Introduction

1. HTTP Protocol

1>HTTP is the foundation of data communication for the in the World Wide Web

2> HTTP functions as a request-response protocol in the client-server computing Model. In HTTP, a web browser for example, acts as a *client*, while an application running on a computer hosting a website functions as a server. The client submits an HTTP *request* message to the server. The server, which stores content, or provides *resources*, such as HTML files or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

It is a stateless protocol, you send request and it responds does not maintain any state. To maintain the state, we can use

1>Session Object in the server side

2>Cookies in the client side.

3>Hidden variables in the form

2 Static and Dynamic Pages

Static Web Pages

A **static web page** (sometimes called a **flat page)** is a web page that is delivered to the user exactly as stored, in contrast to dynamic web pages which are generated by a web application.

Consequently a static web page displays the same information for all users, from all contexts, subject to modern capabilities of a web server to negotiate content-type or language of the document where such versions are available and the server is configured to do so.

Static web pages are often HTML documents stored as files in the file system and made available by the web server over HTTP.

Advantages

- No programming skills are required to create a static page.
- Inherently publicly cacheable (ie. a cached copy can be shown to anyone).
- No particular hosting requirements are necessary.
- Can be viewed directly by a web browser without needing a web server or application server

Disadvantages

- Any personalization or interactivity has to run client-side (ie. in the browser), which is restricting.
- Maintaining large numbers of static pages as files can be impractical without automated tools.

Dynamic Web pages

A **dynamic web page** is a kind of web page that has been prepared with recent information, for each individual viewing. It changes with the time. For ex. News content, events. The content of the site changes when the user logs in and logs out. Page contents are displayed according to the location of the user. In facebook, you can see suggestions to add friends since it knows your friends and those are mostly friend of your friends. Facebook gets updated with recent feeds etc.

Dynamic web pages can be created in two ways

1>**Client-Side Scripting**

Using client-side scripting to change interface behaviors *within* a specific web page, in response to mouse or keyboard actions or at specified timing events. In this case the dynamic behavior occurs within the presentation.

Client-Side scripting languages like JavaScript or Actionscript, used for DHTML and Flash technologies respectively, are frequently used to play with the media types (sound, animations, changing text, etc.) of the presentation

The Client-side content is generated on the user's computer. The web browser retrieves a page from the server, then processes the code embedded in the page (often written in JavaScript) and displays the retrieved page's content to the user.

2>**Server-Side Scripting**

A program running on the web server (server-side scripting) is used to change the web content on various web pages, or to adjust the sequence of or reload of the web pages. Server responses may be determined by such conditions as data in a posted HTML form, parameters in the URL, the type of browser being used, the passage of time, or a database or server state.
Such web pages are often created with the help of server-side languages such as ASP, PHP, JSP etc.

3>**Combined Client Side and Server Side**

Ajax is a web development technique for dynamically interchanging content with the server-side, without reloading the web page. Google Maps is an example of a web application that uses Ajax techniques and database.

Disadvantages

Search engines work by creating indexes of published HTML web pages that were, initially, "static". With the advent of dynamic web pages, often created from a private database, the content is less visible. Unless this content is duplicated in some way (for example, as a series of extra static pages on the same site), a search may not find the information it is looking for. It is unreasonable to expect generalized web search engines to be able to access complex database structures, some of which in any case may be secure.

Introduction to ASP

Microsoft Active Server Pages (ASP) is a server-side scripting technology. ASP is a technology that Microsoft created to ease the development of interactive Web applications.

Using server side scripting language like ASP we can manage the content of any page and such dynamic code (or content) for the web browsers can be generated based on various conditions we set in our ASP program.

ASP engine finishes its job of processing the code and then send the codes to users browser. From this point on words till again the page request comes back to server, there is no control of ASP on the page. So we should not expect ASP to perform some tasks which are likely to happen at the client browser end. This will be clear when we discuss some of the task and where (client or server side) the task is to be completed and which script will take care of it.

ASP provides solutions for transaction processing and managing session state. Asp is one of the most successful language used in web development.

Benefits and Application of ASP

- Dynamically edit, change, or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security - since ASP code cannot be viewed from the browser
- Clever ASP programming can minimize the network traffic

Problems with traditional ASP

1. **Interpreted and Loosely-Typed Code**
   ASP scripting code is usually written in languages such as JScript or VBScript. The script-execution engine that Active Server Pages relies on interprets code line by line, every time the page is called. In addition, although variables are supported, they are all loosely typed as variants and bound to particular types only when the code is run. Both these factors impede performance, and late binding of types makes it harder to catch errors when you are writing code.

2. **Mixes layout (HTML) and logic (scripting code)**
   ASP files frequently combine script code with HTML. This results in ASP scripts that are lengthy, difficult to read, and switch frequently between code and HTML. The interspersion of HTML with ASP code is particularly problematic for larger web applications, where content must be kept separate from business logic.

3. **Limited Development and Debugging Tools**
   Microsoft Visual InterDev, Macromedia Visual UltraDev, and other tools have attempted to increase the productivity of ASP programmers by providing graphical development environments. However, these tools never achieved the ease of use or the level of acceptance achieved by Microsoft Windows application development tools, such as Visual Basic or Microsoft Access. ASP developers still rely heavily or exclusively on Notepad.

   Debugging is an unavoidable part of any software development process, and the debugging tools for ASP have been minimal. Most ASP programmers
   resort to embedding temporary Response. Write statements in their code to trace the progress of its execution.

4. **No real state management**
   Session state is only maintained if the client browser supports cookies. Session state information

can only be held by using the ASP Session object. And you have to implement additional code if you, for example, want to identify a user.

5. **Update files only when server is down**
   If your Web application makes use of components, copying new files to your application should only be done when the Web server is stopped. Otherwise it is like pulling the rug from under your application's feet, because the components may be in use (and locked) and must be registered.

6. **Obscure Configuration Settings**
   The configuration information for an ASP web application (such as session state and server timeouts) is stored in the IIS metabase. Because the metabase is stored in a proprietary format, it can only be modified on the server machine with utilities such as the Internet Service Manager. With limited support for programmatically manipulating or extracting these settings, it is often an arduous task to port an ASP application from one server to another.

Introduction to ASP.NET

ASP.NET Overview

Some point that gives the quick overview of ASP.NET.

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services
- Like ASP, ASP.NET is a server-side technology
- Web Applications are built using Web Forms. ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications

Advantages of ASP.NET

1. **Separation of Code from HTML**
   To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic. This makes it much easier for teams of programmers and designers to collaborate efficiently. This makes it much easier for teams of programmers and designers to collaborate efficiently.

2. **Support for compiled languages**
   Developer can use VB.NET and access features such as **strong typing** and **object-oriented programming**. Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code. ASP.NET pages are precompiled to byte-code and Just In Time (JIT) compiled when first requested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.

3. **Use services provided by the .NET Framework**
   The .NET Framework provides class libraries that can be used by your application. Some of the key classes help you with input/output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.

4. **Graphical Development Environment**
   Visual Studio .NET provides a very rich development environment for Web developers. You can drag and drop controls and set properties the way you do in Visual Basic 6. And you have full Intelligence support, not only for your code, but also for HTML and XML.

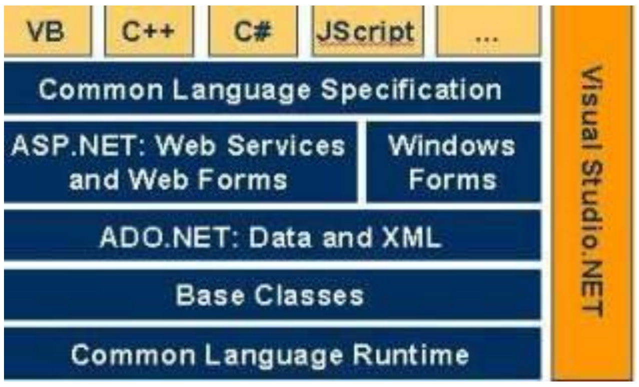
5. **State management**
   To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management. State information can, for example, be kept in memory or stored in a database. It can be shared across Web farms, and state information can be recovered, even if the server fails or the connection breaks down.

6. **Update files while the server is running!**
   Components of your application can be updated while the server is online and clients are connected. The Framework will use the new files as soon as they are copied to the application. Removed or old files that are still in use are kept in memory until the clients have finished.

7. **XML-Based Configuration Files**
   Configuration settings in ASP.NET are stored in XML files that you can easily read and edit. You can also easily copy these to another server, along with the other files that comprise your application.

ASP.NET Architecture

ASP.NET is based on the fundamental architecture of .NET Framework.



Architecture is explained form bottom to top in the following discussion.

1. At the bottom of the Architecture is Common Language Runtime. NET Framework common language runtime resides on top of the operating system services. The common language runtime loads and executes code that targets the runtime. This code is therefore called managed code. The runtime gives you, for example, the ability for cross-language integration.

2. .NET Framework provides a rich set of class libraries. These include base classes, like networking and input/output classes, a data class library for data access, and classes for use by programming tools, such as debugging services. All of them are brought together by the Services Framework, which sits on top of the common language runtime.

3. ADO.NET is Microsoft ActiveX Data Object (ADO) model for the .NET Framework. ADO.NET is not simply the migration of the popular ADO model to the managed environment but a

completely new paradigm for data access and manipulation.

ADO.NET is intended specifically for developing web applications. This is evident from its two major design principles:

1. Disconnected Datasets In ADO.NET, almost all data manipulation is done outside the context of an open database connection.
2. Effortless Data Exchange with XMLDatasets can converse in the universal data format of the Web, namely XML.

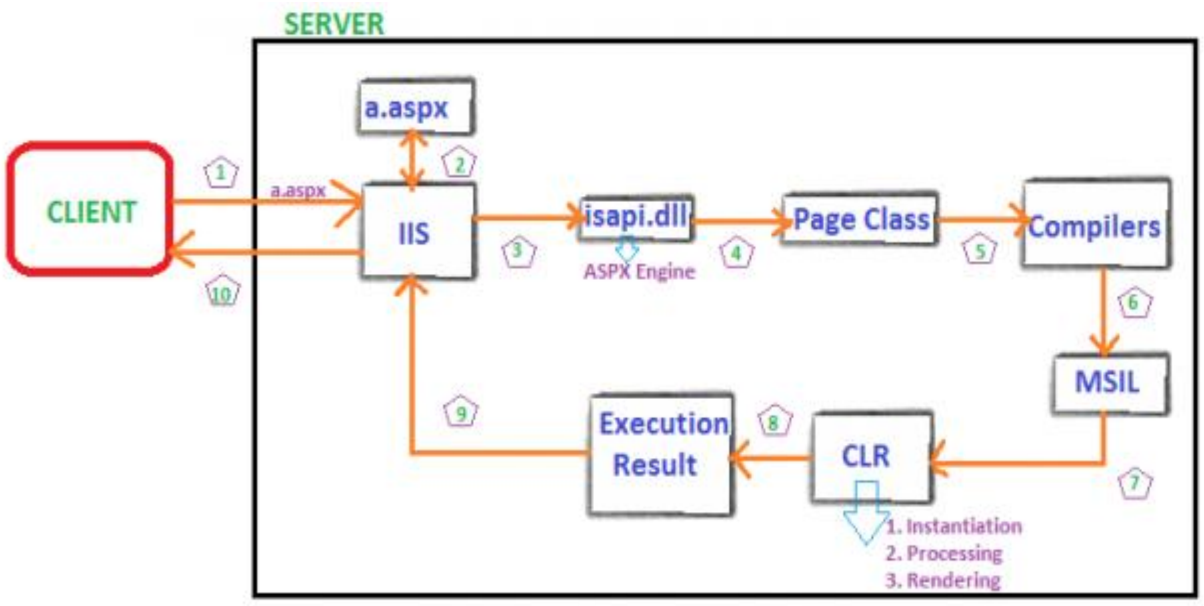
4. The 4th layer of the framework consists of the Windows application model and, in parallel, the Web application model.
   The Web application model-in the slide presented as ASP.NET-includes Web Forms and Web Services.
   ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.

   Web Services brings you a model to bind different applications over the Internet. This model is based on existing infrastructure and applications and is therefore standard-based, simple, and adaptable.

   Web Services are software solutions delivered via Internet to any device. Today, that means Web browsers on computers, for the most part, but the device-agnostic design of .NET will eliminate this limitation.

5. One of the obvious themes of .NET is unification and interoperability between various programming languages. In order to achieve this; certain rules must be laid and all the languages must follow these rules. In other words we can not have languages running around creating their own extensions and their own fancy new data types. CLS(Common Language Specification) is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow.
6. The CLR and the .NET Frameworks in general, however, are designed in such a way that code written in one language can not only seamlessly be used by another language. Hence ASP.NET can be programmed in any of the .NET compatible language whether it is VB.NET, C#, Managed C++ or JScript.NET.

<u>Execution Process</u>



Compilation, when page is requested the first time
The first time a page is requested, the code is compiled. Compiling code in .NET means that a compiler in a first step emits Microsoft intermediate language (MSIL) and produces metadata if you compile your source code to managed code. In a following step MSIL has to be converted to native code.

Microsoft intermediate language (MSIL)
Microsoft intermediate language is code in an assembly language like style. It is CPU independent and therefore can be efficiently converted to native code.

The conversion in turn can be CPU-specific and optimized. The intermediate language provides a hardware abstraction layer.

MSIL is executed by the common language runtime.

Common language runtime
The common language runtime contains just-in-time (JIT) compilers to convert the MSIL into native code. This is done on the same computer architecture that the code should run on.

The runtime manages the code when it is compiled into MSIL the code is therefore called managed code.

IIS(Internet Information Server)

Internet Information Services (**IIS**) – formerly called **Internet Information Server** – is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. It is the most used web server after Apache HTTP Server. IIS 7.5 supports HTTP, HTTPS, FTP, FTPS, SMTP .
Hypertext Transfer Protocol Secure (**HTTPS**) is a combination of the Hypertext Transfer Protocol (HTTP) with SSL/TLS protocol to provide encrypted communication and secure identification of a

network web server. HTTPS connections are often used for payment transactions on the World Wide Web and for sensitive transactions in corporate information systems

It is an integral part of Windows Server family of products, as well as certain editions of Windows XP, Windows Vista and Windows 7. IIS is not turned on by default when Windows is installed.

IIS Features

The architecture of IIS 7 is modular. Modules, also called extensions, can be added or removed individually so that only modules required for specific functionality have to be installed. IIS 7 includes native modules as part of the full installation. These modules are individual features that the server uses to process requests and include the following:

Native modules – Come with the IIS installation

- **HTTP modules** – Used to perform tasks specific to HTTP in the request-processing pipeline, such as responding to information and **inquiries sent in client headers, returning HTTP errors, and redirecting requests.**

- **Security modules** – Used to perform tasks related to security in the request-processing pipeline, such as specifying authentication schemes, performing URL authorization, and filtering requests. Prohibits file listing.

- **Content modules** – Used to perform tasks related to content in the request-processing pipeline, such as processing requests for static files, returning a default page when a client does not specify a resource in a request, and listing the contents of a directory. We can set the default file names in IIS.

- **Compression modules** – Used to perform tasks related to compression in the request-processing pipeline, such as compressing responses, applying Gzip compression transfer coding to responses, and performing pre-compression of static content.

- **Caching modules** – Used to perform tasks related to caching in the request-processing pipeline, such as storing processed information in memory on the server and using cached content in subsequent requests for the same resource.

- **Logging and Diagnostics modules** – Used to perform tasks related to logging and diagnostics in the request-processing pipeline, such as passing information and processing status to HTTP.sys for logging, reporting events, and tracking requests currently executing in worker processes.

Sites, Applications, and Virtual Directories in IIS

IIS formalizes the concepts of sites, applications, and virtual directories. Virtual directories and applications are now separate objects, and they exist in a hierarchical relationship in the IIS configuration schema. Briefly, a site contains one or more applications, an application contains one or more virtual directories, and a virtual directory maps to a physical directory on a computer.

As in IIS 6.0, a site contains all the content, both static and dynamic, that is associated with that site. However, each site must contain at least one application, which is named the root application. And each application (including the root application) must contain at least one virtual directory, which is named the root virtual directory. These objects work together to form the site.

The following sections explain sites, applications, virtual directories, and their related configurations in more detail.

Sites
A site is a container for applications and virtual directories, and you can access it through one or more **unique bindings.**

The binding includes two attributes important for communication: the ***binding protocol*** **and the** ***binding information***. The binding protocol defines the protocol over which communication between the server and client occurs. The binding information defines the information that is used to access the site. For example, the binding protocol of a Web site can be either HTTP or HTTPS, and the binding information is the combination of IP address, port, and optional host header.

A site may contain more than one binding if the site requires different protocols or binding information. In earlier versions of IIS, only the HTTP and HTTPS protocols were supported. For example, a Web site might have had both an HTTP binding and an HTTPS binding when sections of the site required secure communication over HTTPS(for example when you do transactions online, credit card information sharing) .

Applications
An application is a group of files that delivers content or provides services over protocols, such as HTTP. When you create an application in IIS, the application's path becomes part of the site's URL.

In IIS 7, each site must have an application which is named the root application, or default application. However, a site can have more than one application. For example, you might have an online commerce Web site that has several applications, such as a shopping cart application that lets users gather items during shopping and a login application that allows users to recall saved payment information when they make a purchase.

Virtual Directories
A virtual directory is a directory name (also referred to as path) that you specify in IIS and map to a physical directory on a local or remote server. The directory name then becomes part of the application's URL, and users can request the URL from a browser to access content in the physical directory, such as a Web page or a list of additional directories and files. If you specify a different name for the virtual directory than the physical directory, it is more difficult for users to discover the actual physical file structure on your server because the URL does not map directly to the root of the site.

In IIS 7, each application must have a virtual directory, which is named the root virtual directory, and which maps the application to the physical directory that contains the application's content. However, an application can have more than one virtual directory. For example, you might use a virtual directory when you want your application to include images from another location in the file system, but you do not want to move the image files into the physical directory that is mapped to the application's root virtual directory.

By default, IIS uses configuration from Web.config files in the physical directory to which the virtual directory is mapped, as well as in any child directories in that physical directory.

Optionally, when you need to specify credentials and a method to access the virtual directory, you can specify values for the **username**, **password**, and **logonMethod** attributes.