**Chapter-4**

**The Server Tier**

**WEB SERVER CONCEPT**

* A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web.
* The main job of a web server is to display website content through storing, processing and delivering webpages to users. Besides HTTP, web servers also support SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer and storage.
* Web servers are used in web hosting, or the hosting of data for websites and web-based applications.
* The most common type of web server which runs in the web hosting server is Apache. Another very popular Web server is nginx. Internet Information Services (IIS, formally Internet Information Server) is an extensible web server created by Microsoft for use with the Windows NT family.
* IIS Server supports HTTP, HTTP/2, HTTPS, FTP, FTPS, SMTP and NNTP. It has ability to handle server side programming, security characteristics and the particular publishing, search engine and site building tools that come with it.

**The main purpose of web server are as follows:**

* It store web site files and broadcast them over the internet for site visitors to see the content of web pages.
* Web server is simply a powerful computer that stores and transmits data via the Internet.
* When someone visits a web page on your site their browser communicates with your web server, sending and receiving information that ultimately dictates what appears on the visitor’s computer screen.
* Finally, the main purpose of web server is to store and transfer website data upon the request of a visitor’s browser.

**Function/Application of web Server**

* The functions of web server are as follows:

1. **Store and secures website data**

In web hosting services a web server stores all website data and secures it from unauthorized users when it is properly configured.

1. **Provides web database access**

A web server’s responsibility is to provide access to websites that are hosted. Web hosting service providers own some web servers that are used in variables ways to provide different web hosting services such as backend database servers.

1. **Serve the end user request**

Web servers accept requests from different users connected over the internet and serve them accordingly.

1. **Bandwidth controlling to regular network traffic**

It is a feature available in web server to minimize excess network traffic. Web Hosing can set bandwidth values to regulate the rate of data transmission over the internet. This features avoids the down time caused by high web traffic.

1. **Virtual hosting**

Virtual hosting is a type of web hosting service in which a web server is used to host other software based virtual web Servers web sites, data, applications and other services. Virtualized web servers do possess this feature to provide virtual hosting.

1. **Server side web scripting**

This features of web server enables the user to create dynamic web pages. The popular server side Scripting languages include Perl, Ruby, Python, PHP, and ASP etc.

**DYNAMIC CONTENT**

* Dynamic content is the term used to describe site content that automatically adapts to various pre-defined conditions, such as particular user signals. The content featured on the website will adjust dynamically or under the control of CMS, so different users see different content.
* The major benefits are as follows:

1. **It is personalizable**

Every single response can be crafted based on the user’s request or previous visits to the site.

1. **It is extensible**

The scripting languages that power dynamic content can tap into a variety of external resources including the local file system, databases or other servers.

1. **It is interactive**

Dynamic websites are a two-way street: they provide content to users, and they allow users to submit information back to the server.

1. **It is seamless**

By combining server-side scripts with scripts that run in the browser, websites can load dynamic content over an already loaded web page without the user being aware of it.

**CREATING DYNAMIC CONTENT**

Dynamic content is any digital or online content that changes based on data, user behavior and preferences. This can be text, audio or video format content.

In other words, Dynamic content is a web page or email component that changes. Typically, changes are based on user signals that include in-session behavior, user data and user characteristics.

In-session behavior, adapt content based on what pages they visit, which products they add to cart and how long they spend on site.

* **User-data:**

Change content based on past purchases, customer lifecycle, or past engagements with your marketing.

* **User Characteristics**

Adapt content based on demographics such as geo-location or buyer persons.

Web servers are increasingly being used to deliver dynamic content rather than static HTML pages. In order to generate web pages dynamically, servers need to execute a script, which typically connects to DBMS.

To generate dynamic contents, web servers need to execute a program, through some server side scripting mechanism. This script typically connects to a DBMS, performs a query, retrieves the results and formats them in HTML in order to be returned to the user.

Figure below illustrate the dynamic content generation as

www.abc.org

Name

Email

www.abc.org

Thank you for registration

Client

Encode from data

HTTP

Web Server

Read & parse

User input

Perform processing

Print result as HTML

Script

HTTP

Fig: Dynamic Content Generation

**Process how dynamic content generation code works**

* First user brows web application by typing the web address into the web browser.
* Web browser send the request to web server using HTTP methods using HTTP Header.
* Web server fetch requested file or data from server and perform processing to get final HTML output.
* After processing server send HTML code back to web browser.
* When an ASP.Net application is launched, there are a series of steps which are carried out. These series of steps make up the lifecycle of the application.
* Figure below illustrate the lifecycle of the application as.

Application End

Dispose

Application Start

Object Creation

HTTP Application creation

Fig: Lifecycle of Application

* **Application Start**

The life cycle of an ASP.NET application starts when a request is made by a user. This request is to the Web server for the ASP.Net Application. This happens when the first user normally goes to the home page for the application for the first time. During this time, there is a method called Application start which is executed by the web server. Usually, in this method, all global variables are set to their default values.

* **Object Creation**

The next stage is the creation of the HttpContext, HttpRequest & HttpResponse by the web server. This HttpContext is just the container for the HttpRequest and HttpResponse objects. The HttpRequest object contains information about the current request, including cookies and browser information. The HttpRsponse object contains the response that is sent to the client.

* **HTTP Application Creation**

This object is created by the web server. It is this object that is used to process each subsequent request sent to the application. For example, let’s assume we have 2 web applications. One is a shopping cart application and the other is a news website. For each application, we would have 2 HttpApplication objects created. Any further requests to each website would be processed by each HttpApplication respectively.

* **Dispose**

This event is called before the application instance is destroyed. During this time, one can use this method to manually release any unmanaged resources.

* **Application End**

This is the final state of the application. In this part, the application is finally unloaded from memory.

**ASP.NET PAGE LIFECYCLE**

* When an ASP.Net page is called, it goes through a particular lifecycle. This is done before the response is sent to the user. There are a series of steps which are followed for the processing of an ASP.Net page. Figure below illustrate the lifecycle of ASP.Net page as:

Page Request

Page Start

Page Initialization

Post back event handling

Validation

Page Load

Page Rendering

Unload

Fig: ASP.Net Page lifecycle

* **Page Request**

This is when the page is first requested from the server. When the page is requested the server checks if it is requested for the first time. If so, then it needs to compile the page, parse the response and send it across to the user. If it is not the first time the page is requested, the cache is checked to see if the page output exists. If so, that response is sent to the user.

* **Page Start**

During this time two objects known as the Request and Response object are created. The request object is used to hold all the information which was sent when the page was requested. The response object is used to hold the information which is sent back to the user.

* **Page Initialization**

During this time, all the controls on a web page is initialized. So, if you have any label, textbox or any other controls on the web form, they are all initialized.

* **Post back event handling**

This event is triggered if the same page is being loaded again. This happens in response to an earlier event. Sometimes there can be a situation that a user clicks on a submit button on the page. In this case, the same page is displayed again. In such case, the Post back event handler is called.

* **Validation**

Sometimes there can be some validation set on the form. For example, there can be a validation which says that a list box should have a certain set of values. If the condition is false then there should be an error in loading the page.

* **Page Load**

This is when the page is actually loaded with all the default values. So, if a textbox is supposed to have a default value, that value is loaded during the page load time.

* **Page Rendering**

This happens just before all the response information is sent to the user. All the information on the form is saved and the result is sent to the user as a compete web page.

* **Unload**

Once the page output is sent to the user, there is no need to keep the ASP.Net web form objects in memory. So, the unloading process involves removing all unwanted objects from memory.

**HANDLING SESSION AND COOKIE IN ASP.NET**

**Session:**

* Session is a state management technique that is used to manage the state of a page or control throughout the application. So I mean to say that using the session we can store the value and access it in another page or throughout the application.

**Why we use session?**

* As we know that HTTP is a stateless protocol, it means every request is an independent request. Browser does not know about previous request and data. So, every previous request data lost and you cannot retrieve previous data.
* Basically, using the session, we store the data in session and access it anywhere as per requirement. It provide application level state management.

Example: Session[“Name”]= “PMC”;

Note: It help to identify request from the same browser during a time period (session). It used to store value the particular time session. Bydefault ASP.net session state is enable for all ASP.net application.

**Problem with ASP Session State**

* ASP .Net developers know session state as a great feature, but one that is somewhat limited. These limitations include the following factors:
* **Process dependent**

ASP session state exists in the process that hosts ASP, thus the actions that affect the process also affect session state. When the process is recycled or fails, session state is lost.

* **Server farm limitations**

As users move from server to server in a web server farm, their session state does not follow them. ASP session state is machine specific. Each ASP server provides its own session state and unless the user returns to the same server, the session state is inaccessible. While network IP level routing solutions can solve such problems, by ensuring that client IPs are routed to the originating server, some ISP’s choose to use a proxy load-balancing solution for their clients. Most in famous of these is AOL. Solution such as AOL’s prevent network level routing of requests to servers because the IP addresses for the requestor cannot be guaranteed to be unique.

* **Cookie dependent**

Clients that don’t accept HTTP cookies can’t take advantage of session state. Some client s believe that cookies compromise security and privacy and thus disable them, which disables session state on the server.

**ASP.Net Session State**

* ASP .Net session state solves all of the above problems associated with classic ASP session state:
* **Process independent**

ASP.Net session state is able to run in a separate process from the ASP.Net host process. If session state is in a separate process, the ASP .Net process can come and go while the session state process remains available. Of course, you can still use session state in process similar to classic ASP .Net too.

* **Support for server farm configuration**

By moving to an out-of-process model, ASP .Net also solves the server farm problem. The new out-of-process model allows all servers in the farm to share a session state process. You can implement this by changing the ASP.Net configuration to point to a common server.

* **Cookie independent**

Although solution to the problem of cookieless state management do exist for classic ASP, they are not trivial to implement ASP .Net, on the other hand, reduces the complexities of cookieless session state to a simple configuration setting.

**Handling Cookies**

* Cookies is a small piece of information state on the client machine. The file is located on client machine C:\Document and Settings\Currently\_Login User Cookie path.
* It is used to store user performance information like username, password, city, state and phone no etc. on client machine. We need to input namespace called **System.WebHttpCookie** before we use cookie. This information can be read by the web application whenever user visits site.

**Types of Cookies**

1. Persist cookie: A cookie has not have expire time.
2. Non-Persist Cookie: A cookie has expire time.

Note:

* When user request for a web page, web server sends not just a page, but also a cookie containing the date and time, this cookie store in a folder on the user’s hard disk.
* When the user request for the web page again, browser looks on the hard drive for the cookie associated with the web page, browser store separate cookie for each different site user visited.
* The cookie is limited to small size and can be used to store only 4KB (4096 Bytes) text.
* There are two methods to store cookies in ASP.NET application.

1. Cookies Collection
2. HttpCookie

**1. Cookies Collection:**

**Adding cookie:**

Response.Cookies[“UserName”].Value= “PMC”;

**Retrieving Cookie:**

Var Name = Response.Cookies[“UserName”].Value.ToString();

**Expiring Cookie:**

Response.Cookies[“UserName”].Expires=DateTime.Now.AddDays(-1);

**2. HttpCookie**

**Adding Cookie:**

HttpCookie objcookie= new HttpCookie(“UserName”);

Objcookie.Value = “PMC”;

Response.Cookies.Add(objcookie);

**Retriving Cookie:**

Var Name = Response.Cookies[“UserName”].Value;

**Expiring Cookie:**

Objcookie.Expires.Add(new TimeSpan(0,1,0));

Response.Cookies.Add(objcookie);

**Difference between Session and Cookies**

* Cookies are client side files that contain user information whereas session are server side file that contain user information.
* Cookie is not dependent on session whereas session is dependent on cookies.
* Cookie expires depending on the lifetime you set for it whereas a session ends when a user closes browser.
* The maximum cookie size is 4KB whereas in session you can store as much data as you like.
* Cookie does not have a function named unsetcookies () whereas in session you can use session\_destroy(); which is used to destroy all registered data or to unset some.
* You don’t need to start cookie as it is stored in your local machine Whereas in session need to start session like Session\_Start();

**ERROR HANDLING**

* Error handling refers to the anticipation, detection and resolution of programming, application and communications error.
* The best technique for error handling if possible, recover from them when they occur without terminating the application or (if all else fails) gracefully terminate an affected application and save the error information to a log file.
* In order to handle error in ASP .Net we can add error handling settings in the web.config file of the application. It is application-wide error handling. For example, you can add the following lines in the web.config file.

**<configuration>**

**<system.web>**

**<customErrors mode=”RemoteOnly” defaultRedirect = “GenericErrorPage.html”/>**

**<error statusCode = “403” redirect=”NoAccess.html”/>**

**<error statusCode = “404” redirect = “FileNotFound.html”/>**

**</customErrors>**

**</system.web>**

**</configuration>**

**Exception Handling**

* Process to handle runtime error is known as exception handling.
* Exceptions are nothing but unseen error occurs when executing code logic.

**Using try catch finally Block**

**try:**

It contains code that can cause exception.

**catch:**

It contains code that handle it.

**finally:**

code inside it always run. i.e. exception

**throw:{**

User can throw exception on certain condition.

Some exception classes are:

* System.DivideByZeroException
* System.IO.IOException
* System.NullReferenceException

Example:

Using System;

Class Abc

{

Public static void Main(String[] args)

{

Int i=10;

Int j = 0;

// Int result = i/j;

// Console.WriteLine(“End”);

try

{

Int result=i/j;

}

catch(Exception e)

{

Print msg; “divide by zero not allowed”

}

Console.WriteLine(“End”);

Finally

{

}

}

}

Out Put:

Msg

End