ASP.NET: *Web Form fundamentals, Web Controls, State management, Building better web form - Validation, rich controls, user controls and graphics, Data Management with ADO.NET, ASP.NET with Ajax.*

What is a Web Server?

Web servers use HTTP to allow access to the Internet. They search through and use HTML files that are sent to web browsers and translated so the user can understand them. It is also capable of accessing and storing other types of files, but they are often attached in some way to the HTML files it has, such as having images that are placed upon the HTML.

What are Web Servers Used For?

Web servers are primarily used to store process and deliver the pages of a web site to users. In layman's terms, this means that web servers are what make websites appear when you type in a URL.

Types of Web Servers:

There are 4 primary web servers:

- Apache (provided by Apache)
- IIS (provided by Microsoft)
- nginx (provided by NGINX, Inc. and pronounced like "Engine X")
- and GWS (provided by Google and short for Google Web Server)

Apache is the most popular with IIS gaining in popularity and (according to our predictions) soon becoming the most popular web server. nginx is an extremely popular alternative as it is very fast and very lightweight, while GWS is the least used with a small percentage of use.

File Servers

They are often responsible for the availability of stored files and their management as well as security. You can send and receive files at the user's request. These are not like sharing servers; they are more like the filing cabinets of the Internet world.

File servers are often categorized by how the files on the server are accessed. Here are the different methods:

Internet File Servers:

- FTP (File Transfer Protocol)
- HTTP (HyperText Transfer Protocol)

LAN File Servers:

- SMB/CIFS Protocol
- NFS Protocol

File servers are different from web servers because they do not provide dynamic web content like web servers. Instead they only provide static files.

Application Servers

This is a server that is dedicated to serving a certain piece of software. It is often used in conjunction with other servers and software. For example, you may sign up for online gaming and be directed to servers set up solely for the gaming software.

Application Server Advantages:

- Data and Code Integrity
- Centralized Configuration
- Security
- Performance
- Lower Cost of Ownership
- Transaction Support

Types of Application Servers:

- Java Application Servers
- .Net Framework
- PHP Application Servers
- Open Source Application Servers

- Mobile Application Servers

Inter-server level devices

These are devices that bypass servers in order to send signals directly from one place to another in an organized fashion, such as how a person is able to download one file from multiple computers at one time.

Message Servers

These are servers that allow things such as real-time communication between users. They may include IRC servers, chat servers and groupware. The methods of communication are fairly flexible.

Proxy Servers

It will act as a mediator between server-to-filter requests made by users and a client program. It allows the management of emails and shared connections.

There are two types of proxy servers: Open Proxies and Reverse Proxies.

Open Proxies

An open proxy is a forwarding proxy server which is accessible to any user. Anonymous open proxies allow the users which access them to conceal their IP address while using the internet. The level of anonymity varies and these often aren't entirely secretive as their are methods which will cause the client to reeal itself.

Reverse Proxies

Reverse proxies are a type of server which appear to be an ordinary server. A user that requests information from the server will have their request forwarded to one or more proxy servers which allow the user to receive their request as if it were from the original server all while having no information as to the original server.

Proxy Server Uses

Proxy servers have a wide range of uses including both malicious and legitimate uses. For example, large corporations may use proxy servers to protect their data while someone may avoid government, business, or school censorship with the use of a proxy.

Database Servers

They can manage a database, as it is stored on the server. They may use the SQL database management system. The server can search through information and send any requested information back to the client.

Mail Servers

As the name suggests, this is the sort of server you can use to control email. A server may be set up with the expressed idea of it controlling and handling emails only. It sends, receives, stores your emails.

There are two primary types of mail servers: IMAP and POP3 Servers. IMAP is quickly becoming the most popular server as it allows you to read your mail on multiple devices.

Most people will not need to (nor want to) run their own mail server, as Google, Microsoft and plenty of other corporations offer free email.

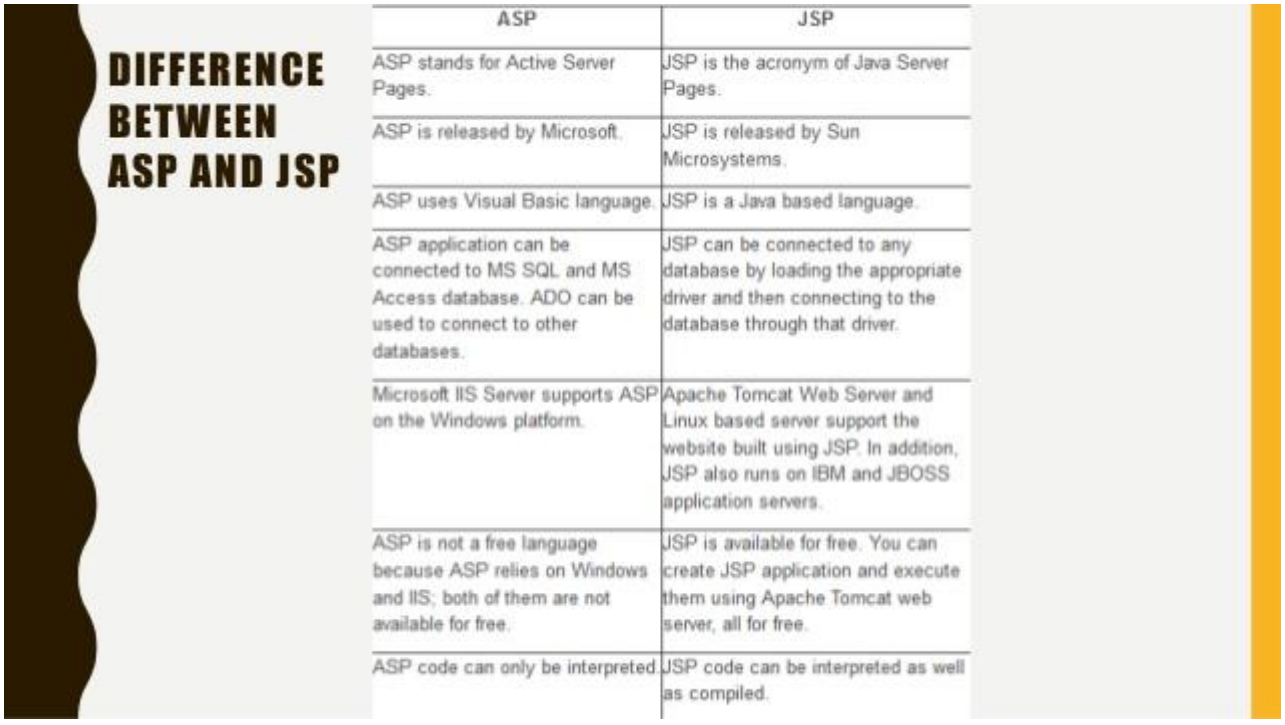
ASP.NET

- ASP stands for **A**ctive **S**erver **P**ages
- ASP is a development framework for building web pages.
- ASP, .net, ASP.Net are different
- Asp.net is part of .net

.NET frame work is A programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies, such as desktop applications and Web services.

The .NET Framework contains three major parts:

- the Common Language Runtime
- the Framework Class Library
- ASP.NET.



ASP	JSP
ASP stands for Active Server Pages.	JSP is the acronym of Java Server Pages.
ASP is released by Microsoft.	JSP is released by Sun Microsystems.
ASP uses Visual Basic language.	JSP is a Java based language.
ASP application can be connected to MS SQL and MS Access database. ADO can be used to connect to other databases.	JSP can be connected to any database by loading the appropriate driver and then connecting to the database through that driver.
Microsoft IIS Server supports ASP on the Windows platform.	Apache Tomcat Web Server and Linux based server support the website built using JSP. In addition, JSP also runs on IBM and JBOSS application servers.
ASP is not a free language because ASP relies on Windows and IIS; both of them are not available for free.	JSP is available for free. You can create JSP application and execute them using Apache Tomcat web server, all for free.
ASP code can only be interpreted.	JSP code can be interpreted as well as compiled.

Components of .Net Framework

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.

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

ASP.NET life cycle specifies how:

- ASP.NET processes pages to produce dynamic output
- The application and its pages are instantiated and processed

 ASP.NET compiles the pages dynamically

ASP.NET life cycle could be divided into two groups:
 Application Life Cycle
 Page Life Cycle

ASP.NET Application Life Cycle
The application life cycle has the following stages:
1. User makes a request for accessing application resource, a page. Browser sends this request to the web server.
2. A unified pipeline receives the first request and the following events take place:
i. An object of the class ApplicationManager is created.
ii. An object of the class HostingEnvironment is created to provide information regarding the resources.
iii. Top level items in the application are compiled.
3. Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.
4. An instance of the HttpApplication object is created and assigned to the request.
5. The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

ASP.NET Page Life Cycle
When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application.

1. **Page request**
   When ASP.NET gets a page request, it decides whether to parse and compile the page, or there would be a cached version of the page; accordingly the response is sent.
2. **Starting of page life cycle**
   At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
3. **Page initialization**
   At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and the themes are applied. For a new request, postback data is loaded and the control properties are restored to the view-state values.
4. **Page load**
   At this stage, control properties are set using the view state and control state values.
5. **Validation**
   Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
6. **Postback event handling.**
   If the request is a postback (old request), the related event handler is invoked.
7. **Page rendering**
   At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of Page.
8. **Unload**

The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done.

DIFFERENCE BETWEEN ASP AND ASP.NET

ASP is interpreted whereas, ASP.NET is compiled. This implies that since ASP uses VBScript; therefore, when an ASP page is executed, it is interpreted. On the other hand, ASP.NET uses .NET languages, such as C# and VB.NET, which are compiled to Microsoft Intermediate Language (MSIL).

1. ASP is interpreted, ASP.NET is compiled.

2. Classic ASP uses a technology called ADO to connect and work with databases. ASP.NET uses the ADO.NET technology

3. ASP has Mixed HTML and coding logic where in asp.net html and coding part are separated by code behind files.

4. ASP.NET purely objects oriented whereas ASP is partially object oriented.

5. For ASP No in-built support for XML whereas in ASP.NET full XML Support for easy data exchange.

FEATURES OF ASP.NET

ASP.NET provides a programming model and infrastructure that facilitates developing new classes of Web applications. Part of this infrastructure is the .NET runtime and framework. Server-side code is written in .NET compiled languages. Two main programming models are supported by ASP.NET.

- **Web Forms** helps you build form-based Web pages. A WYSIWYG development environment enables you to drag controls onto Web pages. Special "server-side" controls present the programmer with an event model similar to what is provided by controls in ordinary Windows programming. This chapter discusses Web Forms in detail.
- **Web services** make it possible for a Web site to expose functionality via an API that can be called remotely by other applications. Data is exchanged using standard Web protocols and formats such as HTTP and XML, which will cross firewalls. We will discuss Web services in the next chapter.

Both Web Forms and Web services can take advantage of the facilities provided by .NET, such as the compiled code and .NET runtime. In addition, ASP.NET itself provides a number of infrastructure services, including state management, security, configuration, caching, and tracing.

ASP.NET Page Life Cycle Events

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.

Following are the page life cycle events:

PreInit

PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.

Init

Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.

InitComplete

InitComplete event allows tracking of view state. All the controls turn on view-state tracking.

LoadViewState

LoadViewState event allows loading view state information into the controls.

LoadPostData

During this phase, the contents of all the input fields are defined with the <form> tag are processed.

PreLoad

PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.

LoadThe Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.

LoadComplete

The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler.

PreRender

The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.

PreRenderComplete

As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.

SaveStateComplete

State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.

UnLoad

The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

WEB FORMS FUNDAMENTALS

ASP.NET Web Forms are:

- Based on Microsoft ASP.NET technology, in which code that runs on the server dynamically generates Web page output to the browser or client device.
- Compatible with any browser or mobile device. An ASP.NET Web page automatically renders the correct browser-compliant HTML for features such as styles, layout, and so on.
- Compatible with any language supported by the .NET common language runtime, such as Microsoft Visual Basic and Microsoft Visual C#.
- Built on the Microsoft .NET Framework. This provides all the benefits of the framework, including a managed environment, type safety, and inheritance.
- Flexible because you can add user-created and third party controls to them.

ASP.NET Web Forms offer:

- Separation of HTML and other UI code from application logic.
- A rich suite of server controls for common tasks, including data access.
- Powerful data binding, with great tool support.
- Support for client-side scripting that executes in the browser.
- Support for a variety of other capabilities, including routing, security, performance, internationalization, testing, debugging, error handling and state management.

ASP.NET Web Forms Helps You Overcome Challenges (challenges faced by web applications)

Web application programming presents challenges that do not typically arise when programming traditional client-based applications. Among the challenges are:

- **Implementing a rich Web user interface** - It can be difficult and tedious to design and implement a user interface using basic HTML facilities, especially if the page has a complex layout, a large amount of dynamic content, and full-featured user-interactive objects.
- **Separation of client and server** - In a Web application, the client (browser) and server are different programs often running on different computers (and even on different operating systems). Consequently, the two halves of the application share very little information; they can communicate, but typically exchange only small chunks of simple information.
- **Stateless execution** - When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then discards all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing the page from scratch. Put another way, a server has no memory of pages that it has processed—page are stateless. Therefore, if an application needs to maintain information about a page, its stateless nature can become a problem.
- **Unknown client capabilities** - In many cases, Web applications are accessible to many users using different browsers. Browsers have different capabilities, making it difficult to create an application that will run equally well on all of them.
- **Complications with data access** - Reading from and writing to a data source in traditional Web applications can be complicated and resource-intensive.
- **Complications with scalability** - In many cases Web applications designed with existing methods fail to meet scalability goals due to the lack of compatibility between the various components of the application. This is often a common failure point for applications under a heavy growth cycle.

Meeting these challenges for Web applications can require substantial time and effort. ASP.NET Web Forms and the **ASP.NET framework address these challenges in the following ways:**

- **Intuitive, consistent object model** - The ASP.NET page framework presents an object model that enables you to think of your forms as a unit, not as separate client and server pieces. In this model, you can program the page in a more intuitive way than in traditional Web applications, including the ability to set properties for page elements and respond to events. In addition, ASP.NET server controls are an abstraction from the physical contents of an HTML page and from the direct interaction between browser and server. In general, you can use server controls the way you might work with controls in a client application and not have to think about how to create the HTML to present and process the controls and their contents.
- **Event-driven programming model** - ASP.NET Web Forms bring to Web applications the familiar model of writing event handlers for events that occur on either the client or server. The ASP.NET page framework abstracts this model in such a way that the underlying mechanism of capturing an event on the client, transmitting it to the server, and calling the appropriate method is all automatic and invisible to you. The result is a clear, easily written code structure that supports event-driven development.
- **Intuitive state management** - The ASP.NET page framework automatically handles the task of maintaining the state of your page and its controls, and it provides you with explicit ways to maintain the state of application-specific information. This is accomplished without heavy use of server resources and can be implemented with or without sending cookies to the browser.
- **Browser-independent applications** - The ASP.NET page framework enables you to create all application logic on the server, eliminating the need to explicitly code for differences in browsers. However, it still enables you to take advantage of browser-specific features by writing client-side code to provide improved performance and a richer client experience.
- **.NET Framework common language runtime support** - The ASP.NET page framework is built on the .NET Framework, so the entire framework is available to any ASP.NET application. Your applications can be written in any language that is compatible that is with the runtime. In addition, data access is simplified using the data access infrastructure provided by the .NET Framework, including ADO.NET.
- **.NET Framework scalable server performance** - The ASP.NET page framework enables you to scale your Web application from one computer with a single processor to a multi-computer Web farm cleanly and without complicated changes to the application's logic.

Web Controls

When you create ASP.NET Web pages, you can use these types of controls:

- **HTML server controls** HTML elements exposed to the server so you can program them. HTML server controls expose an object model that maps very closely to the HTML elements that they render.
- **Web server controls** Controls with more built-in features than HTML server controls. Web server controls include not only form controls such as buttons and text boxes, but also special-purpose controls such as a calendar, menus, and a tree view control. Web server controls are more abstract than HTML server controls in that their object model does not necessarily reflect HTML syntax.
- **Validation controls** Controls that incorporate logic to enable you to what users enter for input controls such as the TextBox control. Validation controls enable you to check for a required field, to test against a specific value or pattern of characters, to verify that a value lies within a range, and so on.
- **User controls** Controls that you create as ASP.NET Web pages. You can embed ASP.NET user controls in other ASP.NET Web pages, which is an easy way to create toolbars and other reusable elements.

Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools. Using these tools, the users can enter data, make selections and indicate their preferences.
Controls are also used for structural jobs, like validation, data access, security, creating master pages, and data manipulation.
ASP.NET uses five types of web controls:
- HTML controls
- HTML Server controls
- ASP.NET Server controls
- ASP.NET Ajax Server controls
- User controls and custom controls

ASP.NET server controls are the primary controls used in ASP.NET. These controls can be grouped into the following categories:
- **Validation controls** - These are used to validate user input and they work by running client-side script.
- **Data source controls** - These controls provides data binding to different data sources.
- **Data view controls** - These are various lists and tables, which can bind to data from data sources for displaying.
- **Personalization controls** - These are used for personalization of a page according to the user preferences, based on user information.
- **Login and security controls** - These controls provide user authentication.
- **Master pages** - These controls provide consistent layout and interface throughout the application.
- **Navigation controls** - These controls help in navigation. For example, menus, tree view etc.
- **Rich controls** - These controls implement special features. For example, AdRotator, FileUpload, and Calendar control.

The syntax for using server controls is:
<asp:controlType ID ="ControlID" runat="server" Property1=value1 [Property2=value2] />

ASP.Net server controls inherit all properties, events, and methods of the WebControl and System.Web.UI.Control class.

The following table shows the inherited properties, common to all server controls:

Property	**Description**
AccessKey	Pressing this key with the Alt key moves focus to the control.
Attributes	It is the collection of arbitrary attributes (for rendering only) that do not correspond to properties on the control.
BackColor	Background color.
BindingContainer	The control that contains this control's data binding.
BorderColor	Border color.
BorderStyle	Border style.
BorderWidth	Border width.
UniqueID	Unique identifier.
ViewState	Gets a dictionary of state information that saves and restores the view state of a server control across multiple requests for the same page.
ViewStateIgnoreCase	It indicates whether the StateBag object is case-insensitive.
ViewStateMode	Gets or sets the view-state mode of this control.

Visible	It indicates whether a server control is visible.
Width	Gets or sets the width of the Web server control.

Methods of the Server Controls

The following table provides the methods of the server controls:

Method	Description
AddAttributesToRender	Adds HTML attributes and styles that need to be rendered to the specified HtmlTextWriterTag.
AddedControl	Called after a child control is added to the Controls collection of the control object.
AddParsedSubObject	Notifies the server control that an element, either XML or HTML, was parsed, and adds the element to the server control's control collection.
ApplyStyleSheetSkin	Applies the style properties defined in the page style sheet to the control.
OnLoad	Raises the Load event.
OnPreRender	Raises the PreRender event.
OnUnload	Raises the Unload event.
OpenFile	Gets a Stream used to read a file.
RemovedControl	Called after a child control is removed from the controls collection of the control object.

Button Controls

ASP.NET provides three types of button control:

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

When a user clicks a button, two events are raised: Click and Command.

Basic syntax of button control:

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

Common properties of the button control:

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

Text Boxes and Labels

Text box controls are typically used to accept input from the user. A text box control can accept one or more lines of text depending upon the settings of the TextMode attribute.

Label controls provide an easy way to display text which can be changed from one execution of a page to the next. If you want to display text that does not change, you use the literal text.

Basic syntax of text control:

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

Common Properties of the Text Box and Labels:

Property	Description
TextMode	Specifies the type of text box. SingleLine creates a standard text box, MultiLIne creates a text box that accepts more than one line of text and the Password causes the characters that are entered to be masked. The default is SingleLine.
Text	The text content of the text box.
MaxLength	The maximum number of characters that can be entered into the text box.
Wrap	It determines whether or not text wraps automatically for multi-line text box; default is true.
ReadOnly	Determines whether the user can change the text in the box; default is false, i.e., the user can not change the text.
Columns	The width of the text box in characters. The actual width is determined based on the font that is used for the text entry.
Rows	The height of a multi-line text box in lines. The default value is 0, means a single line text box.

The mostly used attribute for a label control is 'Text', which implies the text displayed on the label.

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.

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.

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.

HyperLink Control

The HyperLink control is like the HTML <a> element.

Basic syntax for a hyperlink control:

```
<asp:HyperLink ID="HyperLink1" runat="server">
  HyperLink
</asp:HyperLink>
```

It has the following important properties:

Property	Description
ImageUrl	Path of the image to be displayed by the control.
NavigateUrl	Target link URL.
Text	The text to be displayed as the link.
Target	The window or frame which loads the linked page.

Image Control

The image control is used for displaying images on the web page, or some alternative text, if the image is not available.

Basic syntax for an image control:

```
<asp:Image ID="Image1" runat="server">
```

It has the following important properties:

Property	Description
AlternateText	Alternate text to be displayed in absence of the image.
ImageAlign	Alignment options for the control.
ImageUrl	Path of the image to be displayed by the control.

State Management

Hyper Text Transfer Protocol (HTTP) is a stateless protocol. When the client disconnects from the server, the ASP.NET engine discards the page objects. This way, each web application can scale up to serve numerous requests simultaneously without running out of server memory.
However, there needs to be some technique to store the information between requests and to retrieve it when required. This information i.e., the current value of all the controls and variables for the current user in the current session is called the State.

ASP.NET manages four types of states:
- View State
- Control State
- Session State
- Application State

The **view state** is the state of the page and all its controls. It is automatically maintained across posts by ASP.NET framework.
When a page is sent back to the client, the changes in the properties of the page and its controls are determined, and stored in the value of a hidden input field named _VIEWSTATE. When the page is again posted back, the _VIEWSTATE field is sent to the server with the HTTP request.
The view state could be enabled or disabled for:
- **The entire application** by setting the EnableViewState property in the <pages> section of web.config file.
- **A page** by setting the EnableViewState attribute of the Page directive, as <%@ Page Language="C#" EnableViewState="false" %>
- **A control** by setting the Control.EnableViewState property.

It is implemented using a view state object defined by the StateBag class which defines a collection of view state items. The state bag is a data structure containing attribute-value pairs, stored as strings associated with objects.
Properties of state bag class are:

Item(name)	The value of the view state item with the specified name. This is the default property of the StateBag class.
Count	The number of items in the view state collection.
Keys	Collection of keys for all the items in the collection.
Values	Collection of values for all the items in the collection.

The StateBag class has the following methods:

Methods	**Description**
Add(name, value)	Adds an item to the view state collection and existing item is updated.
Clear	Removes all the items from the collection.
Equals(Object)	Determines whether the specified object is equal to the current object.
Finalize	Allows it to free resources and perform other

	cleanup operations.
GetEnumerator	Returns an enumerator that iterates over all the key/value pairs of the StateItem objects stored in the StateBag object.
GetType	Gets the type of the current instance.
IsItemDirty	Checks a StateItem object stored in the StateBag object to evaluate whether it has been modified.
Remove(name)	Removes the specified item.
SetDirty	Sets the state of the StateBag object as well as the Dirty property of each of the StateItem objects contained by it.
SetItemDirty	Sets the Dirty property for the specified StateItem object in the StateBag object.

Control State

Control state cannot be modified, accessed directly, or disabled.

Session State

When a user connects to an ASP.NET website, a new session object is created. When session state is turned on, a new session state object is created for each new request. This session state object becomes part of the context and it is available through the page.

Session state is generally used for storing application data such as inventory, supplier list, customer record, or shopping cart. It can also keep information about the user and his preferences, and keep the track of pending operations.

Sessions are identified and tracked with a 120-bit SessionID, which is passed from client to server and back as cookie or a modified URL. The SessionID is globally unique and random.

The session state object is created from the HttpSessionState class, which defines a collection of session state items.

The HttpSessionState class has the following properties:

Properties	**Description**
SessionID	The unique session identifier.
Item(name)	The value of the session state item with the specified name. This is the default property of the HttpSessionState class.
Count	The number of items in the session state collection.
TimeOut	Gets and sets the amount of time, in minutes, allowed between requests before the session-state provider terminates the session.

The HttpSessionState class has the following methods:

Methods	**Description**
Add(name, value)	Adds an item to the session state collection.
Clear	Removes all the items from session state collection.
Remove(name)	Removes the specified item from the session state collection.
RemoveAll	Removes all keys and values from the session-state collection.
RemoveAt	Deletes an item at a specified index from the session-state collection.

Application State

TheASP.NET application is the collection of all web pages, code and other files within a single virtual directory on a web server. When information is stored in application state, it is available to all the users.

To provide for the use of application state, ASP.NET creates an application state object for each application from the HTTPApplicationState class and stores this object in server memory. This object is represented by class file global.asax.

Application State is mostly used to store hit counters and other statistical data, global application data like tax rate, discount rate etc. and to keep the track of users visiting the site.

The HttpApplicationState class has the following properties:

Properties	Description
Item(name)	The value of the application state item with the specified name. This is the default property of the HttpApplicationState class.
Count	The number of items in the application state collection.

The HttpApplicationState class has the following methods:

Methods	Description
Add(name, value)	Adds an item to the application state collection.
Clear	Removes all the items from the application state collection.
Remove(name)	Removes the specified item from the application state collection.
RemoveAll	Removes all objects from an HttpApplicationState collection.
RemoveAt	Removes an HttpApplicationState object from a collection by index.
Lock()	Locks the application state collection so only the current user can access it.
Unlock()	Unlocks the application state collection so all the users can access it.

Validates

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

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

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

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

	property.

RequiredFieldValidator Control

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

The syntax of the control is as given:

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

RangeValidator Control

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

It has three specific properties:

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

```
ErrorMessage="Enter your class (6 - 12)"
MaximumValue="12"
MinimumValue="6" Type="Integer">
</asp:RangeValidator>
```

CompareValidator Control

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

It has the following specific properties:

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.
Character Escapes	**Description**
\b	Matches a backspace
\t	Matches a tab
\r	Matches a carriage return
\v	Matches a vertical tab
\f	Matches a form feed
\n	Matches a new line
\	Escape character

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Metacharacters	Description
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Quantifiers could be added to specify number of times a character could appear.

Quantifier	Description
*	Zero or more matches
+	One or more matches
?	Zero or one matches
{N}	N matches
{N,}	N or more matches
{N,M}	Between N and M matches

The syntax of the control is as given:

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

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1"
runat="server"
```

ClientValidationFunction=.cvf_func.
ErrorMessage="CustomValidator">
</asp:CustomValidator>

ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

The following two mutually inclusive properties list out the error message:

- **ShowSummary:** shows the error messages in specified format.
- **ShowMessageBox:** shows the error messages in a separate window.

The syntax for the control is as given:

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

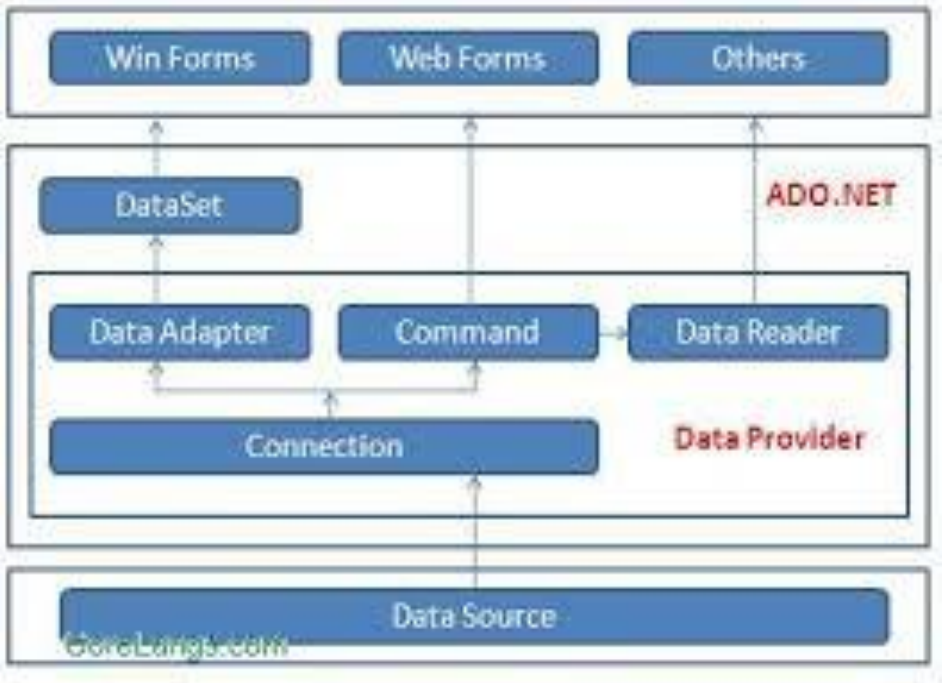
Validation Groups

Complex pages have different groups of information provided in different panels. In such situation, a need might arise for performing validation separately for separate group. This kind of situation is handled using validation groups.

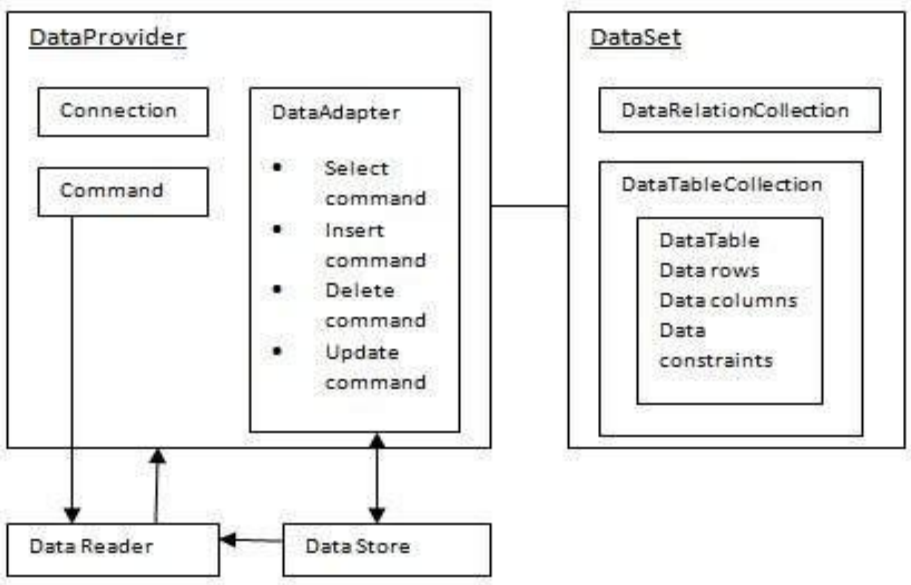
To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup* property.

ADO.NET

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.



The following figure shows the ADO.NET objects at a glance:



The DataSet Class

The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects. The DataRelation objects represent the relationship between two tables.

Following table shows some important properties of the DataSet class:

Properties	**Description**
CaseSensitive	Indicates whether string comparisons within the data tables are case-sensitive.
Container	Gets the container for the component.
HasErrors	Indicates if there are any errors.
IsInitialized	Indicates whether the DataSet is initialized.
Relations	Returns the collection of DataRelation objects.
Tables	Returns the collection of DataTable objects.

The following table shows some important methods of the DataSet class:

Methods	**Description**
AcceptChanges	Accepts all changes made since the DataSet was loaded or this method was called.
BeginInit	Begins the initialization of the DataSet. The initialization occurs at run time.
Clear	Clears data.
Clone	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
Copy	Copies both structure and data.
CreateDataReader()	Returns a DataTableReader with one result set per DataTable, in the same sequence as the tables appear in the Tables collection.
CreateDataReader(DataTable[])	Returns a DataTableReader with one result set per DataTable.
WriteXML()	Writes an XML schema and data from the DataSet. This method has different overloaded forms.
WriteXMLSchema()	Writes the structure of the DataSet as an XML schema. This method has different overloaded forms.

The DataTable Class

The DataTable class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the PrimaryKey property:

Properties	Description
ChildRelations	Returns the collection of child relationship.
Columns	Returns the Columns collection.
Constraints	Returns the Constraints collection.
DataSet	Returns the parent DataSet.
DefaultView	Returns a view of the table.
ParentRelations	Returns the ParentRelations collection.
PrimaryKey	Gets or sets an array of columns as the primary key for the table.
Rows	Returns the Rows collection.

The following table shows some important methods of the DataTable class:

Methods	Description
AcceptChanges	Commits all changes since the last AcceptChanges.
Clear	Clears all data from the table.
GetChanges	Returns a copy of the DataTable with all changes made since the AcceptChanges method was called.
GetErrors	Returns an array of rows with errors.
ImportRows	Copies a new row into the table.
LoadDataRow	Finds and updates a specific row, or creates a new one, if not found any.
Merge	Merges the table with another DataTable.
NewRow	Creates a new DataRow.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
Reset	Resets the table to its original state.
Select	Returns an array of DataRow objects.

The DataRow Class

The DataRow object represents a row in a table. It has the following important properties:

Properties	Description
HasErrors	Indicates if there are any errors.
Items	Gets or sets the data stored in a specific column.
ItemArrays	Gets or sets all the values for the row.
Table	Returns the parent table.

The following table shows some important methods of the DataRow class:

Methods	Description
AcceptChanges	Accepts all changes made since this method was called.
BeginEdit	Begins edit operation.
CancelEdit	Cancels edit operation.
Delete	Deletes the DataRow.
EndEdit	Ends the edit operation.
GetChildRows	Gets the child rows of this row.
GetParentRow	Gets the parent row.
GetParentRows	Gets parent rows of DataRow object.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.

The DataAdapter Object

The DataAdapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.

The DataReader Object

The DataReader object is an alternative to the DataSet and DataAdapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

DbCommand and DbConnection Objects

The DbConnection object represents a connection to the data source. The connection could be shared among different command objects.

The DbCommand object represents the command or a stored procedure sent to the database from retrieving or manipulating data.

AJAX stands for **Asynchronous JavaScript and XML.** This is a cross platform technology which speeds up response time. The AJAX server controls add script to the page which is executed and processed by the browser.

However like other ASP.NET server controls, these AJAX server controls also can have methods and event handlers associated with them, which are processed on the server side.

The control toolbox in the Visual Studio IDE contains a group of controls called the 'AJAX Extensions'

The ScriptManager Control

The ScriptManager control is the most important control and must be present on the page for other controls to work.

It has the basic syntax:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

The UpdatePanel Control

The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Example

Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

The design view looks as follows:

The source file is as follows:

```
<form id="form1" runat="server">
  <div>
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
  </div>

  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Button ID="btnpartial" runat="server" onclick="btnpartial_Click" Text="Partial
PostBack"/>
      <br />
      <br />
      <asp:Label ID="lblpartial" runat="server"></asp:Label>
    </ContentTemplate>
  </asp:UpdatePanel>

  <p> </p>
  <p>Outside the Update Panel</p>
  <p>
    <asp:Button ID="btntotal" runat="server" onclick="btntotal_Click" Text="Total PostBack" />
  </p>

  <asp:Label ID="lbltotal" runat="server"></asp:Label>
</form>
```

Both the button controls have same code for the event handler:

```
string time = DateTime.Now.ToLongTimeString();
lblpartial.Text = "Showing time from panel" + time;
lbltotal.Text = "Showing time from outside" + time;
```

Observe that when the page is executed, if the total post back button is clicked, it updates time in both the labels but if the partial post back button is clicked, it only updates the label within the update panel.

A page can contain multiple update panels with each panel containing other controls like a grid and displaying different part of data.

When a total post back occurs, the update panel content is updated by default. This default mode could be changed by changing the UpdateMode property of the control. Let us look at other properties of the update panel.

Properties of the UpdatePanel Control

The following table shows the properties of the update panel control:

Properties	**Description**

ChildrenAsTriggers	This property indicates whether the post backs are coming from the child controls, which cause the update panel to refresh.
ContentTemplate	It is the content template and defines what appears in the update panel when it is rendered.
ContentTemplateContainer	Retrieves the dynamically created template container object and used for adding child controls programmatically.
IsInPartialRendering	Indicates whether the panel is being updated as part of the partial post back.
RenderMode	Shows the render modes. The available modes are Block and Inline.
UpdateMode	Gets or sets the rendering mode by determining some conditions.
Triggers	Defines the collection trigger objects each corresponding to an event causing the panel to refresh automatically.

Methods of the UpdatePanel Control

The following table shows the methods of the update panel control:

Methods	**Description**
CreateContentTemplateContainer	Creates a Control object that acts as a container for child controls that define the UpdatePanel control's content.
CreateControlCollection	Returns the collection of all controls that are contained in the UpdatePanel control.
Initialize	Initializes the UpdatePanel control trigger collection if partial-page rendering is enabled.
Update	Causes an update of the content of an UpdatePanel control.

The behavior of the update panel depends upon the values of the UpdateMode property and ChildrenAsTriggers property.

UpdateMode	**ChildrenAsTriggers**	**Effect**
Always	False	Illegal parameters.
Always	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back.
Conditional	False	UpdatePanel refreshes if whole page refreshes or a triggering control outside it initiates a refresh.
Conditional	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back or a triggering control outside it initiates a refresh.

The UpdateProgress Control

The UpdateProgress control provides a sort of feedback on the browser while one or more update panel controls are being updated. For example, while a user logs in or waits for server response while performing some database oriented job.

It provides a visual acknowledgement like "Loading page...", indicating the work is in progress.

The syntax for the UpdateProgress control is:

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1" >

  <ProgressTemplate>
    Loading...
```

```
  </ProgressTemplate>

</asp:UpdateProgress>
```

The above snippet shows a simple message within the ProgressTemplate tag. However, it could be an image or other relevant controls. The UpdateProgress control displays for every asynchronous postback unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

Properties of the UpdateProgress Control

The following table shows the properties of the update progress control:

Properties	**Description**
AssociatedUpdatePanelID	Gets and sets the ID of the update panel with which this control is associated.
Attributes	Gets or sets the cascading style sheet (CSS) attributes of the UpdateProgress control.
DisplayAfter	Gets and sets the time in milliseconds after which the progress template is displayed. The default is 500.
DynamicLayout	Indicates whether the progress template is dynamically rendered.
ProgressTemplate	Indicates the template displayed during an asynchronous post back which takes more time than the DisplayAfter time.

Methods of the UpdateProgress Control

The following table shows the methods of the update progress control:

Methods	**Description**
GetScriptDescriptors	Returns a list of components, behaviors, and client controls that are required for the UpdateProgress control's client functionality.
GetScriptReferences	Returns a list of client script library dependencies for the UpdateProgress control.

The Timer Control

The timer control is used to initiate the post back automatically. This could be done in two ways:

(1) Setting the Triggers property of the UpdatePanel control:

```
<Triggers>
  <asp:AsyncPostBackTrigger ControlID="btnpanel2" EventName="Click" />
</Triggers>
```

(2) Placing a timer control directly inside the UpdatePanel to act as a child control trigger. A single timer can be the trigger for multiple UpdatePanels.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Always">
  <ContentTemplate>
    <asp:Timer ID="Timer1" runat="server" Interval="1000">
      </asp:Timer>

    <asp:Label ID="Label1" runat="server" Height="101px" style="width:304px" >
      </asp:Label>
  </ContentTemplate>
  </asp:UpdatePanel>
```

List of Questions:

1. Explain about ASP
2. Explain about .Net
3. What is ASP.Net?
4. Differentiate between ASP and JSP
5. Differentiate between ASP and ASP.Net
6. Explain about components of .Net framework
7. Explain about ASP.Net life cycle in detail
8. List and explain page life cycle events of ASP.Net
9. What are challenges faced by Web Applications?
10. List and explain various web controls of ASP.Net in detail
11. Write about state management in ASP.Net
12. Explain various validators in ASP.Net
13. Draw and explain the purpose of ADO.Net architecture
14. Write shot notes on AJAX Controls