ASP.NET

Unit :- 1 Introduction and Basic Controls

1. IDE (Integrated Development Environment) Of Visual Studio

An integrated development environment (IDE), also known as integrated design environment and integrated debugging environment, is **a type of computer software that assists computer programmers to develop software**.

IDE Components :-

1. **Menu bar**
2. **Toolbar**
3. **Design view**
4. **Toolbox**
5. **Views**
6. **Property window**
7. **Solution Explorer**
8. **Server Explorer**
9. **Class View**

1. menu bar :

Menu bar provides various facilities like ..

- **File(new,open,add,close,saveAll,Exit,etc….)**
- **Edit(cut,copy,paste,delete,undo,redo,etc..)**
- **View(server explorer,solution Explorer,toolbox,etc..)**
- **Website(add new item,add existing item,set as start page,etc..)**
- **Build(build solution,rebuild solution,etc..)**
- **Debug(windows break point,toggle point,etc…)**
- **Format(font,paragram,foreground color,background color,etc..)**
- **Table(insert ,delete,tables,rows,colums,etc..)**
- **Tools(connect to databse,server,etc..)**
- **Test(Run,debug,etc..)**
- **Window(new window,spliting,reset window layout)**
- **Help(various helps)**

2. Toolbar:

Provide quick access to commonly used commands like save,undo,comment,cut,copy paste, run etc.

By default standard toolbar is appear on visual studio.net screen.

We can add other toolbars from view menu or right click on toolbar itself.

3. Toolbox:

Provides a set of controls to place on a form

We can also set our own toolbox, by right click of it.

Short cut is (Ctrl+Alt+X)

4. Design View:

Design view displays asp.net web pages, master pages,HTML pages, and user controls using a near WYSIWYG view.

Design view allows you to add text and elements and then position and size them and set their properties using special menus or the properties window.

Short cut is (Shift+F7)

5. Views:

There are three types of views are there such as

1. **Design**
2. **Split**
3. **Code**

1. **Design views shows the controls or elements of the web page.**
2. **Code view shows the source code(Html or XML)**
3. **Split view shows both design and code view**

6. Property Window:

Lists the properties for the selected conrols.

We can use this window to view and change the design time properties.

We can also set event of the selected controls.

It defines default values of controls where possible, which we can also change.

Short cut is : (F4)

7. Solution Explorer:

Solution explorer contains all our files in our websites.

It consists database file,code file,images,design files,class files,webconfig file,etc.

We can add new item,delete,existing item etc.

Short cut is : (Ctrl+Alt+L)

8. Server Explorer:

It displays all the details of connected databases.

Once connection has been done successfully then it will be appear in server explorer.

It displays list of databases,tables,stored procedures, etc.

We can also manipulate database link from here.

Make database connection to Sql servers and other databases.

Short cut is : (Ctrl+Alt+S)

9. Class View:

It shows all class file of our project or website.

It also consist data members and member functions.

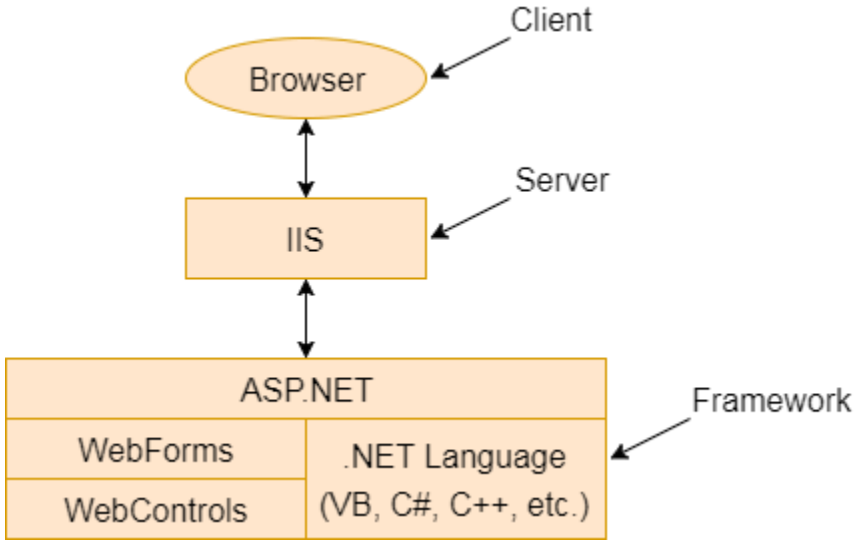
Short cut is : Ctrl+Shift+C

2. ASP.NET WEB FORMS

Web Forms are web pages built on the ASP.NET Technology. It executes on the server and generates output to the browser. It is compatible to any browser to any language supported by .NET common language runtime. It is flexible and allows us to create and add custom controls.

We can use Visual Studio to create ASP.NET Web Forms. It is an IDE (Integrated Development Environment) that allows us to drag and drop server controls to the web forms. It also allows us to set properties, events and methods for the controls. To write business logic, we can choose any .NET language like: Visual Basic or Visual C#.

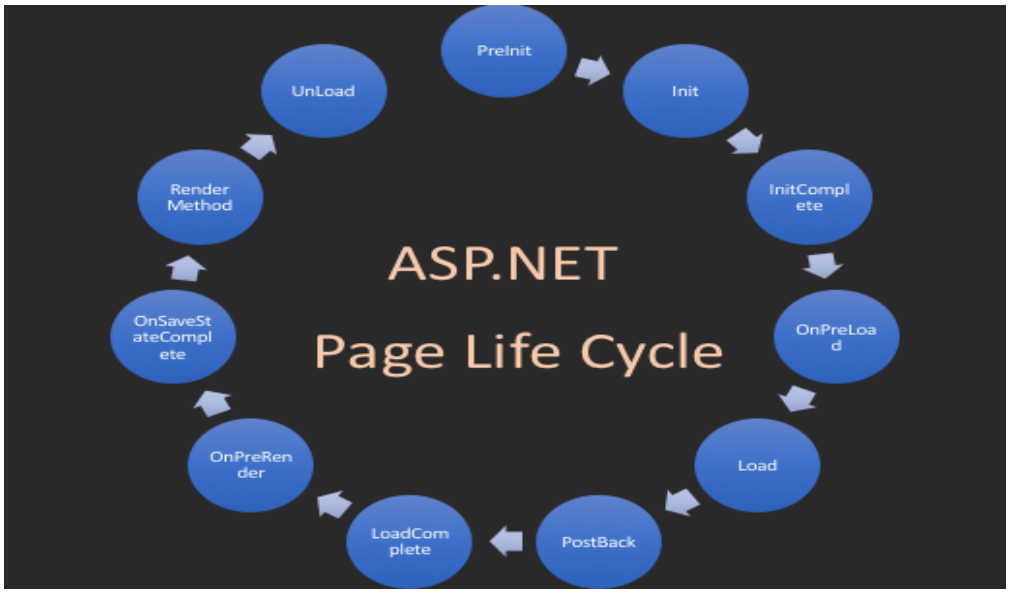
Web Forms are made up of two components: the visual portion (the ASPX file), and the code behind the form, which resides in a separate class file.



3. Page Event Life Cycle:

When an ASP.NET page runs, the page goes through a life cycle in which it performs a series of processing steps.

The following are the various stages or events of ASP.Net page life cycle.



1.PreInit

1. Check the IsPostBack property to determine whether this is the first time the page is being processed.
2. Create or re-create dynamic controls.
3. Set a master page dynamically.
4. Set the Theme property dynamically.

2. Init

1. This event fires after each control has been initialized.
2. Each control's UniqueID is set and any skin settings have been applied.
3. Use this event to read or initialize control properties.
4. The "Init" event is fired first for the bottom-most control in the hierarchy, and then fired up the hierarchy until it is fired for the page itself.

3. InitComplete

1. Until now the viewstate values are not yet loaded, hence you can use this event to make changes to the view state that you want to ensure are persisted after the next postback.
2. Raised by the Page object.
3. Use this event for processing tasks that require all initialization to be complete.

4. OnPreLoad

1. Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.
2. Before the Page instance raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.
3. Loads ViewState: ViewState data are loaded to controls.
4. Loads Postback data: Postback data are now handed to the page controls.

5. Load

1. The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page.
2. This is the first place in the page lifecycle that all values are restored.
3. Most code checks the value of IsPostBack to avoid unnecessarily resetting state.

4. You may also call Validate and check the value of IsValid in this method.
5. You can also create dynamic controls in this method.
6. Use the OnLoad event method to set properties in controls and establish database connections.

6. Control PostBack Event(s)

1. ASP.NET now calls any events on the page or its controls that caused the PostBack to occur.
2. Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.
3. In a postback request, if the page contains validator controls, check the IsValid property of the Page and of individual validation controls before performing any processing.
4. This is just an example of a control event. Here it is the button click event that caused the postback.

7. LoadComplete

1. Raised at the end of the event-handling stage.
2. Use this event for tasks that require that all other controls on the page be loaded.

8. OnPreRender

1. Raised after the Page object has created all controls that are required in order to render the page, including child controls of composite controls.
2. The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.
3. The PreRender event of individual controls occurs after the PreRender event of the page.
4. Allows final changes to the page or its control.
5. This event takes place before saving ViewState, so any changes made here are saved.
6. For example: After this event, you cannot change any property of a button or change any viewstate value.
7. Each data bound control whose DataSourceID property is set calls its DataBind method.
8. Use the event to make final changes to the contents of the page or its controls.

9. OnSaveStateComplete

1. Raised after view state and control state have been saved for the page and for all controls.
2. Before this event occurs, ViewState has been saved for the page and for all controls.
3. Any changes to the page or controls at this point will be ignored.
4. Use this event perform tasks that require the view state to be saved, but that do not make any changes to controls.

10. Render Method

1. This is a method of the page object and its controls (and not an event).
2. The Render method generates the client-side HTML, Dynamic Hypertext Markup Language (DHTML), and script that are necessary to properly display a control at the browser.

11. UnLoad

1. This event is used for cleanup code.
2. At this point, all processing has occurred and it is safe to dispose of any remaining objects, including the Page object.
3. Cleanup can be performed on:

    - Instances of classes, in other words objects
    - Closing opened files
    - Closing database connections.

Example:

```
1. public partial class PageLiftCycle: System.Web.UI.Page {
2.     protected void Page_PreInit(object sender, EventArgs e) {
3.         //Work and It will assign the values to label.
4.         lblName.Text = lblName.Text + "<br/>" + "PreInit";
5.     }
6.     protected void Page_Init(object sender, EventArgs e) {
7.         //Work and It will assign the values to label.
8.         lblName.Text = lblName.Text + "<br/>" + "Init";
9.     }
10.      protected void Page_InitComplete(object sender, EventArgs e)
   {
11.          //Work and It will assign the values to label.
12.          lblName.Text = lblName.Text + "<br/>" + "InitComplete";
13.      }
14.      protected override void OnPreLoad(EventArgs e) {
15.          //Work and It will assign the values to label.
16.          //If the page is post back, then label contrl values will
    be loaded from view state.
17.          //E.g: If you string str = lblName.Text, then str will co
   ntain viewstate values.
18.          lblName.Text = lblName.Text + "<br/>" + "PreLoad";
19.      }
20.      protected void Page_Load(object sender, EventArgs e) {
21.          //Work and It will assign the values to label.
22.          lblName.Text = lblName.Text + "<br/>" + "Load";
23.      }
24.      protected void btnSubmit_Click(object sender, EventArgs e) {

25.          //Work and It will assign the values to label.
26.          lblName.Text = lblName.Text + "<br/>" + "btnSubmit_Click"
   ;
27.      }
28.      protected void Page_LoadComplete(object sender, EventArgs e)
   {
29.          //Work and It will assign the values to label.
30.          lblName.Text = lblName.Text + "<br/>" + "LoadComplete";
31.      }
32.      protected override void OnPreRender(EventArgs e) {
33.          //Work and It will assign the values to label.
34.          lblName.Text = lblName.Text + "<br/>" + "PreRender";
35.      }
36.      protected override void OnSaveStateComplete(EventArgs e) {
37.          //Work and It will assign the values to label.
```

```
38.         //But "SaveStateComplete" values will not be available du
   ring post back. i.e. View state.
39.         lblName.Text = lblName.Text + "<br/>" + "SaveStateComplet
   e";
40.     }
41.     protected void Page_UnLoad(object sender, EventArgs e) {
42.         //Work and it will not effect label contrl, view stae and
    post back data.
43.         lblName.Text = lblName.Text + "<br/>" + "UnLoad";
44.     }
45. }
```

4. Global Application Class:

- The Global.asax is also known as the ASP.NET application file and is used to serve application-level and session-level events.
- It allows us to write code that response to global application events raised by ASP.NET or by HttpModules.
- These events fire at various points during the lifetime of a web application, including when the application domain is first created.
- The Global.asax file resides in the root directory of an ASP.NET-based application
- Global.asax file is optional. If you do not define the file, the ASP.NET page framework assumes that you have not defined any application or session event handlers.

- How to create Global.asax file ?

    1. Open visual studio.
    2. Create a new website.
    3. Go to solution explorer.
    4. Add new item.
    5. Add global application class.
- Global.asax file looks like this…

```
<%@ Application Language="C#" %>
<script runat="server">

    void Application_Start(object sender, EventArgs e)
    {

    }
    void Application_End(object sender, EventArgs e)
    {

    }
    void Application_Error(object sender, EventArgs e)
    {

    }
    void Session_Start(object sender, EventArgs e)
    {

    }
    void Session_End(object sender, EventArgs e)
    {

    }
</script>
```

1.Application_start():-

This method is invoked when the application first starts up and the application domain is created.

This event handler is a useful place to provide application-wide initialization code.

For example, at this point you might load and cache data that will not change throughout the lifetime of an application, such as navigation trees, static product catalogs, and so on.

2.Application_End():-

Code that runs on application shutdown

This method is invoked just before an application ends.

The end of an application can occur because IIS is being restarted or because the application is transitioning to a new application domain in response to updated files or the process recycling settings.

3. Application_Error():-

This method is invoked whenever an unhandled exception occurs in the application.

4.Session_Start():-

This method is invoked each time a new session begins.

This is often used to initialize user-specific information.

Application_Start() and Session_Start() both work same but difference is Session_Start() will re-initialize whenever your web browser is changes or closed.

5.Session_End():-

This method is invoked whenever the user's session ends. A session ends when your code explicitly releases it or when it times out after there have been no more requests received within a given timeout period (typically 20 minutes).

- ## Web.config file

A configuration file (web.config) is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code. In this way you can configure settings independently from your code. Generally a website contains a single Web.config file stored inside the application root directory. However there can be many configuration files that manage settings at various levels within an application.

What Web.config file contains?

There are number of important settings that can be stored in the configuration file. Some of the most frequently used configurations, stored conveniently inside Web.config file are:

- Database connections
- Caching settings
- Session States
- Error Handling
- Security
- Configuration file looks like this

```
<configuration>
    <connectionStrings>
        <add name="myCon" connectionString="server=MyServer;database=puran;uid=puranmehra;pwd=mydata1223" />
  </connectionStrings>
</configuration/>
```

- ## Advantages of Asp.net

1) It allows for separation of concern

ASP.NET follows the MVC architecture, which allows for separate input, process and output of the application. This three-tier architecture, Model-View- Controller has interconnected parts, and can handle specific development aspects of software applications.

2) Reduces coding time

The framework technology is a big help in reducing coding time, especially when you are developing big applications. There are different types of code reviews, so you have no chance of writing a bad code. Code reviews would help you improve code quality.

3) Consists of some of out-of- the box features

ASP.NET delivers enhanced performance and scalability. It also comes with features like just-in- time compilation, early binding, native optimisation and caching services, and they too serve to improve performance several notches higher. The codes here are not interpreted like traditional ASP pages.

4) World class toolbox

The framework comes with incredibly rich toolbox through its Visual Studio integrated development environment. This toolbox acts as a very important building framework for the framework, and aids the developer to create applications very quickly. The toolbox is famous for its features like drag-and-
drop server controls WYSIWYG editing, and automatic deployment.

5) Delivers power and flexibility

The framework language is based on common language runtime, so all the web application developers can enjoy flexibility and power of that entire platform. It is also language independent, so you can choose the language for your application or even divide your application across several languages.

6) Simplicity

Each task can be performed easily, even the most common ones to the complicated and tricky ones. The common language runtime makes development process a simple one, with services like garbage collection and automatic reference counting. The framework lets you build user interfaces that can separate application logic and presentation code.

7) Customizability and Extensibility

The well factored architecture of the framework is a major help to developers. You can easily extend or replace the subcomponent of the ASP.NET runtime with the help of your own custom-prepared components. Implementing those have become even easier.

8) Security

Security is a good feature of the framework language. You can develop secure applications through built in Windows authentication and per-application configuration features.

9) Manageability

The excellent manageability feature of the framework is contributed through its text based hierarchical configuration system. And since these configurations are incorporated as plain texts, you can just make use of the local administration tools to apply the new settings. This makes tasks much easier, with no server restart, or with the necessity to deploy them separately, or replace running compiled code.

10) Benefit of continuous monitoring

Continuous and constant monitoring is an incredible feature of ASP.NET. You don’t have to worry about the status of the applications, components and the pages themselves. The program watches out for any such illegal events, and if anything happens (for example, memory leaps of infinite loops), it would immediately rise into action by destroying the activities, and restarting itself.

11) Cross-platform migration

The framework language allows for easy cross-platform migration, configuration and deployment services.

- ## Features of ASP.NET:-

(1) Common Language Runtime or CLR

It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation. The code that is directly managed by the CLR is called the managed code. When the managed code is compiled, the compiler converts the source code into a CPU independent intermediate language (IL) code. A

Just In Time(JIT) compiler compiles the IL code into native code, which is CPU specific.

(2) .Net Framework Class Library

It contains a huge library of reusable types. classes, interfaces, structures, and enumerated values, which are collectively called types.

(3) Common Language Specification

It contains the specifications for the .Net supported languages and implementation of language integration.

(4) Common Type System

It provides guidelines for declaring, using, and managing types at runtime, and cross-language communication.

(5) Metadata and Assemblies

Metadata is the binary information describing the program, which is either stored in a portable executable file (PE) or in the memory. Assembly is a logical unit consisting of the assembly manifest, type metadata, IL code, and a set of resources like image files.

(6) Windows Forms

Windows Forms contain the graphical representation of any window displayed in the application.

(7) ASP.NET and ASP.NET AJAX

ASP.NET is the web development model and AJAX is an extension of ASP.NET for developing and implementing AJAX functionality. ASP.NET AJAX contains the components that allow the developer to update data on a website without a complete reload of the page.

(8) ADO.NET

It is the technology used for working with data and databases. It provides access to data sources like SQL server, OLE DB, XML etc. The ADO.NET allows connection to data sources for retrieving, manipulating, and updating data.

(9) Windows Workflow Foundation (WF)

It helps in building workflow-based applications in Windows. It contains activities, workflow runtime, workflow designer, and a rules engine.

(10) Windows Presentation Foundation

It provides a separation between the user interface and the business logic. It helps in developing visually stunning interfaces using documents, media, two and three dimensional graphics, animations, and more.

(11) Windows Communication Foundation (WCF)

It is the technology used for building and executing connected systems.

(12) Windows CardSpace

It provides safety for accessing resources and sharing personal information on the internet.

(13) LINQ

It imparts data querying capabilities to .Net languages using a syntax which is similar to the tradition query language SQL.

- ## State Management Types

State management is the technique that is used **to maintain user and page information over multiple requests while browsing the web**. HTTP is a stateless protocol. It does not store any information about user on web page. It is a general requirement that information should be maintained while navigating the website.

There are two type of state management techniques

- Client-side Techniques
  - Cookies
  - Query Strings
  - Viewstate
- Server-side Techniques
  - Session State

Client-side techniques:

1. ViewState:

View State can be used to maintain the State at a page level. The term "Page Level" means that the information is being stored for a specific page and until that specific page is active (i.e. the page which is

being currently viewed by the user). Once the user is re-directed or goes to some other page, the information stored in the View State gets lost.

Some of the features of view state are:

- It is page-level State Management
- Used for holding data temporarily
- Can store any type of data
- Property dependent

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        if (ViewState["count"] != null)
        {
            int ViewstateVal = Convert.ToInt32(ViewState["count"]) + 1;
            View.Text = ViewstateVal.ToString();
            ViewState["count"]=ViewstateVal.ToString();
        }
        else
        {
            ViewState["count"] = "1";
        }
    }
}
// Click Event
protected void Submit(object sender, EventArgs e)
{
      View.Text=ViewState["count"].ToString();
}
```

2.Cookie

Cookies is a small text file that is stored in the user's hard drive using the client's browser. Cookies are just used for the sake of the user's identity matching as it only stores information such as sessions id's, some frequent navigation or post-back request objects.

Whenever we get connected to the internet for accessing a specific service, the cookie file is accessed from our hard drive via our browser for identifying the user. The cookie access depends upon the life cycle or expiration of that specific cookie file.

Some features of cookies are:

- Store information temporarily
- It's just a simple small sized text file
- Can be changed depending on requirements
- User Preferred
- Requires only a few bytes or KBs of space for creating cookies

Two types of cookies are available,

Persistence

This type of cookie works with Date and time.

```
1. Response.Cookies["CookieName"].Value = "Test Cookies";
2. //set expire time
3. Response.Cookies["CookieName"].Expires = DateTime.Today.AddHours(1
   );
```

Non-Persistence

This is a temporary cookie. It is created with access application and discards the close application.

```
1. Response.Cookies["CookieName"].Value = "Test Cookies";
```

3. QueryString:

Query strings are used for some specific purpose. These in a general case are used for holding some value from a different page and move these values to the different page. The information stored in it can be easily navigated to one page to another or to the same page as well.

Some of the features are,

- It is generally used for holding values
- Works temporarily
- Switches info from one to another page
- Increase performance
- Uses real and virtual path values for URL routing

```
if (Request.QueryString["number"] != null)
{
    View.Text = Request.QueryString["number"];
}

// Setting query string
int postbacks = 0;
```

```
if (Request.QueryString["number"] != null)
{
    postbacks = Convert.ToInt32(Request.QueryString["number"]) +
1;
}
else
{
    postbacks = 1;
}

Response.Redirect("default.aspx?number=" + postbacks);
```

- **Server-side techniques:**

1. **Session:**

Session is a very important technique to maintain state. Normally session is used to store information and identity. The server stores information using Sessionid.

Set User Session

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    Session["UserName"] = txtName.Text;

    Response.Redirect("Home.aspx");
}
```

Session Event

Session event can be seen in project *Global.asax* file.

Two types of Session Events

Session_Start

The Session_start event is raised every time a new user requests without a session ID.

```
void Session_Start(object sender, EventArgs e)
{
    Session["master"] = "~/Master.master";
```

```
4. }
```

Session_End

The Session_End event is raised when session is ended by a user or a time out using Session end method.

```
1. void Session_End(object sender, EventArgs e)
2. {
3.    Response.Write("Session_End");
4. }
```

The session is stored in the following for ways in ASP.NET.

- ***InProcMode***
  It is a default session mode and a value store in web server memory (IIS). In this the session value stored with server start and it ends when the server is restarted.

- ***State Server Mode***
  In this mode session data is stored in separate server.

- ***SQL Server Mode***
  In this session is stored in the database. It is a secure mode.

- ***Custom Mode***
  Generally under session data is stored in InProc, Sql Server, State server, etc. If you store session data with other new techniques then provide ASP.NET.

Properties

1. **SessionID :** Returns a unique id for each user. The unique id is generated by the server
2. **Timeout:** Sets or returns the timeout period (in minutes) for the Session object in this application

Methods

1.Abandon(): Destroys a user session

2. Remove(): Deletes an item from the Contents collection

3.RemoveAll(): Deletes all items from the Contents collection

Basic Control

1. Label :- This control is used to display textual information on the web forms. It is mainly used to create caption for the other controls like: textbox.

The example is given below.

```
< asp:LabelID="Label1" runat="server" Text="Label" ></asp:Label>
```

Property	Description
AccessKey	It is used to set keyboard shortcut for the label.
TabIndex	The tab order of the control.
BackColor	It is used to set background color of the label.
BorderColor	It is used to set border color of the label.
BorderWidth	It is used to set width of border of the label.
Font	It is used to set font for the label text.
ForeColor	It is used to set color of the label text.
Text	It is used to set text to be shown for the label.
ToolTip	It displays the text when mouse is over the label.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.

2. Button:- This control is used to perform events. It is also used to submit client request to the server.

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.

3. TextBox:-

Text box controls are typically used to accept input from the user.

Basic syntax of TextBox control:

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

TextChanged is an event of textbox control.

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.

List Controls

ASP.NET provides the following controls

- Drop-down list,
- List box,
- Radio button list,
- Check box list,
- Bulleted list.

These control let a user choose from one or more items from the list. List boxes and drop-down lists contain one or more list items. These lists can be loaded either by code or by the ListItemCollection editor.

Basic syntax of list box control:

```
<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged">
</asp:ListBox>
```

Basic syntax of drop-down list control:

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
</asp:DropDownList>
```

Common properties of list box and drop-down Lists:

Property	Description

Items	The collection of ListItem objects that represents the items in the control. This property returns an object of type ListItemCollection.
Rows	Specifies the number of items displayed in the box. If actual list contains more rows than displayed then a scroll bar is added.
SelectedIndex	The index of the currently selected item. If more than one item is selected, then the index of the first selected item. If no item is selected, the value of this property is -1.
SelectedValue	The value of the currently selected item. If more than one item is selected, then the value of the first selected item. If no item is selected, the value of this property is an empty string ("").
SelectionMode	Indicates whether a list box allows single selections or multiple selections.

Common properties of each list item objects:

Property	Description
Text	The text displayed for the item.
Selected	Indicates whether the item is selected.
Value	A string value associated with the item.

It is important to notes that:

- To work with the items in a drop-down list or list box, you use the Items property of the control. This property returns a ListItemCollection object which contains all the items of the list.
- The SelectedIndexChanged event is raised when the user selects a different item from a drop-down list or list box.

The ListItemCollection

The ListItemCollection object is a collection of ListItem objects. Each ListItem object represents one item in the list. Items in a ListItemCollection are numbered from 0.

When the items into a list box are loaded using strings like: lstcolor.Items.Add("Blue"), then both the Text and Value properties of the list item are set to the string value you specify. To set it differently you must create a list item object and then add that item to the collection.

The ListItemCollection Editor is used to add item to a drop-down list or list box. This is used to create a static list of items. To display the collection editor, select edit item from the smart tag menu, or select the control and then click the ellipsis button from the Item property in the properties window.

Common properties of ListItemCollection:

Property	Description
Item(integer)	A ListItem object that represents the item at the specified index.
Count	The number of items in the collection.

Common methods of ListItemCollection:

Methods	Description
Add(string)	Adds a new item at the end of the collection and assigns the string parameter to the Text property of the item.
Add(ListItem)	Adds a new item at the end of the collection.
Insert(integer, string)	Inserts an item at the specified index location in the collection, and assigns string parameter to the text property of the item.
Insert(integer, ListItem)	Inserts the item at the specified index location in the collection.
Remove(string)	Removes the item with the text value same as the string.
Remove(ListItem)	Removes the specified item.
RemoveAt(integer)	Removes the item at the specified index as the integer.
Clear	Removes all the items of the collection.
FindByValue(string)	Returns the item whose value is same as the string.
FindByValue(Text)	Returns the item whose text is same as the string.

Radio Button list and Check Box list

A radio button list presents a list of mutually exclusive options. A check box list presents a list of independent options. These controls contain a collection of ListItem objects that could be referred to through the Items property of the control.

Basic syntax of radio button list:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
   OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">
</asp:RadioButtonList>
```

Basic syntax of check box list:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack="True"
   OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged">
</asp:CheckBoxList>
```

Common properties of check box and radio button lists:

Property	Description
RepeatLayout	This attribute specifies whether the table tags or the normal html flow to use while formatting the list when it is rendered. The default is Table.
RepeatDirection	It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical. Default is Vertical.
RepeatColumns	It specifies the number of columns to use when repeating the controls; default is 0.

Bulleted lists and Numbered lists

The bulleted list control creates bulleted lists or numbered lists. These controls contain a collection of ListItem objects that could be referred to through the Items property of the control.

Basic syntax of a bulleted list:

```
<asp:BulletedList ID="BulletedList1" runat="server">
</asp:BulletedList>
```

Common properties of the bulleted list:

Property	Description
BulletStyle	This property specifies the style and looks of the bullets, or numbers.
RepeatDirection	It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical. Default is Vertical.
RepeatColumns	It specifies the number of columns to use when repeating the controls; default is 0.

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.