MVC is an architectural pattern provided by ASP.NET. It separates the application logic into three components: the model (data and business logic), the view (user interface), and the controller (handles user input and coordinates the model and view). MVC offers more control and flexibility for web application development.
- **ASP.NET Core:** ASP.NET Core is the **latest version** of the framework, which is an **open-source and cross-platform framework**. It provides improved performance, modular architecture, and support for modern web development practices. ASP.NET Core can be used to build web applications that can run on Windows, macOS, or Linux.

1.1.1 Page Life cycle

- When the user request the page, asp.net checks that the **class for page** is available or not. If it is present, then it creates the instance of the page class and gives response. But if the class is not present then it goes under the page life cycle.
- When the request comes the ASPX page is given for parsing to the ASPX engine. Then from that parsed page and the code behind class, a combined page class is generated and it is compiled. At last the instance of the class is created and rendered for the user. In short, web forms are in the form of page class.

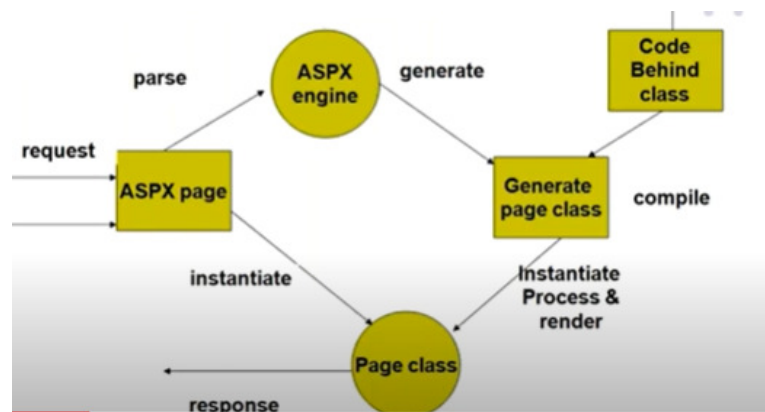


Figure 1: Page Life Cycle

Phase of Asp.Net Page life cycle (IMP)

1. **Page request:** the page request occurs before the page life cycle starts. It checks that the page needs to be parsed and compiled or not.
2. **Page Start:** in this phase, several properties such as **Response, Request, and IsPostBack** are set. Along with, it checks that the request is the postback (the contents of the form are posted back or itself) request or not.
3. **Page Initialization:** During this phase, the controls are available on the page and its **ID property** is set then **themes** are applied. If the request is the postback request, the postback data are not loaded and the values of the controls are not restored from the viewstate.
4. **Page Load:** If the request is the postback request, the postback data is loaded and the values of the controls are not restored from the viewstate.
5. **Validation:** if the validation controls are present on the page then its **validate ()** methods is called, which in turn set **IsValid ()** property of the controls and page.

6. **Postback Event handling:** if the event handlers of the PostBack are need to be called, and then they are called during this phase.
7. **Rendering:** Prior to this phase, the view state for page and controls are saved, during rendering phase **Render ()** method of the controls and page is called which gives the **output** to the **OutPutStream** of the Response Property.(Ready to load).It occurs when all the response information has been sent back to the user. It also stores all the information that is being sent.
8. **Unload** when there is a need to dispose (close) the page as it might not be further needed. A request and response property of the page is also unloaded.

Events of Asp.Net page life cycle

1. **PreInit:** it occurs before the page is initialized. The Page_PreInit event allows you to perform tasks such as dynamically creating or modifying controls before the view state is loaded and before the postback data is processed. Use this event for the following.
 - Check the **IsPostBack** property to determine whether this is the first time the **page is being processed**.
 - Create or recreate **dynamically** controls.
 - Set a **master page, Theme** Property dynamically.
 - Read or set **profile property (server side state mgt technique)** values.
2. **Init:** This event occurs when the page is initialized. You can use this event to initialize controls, set up database connections, or perform other initialization tasks.
3. **PreLoad:** it occurs before the page_load event. It provides a useful opportunity to perform additional tasks or data operations before the page is loaded with data and events are processed.
4. **Load:** This event occurs when the page is loaded into memory. It is a common place to put code that binds data to controls or performs other tasks that should occur on every page load.
5. **Control events:** These events occur when a **user interacts with controls** on the page, such as button clicks or dropdown list selections. Examples of these events include Button_Click, SelectedIndexChanged, etc.
6. **PreRender:** This event occurs just before the page is rendered to the client. You can use this event to make final modifications to the page or its controls.

7. **Render:** This event occurs when the page is rendered to the client. It is the last event in the page life cycle and usually not used for writing custom code.
8. **Unload:** This event occurs when the page is unloaded from memory. It is used to release resources or perform cleanup tasks.

Example:

```
Protected Sub Page_Init(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Init
```

```
    Response.Write("2")
```

```
End Sub
```

```
Protected Sub Page_InitComplete(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.InitComplete
```

```
    Response.Write("3")
```

```
End Sub
```

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
```

```
    Response.Write("5")
```

```
End Sub
```

```
Protected Sub btnsave_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnsave.Click
```

```
End Sub
```

```
Protected Sub Page_LoadComplete(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.LoadComplete
```

```
    Response.Write("6")
```

```
End Sub
```

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.PreInit
```

```
    Response.Write("1")
```

```
End Sub
```

```
Protected Sub Page_PreLoad(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.PreLoad
```

```
    Response.Write("4")
```

```
End Sub
```

Protected Sub Page_PreRender(ByVal sender As Object, ByVal e As System.EventArgs)

Handles Me.PreRender

Response.Write("7")

End Sub

1.1.2 Asp.net files types and subdirectory.

File type	Location	Description
.asmx	Application root or a subdirectory.	An XML Web services file that contains classes and methods that can be invoked by other Web applications.
.aspx	Application root or a subdirectory.	An ASP.NET Web Forms page that can contain Web controls and presentation and business logic.
.config	Application root or a subdirectory.	A configuration file contains XML elements that represent settings for ASP.NET features.
.master	Application root or subdirectory.	A master page that defines the layout for other Web pages in the application.
.mdf	App_Data subdirectory.	A SQL Server Express database file.
.sitemap	Application root.	A sitemap file that defines the logical structure of the Web application. ASP.NET includes a default sitemap provider that uses sitemap files to display a navigational control in a Web page.
.skin	App_Themes subdirectory.	A skin file that contains property settings to apply to Web controls for consistent formatting.
.sln	Visual Studio project directory.	A solution file for a Visual Studio project.

1.2 .Net Framework

- The .NET Framework is a software framework developed by Microsoft that mainly runs on Microsoft Windows. It provides a large library of pre-built code and a runtime

environment for creating and running applications. It supports multiple programming languages and allows developers to build a wide range of applications, including desktop applications, web applications, and much more.

Architecture of .Net Framework

- The .Net framework consists core components Common Language specification, Common Language Runtime, Base class library. It also supports different types of languages for programming. Figure 2 provides an illusion of the .Net Framework.

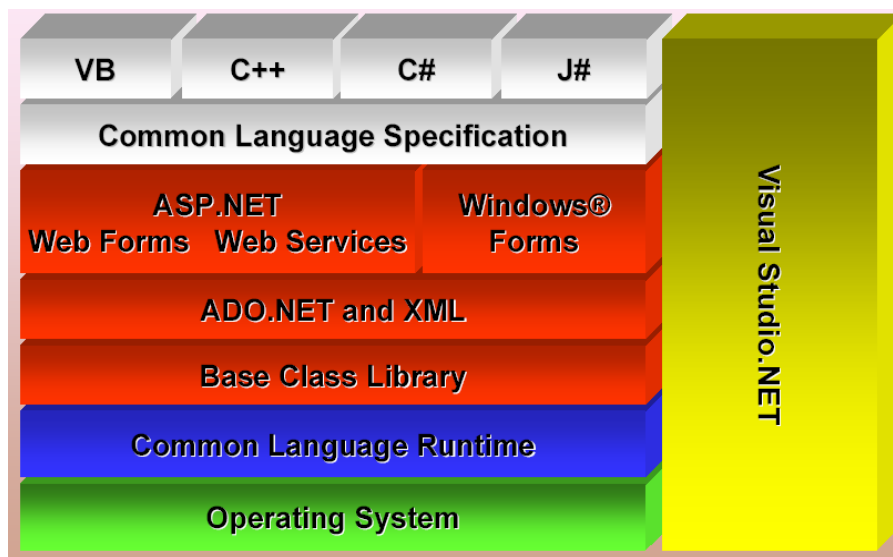


Figure 2: Components of .Net framework

The .NET Framework consists of:

(1) Common Language Runtime (CLR)

- It is the execution environment of the .NET Framework. It is the heart and engine of .net framework. When a .NET application is compiled, it generates an intermediate language code called Common Intermediate Language (CIL) or Microsoft Intermediate Language (MSIL) which is language independent. When the MISL code is executed, the CLR compiles it into machine code (Native Code) and managing the execution of the resulting program. The code which runs under the CLR is called as Managed Code. It provides various services, including automatic memory management, exception handling, security, and type safety.

Execution in CLR

- The provided figure illustrates the programming execution flow when you develop programs in various programming languages, such as Vb, C#, and C++.

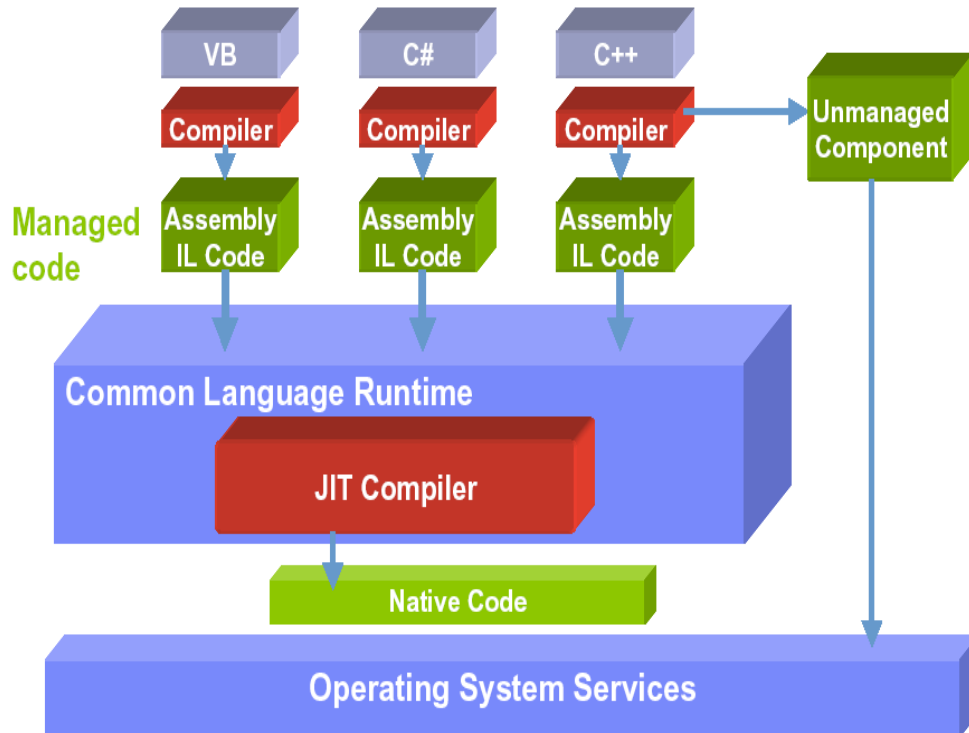


Figure 2: Program execution under CLR

Microsoft Intermediate Language

- Microsoft Intermediate Language (MSIL) also known as Intermediate Language (IL) or Common Intermediate Language (CIL). It is also called Managed Code. While your program is running, the .NET Compiler will generate MSIL, and the Just in Time (JIT) compiler will translate your Intermediate Language into machine code (Native Code). It is a CPU-independent set of instructions that can be efficiently converted to the native code. When a .NET compiler produces MSIL, it also produces Metadata (Data about Data). Metadata is nothing but a description of every namespace, class, method, Property etc. The MSIL and Metadata are contained in a portable executable (PE) file. The MSIL includes instructions for loading, storing, initializing, and calling methods on objects.

(2) Base Class Libraries (BCL)

- It is also known as framework class library (FCL). It is the object-oriented collection of reusable types. It is a Library of prepackaged functionality and Applications. It

provides classes which encapsulate a number of common functions, including file reading & writing, Graphics rendering, database interaction and XML document manipulation.

(3) ADO.Net and XML

- It is also known as data access layer. With the help of this layer, we can access relational databases. It works with XML and provides the disconnected Data Model. It is a part of BCL.it consists of Data Provider and data Set.

(4) Window Forms

- It is also known as Win Forms. It is used to create GUI for windows desktop application. it also provides integrated and unified way of developing GUI. It has a rich variety of windows controls and user interface support like Textbox, Button, Checkbox, Etc. Using visual Studio.NET, we can simply design the GUI by dragging the controls on a form.

(5) Web Forms & web Services

Web Forms

- It provides a tool for web application. It is a part of ASP. Net. It is the forms engine that provides Browser –based user interface. Web Forms are similar to Windows Forms in that they provide properties, methods, and events for the controls that are placed onto them. However, these UI elements render themselves in the appropriate markup language required by the request, e.g. HTML. If you use Microsoft Visual Studio® .NET, you will also get the familiar drag-and-drop interface used to create your UI for your Web application.

Web Services

- Web services are the applications that run on a web server and communicate with other application. It uses a series of XML based communicating protocols that respond to different requests.
- The protocols on which web services are built summarized below:
 - UDDI (Universal Discovery and Description Integration)
 - WSDL (Web services Description Language)
 - SOAP (Simple Object Access Protocol)
 - XML (Extensible Markup Language)
 - HTTP (Hypertext Transfer Protocol)
 - SMTP (Simple Mail Transfer Protocol)

(6) The Common Language Specification (CLS)

- It is a set of rules and constraints that all language must follow which want to be compatible with .NET framework. It is used to support the theme of .NET i.e. unification and interoperability (The ability of computer systems or software to exchange and make use of information). That means, if we want the code which we write in a language to be used by programs in other language (cross-language integration) then it should hold on to the CLS. Thus the CLS describes a set of features that are common different languages.

CLS performs the following functions:

- Establishes a framework that helps enable cross-language integration, type safety, and high-performance code execution. Provides an object-oriented model that supports the complete implementation of many programming languages. Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

1.3 Compile Code

When you write ASP.NET code, it is in human-readable text. Before ASP.NET can run your code, it has to convert it into something that the computer can understand and execute. The process of converting source code into machine code is called compilation. There are two models for compilation process that listed below.

1. Inline Code:

It will provide mix layout means both designing and coding will be done in a single.aspx file using <Script> tag.

Disadvantage:

- It is difficult to debugging when program is very large.
- It is difficult to find the error.

2. Code behind:

It will provide separate layout for designing and coding. Here two separate file is created one for designing (.aspx) and another for coding (.aspx.vb, .aspx.c depending on language).

1.4 The Common Language Runtime

- .NET Framework provides runtime environment called Common Language Runtime (CLR). It is the heart of .net framework. It is the engine that compiles and run the application. It uses MSIL code which is language independent for execution. The MSIL code is translated by JIT compiler.
- It provides an environment to run all the .NET Programs. The code which runs under the CLR is called as Managed Code. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.
- **Execution in CLR**

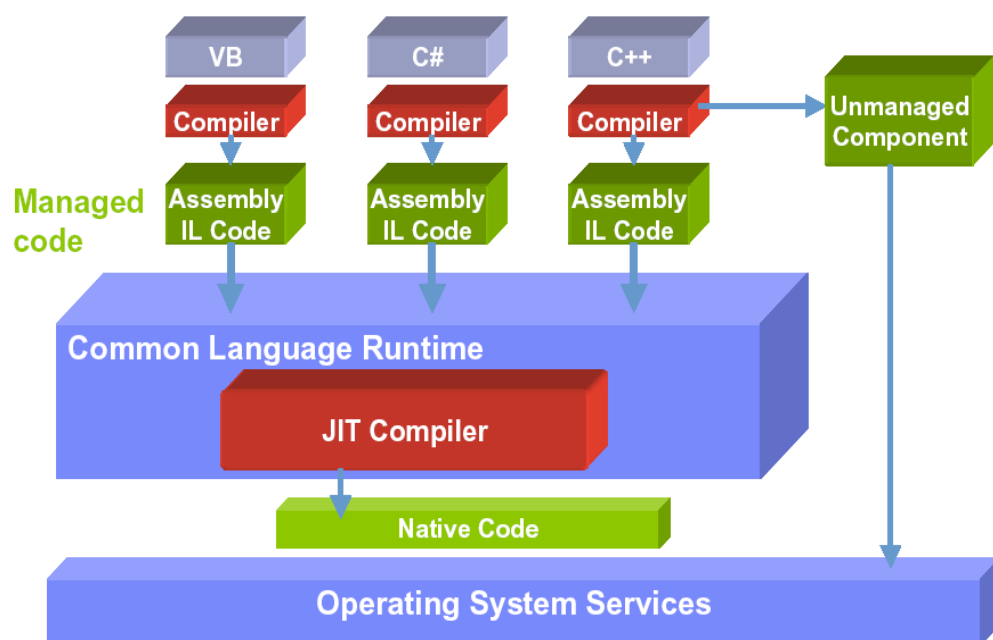


Figure 3 : Program execution flow under CLR

- When the .NET program is compiled, the output of the compiler is not an executable file but a file that constraints a special type of code is called Microsoft intermediate language, which is a low-level set of instructions understand by CLR.
- The MSIL defines a set of portable instructions that are CPU independence.
- It is the functionality of the CLR to translate this MSIL into native code when the program is executed, making the program to run in any environment for which the CLR is implemented. And that's how the .NET framework achieves Portability (run in any environment). This MSIL is converts into Native code using JIT (Just in Time) compiler.

Purpose of CLR

- Thread Support: Threads are managed under the Common Language Runtime. Threading means parallel code execution. Threads are basically light weight processes responsible for multi-tasking within a single application.
- COM Marshaler: It allows the communication between the application and COM objects.
- Type Checker: It will verify types used in the application with CTS or CLS standards supported by CLR, this provides type safety.
- Exception Manager: It handles all the runtime exceptions (Error) thrown by application
- Security Engine: It enforces security permissions at code level security, folder level security, and machine level security using Dot Net Framework setting and tools provided by Dot Net.
- Debug Engine: CLR allows us to perform debugging an application during runtime.
- MSIL: Microsoft Intermediate Language is considered to be the lowest form of human readable language. It is CPU independent and includes instructions of how to load, store, initialize objects. JIT converts this MSIL into native code which is independent on the CPU.
- Code Manager: CLR manages code. When we compile a .NET application you don't generate code that can actually execute on your machine. You actually generate Microsoft Intermediate Language (MSIL or IL). All .NET code is IL code. IL code is also called Managed Code, because the .NET Common Language Runtime manages it.
- Garbage Collector: It handles automatic memory management and it will release memory of unused objects in an application, this provides automatic memory management.
- Class Loader: as and when needed. It loads the class into the system memory.

1.5 Event Driven Programming

- It is a programming paradigm in which the flow of the program is determined by events—such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs.
- It does not follow the predefined path for execution.
- It includes following things:
 - **Properties** mean characteristics of anything. Example Laptop, Mobile, Car Etc.
 - **Events** mean an interaction between the user and application or between the application and the system. (e.g., a button clicks).
 - **Methods** are nothing but function or procedure. Doing something is method.

1.6 Server Controls

- **Server control** is a special type of control that runs on the **server-side** and helps generate HTML output that is sent to the client's browser.
- These controls are executed on server side and allow writing code using different languages. It must have runat attribute with value “server”. (runat=“server”)
- There are two types of server controls one is HTML server controls and another is web server controls.
- All web server controls have following default property
 - ID="txtName": Unique identifier for the control (used in code-behind)
 - runat="server": Required for all ASP.NET server controls
- Here Some to the web server controls are listed below:

1.6.1 Label

- It is used to display the information (Text) on Web Forms. Any end User can not Edit (change) the Label Information.

- **Syntax:**

<code><asp:Label ID="lblname" runat="server"></asp:Label></code>
--

- **Properties:**

Property	Description
----------	-------------

Text	The text to be shown for the label
Font	It is used to sets the font of the Label text.
ForeColor	It is used to sets the text color in the Label.
Height	It is used to Specify the height of the Label Control.
BackColor	It is used to Sets the Background color of the Label control.
BorderWidth	It is used to Sets the Border color of the Label control.
BorderStyle	It is used to Sets the Style of the Label control.

1.6.2 TextBox

- It is used to enter any kind of data like numeric,alphabet etc.
- **Syntax:**

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

- **Properties:**

Property	Description
Text	It is used to gets or sets the text.
TextMode	Controls how text is entered (SingleLine, MultiLine, Password)
Rows	It is used to sets the number of rows display in a TextBox(Multiline).
MaxLength	Maximum number of character allowed
ReadOnly	Makes the text box as Readonly.
AutoPostBack	It is used to handle the event when the TextBox control loses focus.
Wrap	It gets or sets a value indicating whether the text content wraps within a multiline textbox

- **Event:**

TextChanged	It occurs when the end user change the text of the TextBox control.
-------------	---

1.6.3 Button

- It is used to create an event and send request to the web server.
- There are three types of button which are listed below
 1. Button: It displays text within a rectangular area.
 2. Link Button: It displays text that looks like a hyperlink.
 3. Image Button: It displays an image.
- **Syntax:**
- **Properties:**

Property	Description
Text	It is used gets or sets the text to be displayed on the Button Control.
CauseValidation	Determines whether page validation occurs when a user clicks the button. The default is true.
PostBackUrl	The URL of the page that is requested when the user clicks the button
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.

Event:

Click	It is an Event that occurs when the Button is clicked.
-------	--

Example: WAP that takes a user's name as input and displays it on a Label control.

Design the following web form as shown below:

Textbox ,button and label control Example

 Label

Click

Source Code

```
ProtectedSub btnsubmit_Click(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles btnsubmit.Click  
    Dim str As String  
    str = txtname.Text  
    Label1.Text = str  
EndSub
```

1.6.4 Checkbox

- It is used when user have to enter multiple option as input. For example hobbies, Languages.
- It allows the user to select or deselect an option (true/false, yes/no).
- **Syntax**

```
<asp:CheckBox ID="chkAgree" runat="server" Text="I agree to the terms" />
```

- **Properties:**

Property	Description
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to gets or sets text to be shown for the control.
Visible	To set visibility of control on the form.
Checked	It is used to set check state of the control either true or false

Autopostback	Whether a server control causes the page to post back to the server automatically when its value changes.
--------------	---

- **Event:**

Checkchanged	It occurs when someone change the CheckBox true or false state.
--------------	---

Example: WAP that allows the user to select multiple hobbies and displays the selected hobbies on a Label control.

- Design the following web form as shown below:

Source Code:

```
Partial Class Default3
    Inherits System.Web.UI.Page
    Dim strhobbies As String
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnshow.Click
        If chkmusic.Checked = True Then
            strhobbies = chkmusic.Text
        End If
        If chkreading.Checked = True Then
            strhobbies &= chkreading.Text
        End If
        If chktravelling.Checked = True Then
            strhobbies &= chktravelling.Text
        End If
        If chkgaming.Checked = True Then
```

```
        strhobbies &= chkmusic.Text  
    End If  
    lblmsg.Text = strhobbies  
End Sub  
End Class
```

1.6.5 Image Map

- It is used to make image map in a picture on a web page that provides various links called hotspots (clickable areas) to navigate to other web pages depending on the place where the user clicks.
- This control is used to display a map of any country. When a user clicks on specific state of the map, the control navigates to a URL that provides additional data about the selected state.

- **Syntax**

```
<asp:ImageMap ID="ImageMap1" runat="server"> </asp:ImageMap>
```

- **Properties**

Property	Description
HotSpots	It defines hotspots to specify interactive regions on an image. These hotspots can be rectangles(Properties Top,Right,Bottom,Left), circles(Properties Radius X,Y), polygons (properties(Coordinates)
HotSpotMode	It specifies behaviors for the HotSpot objects when the HotSpot objects are clicked. Its value can be Navigate orPostBack Or Inactive.

- **Example**

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="Navigate"
    ImageUrl="~/aspnet_imagemap.jpg">
    <asp:RectangleHotSpot Top="10" Left="20" Bottom="100" Right="120"
        NavigateUrl="~/Default2.aspx" />
    <asp:CircleHotSpot X="200" Y="150" Radius="50"
        NavigateUrl="~/Default4.aspx" />
</asp:ImageMap>
```

1.6.6 Link Button

- It is used when you want to displays text that looks like a link or hyperlink

- **Syntax**

```
<asp:LinkButton ID="LinkButton1" runat="server">LinkButton</asp:LinkButton>
```

- **Properties**

Property	Description
Text	The text to be shown for the link.
PostBackUrl	The URL to post to when the button is clicked.

- **Event**

Click	It is an Event that occurs when the Button is clicked.
-------	--

- **Example: WAP to navigate from one page to another using a LinkButton control**

Design the following web form as shown below

[Google](http://www.google.com)

Source Code:

```
Protected Sub LinkButton1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles LinkButton1.Click
    Response.Redirect("http://www.google.com")
End Sub
```

1.6.7 RadioButton

- It is used to select any one option from multiple. For example Gender, Marital Status.
- you must set the GroupName property to ensure that only one option can be selected at a time.

- **Syntax:**

- **Properties:**

Property	Description
Checked	It Indicate the radiobutton control is checked(true) or unchecked(false)
GroupName	It is used to make a group of radiobutton. If we assign same value to more than one radiobutton then we can select only one radiobutton from that group of radiobutton

- **Event:**

Checkchanged	It occurs when someone change the RadioButton true or false state.
--------------	--

- **Example: WAP that allows the user to select their gender and displays the selected option on a Label control.**

Design the following web form as shown below:

Gender	<input type="radio"/> Male <input type="radio"/> Female
	<input type="button" value="Show"/>
	Label

Source Code:

```
Protected Sub btnshow_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnshow.Click
    If rdbmale.Checked = True Then
        lblmsg.Text = rdbmale.Text
    ElseIf rdbfemale.Checked = True Then
        lblmsg.Text = rdbfemale.Text
    Else
        lblmsg.Text = "Select any one option"
    End If
End Sub
```

1.7 PostBack

- A Post back 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.
- To determine whether the page is posted back or it is the first request for the page, you can use IsPostBack property of the page. Once the page is posted back, IsPostBack property becomes true.
- IsPostBack Property is false first time the page is loaded and is true when the page is submitted and processed.

Example

```
ProtectedSub btnclick_Click(ByVal sender AsObject, ByVal e As System.EventArgs)
Handles btnclick.Click
    MsgBox("Hello India")
EndSub
```

```
ProtectedSub Page_Load(ByVal sender AsObject, ByVal e As System.EventArgs)
HandlesMe.Load
If IsPostBack = FalseThen
    MsgBox("Page Load")
EndIf
EndSub
```

- **AutoPostBack**

- It is the mechanism by which the page will be posted back to the server automatically based on some events in the web controls.
- The AutoPostBack property is used to set or return whether or not an automatic post back occurs when the user presses "ENTER" or "TAB" in the Textbox control. If this property is set to TRUE the automatic post back is enabled, otherwise FALSE. Default is FALSE.
- If we set autopostback property to true of any control then after processing on any control a request (postback) is send to the server.

- **Example:**

```
<form id="form1" runat="server">
<div>
<asp:DropDownList ID="ddlcity" runat="server" AutoPostBack="true" >
</asp:DropDownList>
<asp:Label ID="lblcity" runat="server" Text="Label"></asp:Label>
</div>
</form>

Protected Sub ddlcity_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ddlcity.SelectedIndexChanged
    lblcity.Text = ddlcity.SelectedItem.Text
End Sub
```

1.8 Data Binding

- Data binding is the process of retrieving information from databases, arrays, or collections. It allows data to flow between controls and data sources automatically or manually.
- **Types of Data binding:**
 - 4. Single value binding**

It is used to bind single value. For example label textbox.
 - 5. Multiple value binding.**

It is used to bind multiple values. For example Grid View, ListBox, Repeater, Form View
- Data binding can be done through programmatically binding means that the data for the control (like a GridView) is set **from code in the code-behind** (e.g., Default.aspx.vb) and declaratively binding means you bind the GridView directly in the .aspx file using controls like SqlDataSource, ObjectDataSource, etc.

1.8.1 GridView

- It is used to display data in tabular form on a web page. It displays data in both rows and columns, where each column represents a field, and each row represents a record.
- It provides built-in support for: Paging, Sorting, Editing, Deleting, and Selection.
- Syntax:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="true">  
  
</asp:GridView>
```

- Properties

Property	Description
AutoGenerateColumns	Automatically creates columns from data
DataKeyNames	Set primary key column(s)
AllowPaging	Enables pagination
AllowSorting	Enables column sorting
EditIndex	Sets index of row being edited
SelectedIndex	Sets index of selected row
PageSize	Number of rows per page

- **Event**

Name	Description
RowEditing	Triggered when Edit button clicked
RowUpdating	Handles update operation
RowDeleting	Handles delete operation
PageIndexChanging	Handles pagination

- **Example: Write down the following code in Default.aspx (Programmatically Binding)**

```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:GridView ID="GridView1" runat="server">
      </asp:GridView>
      <asp:Label ID="lblmsg" runat="server" Text="Label"></asp:Label>
    </div>
  </form>
</body>
```

- **Write down the following code in Default.aspx.vb**

```
Imports System.Data
Imports System.Data.SqlClient
Partial Class Default2
  Inherits System.Web.UI.Page
  Dim constring As String =
ConfigurationManager.ConnectionStrings("cns").ToString()
  Dim con As New SqlConnection(constring)
  Dim cmd As New SqlCommand
  Dim da As SqlDataAdapter
  Dim dt As DataTable
```



```

Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load

    Try
        con.ConnectionString = constring
        cmd.Connection = con
    Catch ex As Exception
        lblmsg.Text = ex.Message
    End Try

    If IsPostBack = False Then
        fillgrid()
    End If

End Sub

Sub fillgrid()
    Try
        cmd.CommandText = "select * from tblstudent order by rno"
        da = New SqlDataAdapter(cmd)
        dt = New DataTable
        da.Fill(dt)
        GridView1.DataSource = dt
        GridView1.DataBind()
    Catch ex As Exception
        lblmsg.Text = ex.Message
    End Try
End Sub
End Class

```

- **Another Example (Declarative binding)**

```

<body>
    <form id="form1" runat="server">

```

```

<div>
    <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
    DataKeyNames="rno" DataSourceID="SqlDataSource1">
        <Columns>
            <asp:BoundField DataField="rno" HeaderText="rno"
InsertVisible="False"
                ReadOnly="True" SortExpression="rno" />
            <asp:BoundField DataField="name" HeaderText="name"
SortExpression="name" />
            <asp:BoundField DataField="class" HeaderText="class"
SortExpression="class" />
            <asp:BoundField DataField="div" HeaderText="div"
SortExpression="div" />
        </Columns>
    </asp:GridView>

    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:ConnectionString %>"
        SelectCommand="SELECT * FROM [tblstudent]"></asp:SqlDataSource>

</div>
</form>
</body>

```

1.8.2 List Box

- It is used to create single or multi-selection drop-down list. Each item in a listbox control is defined by a ListItem element.
- Syntax:

```
<asp:ListBox ID="ListBox1" runat="server" ></asp:ListBox>
```

- **Properties**

Property	Description
DataSource	It is used to bind data (list, dataset, etc.)
DataTextField	Field to display as text
DataValueField	Field used as value
SelectionMode	Single or Multiple
AutoPostBack	

- Event:

SelectedIndexChanged	Occurs when the selection changes (used with AutoPostBack="true").
----------------------	--

- **Example: Design web form and write down the following code in your code behind file**

```
Imports System.Data
Imports System.Data.SqlClient
Partial Class Default2
    Inherits System.Web.UI.Page
    Dim constring As String =
ConfigurationManager.ConnectionStrings("cns").ToString()
    Dim con As New SqlConnection(constring)
    Dim cmd As New SqlCommand
    Dim da As SqlDataAdapter
    Dim dt As DataTable

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Try
            con.ConnectionString = constring
            cmd.Connection = con
        Catch ex As Exception
            lblmsg.Text = ex.Message
        End Try
        If IsPostBack = False Then
            Filllistbox()
        End If
    End Sub
End Class
```

```

End If
End Sub
Sub Filllistbox()
Try
    cmd.CommandText = "select * from tblstudent order by rno"
    da = New SqlDataAdapter(cmd)
    dt = New DataTable
    da.Fill(dt)
    ListBox1.DataSource = dt
    ListBox1.DataValueField = "rno"
    ListBox1.DataTextField = "name"
    ListBox1.DataBind()
Catch ex As Exception
    lblmsg.Text = ex.Message
End Try
End Sub
Protected Sub ListBox1_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged
    lblmsg.Text = ListBox1.SelectedItem.Text
End Sub
End Class

```

1.8.3 Data Binding Events

- Data Binding Events are used to bind data from a source (like a database, list, or object) to server-side controls (like GridView, Repeater, DropDownList, etc.). These events allow developers to customize how and when data is loaded and presented in the UI.

Event Name	Applies To	Description
DataBinding	All data-bound controls	Fires before the control are bound to data. You can modify data source here.
DataBound	All data-bound controls	Fires after data have been bound to the control. You can access bound data.
ItemDataBound	Repeater, DataList, ListView	Fires for each item during data binding (used to format or change item appearance).
RowDataBound	GridView	Similar to ItemDataBound, but specific to rows in a GridView.
SelectedIndexChanged	ListBox, DropDownList	Occurs when the selection changes (used with AutoPostBack="true").

1.8.4 Repeater

- It is a data-bound control used to display a repeated list of items that are bound to the control. It's more lightweight than other list controls like GridView or DataList and gives you full control over the HTML markup generated. It does not have any in built layout or styles.
- Paging, sorting or editing must be implemented manually.
- Syntax:

```
<asp:Repeater ID="Repeater1" runat="server">  
  </asp:Repeater>
```

- **Properties**

Property	Description
DataSource	The data source (e.g., List, DataTable, SqlDataReader)
DataSourceID	Binds to a SqlDataSource, ObjectDataSource, etc. by ID
DataMember	Used when DataSource has multiple data tables (like a DataSet)
Items	Collection of all RepeaterItem objects after binding

- **Events**

Event	Description
ItemDataBound	Fired for each item as it's data-bound
ItemCommand	Fired when a control inside ItemTemplate (like a Button) triggers a command

- **Repeater Templates**

Template Type	Description
HeaderTemplate	Rendered once at the top.
ItemTemplate	Rendered for each data item.
AlternatingItemTemplate	Used for alternating rows (like even/odd).
SeparatorTemplate	Renders between each item.
FooterTemplate	Rendered once at the end.

- **Example: write down the following code in .aspx page**

```

<form id="form1" runat="server">
    <div>
        <asp:Repeater ID="Repeater1" runat="server"
DataSourceID="SqlDataSource1">
            <HeaderTemplate>
                <table border="1">
                    <tr><th>Rno</th><th>Name</th></tr>
                </HeaderTemplate>

                <ItemTemplate>
                    <tr>
                        <td><%# Eval("rno")%></td>
                        <td><%# Eval("name") %></td>
                    </tr>
                </ItemTemplate>

                <FooterTemplate>
                    </table>
                </FooterTemplate>
            </asp:Repeater>

            <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                ConnectionString="<%$ ConnectionStrings:cns %>"
                SelectCommand="SELECT * FROM
[tblstudent]"></asp:SqlDataSource>

        </div>
    </form>

```

1.8.5 Formview

- It is a data-bound control that displays a single record at a time. It's commonly used for inserting, editing, viewing, and deleting records from a data source.
- Syntax:

```
<asp:FormView ID="FormView2" runat="server">  
    </asp:FormView>
```

- **Properties:**

Property	Description
DataSourceID	ID of the data source control (SqlDataSource, ObjectDataSource, etc.) FormView1.DataSourceID = "SqlDataSource1";
DataKeyNames	Primary key(s) for identifying a record DataKeyNames="Id"
DefaultMode	Default view mode: ReadOnly, Edit, or Insert DefaultMode="Edit"
AllowPaging	Enables paging for browsing multiple records. AllowPaging="True"
HeaderText, FooterText	Add static text header/footer (not templates) <HeaderTemplate>Header</HeaderTemplate>
DataSource	Programmatically assign data. FormView1.DataSource = myDataTable;

- Events

Event	Description
ItemCreated	When the FormView control creates a new row
ItemInserting	Before inserting a new record
ItemInserted	After a record is inserted
ItemUpdating	Before updating a record
ItemUpdated	After a record is updated
ItemDeleting	Before a delete operation
ItemDeleted	After a record is deleted
PageIndexChanging	When the user changes the
PageIndexChanged	After the page index changes

-
- Example: Write down the following code in .aspx file

```

<form id="form1" runat="server">
    <div>

        <asp:FormView ID="FormView1" runat="server"
            DataKeyNames="rno"
            DataSourceID="SqlDataSource1"
            AllowPaging="True">

            <ItemTemplate>
                <p>ID: <%# Eval("rno")%></p>
                <p>Name: <%# Eval("name")%></p>
            </ItemTemplate>
        </asp:FormView>

        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:cns %>"
            SelectCommand="SELECT * FROM
[tblstudent]"></asp:SqlDataSource>

```

```
</div>  
</form>
```

1.9 Validation Controls, Login Controls

Validation Controls:

- It is used to validate user inputs are correct, complete, and secure before processing.
- There are two types of validation controls: server-side and client-side validation controls.

- **Common Properties of Validators:**

Property	Description
ControlToValidate	ID of the control to validate
ErrorMessage	Error text shown in ValidationSummary
Text	Inline error symbol or message
Display	"Static", "Dynamic", or "None"
EnableClientScript	Enables client-side JavaScript validation
SetFocusOnError	Set true or false for focus on the control when the error is displayed.

- **There are six types of validation controls that are listed below:**

Control	Purpose
RequiredFieldValidator	Ensures a field is not left empty
CompareValidator	Compares values of two inputs (e.g., password and confirm password).
RangeValidator	Ensures value is within a specified range (e.g., age between 18 and 65).
RegularExpressionValidator	Validates input against a regular expression pattern (e.g., email format, phone number).
CustomValidator	Allows custom server-side and/or client-side logic
ValidationSummary	Displays a summary of all errors in one place

1.9.1 Types of validation controls

1.9.1.1 RequiredFieldValidation Control

- It is used to make an input control a required field.
- It checks whether the control is **empty or not** when the form is submitted. Multiple validators can also be associated with the same input control.

Properties	Description
Initial Value	It gets or sets the initial value of the field.

- Example

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
    ControlToValidate="txtname" ErrorMessage="Enter Proper Name"
    SetFocusOnError="True" ForeColor="Red">*</asp:RequiredFieldValidator>
```

1.9.1.2 CompareValidator Control

- It is used to compare the value entered by the user in an input control, such as a TextBox control, with the value entered in another input control or constant value.
- It passes validation if the value of the input control matches the criteria.

Properties	Description
Type	It defines the type of the data.
ControlToCompare	It gets or sets the input control to compare with input control being validated.
ValueToCompare	It gets or sets the constant value to compare with value entered by the user in the input control being validated.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

- Example

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToCompare="txtcpwd" ControlToValidate="txtpwd"
```

```
ErrorMessage="Password does not match with confirm password"
SetFocusOnError="True" ForeColor="Red">*</asp:CompareValidator>
```

1.9.1.3 RangeValidator Control

- It is used to tests whether the value of an input control is within a specified range or not.

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String
MinimumValue	It gets or sets the minimum value of the range.
MaximumValue	It gets or sets the maximum value of the range.

- Example

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="txtage" ErrorMessage="Age must be > 1 and
    less < 100" ForeColor="Red" MaximumValue="100" MinimumValue="1"
    SetFocusOnError="True" Type="Integer">*</asp:RangeValidator>
```

1.9.1.4 RegularExpressionValidator Control

- It is used to check whether the value of an input control matches a pattern defined by a regular expression.
- This type of validation allows you to check **e-mail addresses, telephone numbers, and postal codes.**

Properties	Description
ValidationExpression	It gets or sets regular expression that determine the pattern used to validate filed. There are various predefine expression available. So you can use it.

- Example

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
    ControlToValidate="txtemail" ErrorMessage="Enter proper email id"
    ForeColor="Red" SetFocusOnError="True"
    ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-
.\w+)*"*)></asp:RegularExpressionValidator>
```

1.9.1.5 CustomValidate

- It allows you to create your own validation logic, either on the client side (JavaScript) or server side (VB). It's useful when built-in validators (like RequiredFieldValidator or RangeValidator) don't meet your needs.
- ServerValidate() event is used to perform validation.
- The value of the input control that is to be validated can be accessed by using the **Value** property of the **ServerValidate EventArgs** object passed into the event handler as a parameter.
- Example

```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal
args As System.Web.UI.WebControls.ServerValidateEventArgs) Handles
CustomValidator1.ServerValidate
    Dim age As Integer = Val(args.Value)
    If age >= 1 And age <= 100 Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

1.9.1.6 ValidationSummary

- It is used to summarize the error messages from all validators on a Web page in a single location.
- You can also summarize the error messages from particular group of validators using ValidationGroup property.

Properties	Description
DisplayMode	It gets or sets the display mode of the validation summary(List,BulletList,Sinlgeparagraph)
ShowMessageBox	It gets or sets a value indicating whether the validation summary is displayed in message box or not.
ShowSummary	It gets or sets a value indicating whether the validation summary is displayed or not.

1.9.2 Login Controls

- It is set of built-in web server controls to simplify the process of implementing authentication (logging users in and out) called forms authentication and user account management in web applications. When we use in built login controls, they use ASPNETDB.mdf database which is automatically crated with the default tables.
- They help you manage user login, registration, password recovery, and other membership tasks easily without writing a lot of code.
- **Common Login Controls which are listed below:**

1.9.2.1 Login

- It is used to displays a standard login form which includes fields for username, password, remember me checkbox, and a login button.
- **Properties**

Properties	Description
DestinationPageUrl	It is use to redirects after successful login.
FailureText	It is used to display message on failed login.
LoginButtonText	It is used to set the login button text.
LoginButtonType	It is used to set the login button type.
Orientation	It is used to set the orientation of the form(Horizontal, Vertical)
PasswordLabelText	It is used to set password label text.
RememberMeText	Indicates if "Remember Me" is selected.
Password	It retrieves the password entered by the user

- **Events**

Event	Description
-------	-------------

Authenticate(default)	Used to write custom logic for validating credentials.
LoggedIn	It is used when the user logs in the web site and is authenticated.
LoginIn	It is used when the user submits the login information.
LoginError	It is used when a login error is detected.

1.9.2.2 LoginView

- It is used to display different view based on the user's authentication status (logged in or anonymous). It gives you the ability to customize your content of your web application for the needs of different users.
- The control displays one of two templates: AnonymousTemplate or LoggedInTemplate.
- **Properties**

Properties	Description
AnonymousTemplate	Displayed if the user isn't logged in
LoggedInTemplate	Displayed if the user is logged in

Events:

Event	Description
ViewChanged	It is used when the view is changed

1.9.2.3 LoginStatus

- It is used to display a login link for users who are not authenticated and logout link for users who are authenticated. The login link takes the user to a login page and logout link resets the current user's identity to an anonymous user.
- **Properties**

Properties	Description
LoginText	It is used to get or set login link text.
LogoutText	It is used to get or set logout link text.
LoginPageUrl	It is used to get or set login page URL.
LogoutPageUrl	It is used to get or set logout page URL.

- **Events:**

Event	Description
LoggingOut	It raises the event when the user clicks the logout link of the control.

-

1.9.2.4 *LoginName*

- It is s used to display the username of the currently logged-in user.

- **Properties**

Properties	Description
FormatString	It is used for custom formatting.

1.9.2.5 *PasswordRecovery*

- It helps to recover their password. It typically works by asking for the user's username or email address, then sending a password reset link or showing a security question, depending on the configuration. It sends an e-mail message containing a password to the user.
- It supports the following three types of templates:
 - **UserName view** - Asks the user for his or her registered user name.
 - **Question view** - After Username validated then it will ask for security question and answer. This Question and answer is optional.
 - **Success view** - After successful validation of security questions and answers it displays a message informing to user that his password has been emailed.

- **Properties**

Properties	Description
AnswerLabelText	The text for the “Answer” label field
GeneralFailureText	The text for the “Failure” text field if the password could not be retrieved..
QuestionLabelText	The text for the “Secret Question” label.
SuccessPageUrl	The URL for the Success page if the password has been recovered.
SuccessText	The text for the message if the password retrieval is successful, and this is not displayed if the “SuccessPageUrl” is provided.
MailDefinition	It uses the email server configured by the smtp element in the

	web configuration file.
--	-------------------------

- **Events**

Event	Description
SendingMail()	It allows you to customize the email message before it's sent to the user.

1.9.2.6 ChangePassword

- It allows authenticated users to securely change their existing passwords.
- It supports the following two types of templates:
 - **ChangePasswordTemplate** which contains only 3 controls those are old password, new password, confirm new password.
 - **SuccessTemplate** which defines the messages after a user password has been successfully changed.

- **Properties**

Properties	Description
CancelButtonText	The text for the Cancel button.
ChangePasswordButtonText	The ChangePassword button text
ChangePasswordTitleText	The text for the “ChangePassword” title area.
DisplayUserName	It is used to display the username.
SuccessText	The text for the message if the password change is successful, and this is not displayed if the “SuccessPageUrl” is provided.

- **Events**

Event	Description
ChangedPassword	It is raised after a user's password is successfully changed..

1.9.2.7 CreateUserWizard

- It provides ready-to-use interface for user registration. by default, it adds the new user to the asp.net membership system. It supports a large number of properties that enable you to modify the appearance and behavior of the control.
- It supports the following two types of templates:
 - **Sign up for New Account** which is used to create new account for the user.

- **Complete** which is used to redirect the page after successfully created the account.

- **Properties**

Properties	Description
UserName	Gets the username entered by the user.
Email	Gets the email entered by the user.
Password	Gets the password entered by the user.
Question	Gets the password recovery question (if enabled).
Answer	Gets the answer to the recovery question (if enabled).
ContinueDestinationPageUrl	URL to redirect after account creation.
CreateUserStep	Accesses the CreateUserWizardStep template.
CompleteSuccessText	Message shown after account is successfully created.

- **Events**

Event	Description
CreatedUser	It is raise after the user account is successfully created..

1.10 Master Page, CSS, Themes

1.10.1 Master Page

- It is a template used to define a common layout and structure for multiple content pages in a web application. It allows you to create a consistent look and feel (e.g., header, navigation, footer) across your entire site.
- The **@Master directive** is defines in the master page. The master page contains one or more **<asp:ContentPlaceHolder>** for an individual content. The id attribute identifies the placeholder from all present in a web page. The master page has **.master extension**.
- You can bind the master page to the web form which defines the content of the place holders of the master pages. These web forms are known as content pages. The binding is done with help of MasterPageFile attribute of the @Page directive.
- The syntax of the master directive is as shown below:

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
Inherits="MasterPage" %>
```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Web pages using master page</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>

```

- **Steps to create master Page and to bind web form.**

- Go to File → New → Website → Asp.Net Empty Website → Give proper name to your web site → click on ok button
- This will creates two files (MasterPage.master and MasterPage.master.vb)
- Design master page according to your requirement.
- Right click on your project → Add New Item → Web Form → select your master page → Choose your master Page → ok
- Write Content for the Content Page

```

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">

```

```
</asp:Content>  
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"  
Runat="Server">  
    <h2>Welcome to the Home Page</h2>  
    <p>This content is loaded into the master page layout.</p>  
</asp:Content>
```

- Run the Application

- **Advantages**

- **Code reuse:** Define layout once, reuse everywhere.
- **Consistency:** All pages share a uniform design.
- **Maintainability:** Update the layout in one place.
- **Separation of concerns:** Layout in master page, content in individual pages.

1.10.2 CSS (Cascading style sheets)

- It is used for defining styles to display the elements written in a mark-up language. It helps user separate the HTML or XHTML content from its style. The separation provides flexibility, faster accessibility to content, reduces complexity and repetition of data.

- There are three ways to create CSS that are listed below:

- **External CSS**

- CSS styles are written in a separate .css file and linked to the web page.
- Example : Write down the following code in css file.

```
body
{
    background-color: #CCCC00;
}
```

- Reference the external CSS file in the <head> section of the web form.

```
<head runat="server">
    <title>External CSS in ASP.NET</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
```

- **Internal CSS**

- It is a method of applying styles directly within an HTML (or ASP.NET) document using a <style> tag in the <head> section.
- It is used for only one page.
- Example:

```
<head runat="server">
    <title>Internal CSS in ASP.NET</title>
    <style>
        body
        {
            background-color: #C0C0C0;
        }
    </style>
```

```
</head>
```

- **Inline CSS**

- It is a way to apply CSS styles directly inside an HTML element using the style attribute.
- Example:

```
<form id="form1" runat="server">
<div>
<p style="color: red; font-size: 18px;">This is a inline Css Example</p>
</div>
</form>
```

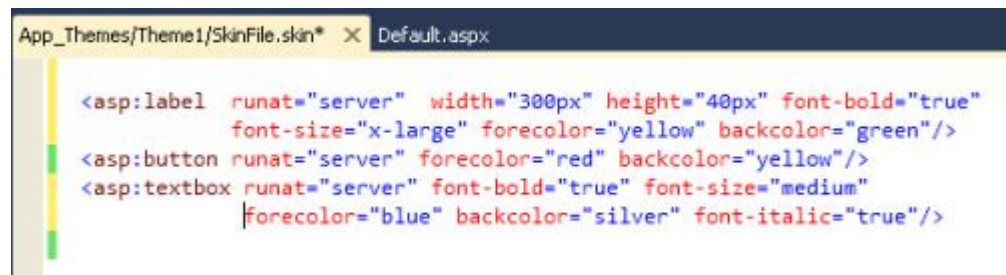
1.10.3 Themes

- It decides the look and feel of the website. It is a collection of files that define the looks of a page. It can include skin files, CSS files & images.
- We define themes in a special App_Themes folder. Inside this folder is one or more subfolders named Theme1, Theme2 etc. that define the actual themes.
- There are 3 different options to apply themes to our website:
- **Page Level**
 - Applied to individual pages.
 - Stored in the App_Themes folder.
 - Set using the Theme or StyleSheetTheme property in the page directive.
 - Example: <%@ Page Theme="Summer" %>
- **Global level**
 - Applies to all pages in all applications.
 - To set the theme for the entire website you can set the theme in the web.config of the website. Open the web.config file and locate the <pages> element and add the theme attribute to it:
<pages theme="Summer" />
- **Coding Level**
 - It should be applied in the page's life cycle i.e. Page_PreInit event should be handled for setting the theme

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles Me.PreInit  
    Page.Theme = "Summer"  
End Sub
```

- **Steps to create theme**

- Right-click in the application name in the Solution Explorer window and select Add
ASP.NET Folder->Theme from the context menu.
- A sub folder named Theme1 is automatically created inside the **APP_Themes** folder in the Solution Explorer.
- Right-click on the Theme1 folder and select the **Add New Item** option.
- Select the Skin file and click the Add button.
- Now write the following code in the skin file.

A screenshot of a code editor window with two tabs: 'App_Themes/Theme1/SkinFile.skin*' and 'Default.aspx'. The 'SkinFile.skin*' tab is active, displaying the following ASP.NET skin file code:

```
<asp:label runat="server" width="300px" height="40px" font-bold="true"  
font-size="x-large" forecolor="yellow" backcolor="green"/>  
<asp:button runat="server" forecolor="red" backcolor="yellow"/>  
<asp:textbox runat="server" font-bold="true" font-size="medium"  
forecolor="blue" backcolor="silver" font-italic="true"/>
```

- Now according to your apply theme at different level.
- Now Press F5 key to run the application
- **Advantages**
 - Centralized control of styling.
 - Easier to switch between different visual styles.
 - Encourages reusability and consistency.
 - Improves maintainability of UI code.