Uttar Pradesh Rajarshi Tandon Open University

Bachelor in Computer Application

BCA-E10

Client Server Technology

Uttar Pradesh Rajarshi Tandon Open University

Bachelor in Computer Application

BCA-E10

Client Server Technology

BLOCK

1

INTRODUCTION TO CLIENT-SERVER COMPUTING

UNIT-1

Introduction to Client Server Computing

UNIT-2

Distributed Computing

UNIT-3

Designing Client Server Applications

Course Design Committee

Dr. Ashutosh Gupta, **Chairman**
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Prof. R.S. Yadav **Member**
Dept. of Computer Science and Engineering, MNNIT, Allahabad

Ms. Marisha **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Course Preparation Committee

Dr. Krishan Kumar **Author**
Assistant Professor,
Department of Comuter Science Faculty of Technology
Gurukula Kangri Vishwavidyalaya, Haridwar (UK)

Dr. V.K. Saraswat **Editor**
Director (IET, Khandare Campus)
Institute of Engineering and Technology
Dr. B.R. Ambedkar University, Agra-282002

Dr. Ashutosh Gupta,
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

©UPRTOU, Prayagraj-2020
ISBN : 978-93-83328-13-0

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2020.
Printed By : Chandrakala Universal Pvt. 42/7 Jawahar Lal Neharu Road, Prayagraj.

BLOCK INTRODUCTION

Block-1 basically contains three units which are mainly intended with Client-Server technology and its design applications. *Unit 1* introduces the history, evolution and notion of Client-Server technology and also gives its fundamental principles. Moreover, Client/server is a program relationship in which one program (the client) requests a service or resource from another program (the server). Although the Client-Server model can be used by programs within a single computer, it is an important concept for networking. In this case, the client establishes a connection to the server over a local area network (LAN) or wide-area network (WAN), such as the Internet. Once the server has fulfilled the client's request, the connection is terminated. Web browser is a client program that has requested a service from a server; in fact, the service and resource the server provided is the delivery of this Web page.

Client-Server networks are basically combination of client and server software which are basically used to respond clients' request. Earlier, the starting of client-server network was with the inception of computer networks which later took the vast shape of Internet and many more technologies. In the beginning, only four cities were connected and the network named as ARPANET (Advance Research Project Agency Network). Things were gradually added giving birth to LAN (local area network), MAN (metropolitan area network), WAN (wide area network) etc. Today, it has changed and has become the network of networks and abbreviated as Internet. Internet has not only given a new shape and direction to new research but has opened doors for several technologies. In such a way, It has changed the simple life of individuals to modern. The present available Client-Server network and its communication is actually a collection of one or more networks, rather than a single network. What characterizes it is the use of the TCP/IP protocol stack throughout.

Unit-2 covers the basic concepts and the need of distributed computing. It also explores its role and change in the today's changing environment. Distributed computing is an important area of computer science and because of distributive nature of database it has become the necessity of world. Most of the Internet applications are having a large database which cannot be handled by a single or simple server computer. Hence, it is required to have the database on multiple satellite servers which needs a similar powerful technology in parallel.

Before studying the other concepts of client-server network and communication one needs to understand its design concepts. Therefore, *Unit 3* describes the principles of designing a Client-Server network. Division of labour and transition from one domain to other domain has also been explained. New Business Requirements, recent government initiatives to expedite the purchase ordering process, improve inventory control and deliver better services to the public have created demands for

applications that would link up the government agencies to their vendors, partners and customers. These types of business systems have to be scalable to accommodate a large and growing number of users (in the range of hundreds or thousands). In addition, not only is multi-platform support essential, these applications also have to be adaptable to emerging client operating systems (e.g. Taligent).

UNIT-1 INTRODUCTION TO CLIENT-SERVER COMPUTING

Structure

1.0 INTRODUCTION

A client-server computing aims at removing the barriers of traditional mainframe centralized computing. As in past, the entire control of any organization was at some central server and high degree of control was limited for system managers. In traditional mainframe systems, end users were not involved. Many barriers of information sharing and access used to exist which caused problems. On the other hand, today, sharing of the information and access has become the main concern for end user computing and developer as well. System managers are not responsible for the entire control. Or Client-server computing has replaced the big mainframe computing. Client-server is any formal system architecture describing technologies that cooperate together on a computer network where users operate PCs (clients) that connect to central computers (servers) over a computer network. Both computers cooperate to split the work of performing various tasks.

One can also say that the set of management strategies for creating systems that improve organizational effectiveness depends upon this client/server request and response. These are strategies for distributing computing resources within an organization to support interpersonal communication, organizational coordination, and business collaboration. Client-server model is an arrangement of the organizing hardware, software, telecommunications, and data resources to put more computing power on the desktop and create a company-wide network linking smaller networks. Moreover, data and processing power are distributed throughout the organization rather than being centrally located. Furthermore it also emphasizes the user interaction with data and hence splits the processing between client and server.

1.1 OBJECTIVES

At the end of this unit one would come to know about:

- The definition of client and server.
- Difference between client and server
- Mainframe computers
- Client server systems
- Model of the client-server system
- Architecture of the client-server system
- Advantages of client-server system

1.2 INTRODUCTION TO CLIENT-SERVER COMPUTING

A client-server network is designed for end-users called clients, used to access resources such as files, songs, video collections, or some other service from a central computer called a server. A server's sole purpose is to do what its name implies - serve its clients. One may have been using this configuration and not even have known it. Have you ever played Xbox Live or used the PlayStation Network? Your Xbox One is the client, and when it logs into the network, it contacts the Xbox Live servers to retrieve gaming resources like updates, video, and game demos.

Moreover; a client-server system is "a networked computing model that distributes processes between clients and servers, which supply the requested service." A client-server network connects many computers, called the clients, to a main computer, called a server. A client can be defined as a networked information solicitor, usually a desktop computer or workstation that can query database and/or other information from a server. The Client handles the presentations logic, processing logic, and much of the storage logic. The client provides the graphical interface,

while the server provides access to shared resources, typically a database. Objects break up the client and server sides of an application into smart components that can work across networks.

A server can be defined as a device that manages applications programs and is shared by each of the client computers that are attached to the local area network (LAN). The server is usually a high-powered workstation, a minicomputer, or a mainframe, that stores information for use by networked clients. A file server is the computer that manages file operations. It is shared by each of the client computers attached to the LAN. This connection allows the client computers to share the server computer's resources, such as printers, files and programs. The server runs software that coordinates the information flow among the other computers, called clients. The file server is like an additional hard drive for each of the computers attached. If most of the processing occurs on the client rather than on a server then client is called a fat client.

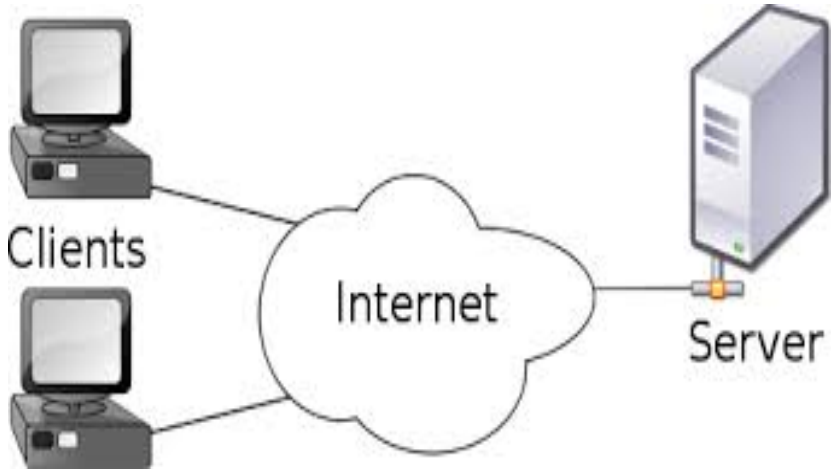


Figure 1.1: Basic Client-Server Diagram

The major difference between the server and the client computers is that the server is ordinarily faster and has more storage space. The server generally performs most of the processing tasks. Some servers are dedicated to performing a specific task such as printing or managing files. A "thin" server is intended for the home user and provides access to the Internet. A client-server network typically provides an efficient means to connect ten or more computers together. Because of the size of a client-server network, most client-server networks have a network administrator who manages this system.

In a file server environment, each client computer is authorized to use the database management system (DBMS) when a database application program runs on that computer. The primary characteristic of file server architecture is that all the data manipulation is performed at the client computers not at the file server. The file server acts solely as a

shared data storage device. Software at the file server queues access requests, but it is up to the application program at each client computer to handle all data management functions.

One of the most used buzzwords of the 1990's is client-server. Nearly all hardware and software vendors have something to say on the subject. New developments in distributed computing and object-orientation together have brought about the creation of a new class of database systems. These systems use a client-server computing model to provide quick response times for users and also support for complex, shared data in a distributed environment. Current relational DBMS products are based on a query-shipping approach in which most query processing is performed within the servers. The clients are mainly used to administer the user interface. Object-oriented database systems (OODBMS) on the other hand, support data-shipping, which allows data request processing to be performed at the clients.

1.2.1 CHARACTERISTICS OF CLIENT

- It initiates the communication
- It sends the request message.
- It waits for server's response.
- It is the first active (or master);
- Sends requests to the server;
- It expects and receives responses from the server.
- Usually connects to a small number of servers at one time
- Typically interacts directly with end-users using a graphical user interface

1.2.2 CHARACTERISTICS OF SERVER

- It acknowledges the communication.
- It sends the response message.
- It processes the request.
- It is initially passive (or slave, waiting for a query);
- It is listening, ready to respond to requests sent by clients;
- When a request comes, he treats it and sends a response.
- Usually accepts connections from a large number of clients
- Typically, does not interact directly with end-users

1.3 EVOLUTION OF CLIENT-SERVER COMPUTING

1970s and 1980s was the era of centralized computing, with IBM mainframe occupied over 70% of the world's computer business. Business transactions, activities and database retrieval, queries and maintenance are all performed by the omnipresent IBM mainframe. We are now in the transition phase towards Client-Server Computing, a totally new concept and technology to re-engineer the entire business world. Someone has called it the wave of the future - the computing paradigm of the 1990s.

One may wonder that how client-server computing is different from traditional mainframe computing and what the benefits from employing it in business are. The main emphasis of client-server architecture is to allow large application to be split into smaller tasks and to perform the tasks among host (server machine) and desktops (client machines) in the network. Client machine usually manages the front-end processes such as GUIs (Graphical User Interfaces), dispatch requests to server programs, validate data entered by the user and also manages the local resources that the user interacts with such as the monitor, keyboard, workstation, CPU and other peripherals.

The evolution of client-server computing has been driven by business needs, as well as the increasing costs for host (mainframe and midrange) machines and maintenance, the decreasing costs and increasing power of micro-computers and the increased reliability of LANs.

In the past twenty-five years, there are dramatic improvements in the hardware and software technologies for micro-computers. Micro-computers become affordable for small businesses and organizations. And at the same time their performances are becoming more and more reliable. On the other hand, the drop in price for mainframe is growing at a slower rate than the drop in its price. Little developments have achieved with mainframes.

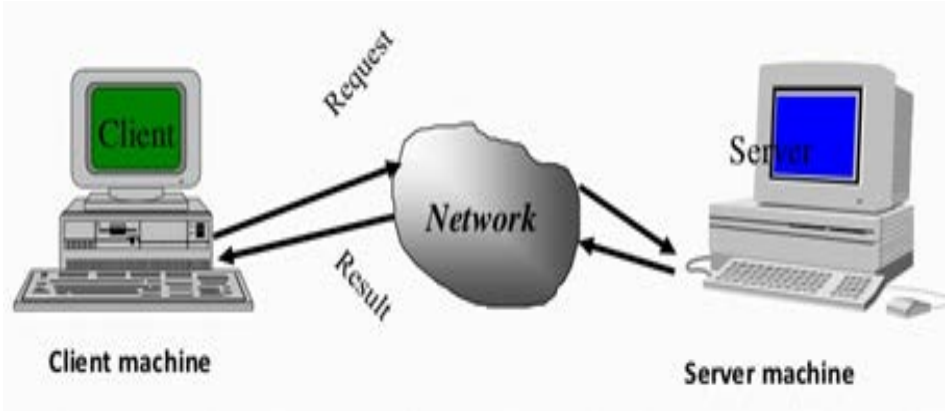


Figure 1.2 : Elements of Client-Server Computing

1.4 CLIENT-SERVER SYSTEMS

Computer system architecture has evolved along with the capabilities of the hardware used to run applications. The simplest (and earliest) of all was the "mainframe architecture" in which all operations and functionality are contained within the central (or "host") computer. Users interacted with the host through 'dumb' terminals which transmitted instructions, by capturing keystrokes, to the host and displayed the results of those instructions for the user. Such applications were typically character based and, despite the relatively large computing power of the mainframe hosts were often relatively slow and cumbersome to use because of the need to transmit every keystroke back to the host.

The introduction and widespread acceptance of the pc, with its own native computing power and graphical user interface made it possible for applications to become more sophisticated and the expansion of networked systems led to the second major type of system architecture, "file sharing". In this architecture the pc (or "workstation") downloads files from a dedicated "file server" and then runs the application (including data) locally. This works well when the shared usage is low, update contention is low, and the volume of data to be transferred is low. However, it rapidly became clear that file sharing choked as networks grew larger, and the applications running on them grew more complex and required ever larger amounts of data to be transmitted back and forth.

The problems associated with handling large, data-centric applications, over file sharing networks led directly to the development of the client-server architecture in the early 1980s. In this approach the file server is replaced by a database server (the "server") which, instead of merely transmitting and saving files to its connected workstations (the" clients") receives and actually executes requests for data, returning only the result sets to the client. By providing a query response rather than a total file transfer; this architecture significantly decreases network traffic. This allowed for the development of applications in which multiple users could update data through GUI front ends connected to a single shared database.

Typically either structured query language (sql) or remote procedure calls (rpcs) are used to communicate between the client and server. There are several variants of the basic client/server architecture as described ahead.

Check Your Progress

- Define the term client software and server software.
- Write a short note on client-server evolution.

1.5 TWO-TIER ARCHITECTURE

In two-tier architecture, the workload is divided between the server (which hosts the database) and the client (which hosts the user interface). Actually these are usually located on separate physical machines but there is no absolute requirement for this to be the case. Providing that the tiers are logically separated they can be hosted (e.g. for development and testing) on the same computer (figure 1.3).

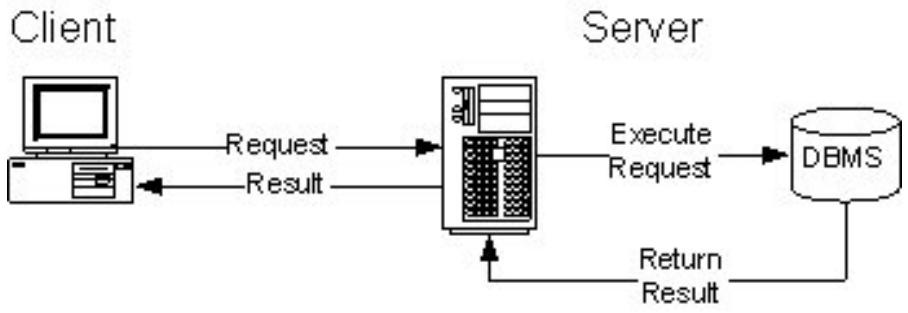


Figure 1.3 : Basic two-tier architecture

The distribution of application logic and processing in this model was problematic. If the client is 'smart' and hosts the main application processing then there are issues associated with distributing, installing and maintaining the application because each client needs its own local copy of the software. If the client is 'dumb' the application logic and processing must be implemented in the database and then becomes totally dependent on the specific DBMS being used. In either scenario, each client must also have a login to the database and the necessary rights to carry out whatever functions are required by the application. However, the two-tier client-server architecture proved to be a good solution when the user population work is relatively small (up to about 100 concurrent users) but it rapidly proved to have a number of limitations. These limitations are explained below.

1.5.1 PERFORMANCE

As the user population grows, performance begins to deteriorate. This is the direct result of each user having their own connection to the server which means that the server has to keep all these connections live (using "keep-alive" messages) even when no work is being done.

1.5.2 SECURITY

Each user must have its own individual access to the database, and be granted whatever rights may be required in order to run the application. Apart from the security issues that this raises, maintaining users rapidly becomes a major task in its own right. This is especially problematic when

new features/functionality has to be added to the application and users' rights need to be updated.

1.5.3 CAPABILITY

No matter what type of client is used, much of the data processing has to be located in the database which means that it is totally dependent upon the capabilities, and implementation, provided by the database manufacturer. This can seriously limit application functionality because different databases support different functionality, use different programming languages and even implement such basic tools as triggers differently.

1.5.4 PORTABILITY

Since the two-tier architecture is so dependent upon the specific database implementation, porting an existing application to a different DBMS becomes a major issue. This is especially apparent in the case of vertical market applications where the choice of DBMS is not determined by the vendor having said that, this architecture found a new lease of life in the internet age. It can work well in a disconnected environment where the browser is essentially dumb. however, in many ways this implementation harks back to the original mainframe architecture and indeed, a browser based, two-tier application, can (and usually does) suffer from many of the same issues.

1.5.5 RELIABILITY

Reliability is an important term of software engineering. As per the software engineering, the software must be reliable enough as per the requirements of the client and server. Client-server architecture, as the basic medium of communication on network is the most reliable technique. The data which is to be send through this mechanism is secured and reliable at any point of time. Normally, all the present technologies in minicomputers and mainframe computers provide services to support reliability. Moreover, reliability needs availability factors to be resolved first. Applications must also be protected from being modified one another. Memory should be shared by only authorized tasks. Furthermore, as far as security is concerned, only authorised users must be allowed to access various available resources. Specifically, the software must automatically handle multiple user contention, provide full recovery after failure of in-flight updates, and provide utility functions to recover a damaged magnetic disk.

1.5.6 REMOTE ACCESS TO DATABASE

Basically remote access is the ability to access the information from a computer residing at some distance. For example, in corporations,

people at their branch offices, telecommuters, and people who are travelling may often need access to the corporation's network. Home users get access to the Internet through remote access to an Internet service provider. Remote access is also possible using a dedicated line between a computer or a remote local area network and the "central" or main corporate local area network. A remote access server is the computer and associated software that is set up to handle users seeking access to network remotely. Data can be accessed from anywhere at any point of time because it is available remotely for all users. It describes the connection of a database client to a database server. It includes features for the following:

- Communicating database operations and parameters from the client to the server.
- Transporting result data from the server to the client.
- Database transaction management.
- Exchange of information.

1.6 THREE-TIER ARCHITECTURE

In an effort to overcome the limitations of the two-tier architecture outlined above, an additional tier was introduced – creating what is now the standard three-tier client-server model. The purpose of the additional tier (usually referred to as the "middle" or "rules" tier) is to handle application execution and database management. As with the two-tier model, the tiers can either be implemented on different physical machines (figure 2), or multiple tiers may be co-hosted on a single machine.

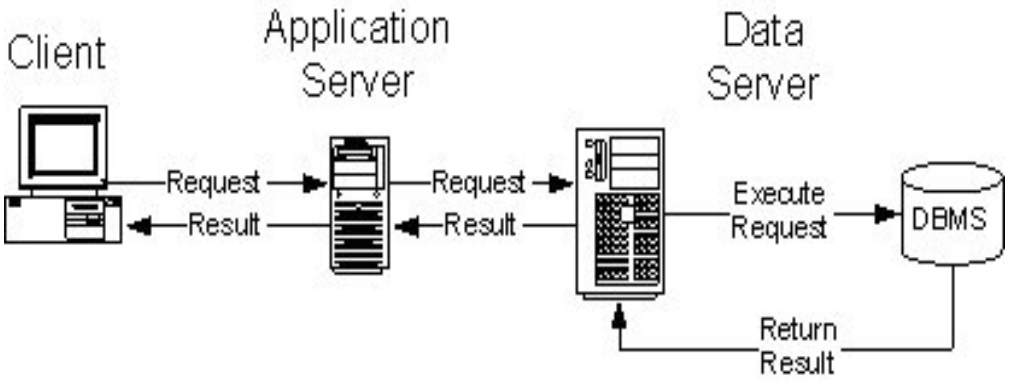


Figure 1.4 : Three-tier architecture

After introducing the middle tier, the limitations of the two-tier architecture are largely removed and the result is a much more flexible, and scalable, system. Since clients now connect only to the application server, not directly to the data server, the load of maintaining connections is removed, as is the requirement to implement application logic within the

database. The database can now be backed to its proper role of managing the storage and retrieval of data, while application logic and processing can be handled in whatever application is most appropriate for the task. The development of operating systems to include such features as connection pooling, queuing and distributed transaction processing has enhanced (and simplified) the development of the middle tier.

It is to be noted here that, in this model, the application server does not drive the user interface, nor does it actually handle data requests directly. Instead it allows multiple clients to share business logic, computations, and access to the data retrieval engine that it exposes. This has the major advantage that the client needs less software and no longer need a direct connection to the database, so there is less security to worry about.

Consequently applications are more scalable, and support and installation costs are significantly less for a single server than for maintaining applications directly on a desktop client or even a two-tier design. There are many variants of the basic three-tier model designed to handle different application requirements. These include distributed transaction processing (where multiple DBMS are updated in a single transaction), message based applications (where applications do not communicate in real-time) and cross-platform interoperability (object request broker or "orb" applications).

1.7 MULTIIN-TIER ARCHITECTURE

With the rapid growth of internet based applications a common enhancement of the basic three-tier client server model has been the addition of extra tiers, such architecture is referred to as 'n-tier' and typically comprises four tiers (figure 3) where the web server is responsible for handling the connection between client browsers and the application server. The main benefit of this is simply that multiple web servers can connect to a single application server, thereby handling more concurrent users.

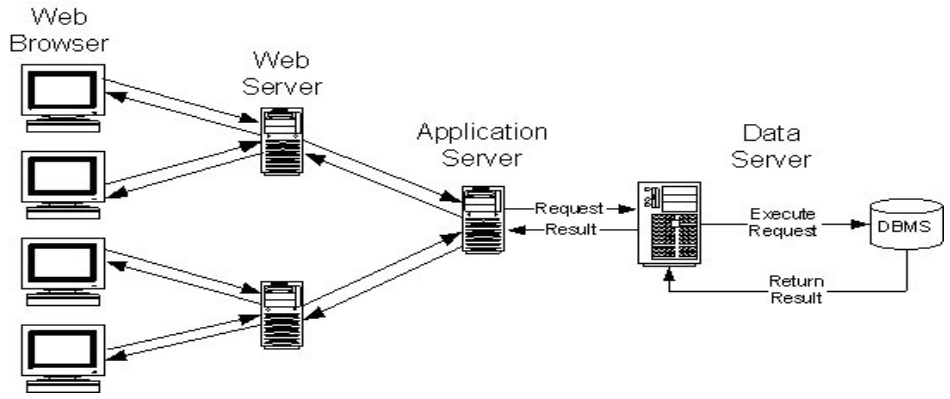


Figure 1.4 : N-tier architecture

Check Your Progress

- Define the term client software and server software.
- Write a short note on client-server evolution.

1.8 TIERS VS LAYERS

These terms are often (regrettably) used interchangeably. Rather they really are distinct and have definite meanings. The basic difference is that tiers are physical, while layers are logical. In other words a tier can theoretically be deployed independently on a dedicated computer, while a layer is a logical separation within a tier (figure 1.5). The typical three-tier model described above normally contains at least seven layers, split across the three tiers. The key concept to remember about a layered architecture is that requests and responses each flow in one direction only and that layers may never be "skipped". Therefore, in the model shown in figure 1.5, the only layer that can address layer "e" (the data access layer) is layer "d" (the rules layer). Similarly layer "c" (the application validation layer) can only respond to requests from layer "b" (the error handling layer).

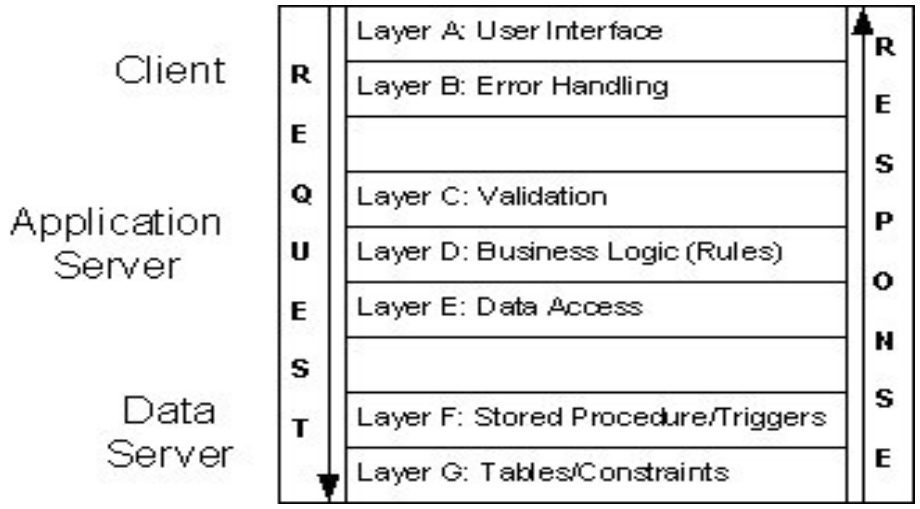


Figure 1.5 : Tiers are divided into logical layers

1.9 CLIENT-SERVER USES / BENEFITS

As client-server is fast-becoming the enabling factor for business process reengineered organizations, because of its flexibility and speedy application development times - it takes around six months to develop a

client-server application compared to around 2 years for a mainframe version. Therefore, time taken to develop client-server applications has reduced remarkably. By adopting the client-server technology, the organizations have changed from steep hierarchies to flattened hierarchies.

Also, network management is replacing vertical management. As a whole, the development and implementation of client-server technology is more complex, more difficult and more expensive than traditional single process applications. However, they are still badly needed because the business demands the increased benefits.

Client-server systems have become the computing architecture for many business organizations. Technically, a client-server system places application processing close to the user and thus increases performance. Due to the recent improvements in the price and performance characteristics of workstations and the networking capabilities, the client-server system architecture has become very popular for database systems.

A client-server DBMS provides the management of a database within a client-server system. The database is stored on disks that can be accessed only by the servers. Copies of database items are cached in the global memory, which consist of all the memories of the computers connected to the system. This reduces the disk access. This efficient global memory design reduces the handling and creating fewer disks input/output during the use of the database.

Some of the advantages are described below:

1) **Centralization:** Unlike P2P, where there is no central administration, here in this architecture there is a centralized control. Servers help in administering the whole set-up. Access rights and resource allocation is done by Servers.

2) **Proper Management :** All the files are stored at the same place. In this way, management of files becomes easy. Also it becomes easier to find files.

3) **Back-up and Recovery possible:** As all the data is stored on server it is easy to make a back-up of it. Also, in case of some break-down if data is lost, it can be recovered easily and efficiently. While in peer computing we have to take back-up at every workstation.

4) **Upgradation and Scalability in Client-server set-up:** Changes can be made easily by just upgrading the server. Also new resources and systems can be added by making necessary changes in server.

5) **Accessibility:** From various platforms in the network, server can be accessed remotely.

6) As new information is uploaded in database; each workstation need not have its own storage capacities increased (as may be the

case in peer-to-peer systems). All the changes are made only in central computer on which server database exists.

7) **Security:** Rules defining security and access rights can be defined at the time of set-up of server.

8) Servers can play different roles for different clients.

Figure 1.6 : Benefits of Client-Server

1.10 DOWNSIZING WITH CLIENT-SERVER COMPUTING

One of the best ways to downsize is by using the new generation of SQL based client-server computing technologies from vendors such as Oracle, Sybase, Gupta and Novell. In the client-server model, the application is split between functions that execute on the client, a PC or workstation, and functions that run on the server, a multiuser data repository. Most application logic runs at the client desktop machine. When the application requires data, it generates the necessary SQL command and then passes high-level code to the communications facility. This facility then directs the SQL commands to the server, where the database request is executed (figure-1.7). The idea of managing data on a separate machine fits well with the management approach of treating data as a corporate resource. In addition to executing the SQL statement, the

server handles security and provides for concurrent access to the data by many queries.

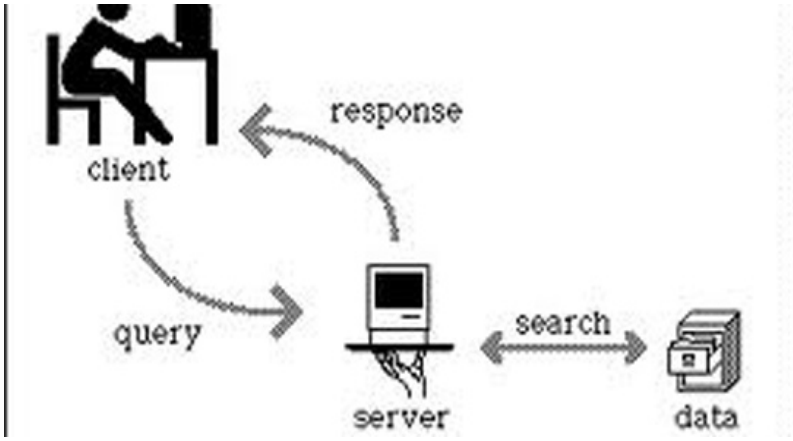


Figure 1.7 : Request/Response in Client-Server Computing

A benefit of using SQL client-server computing is that the hardware and software products supporting this approach are new and take advantage of the latest developments, such as application languages in a windowing environment. Another benefit is network efficiency. In traditional file-serving PC LAN approaches, the entire data file must be transmitted across a network to the client machine. With SQL as the basis for database management, this problem is resolved, since only the necessary query response data (a table) is transmitted to the client machine. SQL on the server also enables the implementation of advanced facilities, such as triggers and automatic procedures in the database.

Check Your Progress

- Write the main benefits of client-server computing.
- Draw and explain the diagram of request-response in client-server environment.

1.11 MAINFRAME COMPUTING

Mainframe computing is used for critical applications, bulk data processing, enterprise resource planning and transaction processing. Most of this is Mainframe/Mid-Range and includes, but is not limited to, operating system, application design & development, operations, support, storage and security. Moreover, It is a very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously. In the hierarchy that starts with a simple microprocessor

(in watches, for example) at the bottom and moves to supercomputers at the top, mainframes are just below supercomputers.

In some ways, mainframes are more powerful than supercomputers because they support more simultaneous programs. But supercomputers can execute a single program faster than a mainframe. The distinction between small mainframes and minicomputers is vague, depending really on how the manufacturer wants to market its machines.

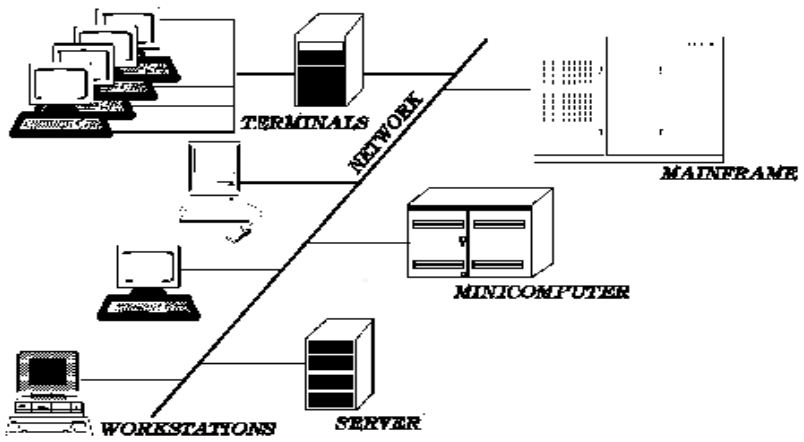


Figure 1.8 : Mainframe Computer in a Network

1.11.1 MAINFRAME COMPUTERS

One of the first types of computers to be used commercially was the mainframe. This system operates by sharing a processor between a large number of 'dumb terminals'. These terminals are composed of a monitor and a keyboard, but they do not have their own processor, hence the term 'dumb'. Large businesses, such as banks and insurance companies, use mainframes to allow their remote branches access to the processor, which is held in a central location. The processor has to be very powerful as huge amounts of data are dealt with. Imagine the number of credit card transactions throughout the country that are processed in a single day. Mainframes support multi-access and multi-programming capabilities for the users. Mainframe computers have the following characteristics :

Processing power : A mainframe computer will have several processors that work together, making the machine extremely powerful.

Memory size : There is usually a vast amount of memory. Some modern mainframes can support more than 32 GB of main memory.

Backing store devices : These are typically greater than 100 GB hard disk. Tape drives are also used for back-up or batch processing.

Input /Output devices : Keyboard, Line printers, page printers and monitors.

1.12 CLIENT-SERVER TECH AND HETEROGENEOUS COMPUTING

1.12.1 CLIENT-SERVER MODEL

Client-Server technology is a means for separating the functions of an application into two or more distinct parts. Client/ server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. The client presents and manipulates data on the desktop computer. The server acts like a mainframe to store and retrieve protected data. It is network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

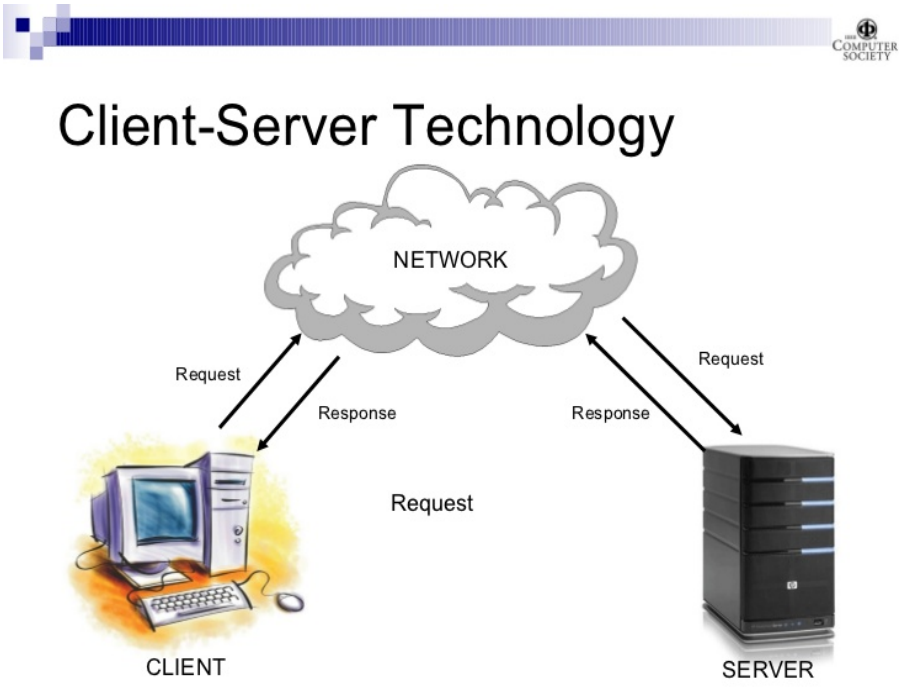


Figure 1.9 : Client-Server Technology

A client/ server model has following three distinct components, each focusing on a specific job:

- Database server
- Client application
- Network.

1.12.1.1 DATABASE SERVER

A server (or "back end") manages the resources such as database, efficiently and optimally among various clients that simultaneously request the server for the same resources. Database server mainly concentrates on the following tasks:

1. Managing a single database of information among many concurrent users.

2. Controlling database access and other security requirements.

3. Protecting database of information with backup and recovery features.

4. Centrally enforcing global data integrity rules across all client applications.

1.12.1.2 CLIENT APPLICATION

A client application (the "front end") is the part of the system that users apply to interact with data. The client application in a client/ server model focuses on the following job:

- Presenting an interface between the user and the resource to complete the job.

- Managing presentation logic.

- Performing application logic and validating data entry.

- Managing the request traffic of receiving and sending information from database server.

1.12.1.3 NETWORK

The third component of a client-server system is network. The communication software is the vehicle that transmits data between the clients and the server. Both the client and the server run communication software that allows them to talk across the network. Client-server is an important idea in a network; however, it can be used by programs within a single computer. In a network, the client-server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client-server model are very common. For example, to check your bank account from your computer, a client program in your computer forwards your request to a server program at the bank. That program may in turn forward the request to its own client program that sends a request to a database server at another bank computer to retrieve your account balance.

1.12.2 HETEROGENEOUS COMPUTING

It refers to systems that use more than one kind of processor or cores. These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar coprocessors, usually incorporating specialized processing capabilities to handle particular tasks. Usually heterogeneity in the context of computing referred to different instruction set architectures (ISA), where the main processor has one and the rest have another, usually a very different architecture (maybe more than one), not just a different micro architecture (floating point number processing is a special case of this not usually referred to as heterogeneous). E.g. ARM big. LITTLE is an exception where the ISAs of cores are the same and heterogeneity refers to the speed of different micro architectures of the same ISA, then making it more like a symmetric multiprocessor system (SMP).

In the past heterogeneous computing meant different ISAs had to be handled differently, while a modern example, Heterogeneous (HSA) systems, eliminate the difference (for the user); use multiple processor types (typically CPUs and GPUs), usually on the same integrated circuit, to provide the best of both worlds: general GPU processing (apart from its well-known 3D graphics rendering capabilities, it can also perform mathematically intensive computations on very large data sets), while CPUs can run the operating system and perform traditional serial tasks.

The level of heterogeneity in modern computing systems is gradually increasing as further scaling of fabrication technologies allows for formerly discrete components to become integrated parts of a system-on-chip (SoC). For example, many new processors now include built-in logic for interfacing with other devices (SATA, PCI, Ethernet, USB, RFID, Radios, UARTs, and memory controllers), as well as programmable functional units and hardware accelerators (GPUs, cryptography co-processors, programmable network processors, A/V encoders/decoders, etc.).

Recent findings show that a heterogeneous-ISA chip multiprocessor that exploits diversity offered by multiple ISAs can outperform the best same-ISA heterogeneous architecture by as much as 21% with 23% energy savings and a reduction of 32% in Energy Delay Product. Heterogeneous computing systems present new challenges not found in typical homogeneous systems. The presence of multiple processing elements raises all of the issues involved with homogeneous parallel processing systems, while the level of heterogeneity in the system can introduce non-uniformity in system development, programming practices, and overall system capability.

Areas of heterogeneity may include:

ISA or instruction set architecture : Compute elements may have different instruction set architectures, leading to binary incompatibility.

ABI or application binary interface: Compute elements may interpret memory in different ways. This may include both endianness, calling convention, and memory layout, and depends on both the architecture and compiler being used.

API or application programming interface: Library and OS services may not be uniformly available to all compute elements.

Low-Level Implementation of Language Features: Language features such as functions and threads are often implemented using function pointers, a mechanism which requires additional translation or abstraction when used in heterogeneous environments.

Memory Interface and Hierarchy: Compute elements may have different cache structures, cache coherency protocols, and memory access may be uniform or non-uniform memory access (NUMA). Differences can also be found in the ability to read arbitrary data lengths as some processors/units can only perform byte-, word-, or burst accesses.

Interconnect: Compute elements may have differing types of interconnect aside from basic memory/bus interfaces. This may include dedicated network interfaces, direct memory access (DMA) devices, mailboxes, FIFOs, and scratchpad memories, etc. Furthermore, certain portions of a heterogeneous system may be cache-coherent, whereas others may require explicit software-involvement for maintaining consistency and coherency.

Performance: A heterogeneous system may have CPUs that are identical in terms of architecture, but have underlying micro-architectural differences that lead to various levels of performance and power consumption.

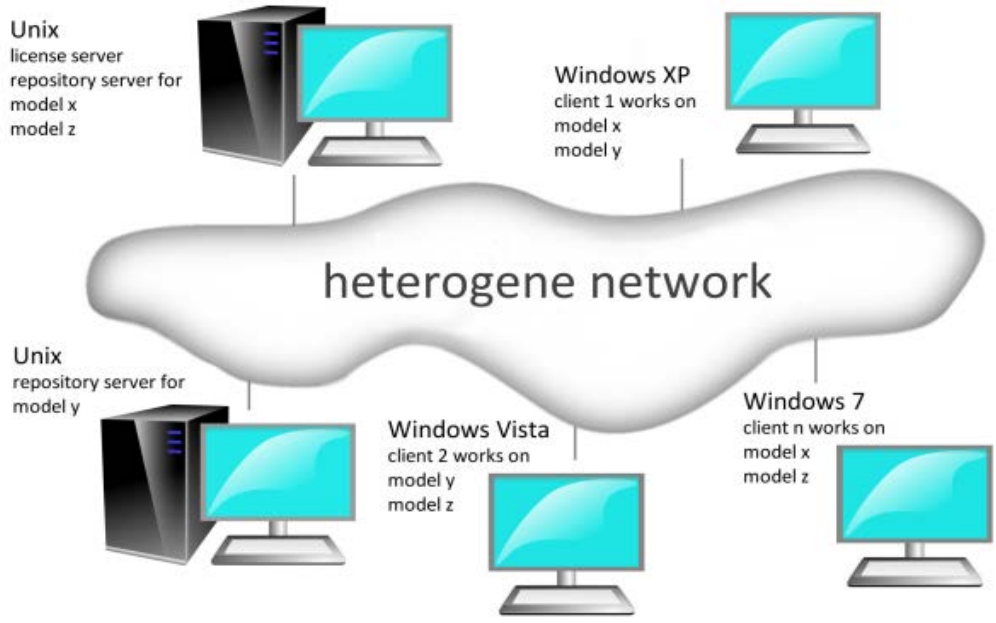


Figure 1.10 : Heterogeneous Network

Check Your Progress

- What is mainframe computing?
- Explain the Heterogeneous network.
- What are the different components used in a client server model?

1.13 SUMMARY

Computer networks can be used for numerous services, both for companies and for individuals. For companies, networks of personal computers using shared server often provide access to corporate information. Typically, they follow the client-server model, with client workstations on employee desktops accessing powerful servers in the machine room. For individuals, networks offer access to a variety of information and entertainment resources. Individuals often access the Internet by calling up an ISP using a modem, although increasingly many people have a fixed connection at home. An up-and-coming area is wireless networking with new applications such as mobile e-mail access and m-commerce.

Mainframe system operates by sharing a processor between a large number of 'dumb terminals'. These terminals are composed of a monitor and a keyboard, but they do not have their own processor, hence the term 'dumb'. Large businesses, such as banks and insurance companies, use mainframes to allow their remote branches access to the processor, which is held in a central location.

Heterogeneous network refers to systems that use more than one kind of processor or cores. These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar coprocessors, usually incorporating specialized processing capabilities to handle particular tasks. Usually heterogeneity in the context of computing referred to different instruction set architectures (ISA), where the main processor has one and the rest have another, usually a very different architecture.

1.14 TERMINAL QUESTIONS

1. What do you understand by Client and server software?
2. Give the evolution of Client-Server Computing.
3. What are the main elements of Client-Server Computing? Draw a diagram also.
4. Why we need Client-Server Computing? Explain its benefits.

5. Define Mainframe computing.
6. Write a short note on Heterogeneous Computing.
7. Compare tier and layers.
8. “Today Client-Server Computing has become the need of almost all computer networks”, justify this statement with respect to arguments of present scenario.
9. Is there any drawback of Client-Server Computing? If yes then discuss.

UNIT-2 DISTRIBUTED COMPUTING

Structure

2.0 INTRODUCTION

Distributed computing is a very important area of Computer Science. The term distributed means the things are scattered but they are directly or indirectly related with each other. Nothing, in computer Science and other technologies is untouched with this word ie distributed computing. Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing.

The word distributed in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers are physically distributed within some geographical area. For example, Bank's ATMs are distributed at several places and whenever money is withdrawn from an ATM the whole Database of the Bank is updated. Hence the load is distributed from one main computer to many computers. In this unit the main emphasis is to know the Distributed computing and similar technologies. This also aims to develop a distributed system using various technologies. Client-server systems and computing is also discussed.

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed

computing, and distributed computing may be seen as a loosely coupled form of parallel computing.

2.1 OBJECTIVE

At the end of this unit you would come to know

- The meaning of distributed computing
- Difference between Distributed systems and Distributed computing
- Client software.
- Server software
- Client-server computing
- File server
- Characteristics of Distributed computing
- Advantages of distributed computing
- Microprocessor integration
- Scaling
- Main issues of Distributed Computing

2.2 INTRODUCTION TO DISTRIBUTED COMPUTING

Distributed Computing is a field of computer science that studies distributed systems. A *distributed system* is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are:

- Concurrency of components
- Lack of a global clock
- Independent failure of components.

Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications. A computer program that runs in a distributed system is called a *distributed program*, and *distributed programming* is the process of writing such programs. There are many alternatives for the message passing mechanism, including pure HTTP, RPC-like connectors and message queues. A goal and challenge pursued each other by message passing.

There is no single definition of a distributed system, however some computer scientists and practitioners in distributed systems is location transparency; this goal has fallen out of favour in industry, as distributed systems are different from conventional non-distributed systems, and the differences, such as network partitions, partial system failures, and partial upgrades, cannot simply be "papered over" by attempts at "transparency".

Distributed computing also refers to the use of distributed systems to solve computational problems. In *distributed computing*, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing. The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers are physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with There are several autonomous computational entities, each of which has its own local memory. The entities communicate with each other by message passing. In this article, the computational entities are called *computers* or *nodes*.

A distributed system may have a common goal, such as solving a large computational problem. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users. Other typical properties of distributed systems include the following: The system has to tolerate failures in individual computers.

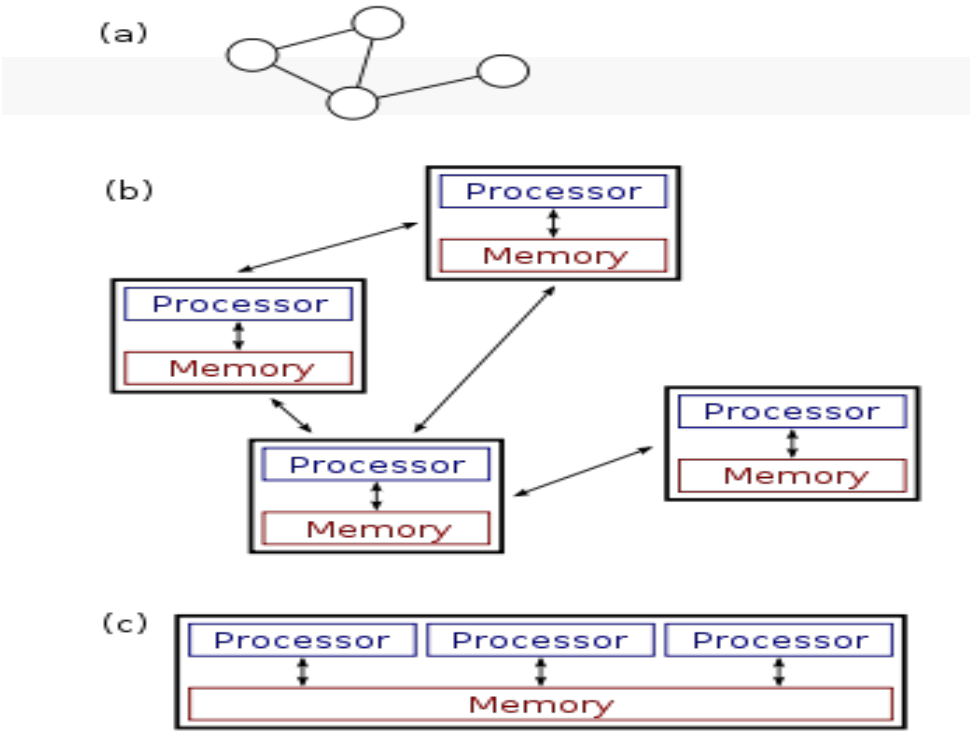


Figure 2.1: (a) Simple System (b) Distributed System (c) Parallel System

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing.

Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory called distributed memory. Information is exchanged by passing messages between the processors.
- Figure (a) is a schematic view of a typical distributed system; as usual, the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link.
- Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links.
- Figure (c) shows a parallel system in which each processor has a direct access to a shared memory.

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems*. Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithm.

2.2.1 PARALLEL ALGORITHMS IN SHARED-MEMORY MODEL

- All processors have access to a shared memory. The algorithm designer chooses the program executed by each processor.
- One theoretical model is the parallel random access machines (PRAM) that are used. However, the classical PRAM model assumes synchronous access to the shared memory.
- Shared-memory programs can be extended to distributed systems if the underlying operating system encapsulates the

communication between nodes and virtually unifies the memory across all individual systems.

- A model that is closer to the behavior of real-world multiprocessor machines and takes into account the use of machine instructions, such as Compare-and-swap (CAS), is that of *asynchronous shared memory*. There is a wide body of work on this model, a summary of which can be found in the literature.

2.2.2 ALGORITHMS IN MESSAGE-PASSING MODEL

- The algorithm designer chooses the structure of the network, as well as the program executed by each computer.
- Models such as Boolean circuits and sorting networks are used.A Boolean circuit can be seen as a computer network: each gate is a computer that runs an extremely simple computer program. Similarly, a sorting network can be seen as a computer network: each comparator is a computer.

2.2.3 DISTRIBUTED ALGORITHMS IN MESSAGE-PASSING MODEL

- The algorithm designer only chooses the computer program. All computers run the same program. The system must work correctly regardless of the structure of the network.
- A commonly used model is a graph with one finite-state machine per node.

In the case of distributed algorithms, computational problems are typically related to graphs. Often the graph that describes the structure of the computer network is the problem instance.

Example: Consider the computational problem of finding a coloring of a given graph *G*. Different fields might take the following approaches.

*Centralized algorithms:*The graph *G* is encoded as a string, and the string is given as input to a computer. The computer program finds a coloring of the graph, encodes the coloring as a string, and outputs the result.

*Parallel algorithms:*Again, the graph *G* is encoded as a string. However, multiple computers can access the same string in parallel. Each computer might focus on one part of the graph and produce a coloring for that part. The main focus is on high-performance computation that exploits the processing power of multiple computers in parallel.

*Distributed algorithms:*The graph *G* is the structure of the computer network. There is one computer for each node of *G* and one communication link for each edge of *G*. Initially, each computer only knows about its immediate neighbors in the graph *G*; the computers must exchange messages with each other to discover more about the structure

of *G*. Each computer must produce its own color as output. The main focus is on coordinating the operation of an arbitrary distributed system. While the field of parallel algorithms has a different focus than the field of distributed algorithms, there is a lot of interaction between the two fields.

E.g., the Cole–Vishkin algorithm for graph coloring was originally presented as a parallel algorithm, but the same technique can also be used directly as a distributed algorithm. So far the focus has been on *designing* a distributed system that solves a given problem. A complementary research problem is *studying* the properties of a given distributed system.

Example: The halting problem is an analogous example from the field of centralized computation: we are given a computer program and the task is to decide whether it halts or runs forever.

The halting problem is undecidable in the general case, and naturally understanding the behaviour of a computer network is at least as hard as understanding the behaviour of one computer. However, there are many interesting special cases that are decidable. In particular, it is possible to reason about the behaviour of a network of finite-state machines. One example is telling whether a given network of interacting (asynchronous and non-deterministic) finite-state machines can reach a deadlock. This problem is PSPACE-complete,it is decidable, but it is not likely that there is an efficient (centralized, parallel or distributed) algorithm that solves the problem in the case of large networks.

2.3 FILE SERVER VS CLIENT-SERVER DATABASE

2.3.1 FILE SERVER

In computing, a file server (or fileserver) is a computer attached to a network that has the primary purpose of providing a location for shared disk access, i.e. shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are attached to the same computer network. The term server highlights the role of the machine in the client–server scheme, where the clients are the workstations using the storage. A file server is not intended to perform computational tasks, and does not run programs on behalf of its clients. It is designed primarily to enable the storage and retrieval of data while the computation is carried out by the workstations File servers are commonly found in schools and offices, where users use a LAN to connect their client computers.

2.3.1.1 TYPES OF FILE SERVERS

A file server may be dedicated or non-dedicated. A dedicated server is designed specifically for use as a file server, with workstations

attached for reading and writing files and databases. File servers may also be categorized by the method of access: Internet file servers are frequently accessed by File Transfer Protocol (FTP) or by HTTP (but are different from web servers, that often provide dynamic web content in addition to static files). Servers on a LAN are usually accessed by SMB/CIFS protocol (Windows and Unix-like) or NFSprotocol (Unix-like systems). Database servers, that provide access to a shared database via a database device driver, are not regarded as file servers as they may require Record locking.

2.3.1.2 DESIGN OF FILE SERVERS

In modern businesses the design of file servers is complicated by competing demands for storage space, access speed, recoverability, administration, security, and budget. This is further complicated by a constantly changing environment, where new hardware and technology rapidly obsolesces old equipment, and yet must seamlessly come online in a fashion compatible with the older machinery. To manage throughput, peak loads, and response time, vendors may utilize queuing theory to model how the combination of hardware and software will respond over various levels of demand. Servers may also employ dynamic load balancing scheme to distribute requests across various pieces of hardware. The primary piece of hardware equipment for servers over the last couple of decades has proven to be the hard disk drive. Although other forms of storage are viable (such as magnetic tape and solid-state drives) disk drives have continued to offer the best fit for cost, performance, and capacity.

2.3.1.3 STORAGE

Since the crucial function of a file server is storage, technology has been developed to operate multiple disk drives together as a team, forming a disk array. A disk array typically has cache, as well as advanced functions like RAID and storage virtualization. Typically disk arrays increase level of availability by using redundant components other than RAID, such as power supplies. Disk arrays may be consolidated or virtualized in a SAN.

2.3.1.4 NETWORK-ATTACHED STORAGE

Network-attached storage (NAS) is file-level computer data storage connected to a computer network providing data access to a heterogeneous group of clients. NAS devices specifically are distinguished from file servers generally in a NAS being a computer appliance – a specialized computer built from the ground up for serving files – rather than a general purpose computer being used for serving files (possibly with other functions). In discussions of NASs, the term "file server" generally stands for a contrasting term, referring to general purpose computers only. As of 2010 NAS devices are gaining popularity,

offering a convenient method for sharing files between multiple computers.

Potential benefits of network-attached storage, compared to non-dedicated file servers, include faster data access, easier administration, and simple configuration. NAS systems are networked appliances containing one or more hard drives, often arranged into logical, redundant storage containers or RAID arrays. Network Attached Storage removes the responsibility of file serving from other servers on the network. They typically provide access to files using network file sharing protocols such as NFS, SMB/CIFS (Server Message Block/Common Internet File System), or AFP.

2.3.1.5 SECURITY

File servers generally offer some form of system security to limit access to files to specific users or groups. In large organizations, this is a task usually delegated to what is known as directory services such as open LDAP, Novell's e-Directory or Microsoft's Active Directory. These servers work within the hierarchical computing environment which treat users, computers, applications and files as distinct but related entities on the network and grant access based on user or group credentials. In many cases, the directory service spans many file servers, potentially hundreds for large organizations. In the past, and in smaller organizations, authentication could take place directly at the server itself.

2.3.2 CLIENT-SERVER

The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer networkon separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

2.3.2.1 CLIENT AND SERVER ROLE

The Client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For instance, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and

electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

Whether a computer is a client, a server, or both, is determined by the nature of the application that requires the service functions. For example, a single computer can run web server and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software within the same computer. Communication between servers, such as to synchronize data, is sometimes called inter-server or server-to-server communication.

2.3.2.2 CLIENT AND SERVER COMMUNICATION

The client only has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service. Clients and servers exchange messages in a request–response messaging pattern: The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication.

To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application-layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an API. The API is an abstraction layer for such resources as databases and custom software. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

A server may receive requests from many different clients in a very short period of time. Because the computer can perform a limited number of tasks at any moment, it relies on a scheduling system to prioritize incoming requests from clients in order to accommodate them all in turn.

Example

When a bank customer accesses online banking services with a web browser, the client initiates a request to the bank's web server. The customer's login details may be stored in a database, and the web server accesses the database server as a client. An application server interprets the returned data by applying the bank's business logic, and provides the output to the web server. Finally, the web server returns the result to the client web browser for display. In each step of this sequence of client–server communication, a computer processes a request and returns data. This is the request-response messaging pattern. When all the requests are met, the sequence is complete and the web browser presents the data to the

customer. This example illustrates a design pattern applicable to the client–server model.

Check Your Progress

- Compare file server and client server database.
- What is distributed computing?

2.4 COMPUTING PLATFORM

A computing platform is whatever a pre-existing piece of computer software or code object is designed to run within, obeying its constraints, and making use of its facilities. The term computing platform can refer to different abstraction levels, including a certain hardware architecture, an operating system (OS), and runtime libraries. In total it can be said to be the stage on which computer programs can run. Binary executables have to be compiled for a specific hardware platform, since different central processor units have different machine codes.

In addition, operating systems and runtime libraries allow re-use of code and provide abstraction layers which allow the same high-level source code to run on differently configured hardware. For example, there are many kinds of data storage device, and any individual computer can have a different configuration of storage devices; but the application is able to call a generic save or write function provided by the OS and runtime libraries, which then handle the details themselves.

2.4.1 COMPONENTS

Computing platforms may also include:

- Hardware alone, in the case of small embedded systems. Embedded systems can access hardware directly, without an OS; this is referred to as running on "bare metal".
- A browser in the case of web-based software. The browser itself runs on a hardware + OS platform, but this is not relevant to software running within the browser.
- An application, such as a spreadsheet or word processor, which hosts software written in an application-specific scripting language, such as an Excel macro. This can be extended to writing fully-fledged applications with the Microsoft Office suite as a platform.
- Software frameworks that provide ready-made functionality.
- Cloud computing and Platform as a Service. Extending the idea of a software framework, these allow application developers to build

software out of components that are hosted not by the developer, but by the provider, with internet communication linking them together. The social networking sites Twitter and Facebook are also considered development platforms.

- A virtual machine (VM) such as the Java virtual machine. or .NET CLR. Applications are compiled into a format similar to machine code, known as byte code, which is then executed by the VM.
- A virtualized version of a complete system, including virtualized hardware, OS, software and storage. These allow, for instance, a typical Windows program to run on what is physically a Mac.
- Some architectures have multiple layers, with each layer acting as a platform to the one above it. In general, a component only has to be adapted to the layer immediately beneath it. For instance, a Java program has to be written to use the Java virtual machine (JVM) and associated libraries as a platform, but does not have to be adapted to run for the Windows, Linux or Macintosh OS platforms. However, the JVM, the layer beneath the application, does have to be built separately for each OS.

2.4.2 HARDWARE EXAMPLE

- Wintel, that is, Intel x86 or compatible personal computer hardware with Windows operating system.
- Macintosh, custom Apple Computer hardware and Classic Mac OS
- Newton devices running the Newton OS, also from Apple
- ARM architecture used in mobile devices
- Video game consoles
- Multimedia player platform for video game console development
- RISC processor based machines running Unix variants
- Midrange computers with their custom operating systems, such as IBM OS/400
- Mainframe computers with their custom operating systems, such as IBM z/OS
- Supercomputer architectures

2.4.3 SOFTWARE EXAMPLE

- Windows 7
- iOS 9
- Android Lolipop

- Android 4.0
- Windows 10
- Windows 8.1
- mac OS 4.1
- iOS 8
- Windows XP
- Linux

2.5 MICROPROCESSOR INTEGRATION & CLIENT-SERVER COMPUTING

2.5.1 MICROPROCESSOR INTEGRATION

A microprocessor is a computer processor which incorporates the functions of a computer's CPU on a single integrated circuit (IC), or at most a few integrated circuits. The microprocessor is a multipurpose, clock driven, register based, programmable electronic device which accepts digital or binary data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system. The integration of a whole CPU onto a single chip or on a few chips greatly reduced the cost of processing power.

Integrated circuit processors are produced in large numbers by highly automated processes resulting in a low per unit cost. Single-chip processors increase reliability as there are many fewer electrical connections to fail. As microprocessor designs get faster, the cost of manufacturing a chip (with smaller components built on a semiconductor chip the same size) generally stays the same. Before microprocessors, small computers had been built using racks of circuit boards with many medium- and small-scale integrated circuits. Microprocessors combined this into one or a few large-scale ICs. Continued increases in microprocessor capacity have since rendered other forms of computers almost completely obsolete, with one or more microprocessors used in everything from the smallest embedded systems and handheld devices to the largest mainframes and supercomputers.

2.5.1.1 ORGANIZATION

A microprocessor normally has the arithmetic and logic section, register file, control logic section, and buffers to external address and data lines The internal arrangement of a microprocessor (eg Z80) varies depending on the age of the design and the intended purposes of the microprocessor. The complexity of an integrated circuit (IC) is bounded

by physical limitations of the number of transistors that can be put onto one chip, the number of package terminations that can connect the processor to other parts of the system, the number of interconnections it is possible to make on the chip, and the heat that the chip can dissipate.

A minimal hypothetical microprocessor might only include an arithmetic logic unit (ALU) and a control logic section. The ALU performs operations such as addition, subtraction, and operations such as AND or OR. Each operation of the ALU sets one or more flags in a status register, which indicate the results of the last operation (zero value, negative number, overflow, or others). The control logic retrieves instruction codes from memory and initiates the sequence of operations required for the ALU to carry out the instruction. A single operation code might affect many individual data paths, registers, and other elements of the processor. As integrated circuit technology advanced, it was feasible to manufacture more and more complex processors on a single chip.

Additional features were added to the processor architecture; more on-chip registers speed up programs, and complex instructions could be used to make more compact programs. Floating-point arithmetic, for example, was often not available the same microprocessor chip, speed up floating point calculations. Instead of processing all of a long word on one integrated circuit, multiple circuits in parallel processed subsets of each data word. While this required extra logic to handle, for example, carry and overflow within each slice, the result was a system that could handle, for example, 32-bit words using integrated circuits with a capacity for only four bits each.

2.5.1.2 SPECIAL-PURPOSE DESIGNS

A microprocessor is a general purpose system. Several specialized processing devices have followed from the technology. A digital signal processor (DSP) is specialized for signal processing. Graphics processing units (GPUs) are processors designed primarily for real time rendering of 3D images. They may be fixed function (as was more common in the 1990s), or support programmable shaders. With the continuing rise of GPU, GPUs are evolving into increasingly general purpose stream processors (running compute shaders), whilst retaining hardware assist for rasterizing, but still differ from CPUs in that they are optimized for throughput over latency, and are not suitable for running application or OS code. Other specialized units exist for video processing and machine vision. Microcontrollers integrate a microprocessor with peripheral devices in embedded systems. These tend to have different tradeoffs compared to CPUs.32-bit processors have more digital logic than narrower processors, so 32-bit (and wider) processors produce more digital noise and have higher static consumption than narrower processors.

Reducing digital noise improves ADC conversion results. So, 8- or 16-bit processors are better than 32-bit processors for system on a chip and microcontrollers that require extremely low-power electronics, or are part of a mixed-signal integrated circuit with noise-sensitive on-chip

analog electronics such as high-resolution analog to digital converters, or both. Nevertheless, trade-offs apply: running 32-bit arithmetic on an 8-bit chip could end up using more power, as the chip must execute software with multiple instructions. Modern microprocessors go into low power states when possible, and the 8-bit chip running 32-bit software is active most of the time. This creates a delicate balance between software, hardware and use patterns, plus costs. When manufactured on a similar process, 8-bit microprocessors use less power when operating and less power when sleeping than 32-bit microprocessors. However, some people say a 32-bit microprocessor may use less average power than an 8-bit microprocessor when the application requires certain operations such as floating-point math that take many more clock cycles on an 8-bit microprocessor than a 32-bit microprocessor so the 8-bit microprocessor spends more time in high-power operating mode.

2.5.1.3 EMBEDDED APPLICATIONS

Thousands of items that were traditionally not computer-related include microprocessors. These include large and small household appliances, cars (and their accessory equipment units), car keys, tools and test instruments, toys, light switches/dimmers and electrical circuit breakers, smoke alarms, battery packs, and hi-fi audio/visual components (from DVD players to phonograph turntables). Such products as cellular telephones, DVD video system and HDTV broadcast systems fundamentally require consumer devices with powerful, low-cost, microprocessors. Increasingly stringent pollution control standards effectively require automobile manufacturers to use microprocessor engine management systems, to allow optimal control of emissions over widely varying operating conditions of an automobile. Non-programmable controls would require complex, bulky, or costly implementation to achieve the results possible with a microprocessor.

A microprocessor control program (embedded software) can be easily tailored to different needs of a product line, allowing upgrades in performance with minimal redesign of the product. Different features can be implemented in different models of a product line at negligible production cost. Microprocessor control of a system can provide control strategies that would be impractical to implement using electromechanical controls or purpose-built electronic controls. For example, an engine control system in an automobile can adjust ignition timing based on engine speed, load on the engine, ambient temperature, and any observed tendency for knocking—allowing an automobile to operate on a range of fuel grades.

2.5.2 CLIENT-SERVER COMPUTING

2.5.2.1 INTRODUCTION

In the 1970s and 1980s was the era of centralized computing, with IBM

mainframe occupied over 70% of the world's computer business. Business transactions, activities and database retrieval, queries and maintenance are all performed by the omnipresent IBM mainframe. We are now in the transition phase towards Client-Server Computing, a totally new concept and technology to re-engineer the entire business world. Someone has called it the wave of the future - the computing paradigm of the 1990s.You may start to wonder how is Client-Server computing is different from traditional mainframe computing and what are the benefits from employing it in business. The main emphasis of Client-Server Architecture is to allow large application to be split into smaller tasks and to perform the tasks among host (server machine) and desktops (client machine) in the network. Client machine usually manages the front-end processes such as GUIs (Graphical User Interfaces), dispatch requests to server programs, validate data entered by the user and also manages the local resources that the user interacts with such as the monitor, keyboard, workstation, CPU and other peripherals. On the other hand, the server fulfills the client request by performing the service requested. After the server receives requests from clients, it executes database retrieval, updates and manages data integrity and dispatches responses to client requests.

The goals of Client-Server Computing are to allow every networked workstation (Client) and host (Server) to be accessible, as needed by an application, and to allow all existing software and hardware components from various vendors to work together. When these two conditions are met, the environment can be successful and the benefits of client/server computing, such as cost savings, increased productivity, flexibility, and resource utilization, can be realized.

The evolution of Client-Server Computing has been driven by business needs, as well as the increasing costs for host (mainframe and midrange) machines and maintenance, the decreasing costs and increasing power of micro-computers and the increased reliability of LANs (Local Area Networks).

In the past twenty years, there are dramatic improvements in the hardware and software technologies for micro-computers. Micro-computers become affordable for small businesses and organizations. And at the same time their performances are becoming more and more reliable. On the other hand, the drop in price for mainframe is growing at a slower rate than the drop in its price. Little developments have achieved with mainframes.

The following are the improvements made by micro-computers:

Hardware: The speed of desktop microprocessors has grown exponentially, from a 8MHz 386-based computers to 100Hz-based Pentium-based microprocessors. These mass-produced microprocessors are cheaper and more powerful than those used in mainframe and midrange computers. On the other hand, the capacity of main memory in micro-computers has been quadrupling every three years. Typically main memory size is 16 Megabytes nowadays. Besides, the amount of backup

storage and memory such as hard disks and CD-ROMs that are able to support micro-computers has also puts an almost unlimited amount of data in reach for end-users.

Software: The development and acceptance of GUIs (Graphical User Interfaces) such as Windows 3.1 and OS/2 has made the PC working environment more user-friendly. And the user are more efficient in learning new application software in a graphical environment. Besides GUIs, the use of multithreaded processing and relational databases has also contributed to the popularity of Client-Server Computing.

Configurations in Client-Server Computing: Client-Server Computing is divided into three components, a Client Process requesting service and a Server Process providing the requested service, with a Middleware in between them for their interaction.

Client : A Client Machine usually manages the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs. It is the front-end of the application that the user sees and interacts with. Besides, the Client Process also manages the local resources that the user interacts with such as the monitor, keyboard, workstation, CPU and other peripherals.

Server: On the other hand, the Server Machine fulfils the client request by performing the service requested. After the server receives requests from clients, it executes database retrieval, updates and manages data integrity and dispatches responses to client requests. The server-based process may run on another machine on the network; the server is then provided both file system services and application services. Or in some cases, another desktop machine provides the application services. The server acts as software engine that manages shared resources such as databases, printers, communication links, or high powered-processors. The main aim of the Server Process is to perform the back-end tasks that are common to similar applications. The simplest forms of servers are disk servers and file servers. With a file server, the client passes requests for files or file records over a network to the file server. This form of data service requires large bandwidth and can slow a network with many users. The more advanced form of servers are Database servers, Transaction server and Application servers.

The Four Dominant Client/Server Application Models: Having had a deeper look into the terms and architectures of client/server technology, let's consider the dominant application models available. Nowadays, there are four client/server application models that are widely used in the market. They are Structured Query Language (SQL) databases, Transaction Processing (TP) monitors, groupware and distributed objects. Each one of them is capable of creating its own complete client/server applications with its own tools. Moreover, they also introduce their own favourable form of middleware (all this will be further discussed later). But first, what is the reason for having different models

instead of having just one model, and what is the advantages/disadvantages of having just one particular model. The reason why we need different models for different applications is because each one of them have their own advantages and disadvantages, and sometimes one model performs better than the others in one particular situation. Furthermore, standardizing the whole market with one particular model will not only discourage the vendors from developing other new (and better) models, but also put off other potential small companies from competing with those gigantic ones. Having said that, standardizing the market with one particular model does have the advantage of concentrating the development of that particular model-based software, and hence improvements can be achieved much faster and as a result, cost of running/implementing/services will reduce significantly.

2.6 IMPLEMENTATION AND SCALABILITY

2.6.1 IMPLEMENTATION

Implementation is the carrying out, execution, or practice of a plan, a method, or any design, idea, model, specification, standard or policy for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of WWW Consortium-recommended specifications, and software development tools contain implementations of programming languages. A special case occurs in object-oriented programming, when a concrete class implements an interface; in this case the concrete class is an implementation of the interface and it includes methods which are implementations of those methods specified by the interface.

In the Information Technology (IT) industry, implementation refers to post-sales process of guiding a client from purchase to use of the software or hardware that was purchased. This includes requirements analysis, scope analysis, customizations, systems integrations, user policies, user training and delivery. These steps are often overseen by a project manager using project management methodologies. Software Implementations involve several professionals that are relatively new to the knowledge based economy such as business analysts, technical analysts, solutions architects, and project managers. To implement a system successfully, a large number of inter-related tasks need to be carried out in an appropriate sequence.

In political science, implementation refers to the carrying out of public policy. Legislatures pass laws that are then carried out by public servants working in bureaucratic agencies. This process consists of rule-making, rule-administration and rule-adjudication. Factors impacting

implementation include the legislative intent, the administrative capacity of the implementing bureaucracy, interest group activity and opposition, and presidential or executive support.

2.6.1 SCALABILITY

Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth. For example, it can refer to the capability of a system to increase its total output under an increased load when resources (typically hardware) are added.

An analogous meaning is implied when the word is used in an economic context, where scalability of a company implies that the underlying business model offers the potential for economic growth within the company. Scalability, as a property of systems, is generally difficult to define and in any particular case it is necessary to define the specific requirements for scalability on those dimensions that are deemed important. It is a highly significant issue in electronics systems, databases, routers, and networking. A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a scalable system. An algorithm, design, networking protocol, program, or other system is said to scale if it is suitably efficient and practical when applied to large situations (e.g. a large input data set, a large number of outputs or users, or a large number of participating nodes in the case of a distributed system).

Check Your Progress

- What is computing platform?
- What is scalability?
- Explain the term Implementation.

2.7 SUMMARY

The term distributed means the things are scattered but they are directly or indirectly related with each other. Nothing, in computer Science and other technologies is untouched with this word ie distributed computing. Distributed computing also refers to the use of distributed systems to solve computational problems.

A computer program that runs in a distributed system is called a *distributed program*, and *distributed programming* is the process of writing such programs. There are many alternatives for the message passing mechanism, including pure HTTP, RPC-like connectors and message queues. A goal and challenge pursued each other by message passing.

The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing.

In computing, a file server (or fileserver) is a computer attached to a network that has the primary purpose of providing a location for shared disk access, i.e. shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are attached to the same computer network.

2.8 TERMINAL QUESTIONS

1. What do you understand by Distributed Computing?
2. Give the meaning of file server database.
3. What are the main benefits of Distributed Computing?
4. Discuss the term parallel computing with reference to Distributed computing.
5. Write a short note on Four Dominant Client/Server Application Models.
6. Compare implementation and scalability.
7. Explain embedded applications.
8. Is there any drawback of Distributed Computing? If any then discuss.

UNIT-3 DESIGNING CLIENT-SERVER APPLICATIONS

Structure

3.0 INTRODUCTION

The trend of business process reengineering has generated the need to improve communications, reduce overhead, enhance work-process efficiencies and facilitate information sharing across departmental and organizational boundaries. As a result, information technology is faced with a new set of application requirements and pressures toward distributed computing.

New Business Requirements Recent government initiatives to expedite the purchase ordering process, improve inventory control and deliver better services to the public have created demands for applications that would link up the government agencies to their vendors, partners and customers. These types of business systems have to be scalable to accommodate a large and growing number of users (in the range of hundreds or thousands). In addition, not only is multi-platform support essential, these applications also have to be adaptable to emerging client operating systems (e.g., Taligent). Finally, considerations have to be made for dial-up (or remote) users.

Current Approach to Application Design Today's most popular approach to application design cannot meet the aforementioned business needs. Most of the commonly used client/server development tools allow only client-side processing. In most organizations, client/server applications are developed using one of these tools in addition to a SQL database server that supports stored procedures. A stored procedure is written in a vendor-specific SQL dialect. It resides in a database for processing data requests sent from clients. Application created using this approach generally has two-tier architecture. The first tier is a single process running on the user machine. It consists of the presentation and application layers bundled together to form one executable, generated by the client/server development software.

The second tier is the database server with stored procedures residing in it. This type of application architecture performs well in single-vendor/single-database environments with fewer than 100 users. Each user connection takes up some resources on the database server. As the number of users increases, the single database server will sooner or later run out of resources. More database servers will be required to accommodate the growing install base. One way to solve the problem is with a two-tier model and database replication. By replicating data on one server to another server, the number of users supported can be doubled. Data replication, however, is only suited for certain types of applications - those with a low chance of simultaneous record updates. In other words, when it is unlikely that two or more users will be updating the same record at the same time, data replication could be an appropriate approach to solving the scalability problem. However, for those applications with a high volume of transactions and with potentially hundreds or thousands of users, one may wish to find an alternative solution.

Some of the things that you may need to think about include:

- Designing a Distributed Database: This brings a new level of complexity to applications. What's on the client(s)? What's on the server(s)? How much disk space? What client/server model is to be used? Object Locking?

- Designing a Distributed Application: What runs on the client(s)? What runs on the server(s)? What runs on both? What client/server model is to be used?

- Performance and Communications Loads: How much load on the LAN? Can it run on twinax? What about slow leased line and dial-up access?

- Security and Integrity Considerations: Backup what? Backup when? Backup where? Software distribution and upgrade procedures across "n" PCs? Security administration across "n" PCs?

- The User Interface: What about OOD? GUI WIMP constructs? Action bar object-action designs? This is not a traditional i5/OS "menu driven" system.

3.1 OBJECTIVES

At the end of this unit you would come to know

- The meaning of designing client-server application
- Types of logic
- Division of Labor
- Client-Server communication
- Basic idea of protocols like SMTP
- Interaction of Client-Server with Protocols
- Goals for Client/Server Design
- Client/Server Performance Optimization
- Implementing a Client/Server Application

3.2 FUNDAMENTALS OF CLIENT-SERVER APPLICATION

Client-server is a program relationship in which one program (the client) requests a service or resource from another program (the server). Although the client/server model can be used by programs within a single computer, it is a more important concept for networking. In this case, the client establishes a connection to the server over a local area network (LAN) or wide-area network (WAN), such as the Internet. Once the server has fulfilled the client's request, the connection is terminated. Your Web browser is a client program that has requested a service from a server; in fact, the service and resource the server provided is the delivery of this Web page.

3.2.1 BASIC CLIENT/SERVER ARCHITECTURE

The most basic form of the client/server architecture involves two computers: one computer, the server, is responsible for storing some sort of data and handing it to the other computer, the client, for user interaction. The user can modify that data and save it back to the server. The Web implements this simple form of client/server architecture for multiple client machines. Your computer, the client, uses a Web browser to display HTML documents stored across the Internet on a Web server.

There are four software components to the Web system:

- A browser such as Netscape that displays HTML documents on a client machine.
- A server program running on the server that hands HTML documents to client browsers.
- The HTML documents stored on the server machine.
- The communications protocol that handles the communication of data between the client and server.

The Figure 3.1 shows how this architecture fits together.

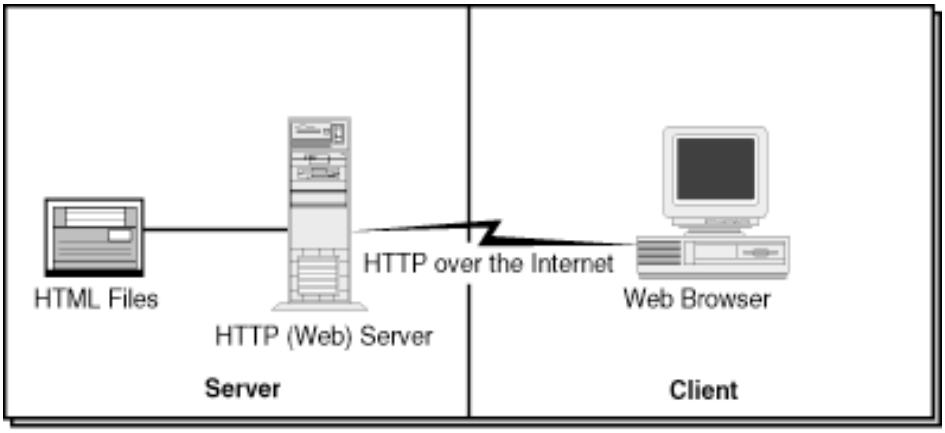


Figure 3.1 : The Client/Server Architecture of the Web

3.2.2 PROCESS CLASSIFICATION

Client process, the process that requires a service is called a client process and on the other hand the process that provides the required service is called server process. The client requires a service and the server provides the service and makes the results available to the client. In general, client software:

- It is an arbitrary application program that becomes a client temporarily when remote access is needed, but also performs other computation locally.
- It is invoked locally by a user, and executes only for one session.
- It runs locally on a user personal computer.
- It actively initiates contact with a server.
- It can access multiple services as needed, but actively contacts one remote server at a time.
- It does not require special hardware or a sophisticated operating system.

3.2.3 SERVER FUNCTIONS

Any server function has the following properties:

- Is a special purpose, privileged program dedicated to providing one service, but can handle multiple remote clients at the same time.
- It runs on a shared computer (i.e. not a user's personal computer).
- Waits for contact from arbitrary remote clients.
- Accepts contact from arbitrary clients, but offers a single service.
- Requires powerful hardware and a sophisticated operating system.

3.2.4 APPLISCATION SERVER

3.2.4.1 KINDS OF CLIENT SERVICES

- mail server
- file server
- terminal server
- name server
- authentication server
- gateway server
- administration server

3.2.4.2 A SERVER MUST GUARANTEE

- *Authentication:* client identity verification
- *Authorization:* verification of the possibility for a client to access to a particular service
- *Data security:* guarantee that specific data cannot be read and/or modified.

3.2.5 CHARACTERISTICS OF CLIENT-SERVER ARCHITECTURE

- Client and server machines need different types of hardware and software resources.
- Client and server machines may belong to different vendors.
- Horizontal scalability (increase of the client machines) and vertical scalability (migration to a more powerful server or to a multi-server solution).
- A client or server application interacts directly with a transport layer protocol to establish communication and to send or receive information.

- The transport protocol then uses lower layer protocols to send or receive individual messages. Thus, a computer needs a complete stack of protocols to run either a client or a server.
- A single server-class computer can offer multiple services at the same time; a separate server program is needed for each service.
- Identifying a particular service TCP uses 16-bit integer values (protocol port numbers) to identify services, and assign a unique port number to each service.
- A client specifies the protocol port number of the desired service when sending a request.

3.2.6 CLASSIFICATION OF CLIENT-SERVER STRUCTURES

In a client/server application three functions are present: user interface, application programs, data management. Following the assignment of functions among client and server we have three possible types of structures:

Host- based processing: is not true client server computing. It refers to the traditional mainframe environment in which all or virtually all of the processing is done on a central host. The user's station is generally limited to the role of a terminal emulator.

Server-based processing: The client is principally responsible for providing a graphical user interface, while virtually all the processing is done on the server.

Client-based processing: Virtually all application processing may be done at the client, with the exception of database logic functions that are best performed at the server. This configuration is perhaps the most common client server approach in current use.

Cooperative processing: That application processing is performed in an optimized fashion, taking advantage of the strength of both client and server machines and of distribution of data.

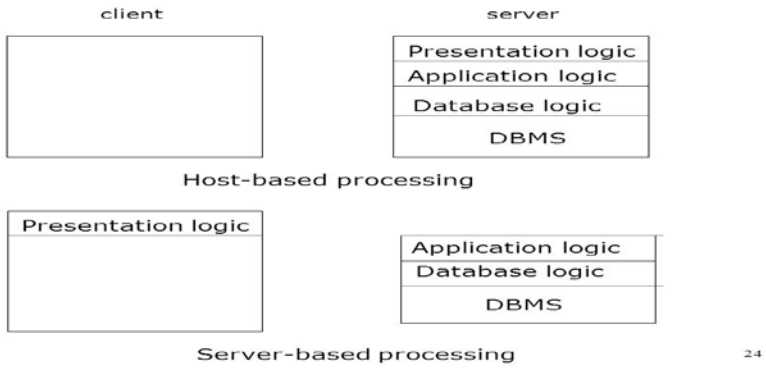


Figure 3.2: Host-Based and Server-Based Processing

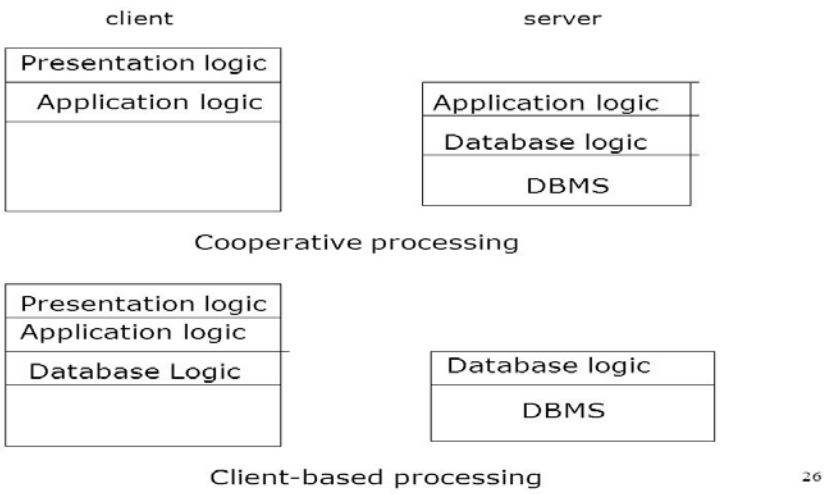


Figure 3.2: Cooperative-Based and Client-Based Processing

3.3 TYPES OF LOGIC

For purposes of distribution, it is possible to divide application software into three distinct types of logic:

- Presentation, which is associated with the user interface
- Business rule, which is associated with accomplishing a basic business process
- Data manipulation, which is associated with reading and writing persistent data

Presentation logic is associated with a procedure step definition. Business rule logic and data manipulation logic are generally found in elementary processes or process action blocks. In some applications, little or no business logic exists except for that directly governing data.

In simple client/server applications, these three types of logic can be divided into two components executing on different machines, one for presentation and one for data manipulation. Business rule logic is generally embedded in whichever component it appears to make sense, or it may be segmented into reusable action blocks. One reason for putting business rule logic in a reusable component is that you can easily generate and install the component for a different platform.

You do not have to decide what processing logic is required to support network communications. CA Gen for client/server products includes a communications runtime for you. For more information about the communications runtime components, see Understanding Distributed Processing.

The important task to remember is to design client/server applications to be modular and flexible. Maintain a clear distinction

between presentation logic, business rule logic, and data manipulation logic so that you can change between client/server styles as needed. Maintain data manipulation logic as separate action diagrams or common action blocks to be used by server procedures. This approach also allows many client procedures to use the server procedures.

3.3.1 PRESENTATION LOGIC

The client component of a client/server application requires a graphical user interface. For information about the special action statements for GUI applications, see Designing Action Diagrams.

3.3.2 BUSINESS RULE LOGIC

A major design consideration for the processing logic is *what* logic is required to fully support the tasks performed by the user. After you determine this, you can decide if the logic must be distributed or not. If the logic is distributed, you have considerations of which platform, client or server, is best suited for the particular functionality.

3.3.3 DATA MANIPULATION LOGIC

A key design consideration concerning data is how important is access to *current* data. Depending on the application and business needs, data that is updated periodically, instead of continually, may be adequate. Data that is replicated on the client platform is often used only for read access (data look up). If other actions are required, such as create, update, and delete, the database management system (DBMS) must have the capability to handle the data integrity, node directories, and two-phase commits. Typically, the DBMS provides these capabilities and not the generated application. You can, however, build with CA Gen the time stamping and locking logic in your application, but you must also design for various failure conditions.

Dividing the logic also gives rise to three basic design alternatives for client-server:

- remote presentation
- distributed process
- remote data access

3.4 DIVISION OF LABOR

Businesses of various sizes have various computer needs. Larger businesses necessarily need to use more computers than smaller businesses do. Large businesses routinely have large computer setups, such as mainframes and networks. A network for a large business

commonly has client-server architecture, also known as two-tier architecture. No matter what it is called, this type of architecture is a division of labor for the computing functions required by a large business.

Under the structure of the client-server architecture, a business's computer network will have a server computer, which functions as the "brains" of the organization, and a group of client computers, which are commonly called *workstations*. The server part of this architecture will be a large-capacity computer, perhaps even a mainframe, with a large amount of data and functionality stored on it. The client portions are smaller computers that employees use to perform their computer-based responsibilities.

Servers commonly contain data files and applications that can be accessed across the network, by workstations or employee computers. An employee who wants to access company-wide data files, for instance, would use his or her client computer to access the data files on the server. Other employees may use a common-access application by accessing the server through their client computers.

Check Your Progress

- What is implementation of client-server application?
- Give the important goals of client/server design.

3.5 CLIENT-SERVER COMMUNICATION

A key part of any Asynchronous JavaScript and XML (Ajax)-based web application is the communication layer between the client and the server. Modern web applications are all based on various Ajax-related concepts. The use of Ajax techniques led to an increase in interactive or dynamic interfaces on web pages. The Ajax revolution began with the notion that web applications can retrieve data from the server asynchronously in the background, and interaction between the web page and the server is not limited to the moment when the page is fetched. The web page concept extended into a long-living web application that interacts with the user through ongoing communication with the application's back end. A few examples for what this ongoing communication allows are

- Sending and receiving of information
- Ad-hoc input validation (for example, password strength)
- Auto-completion of user input based on rules and analysis done on the server

To perform the tasks related to the client-server interaction, an application needs an optimal communication layer that provides the proper communication mechanism for each communication task.

3.6 INTERACTION OF CLIENT-SERVER WITH PROTOCOLS

3.6.1 TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL

The TCP/IP protocol suite is now being used in many commercial applications. It is particularly evident in internetworking between different LAN environments. TCP/IP is specifically designed to handle communications through "networks of interconnected networks." In fact, it has now become the de facto protocol for LAN-based Client/Server connectivity and is supported on virtually every computing platform. More importantly, most inter-process communications and development tools embed support for TCP/IP where multiplatform interoperability is required. It is worth noting that IBM has followed this growth and not only provides support for TCP/IP on all its platforms, but now enables the transport of its own interoperability interfaces (such as CPIC, APPC) on TCP/IP.

3.6.2 TCP/IP'S ARCHITECTURE

The TCP/IP protocol suite is composed of the following components: a network protocol (IP) and its routing logic, three transport protocols (TCP, UDP, and ICMP), and a series of session, presentation and application services. The following sections highlight those of interest.

3.6.3 INTERNET PROTOCOL

IP represents the network layer and is equivalent to OSI's IP or X.25. A unique network address is assigned to every system, whether the system is connected to a LAN or a WAN. The system comes with its associated routing protocols and lower level functions such as network-to-physical address resolution protocols (ARP). Commonly used routing protocols include RIP, OSPF, IGRP, and Cisco's proprietary protocol. OSPF has been adopted by the community to be the standards-based preferred protocol for large networks.

3.6.3.1 INTERNET ADDRESSING SCHEME

An addressing scheme is clearly a requirement for communications in a computer network. With an addressing scheme, packets are forwarded from one location to another.An IP address is a unique identifier used to locate a device on the IP network. Conceptually an IP address is a unique global address for a network interface. It is a **32 bit long** identifier and encodes a network number (**network prefix**) and a **host number.** IP has two versions i.e; IPv4 and IPv6, out of which IPv4 is currently used in

India. IPv4 has 32- bit address which is divided into four octates. It is represented in a so-called Dotted Decimal notation. Eg. 123. 65. 1. 92. Each byte is identified by a decimal number in the range [0..255].

To make the system scalable, the address structure is subdivided into the *network* ID and the *host* ID. The network ID identifies the network the device belongs to; the host ID identifies the device. This implies that all devices belonging to the same network have a single network ID. Based on the bit positioning assigned to the network ID and the host ID, the IP address is further subdivided into classes A, B, C, D (multicast), and E (reserved).

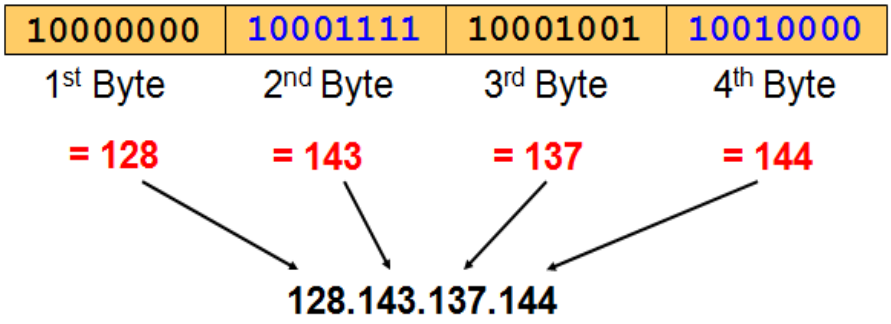


Figure 3.3 : Dotted Decimal Notation

3.6.4 TRANSPORT PROTOCOLS

TCP provides Transport services over IP. It is connection-oriented, meaning it requires a session to be set up between two parties to provide its services. It ensures end-to-end data transmission, error recovery, ordering of data, and flow control. TCP provides the kind of communications that users and programs expect to have in locally connected sessions. UDP provides connectionless transport services, and is used in very specific applications that do not require end-to-end reliability such as that provided by TCP.

3.6.4.1 TCP CONNECTION

TCP is a connection oriented protocol due to 3-way handshaking. TCP has 3 phases:

1. Connection establishment
2. Data transfer
3. Connection release

Out of which connection establishment and connection release both takes place through 3 way

Handshaking (figure 3.4)

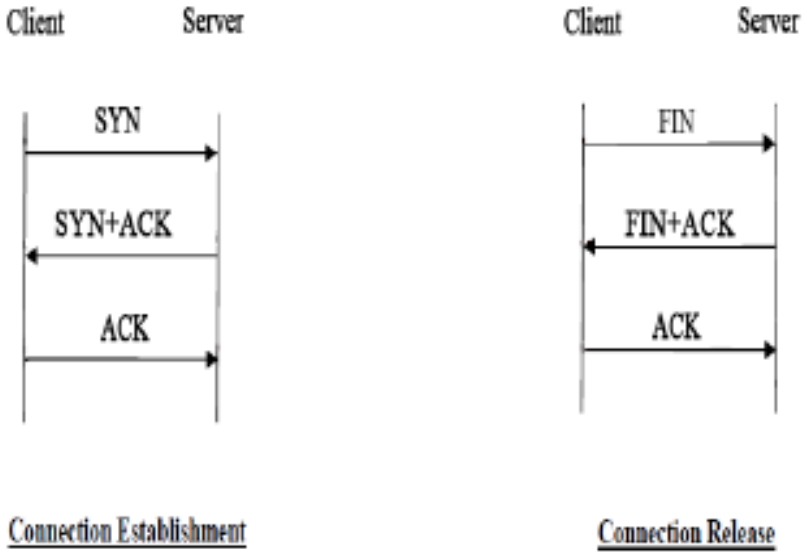


Figure 3.4 : TCP Connection

3.6.5 TELNET

Telnet is an application service that uses TCP. It provides terminal emulation services and supports terminal-to-host connections over an internetwork. It is composed of two different portions: a client entity that provides services to access hosts and a server portion that provides services to be accessed by clients. Even workstation operating systems such as OS/2 and Windows can provide telnet server support, thus enabling a remote user to log onto the workstation using this method.

3.6.6 FILE TRANSFER PROTOCOL (FTP)

FTP uses TCP services to provide file transfer services to applications. FTP includes a client and server portion. Server FTP listens for a session initiation request from client FTP. Files may be transferred in either direction, and ASCII and binary file transfer is supported. FTP provides a simple means to perform software distribution to hosts, servers, and workstations.

3.6.7 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

SNMP provides intelligence and services to effectively manage an internetwork. It has been widely adopted by hub, bridge, and router manufacturers as the preferred technology to monitor and manage their devices. SNMP uses UDP to support communications between agents—intelligent software that runs in the devices—and the manager, which runs in the management workstation. Two basic forms of communications can occur: SNMP polling (in which the manager periodically asks the agent to

provide status and performance data) and trap generation (in which the agent proactively notifies the manager that a change of status or an anomaly is occurring).

3.6.8 NETWORK FILE SYSTEM (NFS)

The NFS protocol enables the use of IP by servers to share disk space and files the same way a Novell or LAN Manager-network server does. It is useful in environments in which servers are running different operating systems. However, it does not offer support for the same administration facilities that a NetWare environment typically provides.

3.6.9 SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

SMTP uses TCP connections to transfer text-oriented electronic mail among users on the same host or among hosts over the network. Developments are under way to adopt a standard to add multimedia capabilities (MIME) to SMTP. Its use is widespread on the Internet, where it enables any user to reach millions of users in universities, vendor organizations, standards bodies, and so on. Most electronic mail systems today provide some form of SMTP gateway to let users benefit from this overall connectivity.

3.7 GOALS FOR CLIENT-SERVER DESIGN

When you design a client-server application, you're balancing several sets of requirements. You want to build the fastest, most productive application for your users. You also want to ensure the integrity of application data, make the most of existing hardware investments, and build in scalability for the future. In addition, as a developer, you want to make the development process as streamlined and cost-efficient as possible. Some common goals for the client-server systems are:

- **Portability:** Server can be installed on a variety of machines and operating systems and functions in a variety of networking environments.
- **Transparency:** The server might itself be distributed (why?), but should provide a single "logical" service to the user.
- **Performance:** Client should be customized for interactive display-intensive tasks and Server should provide CPU-intensive operations.
- **Scalability:** Server has spare capacity to handle larger number of clients.
- **Flexibility:** Should be usable for a variety of user interfaces.
- **Reliability:** System should survive individual node and/or communication link problems.

3.8 CLIENT-SERVER PERFORMANCE OPTIMIZATION

When you have implemented your client-server application, you might find areas where you'd like to improve performance. For example, you can fine-tune your application to gain maximum performance by speeding up forms and queries and increasing data throughput. This section discusses optimization strategies for application performance on the client, network, and server.

3.8.1 CONNECTION USE OPTIMIZATION

Establishing a connection uses time and memory on both the client and the server. When you optimize connections, you balance your need for high performance against the resource requirements of your application.

3.8.2 SPEEDING UP DATA RETRIEVAL

You can speed up data retrieval by managing the number of rows fetched during progressive fetching, controlling fetch size, and using delayed Memo fetching.

3.8.3 QUERY AND VIEW ACCELERATION

You can improve query and view performance by adding indexes, optimizing local and remote processing, and optimizing parameter expressions.

3.8.4 FORM ACCELERATION

When you design a form based primarily on server data, take a minimalist approach for the best performance.

3.8.5 PERFORMANCE IMPROVEMENT ON UPDATES AND DELETES

You can speed up Update and Delete statements by, adding timestamps to your remote tables, using the CompareMemo property, using manual transaction mode, using server stored procedures, and batching updates.

3.8.6 RELATED SECTIONS

Creating Client/Server Solutions

Client/server applications combine the functionality of Microsoft® Visual FoxPro® on your local computer with the storage and security benefits provided by a remote server.

Client/Server Application Design

Building on multi-user development technologies, learn how to design a powerful client/server application.

Upsizing Visual FoxPro Databases

Creating local prototypes of your design can reduce development time and costs. When you have a tested local prototype, it is easy and beneficial to upsize your application, so it can take advantage of all the features provided by the remote server.

3.9 IMPLEMENTION OF CLIENT/SERVER APPLICATION

Whether you have created and upsized a working local prototype or developed your application against remote data using remote views, you have gained access to the large data stores typically available in a server database. In addition, you can take advantage of the security and transaction processing capabilities of the remote server. While remote views handle the main data management tasks, you can enhance your application by using SQL pass-through (SPT) technology to create objects on the server, run server stored procedures, and execute commands using native server syntax.

The techniques for implementing client/server technology in a working application that uses remote views are given below:

3.9.1 USING SQL PASS-THROUGH TECHNOLOGY

Remote views provide the most common and easiest method for accessing and updating remote data. The upsizing wizards can create remote views automatically in your database as part of upsizing, or you can use Microsoft Visual FoxPro to create remote views after upsizing.

3.9.2 WORKING WITH REMOTE DATA USING SQL PASS-THROUGH

After you retrieve a result set using SQL pass-through, you can view and control the properties of your result set cursor using the Microsoft Visual FoxPro functions CURSORGETPROP( ) and CURSORSETPROP( ).

3.9.3 HANDLING SQL PASS-THROUGH ERRORS

If a SQL pass-through function returns an error, Microsoft Visual FoxPro stores the error message in an array.

3.9.4 RELATED SECTIONS

Creating Client/Server Solutions

Client/server applications combine the functionality of Microsoft Visual FoxPro on the local computer with the storage and security benefits provided by a remote server.

Client/Server Application Design

Building on multi-user development technologies, learn how to design a powerful client/server application.

Upsizing Visual FoxPro Databases

Creating local prototypes of your design can reduce development time and costs. When you have a tested local prototype, it is easy and beneficial to upsize your application, so it can take advantage of all the features provided by the remote server.

Client/Server Performance Optimization

After upsizing and implementing, you can take additional steps to optimize the performance of your application. Find out what you can do in Microsoft Visual FoxPro and on the remote server to optimize your client-server application.

Check Your Progress

- What is implementation of client-server application?
- Give the important goals of client/server design.

3.10 SUMMARY

Today, over 80 percent of the applications running on a Microsoft Windows platform access data. More and more of these applications are used where the client/server architecture is not only recommended, it is a requirement. Unfortunately, most of these applications fail to succeed for a variety of reasons, including poor planning, design, and implementation. Here, we have examined the most common mistakes, and discussed the benefits of using the Active Platform. Developers use the Active Platform so that client/server applications work seamlessly over the Internet, an Intranet, or corporate network.

There are many answers about what differentiates client/server architecture from some other design. There is no single correct answer, but generally, an accepted definition describes a *client* application as the user interface to an intelligent database engine—the *server*. Well-designed client applications do not hard code details of how or where data is

physically stored, fetched, and managed, nor do they perform low-level data manipulation. Instead, they communicate their data needs at a more abstract level, the server performs the bulk of the processing, and the result set isn’t raw data but rather an intelligent answer. The word "design" here does not simply refer to the design of the User Interface with its action bars, push buttons, drop downs, etc. but to much larger and infinitely more complex issues that you will have to resolve as part of the design process.

3.11 TERMINAL QUESTIONS

1. What do you understand by Client-server design?
2. Discuss the main goals of Client-Server design.
3. Define the term communication and protocol.
4. Explain the few different communication protocols important in data communication.
5. Explain different characteristics of Client-Server architecture.
6. What are the main elements of Client-Server Computing?
7. Write a short note on Client-Server classification.
8. Why we need Client-Server performance optimization? Explain.
9. Give the dos and don’ts during the design of Client-Server application.
10. Write a short note on types of logic.

Uttar Pradesh Rajarshi Tandon Open University

Bachelor in Computer Application

BCA-E10

Client Server Technology

BLOCK

2

INTRODUCTION TO ASP.NET

UNIT-4

Introduction to.NET Framework

UNIT-5

Traditional ASP Basics

UNIT-6

ASP.NET Introduction and Controls

Course Design Committee

Dr. Ashutosh Gupta, **Chairman**
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Prof. R.S. Yadav **Member**
Dept. of Computer Science and Engineering, MNNIT, Allahabad

Ms. Marisha **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Course Preparation Committee

Dr. Krishan Kumar **Author**
Assistant Professor,
Department of Comuter Science Faculty of Technology
Gurukula Kangri Vishwavidyalaya, Haridwar (UK)

Dr. V.K. Saraswat **Editor**
Director (IET, Khandare Campus)
Institute of Engineering and Technology
Dr. B.R. Ambedkar University, Agra-282002

Dr. Ashutosh Gupta,
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

©UPRTOU, Prayagraj-2020
ISBN : 978-93-83328-13-0

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2020.
Printed By : Chandrakala Universal Pvt. 42/7 Jawahar Lal Neharu Road, Prayagraj.

BLOCK INTRODUCTION

Block 2 basically contains three units which are mainly intended with ASP.NET technology and its applications. *Unit 4* introduces the history, evolution and notion of .NET framework and also gives its overview. It overview contains features, advantages, disadvantages, main challenges etc. Moreover, .NET Framework (pronounced *dot net*) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. The .Net frame work Provide and environment for building and running Web services and other application. It consist of components such as common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support wide range of technologies.

As Dot Net programming logic can be developed in any Dot Net framework compatible languages; hence Dot Net is called as language independent. Microsoft is introducing approximately 40 languages into Dot Net framework, out of which as of now approximately 24 languages and one specification are released.

Unit 5 covers the basics of Active Server Pages (ASP). It explores ASP's role and change in the today's changing environment. ASP is basically a language used to develop server side programs. It is normally used along with the .NET framework and becomes ASP.NET. Moreover, Active Server Pages were introduced by Microsoft in 1996 as a downloadable feature of Internet Information Server 3.0. The concept is pretty simple: an Active Server Page allows code written in the JavaScript or VBScript languages to be embedded within the HTML tags of a Web page and executed on the Web server. There are great advantages to this, not the least of which is security. Since your code is executed on the Web server, only HTML tags are sent to the browser. The result is that the ASP code is "invisible" to the end user.

Last *Unit 6* describes the principles of designing a application using ASP.NET and Web form controls. Before proceeding with this tutorial, you must also have a basic understanding of .NET programming language. As you are going to develop web-based applications using ASP.NET web application framework, it will be good if you have an understanding of other web technologies such as HTML, CSS, AJAX, etc.

We also need to understand about the Web application. A Web application consists of document and code pages in various formats. The simplest kind of document is a static HTML page, which contains information that will be formatted and displayed by a Web browser. An HTML page may also contain hyperlinks to other HTML pages. A hyperlink (or just *link*) contains an address, or a Uniform Resource Locator (URL), specifying where the target document is located. The resulting combination of content and links is sometimes called *hypertext* and provides easy navigation to a vast amount of information on the World Wide Web.

UNIT-4 INTRODUCTION TO .NET FRAMEWORK

Structure

4.0 INTRODUCTION

The .NET Framework is a class of reusable libraries (collection of classes) given by Microsoft to be used in other .Net applications and to develop, build and deploy many types of applications on the Windows platform including the following:

- Console Applications
- Windows Forms Applications
- Windows Presentation Foundation (WPF) Applications
- Web Applications
- Web Services
- Windows Services
- Services-oriented applications using Windows Communications Foundation (WCF)
- Workflow-enabled applications using Windows Workflow Foundation(WF)
- Silverlight Application
- WCF workflow service application
- Crystal Reports Application

That primarily runs on the Microsoft Windows operating system. What really happens when we compile a .NET program?

- The exe file that is created doesn't contain executable code, rather it's MicroSoft Intermediate Language (MSIL) code.
- When you run the EXE, a special runtime environment (the Common Language Runtime or CLR) is launched and the IL instructions are executed by the CLR to the machine language.
- The CLR comes up with a Just In Time Compiler that translates the IL to native language the first it is encountered.

Therefore the process of programming goes like:

- We write a program in C#, VB.Net and other languages.
- We compile our code to IL code based on the language compiler (csc.exe, vbc.exe and so on).
- Run your IL program that launches the CLR to execute your IL, using its JIT to translate your program into native code as it executes.

4.1 OBJECVTIVES

At the end of this unit you would come to know

- The meaning of dot net framework
- Basic knowledge of Common Type System
- What is Common Language Runtime (CLR)
- Common Language Specification (CLS)
- Microsoft Intermediate Language (MSIL)
- What is Just in time compilation
- platform independent concepts
- language independent concepts
- To simplify Web application development

4.2 FUNDAMENTAL OF .NET FRAMEWORK

.NET Framework (pronounced *dot net*) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. The .Net frame work Provide and environment for building and running Web services and other application. It consist of components such as common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support wide range of technologies.

As Dot Net programming logic can be developed in any Dot Net framework compatible languages; hence Dot Net is called as language independent. Microsoft is introducing approximately 40 languages into Dot Net framework, out of which as of now approximately 24 languages and one specification are released.

Eg:C#.Net,VB.Net,VC++,VJ#,VF#,PHP,COBOL,PERL,PHYTHON,JSCRIPT...etc
One specification is ASP.Net. VC#.Net is case sensitive, VB.Net is not case sensitive, and ASP.Net case sensitivity depends on integrated language.

The .NET development framework provides a new and simplified model for programming and deploying applications on the Windows platform. It provides such advantages as multiplatform applications, automatic resource management, and simplification of application deployment. As security is an essential part of .NET, it provides security support, such as code authenticity check, resources access authorizations, declarative and imperative security, and cryptographic security methods for embedding into the user's application.

.NET provides a simple object-oriented model to access most of the Windows application programming interfaces (APIs). It also provides mechanisms by which you can use the existing native code. In addition, it significantly extends the development platform by providing tools and technologies to develop Internet-based distributed applications. .Net framework support more than 57 language eg:VC++, VB.NET, C#.NET, J# , WPF, WCF, LINQ, AJAX e.t.c. The .NET Framework consists of three main parts:

1. Common Language Runtime,
2. Unified class libraries
3. Active Server Pages called ASP.NET.

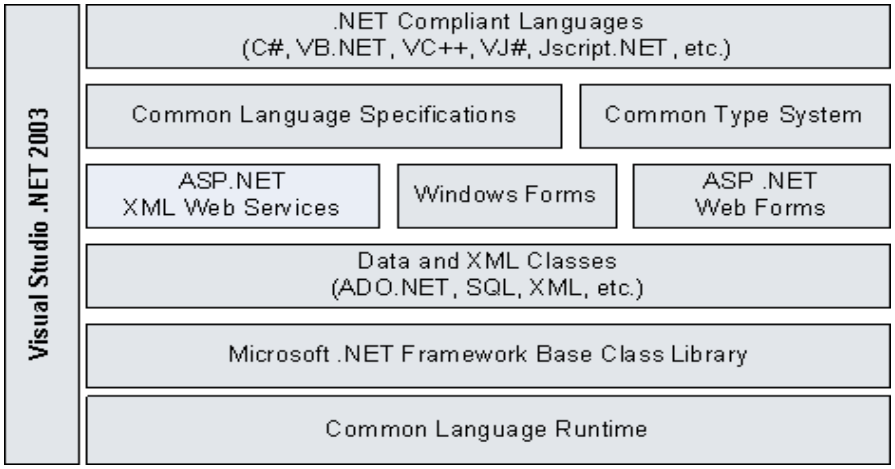


Figure 4.1: The .NET Framework Architecture

4.1.1 THE ORIGIN OF .NET TECHNOLOGY

.NET technology was originally known as NGWS which means Next Generation Windows Services. .Net is actually a thoughtful combination of technologies and development tools and the infrastructure is well integrated into Microsoft's operating systems and other products.

If we talk about technological advancements, there are various changes in the IT world in terms of infrastructure, hardware and advanced programming services. But for those seeking high end web application services, they are mostly mightily pleased with advanced .Net development services. It is one of the most used web technologies which is in vogue among web developers today. .NET is a combination of a number of technologies, tools as well as international coding standards. The evolution in web development is commonly attributed to .NET development which has generated tremendous interest in creating cloud-based solutions too.

There are a plethora of .NET applications in the online world too which are connected to diverse domains and industries. There is intense competition in the field but there is also the need to generate more business online with expertise and experience. To achieve this, optimal use of .Net development services is of huge importance. This latest web technology has encapsulated huge amount of information and critical data which include facts about customers, operations, services and the like. If one wants to create an impressive and marketable website with dynamic features and incredible functionalities, .Net application development is what you must resort to.

4.1.2 WHY .NET DEVELOPMENT RULES THE WEB DEVELOPMENT WORLD

- .NET technology gives varied support to different authentication services including e-wallets, passwords, and various types of smart cards.
- It delivers smooth browsing capabilities, navigation and innovative website functionalities for software development.
- By employing .NET technology one can take advantage of a plethora of directory services that can solve XML queries.
- One can develop web services using the SOAP toolkit and making optimum use of .Net framework.

Some of the .Net development services provided by vendors include

- .Net Ajax development solutions
- Web Services /WCF development
- Crystal reports development and implementation

- Windows forms development
- WPF and Windows Services application development
- Legacy modernization services and migration from ASP classic to .NET platform
- Custom web parts and SharePoint development
- Azure and custom cloud computing services

It is important for vendors to have a cohesive methodology in place derived from combining industry experience, standard principles and concepts, and analytical expertise of creating industry-specific solutions. It is important for clients to choose the best software development methodology considering the usefulness of the project such as project scope, deliverables, and the like. Different vendors have different methodologies to follow most of them assign dedicated teams for each of the .NET development projects. Usually every project has a Project Lead, senior .NET Developer and software engineers specializing in .NET working for every project.

For more information about .NET Development, visit Elan Emerging Technologies who have been one of the leading Offshore .NET Application Development company providing innovative ASP.NET Web Apps by their expert .NET Application Developers to meet unique business challenges in the web.

Components of .Net Framework

It has many components but mainly used in two components:

1. System class Library
2. Common Type System
3. Garbage Collection
4. Class Loader
5. Common Language Run Time

Figure 4.2 : CLR Components

1. **System Class Library (BCL)**:

   This is also called as *Base Class Library* and it is common for all types of applications i.e. the way you access the Library Classes and Methods in VB.NET will be the same in C#, and it is common for all other languages in .NET.

The following are different types of applications that can make use of .net class library.

- Windows Application.
- Console Application
- Web Application.
- XML Web Services.
- Windows Services.

In short, developers just need to import the BCL in their language code and use its predefined methods and properties to implement common and complex functions like reading and writing to file, graphic rendering, database interaction, and XML document manipulation. The base class library contains standard programming features such as Collections, XML, Data Type definitions, IO (for reading and writing to files), Reflection and Globalization to name a few. All of which are contained in the System namespace. As well, it contains some non-standard features such as LINQ, ADO.NET (for database interactions), drawing capabilities, forms and web support.

The below table provides a list each class of the base class library and a brief description of what they provide.

Table 4.1 : Base Class Library Namespaces & Their Meaning

Base Class Library Namespace	**Brief Description**
System	Contains the fundamentals for programming such as the data types, console, match and arrays, etc.
System.CodeDom	Supports the creation of code at runtime and the ability to run it.
System.Collections	Contains Lists, stacks, hashtables and dictionaries
System.ComponentModel	Provides licensing, controls and type conversion capabilities

System.Configuration	Used for reading and writing program configuration data
System.Data	Is the namespace for ADO.NET
System.Deployment	Upgrading capabilities via ClickOnce
System.Diagnostics	Provides tracing, logging, performance counters, etc. functionality
System.DirectoryServices	Is the namespace used to access the Active Directory
System.Drawing	Contains the GDI+ functionality for graphics support
System.EnterpriseServices	Used when working with COM+ from .NET
System.Globalization	Supports the localization of custom programs
System.IO	Provides connection to file system and the reading and writing to data streams such as files
System.Linq	Interface to LINQ providers and the execution of LINQ queries
System.Linq.Expressions	Namespace which contains delegates and lambda expressions
System.Management	Provides access to system information such as CPU utilization, storage space, etc.
System.Media	Contains methods to play sounds
System.Messaging	Used when message queues are required within an application, superseded by WCF

System.Net	Provides access to network protocols such as SSL, HTTP, SMTP and FTP
System.Reflection	Ability to read, create and invoke class information.
System.Resources	Used when localizing a program in relation to language support on web or form controls
System.Runtime	Contains functionality which allows the management of runtime behavior.
System.Security	Provides hashing and the ability to create custom security systems using policies and permissions.
System.ServiceProcess	Used when a windows service is required
System.Text	Provides the StringBuilder class, plus regular expression capabilities
System.Threading	Contains methods to manage the creation, synchronization and pooling of program threads
System.Timers	Provides the ability to raise events or take an action within a given timer period.
System.Transactions	Contains methods for the management of transactions
System.Web	Namespace for ASP.NET capabilities such as Web Services and browser communication.
System.Windows.Forms	Namespace containing the interface into the Windows API for the creation of Windows Forms programs.

System.Xml	Provides the methods for reading, writing, searching and changing XML documents and entities.

4.3 COMMON TYPE SYSTEM (CTS)

It describes set of data types that can be used in different .Net languages in common. The Common Type System (CTS) standardizes the data types of all programming languages using .NET under the umbrella of .NET to a common data type for easy and smooth communication among these .NET languages.

How CTS converts the data type to a common data type

To implement or see how CTS is converting the data type to a common data type, for example, when we declare an int type data type in C# and VB.Net then they are converted to int32. In other words, now both will have common data type that provides flexible communication between these two languages.

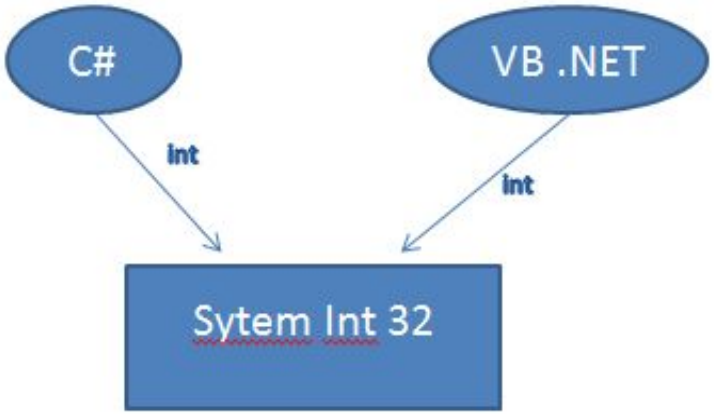


Figure 4.3 : Conversion of data type to Common data type

The common type system supports two general categories of types:
Value types

Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.

Reference types

Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types,

pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and delegates

Functions of the Common Type System (CTS)

- To establish a framework that helps enable cross-language integration, type safety, and high performance code execution.
- To provide an object-oriented model that supports the complete implementation of many programming languages.
- To define rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
- The CTS also defines the rules that ensure that the data types of objects written in various languages are able to interact with each other.
- The CTS also specifies the rules for type visibility and access to the members of a type, i.e. the CTS establishes the rules by which assemblies form scope for a type, and the Common Language Runtime enforces the visibility rules.
- The CTS defines the rules governing type inheritance, virtual methods and object lifetime.
- Languages supported by .NET can implement all or some common data types…
- When rounding fractional values, the halfway-to-even ("banker's") method is used by default, throughout the Framework. Since version 2, "Symmetric Arithmetic Rounding" (round halves away from zero) is also available by programmer's option.
- It is used to communicate with other language.

4.4 COMMON LANGUAGE RUNTIME (CLR)

.Net Framework provides runtime environment called Common Language Runtime (CLR).It provides an environment to run all the .Net Programs. The code which runs under the CLR is called as Managed Code. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management. Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed.

- Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to **Microsoft Intermediate Language** (MSIL) intern this will be converted to**Native Code** by CLR. See the below Fig.

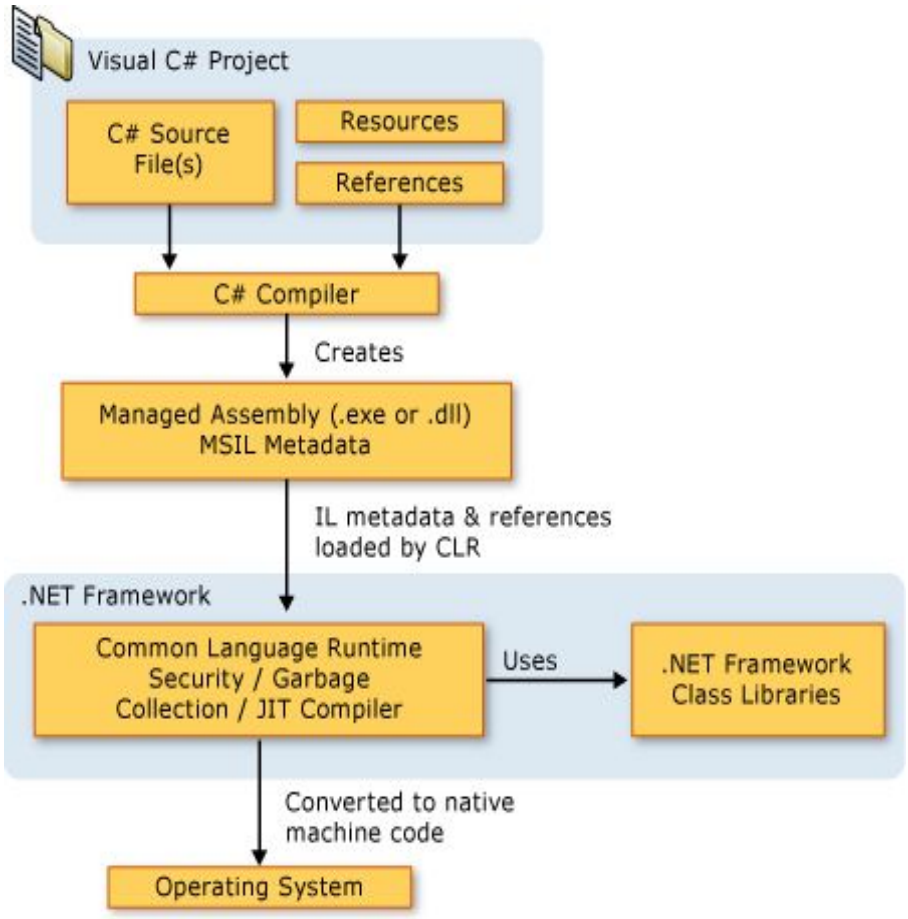


Figure 4.4 : Converting Code to Native Code

- There are currently over 15 language compilers being built by Microsoft and other companies also producing the code that will execute under CLR

4.4.1 FUNCTIONS OF THE CLR

1. Convert code into CLI.
2. Exception handling
3. Type safety
4. Memory management (using the Garbage Collector)
5. Security
6. Improved performance
7. Language independency

8. Platform independency
9. Architecture independency

4.4.2 COMPONENTS OF THE CLR

- **Class Loader:** Used to load all classes at run time.
- **MSIL to Native code:** The Just in Time (JIT) compiler will convert MSIL code into native code.
- **Code Manager:** It manages the code at run time.
- **Garbage Collector:** It manages the memory. Collect all unused objects and deallocate them to reduce memory.
- **Thread Support:** It supports multithreading of our application.
- **Exception Handler:** It handles exceptions at run time.

4.5 COMMON LANGUAGE SPECIFICATION (CLS)

The Common Language Specification (CLS) is a fundamental set of language features supported by the Common Language Runtime (CLR) of the .NET Framework. CLS is a part of the specifications of the .NET Framework. CLS was designed to support language constructs commonly used by developers and to produce verifiable code, which allows all CLS-compliant languages to ensure the type safety of code. CLS includes features common to many object-oriented programming languages. It forms a subset of the functionality of common type system (CTS) and has more rules than defined in CTS.

It is a sub set of CTS and it specifies a set of rules that needs to be adhered or satisfied by all language compilers targeting CLR. It helps in cross language inheritance and cross language debugging.

4.5.1 COMMON LANGUAGE SPECIFICATION RULES

It describes the minimal and complete set of features to produce code that can be hosted by CLR. It ensures that products of compilers will work properly in .NET environment. CLS defines the base rules necessary for any language targeting common language infrastructure to interoperate with other CLS-compliant languages. For example, a method with parameter of "unsigned int" type in an object written in C# is not CLS-compliant, just as some languages, like VB.NET, do not support that type.

CLS represents the guidelines to the compiler of a language, which targets the .NET Framework. CLS-compliant code is the code exposed and

expressed in CLS form. Even though various .NET languages differ in their syntactic rules, their compilers generate the Common Intermediate Language instructions, which are executed by CLR. Hence, CLS allows flexibility in using non-compliant types in the internal implementation of components with CLS-compliant requirements. Thus, CLS acts as a tool for integrating different languages into one umbrella in a seamless manner.

4.5.2 SAMPLE RULES

1. Representation of text strings
2. Internal representation of enumerations
3. Definition of static members and this is a subset of the CTS which all .NET languages are expected to support.
4. Microsoft has defined CLS which are nothing but guidelines that language to follow so that it can communicate with other .NET languages in a seamless manner

4.6 MICROSOFT INTERMIDIATE LANGUAGE (MSIL)

A.NET programming language (C#, VB.NET, J# etc.) does not compile into executable code; instead it compiles into an intermediate code called Microsoft Intermediate Language (MSIL). As a programmer one need not worry about the syntax of MSIL - since our source code in automatically converted to MSIL. The MSIL code is then send to the CLR (Common Language Runtime) that converts the code to machine language, which is then run on the host machine. MSIL is similar to Java Byte code. MSIL is the CPU-independent instruction set into which .NET Framework programs are compiled. It contains instructions
for loading, storing, initializing, and calling methods on objects. Combined with metadata and the common type system, MSIL allows for true cross-language integration Prior to execution, MSIL is converted to machine code. It is not interpreted.

Microsoft Intermediate Language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to the native code. During the runtime the Common Language Runtime (CLR)'s Just In Time (JIT) compiler converts the Microsoft Intermediate Language (MSIL) code into native code to the Operating System.

MSIL stands for Microsoft Intermediate Language. We can call it as Intermediate Language (IL) or Common Intermediate Language (CIL). During the compile time , the compiler convert the source code into Microsoft Intermediate Language (MSIL) .Microsoft Intermediate Language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to the native code. During the runtime the Common Language Runtime (CLR)'s Just In Time (JIT) compiler converts the

Microsoft Intermediate Language (MSIL) code into native code to the Operating System. When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata. The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file.

Microsoft Intermediate Language (MSIL) includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations Just In Time Compiler The .Net language, which conforms to the Common Language Specification (CLS), uses its corresponding runtime to run the application on different Operating Systems. During the code execution time, the Managed Code compiled only when it is needed, that is it converts the appropriate instructions to the native code for execution just before when each function is called. This process is called Just in Time (JIT) compilation, also known as Dynamic Translation.

With the help of Just in Time Compiler (JIT) the Common Language Runtime (CLR) doing these tasks. The Common Language Runtime (CLR) provides various Just In Time compilers (JIT) and each works on a different architecture depending on Operating System. That is why the same Microsoft Intermediate Language (MSIL) can be executed on different Operating Systems without rewrite the source code. Just In Time (JIT) compilation preserves memory and save time during application initialization. Just In Time (JIT) compilation is used to run at high speed, after an initial phase of slow interpretation. Just In Time Compiler (JIT) code generally offers far better performance than interpreter.

4.7 JUST IN TIME (JIT)

Before the Microsoft Intermediate Language (MSIL) can be executed, it must be converted by a .NET Framework Just-In-Time (JIT) compiler to native code, which is CPU-specific code that runs on the same computer architecture as the JIT compiler. The JIT compiler is part of the Common Language Runtime (CLR). The CLR manages the execution of all .NET applications. In addition to JIT compilation at runtime, the CLR is also responsible for garbage collection, type safety and for exception handling.

4.7.1 JIT COMPILER

A Web Service or Web Forms file must be compiled to run within the CLR. Compilation can be implicit or explicit. Although you could explicitly call the appropriate compiler to compile your Web Service or Web Forms files, it is easier to allow the file to be complied implicitly. Implicit compilation occurs when you request the .asmx via HTTP-SOAP, HTTP-GET, or HTTP-POST. The parser (xsp.exe) determines

whether a current version of the assembly resides in memory or in the disk. If it cannot use an existing version, the parser makes the appropriate call to the respective compiler (as you designated in the Class property of the .asmx page).

When the Web Service (or Web Forms page) is implicitly compiled, it is actually compiled twice. On the first pass, it is compiled into IL. On the second pass, the Web Service (now an assembly in IL) is compiled into machine language. This process is called Just-In-Time JIT compilation because it does not occurs until the assembly is on the target machine. The reason you do not compile it ahead of time is so that the specific JIT for your OS and processor type can be used. As a result, the assembly is compiled into the fastest possible machine language code, optimized and enhanced for your specific configuration. It also enables you to compile once and then run on any number of operating systems.

4.7.2 HOW JIT WORKS?

Before MSIL(MS Intermediate Language) can be executed, it must converted by .net Framework Just in time (JIT) compiler to native code, which is CPU specific code that run on some computer architecture as the JIT compiler. Rather than using time and memory to convert all the MSIL in portable executable (PE) file to native code, it converts the MSIL as it is needed during execution and stored in resulting native code so it is accessible for subsequent calls.

The runtime supplies another mode of compilation called install-time code generation. The install-time code generation mode converts MSIL to native code just as the regular JIT compiler does, but it converts larger units of code at a time, storing the resulting native code for use when the assembly is subsequently loaded and executed. As part of compiling MSIL to native code, code must pass a verification process unless an administrator has established a security policy that allows code to bypass verification. Verification examines MSIL and metadata to find out whether the code can be determined to be type safe, which means that it is known to access only the memory locations it is authorized to access.

4.7.3 JIT TYPES

In Microsoft .NET there are three types of JIT (Just-In-Time) compilers which are Explained as Under:

1. Pre-JIT Compiler (Compiles entire code into native code completely)

2. Econo JIT Compiler (Compiles code part by part freeing when required)

3. Normal JIT Compiler (Compiles only that part of code when called and places in cache

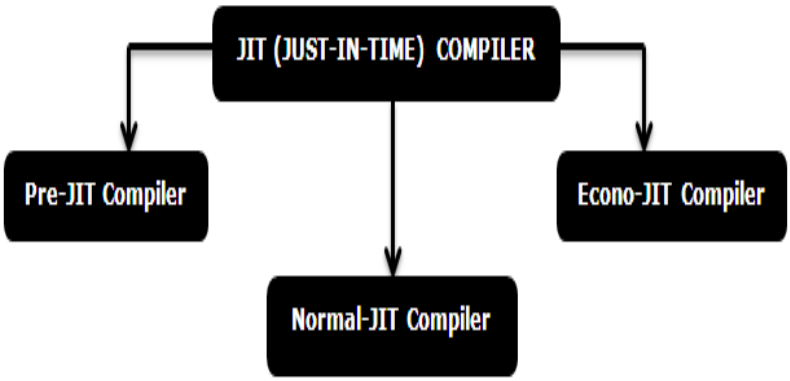


Figure 4.5 : Just In Time Types

4.7.4 DESCRIPTION

Pre-JIT Compiler

Pre-JIT compiles complete source code into native code in a single compilation cycle. This is done at the time of deployment of the application.

Econo-JIT Compiler

Econo-JIT compiles only those methods that are called at runtime. However, these compiled methods are removed when they are not required.

Normal-JIT Compiler

Normal-JIT compiles only those methods that are called at runtime. These methods are compiled the first time they are called, and then they are stored in cache. When the same methods are called again, the compiled code from cache is used for execution. These methods are compiled the first time they are called, and then they are stored in cache. When the same methods are called again, the compiled code from cache is used for execution

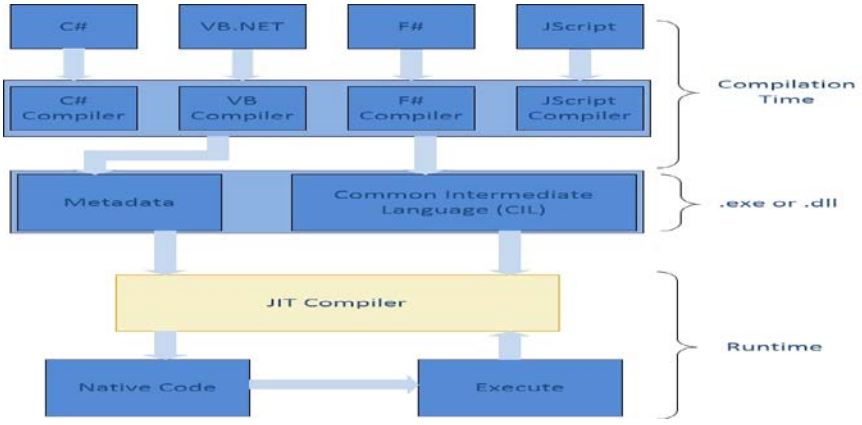


Figure 4.6 : .NET compiler

Different machine configurations use different machine level instructions. As Figure 1shows, the source code is compiled to *exe* or *dll* by the .NET compiler. Common Intermediate Language (CIL) consists of instructions that any environment supporting .NET can execute and includes metadata describing structures of both data and code. The JIT Compiler processes the CIL instructions into machine code specific for an environment. Program portability is ensured by utilizing CIL instructions in the source code. The JIT compiler compiles only those methods called at runtime. It also keeps track of any variable or parameter passed through methods and enforces type-safety in the runtime environment of the .NET Framework.

4.8 SUMMARY

- NET Framework is a code execution platform – the environment which .NET programs run
- .NET Framework consists of two primary parts: Common Language Runtime and .NET Class Libraries
- The CLS (Common Language Specification) allows different languages to interact seamlessly.
- The CTS (Common Type System) allows all languages to share base data types.
- .NET languages are compiled to MSIL by their respective compilers
- MSIL code is compiled to machine code by the JIT compiler
- All .NET languages have equal access to the FCL (Framework Class Library) which is a rich set of classes for developing software
- Base Class Library is set of basic classes: Collections, I/O, Networking, Security, etc.
- ADO.NET provides .NET applications with access to relational databases
- .NET has great XML support including: DOM, XSLT, XPath, and XSchema
- Windows Forms provides GUI interface for the .NET applications
- ASP.NET allows creating web interface to .NET applications
- Web Services expose functionality from web sites and make it remotely accessible through standard XML-based protocols
- Visual Studio .NET is powerful development IDE for all .NET languages and technologies

4.9 TERMINAL QUESTIONS

1. What is .NET Framework?
2. What are the main components of .NET Framework?
3. What is an IL?
4. What is DLL as per Dot Net?
5. What are the advantages of .Net?
6. How to invoke garbage collector programmatically?
7. What is a Managed Code?
8. What are the different types of JIT's?
9. What are Value types and Reference types?
10. Explain the concept of Boxing and Unboxing?
11. What is Code Document Object Model (CodeDom)?
12. Difference between .exe and .dll?
13. What is a Dll Hell?

UNIT-5 INTRODUCTION TO ASP BASIC

Structure

5.0 INTRODUCTION

Basically ASP is a development framework for building web pages. Do you have any problem with the development of static HTML pages? Do you want to create dynamic web pages? Do you want to enable your web pages with database access? If your answer is "Yes", ASP might be a solution for you. In May 2000, Microsoft estimated that there are over 800,000 ASP developers in the world. You may come up with a question what the heck ASP is. After reading this unit, you will be able to know what it is, how it works and what it can do for you.

Active Server Pages were introduced by Microsoft in 1996 as a downloadable feature of Internet Information Server 3.0. The concept is pretty simple: an Active Server Page allows code written in the JavaScript or VBScript languages to be embedded within the HTML tags of a Web page and executed on the Web server. There are great advantages to this, not the least of which is security. Since your code is executed on the Web server, only HTML tags are sent to the browser. The result is that the ASP code is "invisible" to the end user.

Another upside to the "server-side script" concept is that it allows things like database connections to be made from the Web server rather

than from the client. Therefore, any special configurations that might need to be set up, like ODBC data sources, only have to exist on the server. Of course, before you can create an Active Server Page (ASP), you'll need to look at the software requirements.

ASP is in very high demand in web development field. it is a server-side scripting language which is very popular due to its features. it runs on IIS (Internet Information Services) server. So coming to the topic this has been designed in a very efficient way and in a very understandable way. Before you continue, you should also have a basic understanding of HTML, CSS, JavaScript and SQL. In this course you will learn the basics of ASP. This is a beginners' level course and this course is not for experts in ASP.

ASP supports different development models like Classic ASP, ASP.NET Web Forms, ASP.NET MVC, ASP.NET Web Pages, ASP.NET API, ASP.NET Core.

5.1 OBJECVTIVES

At the end of this unit you would come to know

- Fundamental meaning of ASP
- The difference between client side programming and server side programming
- The ASP setup
- Why we choose ASP
- How ASP works
- Basic syntax rule of ASP
- ASP'S object model
- Working with Database in ASP
- Limitations of ASP
- Summary

5.2 ASP & THE SETUP

ASP stands for Active Server Pages. Microsoft introduced Active Server Pages in December 1996, beginning with Version 3.0. Microsoft officially defines ASP as: "Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server side scripting for IIS with native support for both VBScript and Jscript". In other words, ASP is a Microsoft technology that enables you to create dynamic web sites with the help of server side script, such as VBScript and Jscript. ASP

technology is supported on all Microsoft Web servers that are freely available. If you have Window NT 4.0 Server installed, you can download IIS (Internet Information Server) 3.0 or 4.0. If you are using. Window 2000, IIS 5.0 comes with it as a free component. If you have Window 95/98, you can download Personal Web Server (PWS), which is a smaller version of IIS, from Window 95/98 CD.

Before you can create an Active Server Page, you'll need a Web server that supports Active Server Pages. The most obvious choice would be Microsoft's Internet Information Server (IIS) version 3.0 or higher. IIS is available for Windows NT 4.0 or higher as part of the Windows NT option pack, which can be downloaded from Microsoft's Web site. For the highest level of compatibility and functionality, you'll want to use the most recent version of IIS.

Another option that you might not have considered is Microsoft's Personal Web Server for Windows 9*x*and Windows ME. If you're running Windows 95 or above, Personal Web Server can be installed by running "setup.exe" from the setup CD. Alternatively, it can be downloaded from Microsoft's Web site as part of the Windows NT option pack. Earlier this download is the only choice for Windows 95 or Windows ME. It's important to note that Microsoft does not support running Personal Web Server under Windows ME. While Personal Web Server is not the optimal choice for a production Web server, it is a great option for developing and testing your ASP scripts. If you're running IIS or Personal Web Server, no additional software is required to support Active Server Pages. To allow a user to access an ASP, the ability to do so must be enabled on the IIS server. This is done by selecting "Scripts" or "Execution (Including Scripts)" from the "Home Directory" tab of the Properties window for your Web site, as shown in Figure 5.1.

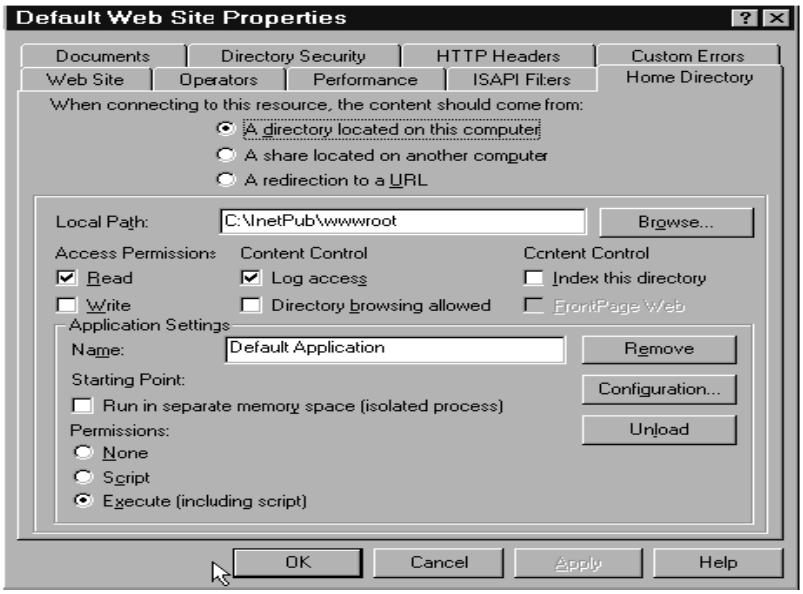


Figure 5.1: Active Server Pages are enabled by adding script permission

For other operating systems or Web servers, it gets a little tricky, but is possible. For Unix or Linux servers running the Apache Web server, you can use a bolt-on product to add ASP support. Sun Microsystems' Sun ONE Active Server Pages (formerly called Chili!Soft ASP) is one of these products. This product supports most, but not all, of the controls available in IIS. This is just one product that can add Active Server Page support to non-Microsoft Web servers. Table 2.1 has a more complete list of ASP compatibility products and the operating systems and Web servers they run on. There are products to allow ASPs to be used on just about any Web server out there. This fact makes using ASPs that much more attractive because you aren't limited in the choice of hardware, operating system, or Web server to host your Web pages. As you can see, there are even ASP-compatibility products for the iSeries.

5.3 ASP FILE

An ASP file is quite like an HTML file. It contains text, HTML tags and scripts, which are executed on the server. The two widely used scripting languages for an ASP page are VBScript and JScript. VBScript is pretty much like Visual Basic, whereas Jscript is the Microsoft's version of JavaScript. However, VBScript is the default scripting language for ASP. Besides these two scripting languages, you can use other scripting language with ASP as long as you have an ActiveX scripting engine for the language installed, such as PerlScript.

ASP contains the server scripts, which can contain any expressions, statements, procedures, operators valid for the scripting language. These server scripts are enclosed by the delimiters <% and %>

1. VBScript
2. JavaScript
3. Other Scripting Languages

The difference between an HTML file and an ASP file is that an ASP file has the ".asp" extension. Furthermore, script delimiters for HTML tags and ASP code are also different. A script delimiter is a character that marks the starting and ending of a unit. HTML tags begins with lesser than (<) and greater than (>) brackets, whereas ASP script typically starts with <% and ends with %>. In between the delimiters are the server-side scripts.

To write an ASP script, you don't need any additional software because it can be written with any HTML editor, such as Notepad. Nonetheless, if you feel bored with the plain text and would like to use some special software, you can use Microsoft visual InterDev, which helps you to easily create an ASP page by giving you nice highlights and debugging dialogue boxes. I hope that you already have an idea of what an ASP file is and how it is different from an HTML file. In the next step, you will learn how ASP works.

5.4 HOW DOES IT WORK?

As you have learned, scripts in an ASP file are server-side scripts, which means that the scripts are processed on the server and then the result of the scripts will be converted to HTML before sending to the web browser. To illustrate, let's take a look at this table to compare the process of retrieving an HTML page and an ASP page .

HTML process	ASP process
A user requests a web page (i.e., *http://www.gkv.ac.in/index.html* in the web browser.	A user requests a web page (i.e., *http://www.gkv.ac.in/index.html* in the web browser.
The browser finds the appropriate web server, and asks for the required page.	The browser finds the appropriate web server (like IIS or PWS), and asks for the required page.
The web server locates the required page and sends it back to the browser as HTML text.	The web server locates the required page, and parses out the ASP code within the ASP script delimiters (<%…%>), produces a standard HTML page. The server sends that HTML page back to the browser, so the user cannot see ASP code.
The browser executes the client side scripting (like JavaScript) determining how to display the results	The browser executes the client side scripting (like JavaScript) determining how to display the results
Web page is not coupled with any other programming language.	Web page is coupled with another programming language for database maintenance.
It is used for web designing purpose so dynamic websites are not possible.	It is used for web development purposes so dynamic web sites are possible.

As you can see, the whole process of the two is quite similar. Since ASP is a server-side technology, the required page is executed on the server before the HTML is created and served to the client. To make it clearer, Figure1 shows the processing behind a browser request to an ASP page . For example, a client types in a URL into your browser. The browser requests the ASP page from the web server. The server proceeds the file with ".asp" extension to ASP Engine in which Objects or ActiveX Components can be used to extend the web server with application-specific functionality. In addition, ASP will use ADO to connect to a Client Browser Server ASP Engine ADO ODBC Object / Component Provider Driver ASP Script Oracle SQL Server Access FoxPro Request ASP page

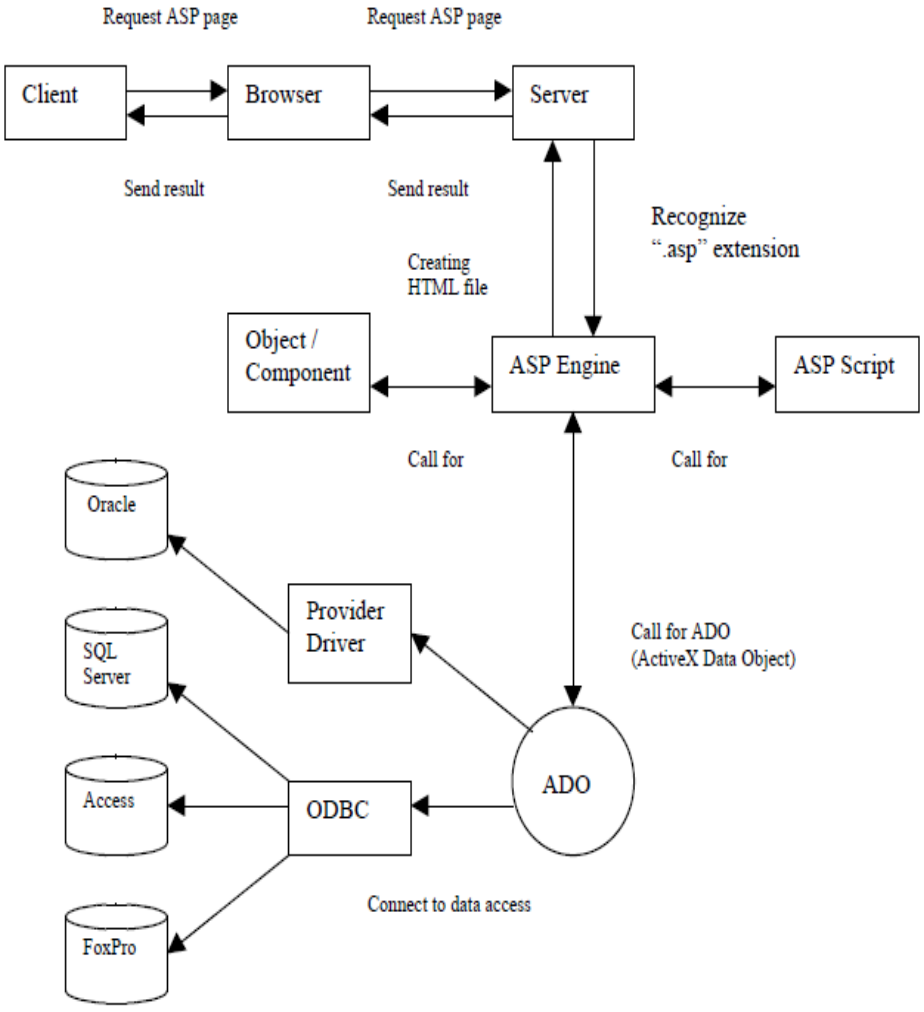


Figure 5.2 : How ASP works

For example, a client types in a URL into your browser. The browser requests the ASP page from the web server. The server proceeds the file with ".asp" extension to ASP Engine in which Objects or ActiveX

Components can be used to extend the web server with application-specific functionality. In addition, ASP will use ADO to connect to a database (SQL, Access, Oracle, etc.) to pull out the relevant data, such as the current weather in a specific area. Thus, a different page is generated according to the area specified and time that the page is accessed. Then, the server generates HTML tags before sending it back to the client. Therefore, when you view the source of an ASP file, you will not see any different from a standard HTML file.

ASP includes several build-in objects:

- Application Objects- Describes the methods, properties, and collections of the object that stores information related to the entire Web application, including variables and objects that exist for the lifetime of the application.
- ASP Error Object- Describes the properties of the object that stores information about an error condition.
- Object Context Objects- Describes a wrapper for the COM + ObjectContext object, which provides methods and events that are used only for transaction processing.
- Request Object- Describes the methods, properties, and collections of the object that stores information related to the HTTP request. This includes forms, cookies, server variables, and certificate data.
- Response Object- Describes the methods, properties, and collections of the object that stores information related to the server's response. This includes displaying content, manipulating headers, setting locales, and redirecting requests.
- Scripting Context Object- In a component, the **ScriptingContext** object returns references to the ASP built-in objects; however, this is an obsolete and unsupported method, removed in IIS 4.0. Use the COM+ ObjectContext object to return references to the ASP built-in objects. For more information, see COM+ ObjectContext Reference to the ASP Built-In Objects.

5.5 ADVANTAGES

Certainly, ASP must have some strength; otherwise, it won't be popular as such. Let's count on its strong points and functionality.

- Dynamic web page – Since ASP supports scripting languages, which run on the web server, your web page can be dynamically created. For example, you can create your web page so as to greeting each user when they log into your page differently.
- Browser independent – ASP is browser independent because all the scripting code runs on the server. The browser only gets the results from the server in a standard HTML page.

- Database Access – One of the greatest assets of ASP is database connectivity. ASP enables you to easily build rich database functionality into your web site, such as form processing.
- Building-in objects – The five built-in objects that come with ASP facilitate Web application development. For example, you can use Request object to retrieve browser request information.
- Free availability – Yes, it's free. You can download web server (IIS or PWS) for free from Microsoft's web site. You don't even have to have any special tool to write an ASP file. In other words, you can simply use any text editor, like NotePad.

Check Your Progress

- Define ASP.
- Write the benefits of ASP.

5.6 ASP OBJECT

An application on the Web may consists of several ASP files that work together to perform some purpose. The Application object is used to tie these files together. The Application object is used to store and access variables from any page, just like the Session object. The difference is that ALL users share ONE Application object (with Sessions there is ONE Session object for EACH user). The Application object holds information that will be used by many pages in the application (like database connection information). The information can be accessed from any page. The information can also be changed in one place, and the changes will automatically be reflected on all pages. The Application object's collections, methods, and events are described below:

5.6.1 COLLECTIONS

Collections	Description
Contents	The Contents collection contains all the items added to the application through the use of scripts (not through the use of the <OBJECT> tag).
StaticObjects	The StaticObjects collection contains all session-level objects added to the application through the use of the <OBJECT> tag. The collection can be used to retrieve the value of a specific property for an object, or to retrieve all properties for all static objects.

5.6.2 METHODS

Methods	Description
Contents.Remove	The Contents.Remove method deletes the specified item from the Application object Contents collection.
Contents.RemoveAll	The Contents.RemoveAll method deletes all the items from the Application object Contents collection.
Lock	The Lock method locks the Application Object, preventing other users from modifying the same application-level variables at the same time. The individual user retains control of the Application object until the Application.UnLock method is declared. If the Unlock method is not called explicitly, IIS will unlock the locked Application object when the script ends or times out.
Unlock	The Unlock method releases control of the locked application variables. Once Unlock has been called, other clients can again alter the values of the variables in the Application object. If the Unlock method is not called explicitly, IIS will unlock the locked Application object when the script ends or times out.

5.6.3 EVENTS

Events	**Description**
Application_OnStart	The Application_OnStart event occurs before the first new session is created (when the first client request is received).
Application_OnEnd	The Application_OnEnd event occurs when the ASP application is explicitly unloaded from the web server or when the web service on the web server is stopped.

5.7 WORKING WITH DATABASE IN ASP

One of ASP's greatest assets is that it allows you to tap into a database with ease. It's common to work with either an Access or a SQL database. Since Access is the easiest to start with, and is a tool you may already have, we'll use it for these examples. Once you learn core ASP techniques for working with your Access database, you'll find that many of the same skills will be necessary when you start working with SQL server.

When you want to tap into a database, you have to open it on the server. You can connect to and open the database by using either a Data Source Name (DSN) or by making a DSN-less connection directly in your script.

The normal way to access a database from inside an ASP page is to:

1. Create an ADO connection to a database
2. Open the database connection
3. Create an ADO recordset
4. Open the recordset
5. Extract the data you need from the recordset
6. Close the recordset
7. Close the connection

The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft SQL database on your web site. If you have a database called "northwind " located in your SQL Server, you can connect to the database with the following ASP code:

5.7.1 CREATING A DATA SOURCE NAME (DSN)

You can prep your database for use with ASP by setting up a System DSN for it in the control panel. On your local machine you can set up DSNs for any of the databases with which you are working. Then you can test your pages on your local server. If your site is being hosted by an ISP, and the ISP supports ASP, then it's likely you'll be provided with a GUI interface for creating a DSN for your database.

- In Windows 95/98/NT, open the Control Panel (Start/Settings/Control Panel) and double-click the ODBC entry.
- Select the System DSN tab and click Add.
- Select Microsoft Access Driver and click Finish.

- Fill in the Data Source Name. This is the name with which you'll refer to the database, so it operates as an alias.
- Click the Select button in the Database section and browse to find the Access database on your system.
- Click OK.

The new DSN will now be in the list of System DSNs and ready to use on your local server.

5.7.2 CONNECTING TO A DATABASE

Let's run through creating a DSN-less connection and look at how you connect to the database. When you create a DSN, you are storing a chunk of information about the database, so you don't have to repeat it every time you need it: type of database, name, location, and, optionally, user and password. To create a DSN-less connection, you'll have to supply the same information the long way. This sample, for example, shows a DSN-less connection being made to an Access database called *products*:

```
<%

StrConnect = "Driver={Microsoft Access Driver (*.mdb)};

DBQ=C:\db\products.mdb" Set objConn = Server.CreateObject
("ADODB.Connection")
objConn.Open StrConnect
%>
```

The second line defines the driver and the physical path for the database. To use a DSN-less connection, you need to know the actual location of the file from the root. Server.MapPath offers an easy work-around for anyone using a hosting service where the actual path can be hard to track down.

If we had set up a System DSN called *products*, the connection string would simply be:

```
< %
  Set objConn = Server.CreateObject ("ADODB.Connection")
  objConn.Open "products"
  %>
```

Now that the database is open, what can you do? Lots. The first thing to try, of course, is to read a set of records from the database and plop them onto your page. Before that, however, you'll need a recordset.

5.7.3 RECORDSET

A *recordset* is a collection of all the information stored in a specific database table. So, all the rows and columns in the table are available when you open the recordset. You need to open the recordset

just as you needed to open the database connection. The commands are similar:

```
Set objRec = Server.CreateObject ("ADODB.Recordset")
objRec.Open "downloadable", strConnect, 0,1,2
```

This creates a recordset (objRec) of the table named *downloadable* that sits in the *products* database defined in strConnect. With the Recordset open, we can loop through the table and write all the contents to the screen. Or, we can test for specific contents and only write the data that matches our criteria to the screen.

Writing the Recordset contents

It's really easy to work with a Recordset. If you wanted to loop through the database and print all the information to the screen, you could do this:

```
  While NOT objRec.EOF
' says to do this as long as we haven't reached the end of the file
  Response.Write objRec("ProductID") & ", "
  Response.Write objRec("SKU") & ", "
  Response.Write objRec("Name") & ", "
  Response.Write objRec("File") & "<BR>"
    objRec.MoveNext
     Wend
```

Even if you haven't used a loop like this before, you can probably tell by looking that this prints the information in a comma-delimited string and starts a new row for each new row in the table. You could use this same technique to write the data to an HTML table. Just insert your TABLE tags appropriately using Response.Write and keep a few things in mind:

1. Your HTML tags and content go in quote marks.
2. If your tag or content uses quote marks, double them: <FONT SIZE=""+2"">.
3. Concatenate the variables and HTML/content information with ampersands

Picking and choosing within the Recordset

Imagine our products database also contains a field called *OS* for, you guessed it, a platform delimiter. Let's also imagine that the data stored in that field can only be one of the following values: Windows NT, Windows 95, Windows 98, Windows, Mac, Unix, or Linux. We then could specify which records we wanted to write to the screen and ignore the others. Or, we could choose to format some results one way and others another—in different colors, for example.

A simple If...Then loop will quickly give us more control over the database. Let's first print out the records that are tagged as Windows NT products:

```
< TABLE BORDER=0 WIDTH=600>

<TR><TD COLSPAN=4 ALIGN=CENTER><FONT
SIZE="+1"<<B>Windows NT Products</B></FONT></TD></TR>

<%
  While NOT objRec.EOF

  If objRec("OS") = "Windows NT" THEN ' specifies the criteria

Response.Write "<TR><TD BGCOLOR=""#FFFF66"">" &
objRec("ProductID") & "</TD>"
Response.Write "<TD>" & objRec("SKU") & "</TD>"
Response.Write "<TD>" & objRec("Name") & "</TD>"
Response.Write "<TD>" & objRec("File") & "</TD></TR>"

  end if
  objRec.MoveNext
  Wend

%>
< /TABLE>
```

Adding a record

Once you start working with the recordset and ASP, you'll soon be itching to add to a database via the Web. Adding is important whether you want your site visitors to be able to add comments or if you want to be able to make administrative updates.

The following code opens the recordset for a table that holds a list of books and their authors. You've seen this before, but this time the last three specifications defining the cursor type are different: adOpenStatic, adLockOptimistic, adCmdTable:

```
< %   ' database connection already made; code not shown here
Set objRec = Server.CreateObject ("ADODB.Recordset")
  objRec.Open "books", bookdb, adOpenStatic, adLockOptimistic,
adCmdTable
%>
```

(If you are not using a copy of adovbs.inc, the line would be: objRec.Open "books", bookdb, 3,3,2).

The recordset is now ready to accept data. You just need to tell it what to add. In this case, let's assume we have pulled variables from a form: strBookTitle and strBookAuthor. Our table, *books*, has two fields, called Title and Author, so we can add a new record by doing this:

```
< %
objRec.AddNew
```

```
  ObjRec("Title") = strBookTitle
  objRec("Author") = strBookAuthor
  objRec.Update
%>
```

strBookTitle and strBookAuthor represent values, possibly entered by a user in a form. If you just want to test the add function, you can slot in a title and author in place of the variables—just remember to use quote marks. The first few times you try it, you'll probably want to open your database immediately to ensure that the update happened.

Recordset types

In the sample objRec.Open statement shown, you'll see the 0,1,2 at the end. These numbers stand for various cursor types. The *type* you use depends on what you are going to do. For example, if you're not going to add or edit any records, you use one *Lock* type. You'll use a different type when you plan to make changes or updates to the database.

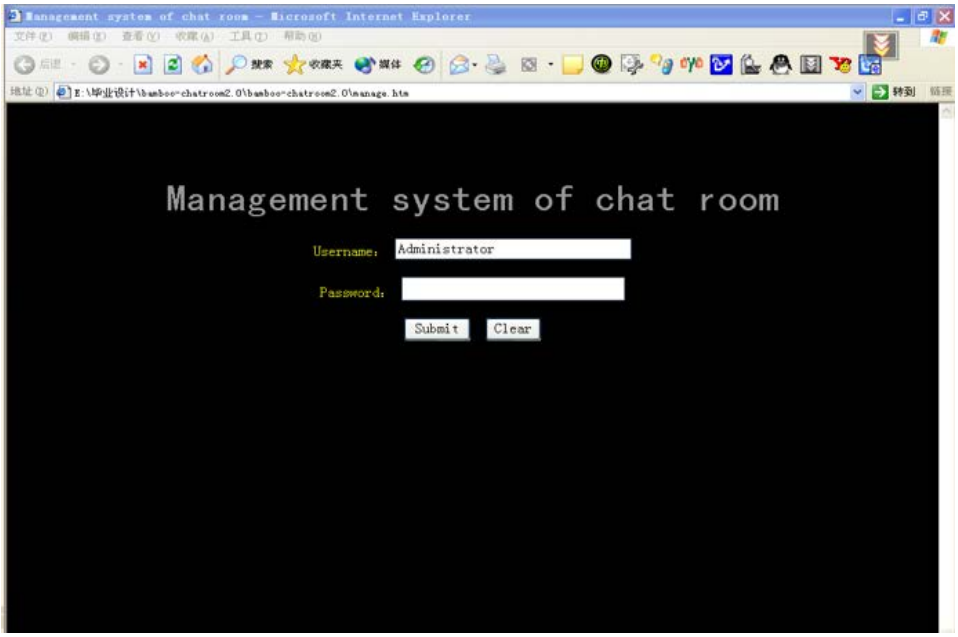
0,1,2 actually stands for:

adOpenForwardOnly, adLockReadOnly, adCmdTable

You can use the words rather than the numbers if you have a copy of adovbs.inc stored on your server. It contains a list of these constants and many others.

5.8 EXAMPLE

A management system of a small chat room: In this management system of the chat room, files are used to record the data. This management system is used to kick out some users.



source code

```
<%Response.Expires=0

manage=Session("hqtchatmanage")

if manage<>"Administrator" then Response.Write “visit declined" :
Response.end

username=Request.QueryString("username")

dim online()

onfile=server.mappath("online.asp")    get the path of the online.asp

Set fs=CreateObject("Scripting.FileSystemObject")

establish the “Filesystemobject” objection,   and set the fs as the
variable

Set thisfile = fs.OpenTextFile(onfile,1,False)

counton = 0

do while not thisfile.AtEndOfStream

thisline = thisfile.readline        Get a line of data from the file

Redim preserve online(counton)   define an array

online(counton) = thisline

counton = counton + 1

loop

thisfile.Close

Set outfile = fs.CreateTextFile(onfile)

for i=0 to counton-1 step 3

if online(i)<>username then

If the user are not deleted

outfile.WriteLine online(i)output all of the data of this user, and rewrite
the

outfile.WriteLine online(i+1)data to the online.asp. And replace the
original

outfile.WriteLine online(i+2)data ofonline.asp. So you can just see the
users

end if  except the name of this user who was kicked out!

next

outfile.CloseClose this file
```

```
set fs=nothing
Response.Redirect "onlinelist.asp"
%>
```

This function is achieved all by files instead of the database. At first, establish a temporary file folder (outfile). Output the data of the users' information such as username, IP address and so on and store in this folder. Then write into the file (online.asp) that recorded the users' information orderly and replace the original data in this file.

5.9 LIMITATIONS OF ASP

- Many of the functions of ASP will perform often in a data driven website take several lines of repetitive code each time to perform.
- Sometimes, the code of ASP will become repetitive and is somewhat time consuming.
- ASP is the lack of a good development environment. ASP is extremely robust, working well with the Windows platform on many levels, but is often convoluted and finding good support documentation and code examples can be excruciating.
- And if you are not proficient in Visual Basic or programming, ASP is a very difficult language to learn.

Check Your Progress

- How ASP can interact with the database?
- Discuss about the limitations of ASP.

5.10 SUMMARY

ASP was designed as a faster and easier alternative to CGI scripting using Perl or C. It provides an easy-to-learn scripting interface (including native support for VBScript and JScript), along with a number of predefined objects that simplify many development tasks, such as maintaining user state and defining global variables within an application. ADO components can be used to perform additional functions, including accessing ODBC-compliant databases, and outputting data to text files. Java components and XML can be used to extend ASP scripts.

ASP technology is a server-side programming developed to enable dynamic web pages. With its build-in objects and ActiveX components, ASP developers can create a dynamic web page and make use of database access. Now that you have basic knowledge about ASP, it is better to have a hands-on experience, which is the best way to learn to learn how to write an ASP file. There are tons of ASP tutorials out there on the Web. You can also find some tutorials included at the end of this paper.

5.11 TERMINAL QUESTIONS

1. What is the basic difference between ASP and ASP.NET?
2. What are the Major Built- in object in ASP.Net?
3. Which is the parent class of the Web server control?
4. What are the advantages of the code-behind feature?
5. What is IIS? Why is it used?
6. What is actually returned from server to the browser when a browser requests an .aspx file and the file is displayed?
7. Which ASP.NET objects encapsulate the state of the client and the browser?
8. Which method is used to force all the validation controls to run?
9. Differentiate between client-side and server-side validations in Web pages.?
10. What is the difference between HTML and Web Server Control?

UNIT-6 ASP.NET INTRODUCTION & CONTROLS

Structure

6.0 INTRODUCTION

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites. It allows you to use a full-featured programming language such as C# or VB.NET to build web applications easily. This tutorial covers all the basic elements of ASP.NET that a beginner would require to get started.

This tutorial is prepared for the beginners to help them understand basic ASP.NET programming. After completing this tutorial, you will find yourself at a moderate level of expertise in ASP.NET programming from where you can take yourself to next levels.

Before proceeding with this tutorial, you should have a basic understanding of .NET programming language. As you are going to develop web-based applications using ASP.NET web application framework, it will be good if you have an understanding of other web technologies such as HTML, CSS, AJAX, etc.

We also need to understand about the Web application. A Web application consists of document and code pages in various formats. The simplest kind of document is a static HTML page, which contains information that will be formatted and displayed by a Web browser. An HTML page may also contain hyperlinks to other HTML pages. A

hyperlink (or just *link*) contains an address, or a Uniform Resource Locator (URL), specifying where the target document is located. The resulting combination of content and links is sometimes called *hypertext* and provides easy navigation to a vast amount of information on the World Wide Web.

6.1 OBJECVTIVES

This unit aims to achieve the following objectives:

- About ASP
- About ASP.NET
- Main features of ASP.NET
- Main challenges of ASP.NET
- Advantages of ASP.NET
- Web form architecture
- Basic syntax of ASP.NET
- A small program
- Role of different types of controls.
- Life cycle of ASP.Net page
- Life cycle events
- ADO.NET and its working

6.2 FUNDAMENTAL OF ASP.NET

ASP.NET is a programming framework developed by Microsoft for building powerful Web-based applications. While not a programming language, ASP.NET is the cornerstone of the .NET platform's Internet-centric orientation. It is the latest version of Microsoft's ASP, a powerful, server-based technology for creating dynamic Web pages. Being a core element in the .NET platform, all .NET languages utilize ASP.NET as their entry point for Internet development.

Perhaps the biggest difference between earlier versions of ASP and ASP.NET is the way in which code runs on the Web server. With ASP.NET, code is compiled into executable classes that are compiled to native code by the common language runtime layer of the .NET Framework. All the earlier versions of ASP supported only scripting languages, which were interpreted. Since ASP.NET pages are now executable classes, they have access to all the other powerful features of the common language runtime, such as memory management, debugging, security, and strong data typing.

ASP.NET has built-in support for three languages: Visual Basic .NET, C#, and JScript.NET. The user can install support for other .NET-compatible languages as well. JScript.NET is Microsoft's latest version and implementation of JavaScript. Although this version still maintains its "scripting" feel, it is fully object-oriented and compiles to MSIL.

Because all ASP.NET code must be written in a .NET language, the ASP.NET programming framework is the foundation of Web services and Web Forms, the two components of ASP.NET Application Services.

The following list illustrates how developers can exploit ASP.NET in developing Internet-centric applications:

- XML Web services give developers the ability to access server functionality remotely. Using Web services, organizations can share their data and component libraries with other applications. Web services enable client/server and server/server communication through the HTTP protocol by using XML. Consequently, programs written in any language, using a component model and running on any operating system, can access Web services.
- ASP.NET Web Forms gives the developer the ability to create Web pages on the .NET platform. Web Forms enables developers to program embedded controls on either the client or the server. Using ASP Server Controls, Web applications enable the easy maintenance of state as information is communicated between the browser and the Web server.
- ASP.NET and the .NET Framework provide default authorization and authentication schemes for Web applications.
- Developers can easily remove, add to, or replace these schemes, depending on the needs of the application.
- Simple ASP pages can be migrated easily to ASP.NET applications. ASP.NET offers complete syntax and processing compatibility with ASP applications. Developers simply need to change the file extensions from .asp to .aspx to migrate their files to the ASP.NET page framework.

6.2.1 CLASS LIBRARY

ASP.NET includes an enormous class library which was built by Microsoft. Because this class library is so large, it encapsulates a huge number of common functions. For example, if you want to retrieve data from a database and display that data in a simple grid control through classic ASP, then you would have to write a lot of code.

In ASP.NET, we don't write any code to display the data, we just write the code to bind the data to an object called a DataGrid (which can be done in just a couple of lines). Then, we just have to create a reference on our page to where that DataGrid should go. The DataGrid will be

rendered as a table and will contain all of the data extracted from the database.

Microsoft has created an amazingly well designed MSDN library for ASP.NET and all of the other .NET languages. It includes full class library containing information and examples on every class, function, method and property accessible through ASP.NET.

The MSDN library also includes some tutorials and examples to get us started. It may take us a while to get used to the format and layout of the MSDN ASP.NET library, however, once we do, we will find that it is an invaluable resource to aid us throughout our ASP.NET learning experience. The .NET MSDN library can be found at *http://msdn.microsoft.com/net/*

6.2.2 COMPLETE COMPATIBILITY

One of the most important goals of .NET was to allow developers to write an ASP.NET application using multiple programming languages. As long as each ASP.NET page contains only one programming language, we can mix and match different pages using different languages and they will work together seamlessly. This means that we can now have a team of developers with half programming in C# and the other half in VB.NET, with no need to worry about language incompatibilities, etc. A cool little side-effect of all this is that all the programming languages look very similar, and differ only by their language syntax.

For example, take the following code snippets. They both do exactly the same thing but the first is written in C# and the second in VB.NET.

The C# version:

```
void Page_Load (object S, EventArgs E) { myLabel.Text=”Hello world!!” ;}

</script>
```

The VB.NET version:

```
Sub Page_Load(S As Object, E As EventArgs) myLabel.Text= “Hello world!!”

End Sub

</script>
```

6.2.3 PROGRAMMING BASICS

Page Syntax

- The most basic page is just static text. Any HTML page can be renamed .aspx

- Pages may contain:
  i) Directives: <%@ Page Language="C#" %>
  ii) Server controls: <asp:Button runat="server">
  iii) Code blocks: <script runat="server">…</script>
  iv) Data bind expressions: <%# %>
  v) Server side comments: <%-- --%>
  vi) Render code: <%= %> and <% %>, Use is discouraged; use <script runat=server> with code in event handlers instead

The Page Directive

Lets you specify page-specific attributes, e.g.

- Asp Compat: Compatibility with ASP
- Buffer: Controls page output buffering
- CodePage: Code page for this .aspx page
- ContentType: MIME type of the response
- ErrorPage: URL if unhandled error occurs
- Inherits: Base class of Page object
- Language: Programming language
- Trace: Enables tracing for this page
- Transaction: COM+ transaction setting

Note: Only one page directive per .aspx file

Page Events

Pages are structured using events

- Enables clean code organization
- Avoids the "Monster IF" statement
- Less complex than ASP pages

Code can respond to page events

- E.g. Page_Load, Page_Unload

Code can respond to control events

- Button1_Click
- Textbox1_Changed

6.2.4 FIRST PROGRAM

Example: HelloWorld.aspx

```
<html>
<head><title>HelloWorld.asp</title></head>
<body>
<form method=“post">
<input type="submit" id=button1 name=button1
value="Push Me" />
<%
if (Request.Form("button1") <> "") then
Response.Write "<p>Hello, the time is "&Now()
end if
%>
</form>
</body>
</html>
```

6.3 ASP.NET KEY FEATURES

Following are the main features of ASP.NET:

1. Web Forms
2. Web Services
3. Built on .NET Framework
4. Simple programming
5. model
6. Maintains page state
7. Multibrowser support
8. XCOPY deployment
9. XML configuration
10. Complete object model
11. Session management
12. Caching
13. Debugging
14. Extensibility

15. Separation of code and UI
16. Security
17. Simplified form validation
18. Cookieless sessions
19. Bundling and magnification feature
20. Strongly typed data controls
21. Model binding- isolation of web form from Model
22. Value providers
23. Support for OpenID in OAuth Logins
24. Support for improved paging in ASP.NET 4.5 GridView control
25. Enhanced support for asynchronous programming
26. Support for web sockets
27. Support for HTML5 form types
28. ASP.NET Web API

6.4 ADVANTAGES OF ASP.NET

ASP.NET has many advantages over other platforms when it comes to creating Web applications. Probably the most significant advantage is its integration with the Windows server and programming tools. Web applications created with ASP.NET are easier to create, debug, and deploy because those tasks can all be performed within a single development environment—Visual Studio .NET.

ASP.NET delivers the following other advantages to Web application developers:

- Executable portions of a Web application compiled so they execute more quickly than interpreted scripts.
- On-the-fly updates of deployed Web applications without restarting the server.
- Access to the .NET Framework, which extends the Windows API.
- Use of the widely known Visual Basic programming language, which has been enhanced to fully support object-oriented programming.
- Introduction of the new Visual C# programming language, which provides a type-safe, object-oriented version of the C programming language.

- Automatic state management for controls on a Web page (called server controls) so that they behave much more like Windows controls.
- The ability to create new, customized server controls from existing controls.
- Built-in security through the Windows server or through other authentication/authorization methods.
- Integration with Microsoft ADO.NET to provide database access and database design tools from within Visual Studio .NET.
- Full support for Extensible Markup Language (XML), cascading style sheets (CSS), and other new and established Web standards.
- Built-in features for caching frequently requested Web pages on the server, localizing content for specific languages and cultures, and detecting browser capabilities.

Check Your Progress

- Write a program for HelloWorld.
- What are the advantages of ASP.NET?

6.5 ASP CHALLENGES

- Coding overhead (too much code)- Everything requires writing code!
- Code readability (too complex; code and UI intermingled)
- Maintaining page state [After submit button is clicked, if we click the back button, we expect to maintain scroll position, maintain which control had focus, and restore focus, or allow server code to focus a new control] requires more code
- Session state scalability and availability
- Limited support for caching, tracing, debugging, etc.
- Performance and safety limitations of script

6.6 CLIENT-SIDE SCRIPTING AND SERVER-SIDE SCRIPTING

Client-side refers to the browsers and the machine running the browser. Server-side on the other hand refers to a Web server.

6.6.1 CLIENT-SIDE SCRIPTING

JavaScript and VBScript are generally used for Client-Side scripting. Client-Side scripting executes in the browser after the page is loaded. Using Client-Side scripting, we can add cool features to our page. Both, HTML and the script are together in the same file and the script is downloading as part of the page which anyone can view. A Client-Side script runs only on a browser that supports scripting and specifically the scripting language that is used. Since the script is in the file as the HTML and as it executes on the machinery we use, the page may rake longer time to download.

6.6.2 SERVER-SIDE SCRIPTING

ASP.NET is purely server-side technology. ASP.NET code executes on the server before it is sent to the browser. The code that is sent back to the browser is pure HTML and not ASP.NET code. Like client-side scripting, ASP.NET code is similar in a way that it allows us to write our code alongside HTML. Unlike client-side scripting, ASP.NET code is executed on the server and not in the browser. The script that we write alongside our HTML is not sent back to the browser and that prevents others from stealing the code we developed.

6.7 WEB FORM ARCHITECTURE

A Web Form consists of two parts:

- The visual content or presentation, typically specified by HTML elements.
- Code that contains the logic for interacting with the visual elements.

A Web Form is physically expressed by a file with the extension **.aspx**. Any HTML page could be renamed to have this extension and could be accessed using the new extension with identical results to the original. Thus Web Forms are upwardly compatible with HTML pages.

The way code can be separated from the form is what makes a Web Form special. This code can be either in a separate file (having an extension corresponding to a .NET language, such as **.vb** for VB.NET) or in the **.aspx** file, within a <SCRIPT RUNAT="SERVER"> ... /SCRIPT> block. When your page is run in the Web server, the user interface code runs and dynamically generates the output for the page.

We can understand the architecture of a Web Form most clearly by looking at the code-behind version of our "Echo" example. The visual content is specified by the **.aspx** file.

HelloCodebehind.aspx.

```
<!-- HelloCodebehind.aspx -->
```

```
<%@ Page Language="VB#" Src="HelloCodebehind.aspx.vb"
Inherits= MyWebPage %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM RUNAT="SERVER">YOUR NAME: 
<asp:textbox id=txtName Runat="server"></asp:textbox>
<p><asp:button id=cmdEcho onclick=cmdEcho_Click Text="Echo"
runat="server" tooltip="Click to echo your name">
</asp:button></p>
<asp:label id=lblGreeting runat="server"></asp:label>
<P></P>
</FORM>
</BODY>
</HTML>
```

The user interface code is in the file **HelloCodebehind.aspx.vb**,

'

HelloCodebehind.aspx.vb

```
Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
Public Class MyWebPage
Inherits System.Web.UI.Page
Protected txtName As TextBox
Protected cmdEcho As Button
Protected lblGreeting As Label
Protected Sub cmdEcho_Click(Source As Object, _
e As EventArgs)
lblGreeting.Text="Hello, "& txtName.Text
```

End Sub

End Class

6.7.1 PAGE CLASS

The key namespace for Web Forms and Web services is **System.Web**. Support for Web Forms is in the namespace **System.Web.UI**. Support for server controls such as textboxes and buttons is in the namespace **System. Web.UI.WebControls**. The class that dynamically generates the output for an **.aspx** page is the **Page** class, in the **System.Web.UI** namespace, and classes derived from **Page**, as illustrated in the code-behind page in this last example.

Inheriting From Page Class

The elements in the **.aspx** file, the code in the code-behind file (or script block), and the base **Page** class work together to generate the page output. This cooperation is achieved by ASP.NET's dynamically creating a class for the **.aspx** file, which is derived from the code-behind class, which in turn is derived from **Page**. This relationship is created by the Inherits attribute in the .aspx file. Here **MyWebPage** is a class we implement, derived from **Page**. The most derived page class, shown as *My .aspx Page*, is dynamically created by the ASP.NET runtime. This class extends the page class, shown as *MyWebPage* in the figure, to incorporate the controls and HTML text on the Web Form. This class is compiled into an executable, which is run when the page is requested from a browser. The executable code creates the HTML that is sent to the browser.

6.7.2 WEB FORMS PAGE LIFE CYCLE

We can get a good high-level understanding of the Web Forms architecture by following the life cycle of our simple Echo application. We will use the code behind version (the example), **HelloCodebehind.aspx**.

1. User requests the **HelloCodebehind.aspx** Web page in the browser.

2. Web server compiles the page class from the **.aspx** file and its associated code-behind page. The Web server executes the code, creating HTML, which is sent to the browser. (In Internet Explorer you can see the HTML code from the menu View | Source.) Note that the server controls are replaced by straight HTML. The following code is what arrives at the browser, *not the original code on the server*.

   <!-- HelloCodebehind.aspx -->

   <HTML>

   <HEAD>

   </HEAD>

<BODY>

<form name="ctrl0" method="post"
action="HelloCodebehind.aspx" id="ctrl0">

<input type="hidden" name="__VIEWSTATE"
value="dDwxMzc4MDMwNTk1Ozs+" />YOUR NAME:
<input name="txtName" type="text" id="txtName" />

<input type="submit" name="cmdEcho" value="Echo"
id="cmdEcho" title="Click to echo your name" />

<P></P>

</form>

</BODY>

</HTML>

3. The browser renders the HTML, displaying the simple form shown in Figure. To distinguish this example from the first one, we show "YOUR NAME" in all capitals. Since this is the first time the form is displayed, the text box is empty, and no greeting message is displayed.

4. The user types in a name (e.g., Mary Smith) and clicks the Echo button. The browser recognizes that a Submit button has been clicked. The method for the form is POST1 and the action is HelloCodebehind.aspx. We thus have what is called a *postback* to the original **.aspx** file.

5. The server now performs processing for this page. An event was raised when the user clicked the Echo button, and an event handler in the **MyWebPage** class is invoked.

6. The **Text** property of the **TextBox** server control **txtName** is used to read the name submitted by the user. A greeting string is composed and assigned to the **Label** control **lblGreeting**, again using property notation.

7. The server again generates straight HTML for the server controls and sends the whole response to the browser. Here is the HTML.

....

<form name="ctrl0" method="post"
action="HelloCodebehind.aspx" id="ctrl0">

<input type="hidden" name="__VIEWSTATE"
value="dDwxMzc4MDMwNTk1O3Q8O2w8aTwyPjs+O2w8dDw
7bDxpPDU+Oz47

bDx0PHA8cDxsPFRleHQ7PjtsPEhlbGxvLCBNYXJ5IFNtaXRoO
z4+Oz47Oz4 7Pj47Pj47Pg==" />

YOUR NAME: <input name="txtName" type="text"
value="KRISHAN KUMAR" id="txtName" />

<input type="submit" name="cmdEcho" value="Echo"
id="cmdEcho" title="Click to echo your name" /> <span
id="lblGreeting">**Hello, KRISHAN KUMAR**

...

8. The browser renders the page, now a greeting message is displayed.

6.7.3 VIEW STATE

An important characteristic of Web Forms is that all information on forms is "remembered" by the Web server. Since HTTP is a stateless protocol, this preservation of state does not happen automatically but must be programmed. A nice feature of ASP.NET is that this state information, referred to as "view state," is preserved automatically by the framework, using a "hidden" control.

...

<input type="hidden" name="__VIEWSTATE"

value="dDwxMzc4MDMwNTk1O3Q8O2w8aTwyPjs+O2w8dDw7bDxp
PDU+Oz47

bDx0PHA8cDxsPFRleHQ7PjtsPEhlbGxvLCBNYXJ5IFNtaXRoOz4+Oz
47Oz4

7Pj47Pj47Pg==" />

...

6.7.4 WEB FORMS EVENT MODEL

From the standpoint of the programmer, the event model for Web Forms is very similar to the event model for Windows Forms. Indeed, this similarity is what makes programming with Web Forms so easy. What is actually happening in the case of Web Forms, though, is rather different. The big difference is that events get raised on the client and processed on the server.

Our simple form with one textbox and one button is not rich enough to illustrate event processing very thoroughly. Let's imagine a more elaborate form with several textboxes, listboxes, checkboxes, buttons, and the like. Because round trips to the server are expensive, events do not automatically cause a postback to the server. Server controls have what is known as an intrinsic event set of events that automatically

cause a postback to the server. The most common such intrinsic event is a button click. Other events, such as selecting an item in a list box, do not cause an immediate postback to the server. Instead, these events are cached, until a button click causes a post to the server. Then, on the server the various change events are processed, in no particular order, and the button-click event that caused the post is processed.

6.7.5 PAGE PROCESSING

Processing a page is a cooperative endeavour between the Web server, the ASP.NET runtime, and your own code. The **Page** class provides a number of events, which you can handle to hook into page processing. The **Page** class also has properties and methods that you can use. We cover some of the major ones here. For a complete description, consult the .NET Framework documentation. The example programs in this chapter will illustrate features of the **Page** class.

Page Events

A number of events are raised on the server as part of the normal processing of a page. These events are actually defined in the **Control** base class and so are available to server controls also. The most important ones are listed below.

- *Init*is the first step in the page's life cycle and occurs when the page is initialized. There is no view-state information for any of the controls at this point.
- *Load*occurs when the controls are loaded into the page. View-state information for the controls is now available.
- *PreRender*occurs just before the controls are rendered to the output stream. Normally this event is not handled by a page but is important for implementing your own server controls.
- *Unload*occurs when the controls are unloaded from the page. At this point it is too late to write your own data to the output stream.

Page Properties

The **Page** class has a number of important properties. Some of the most useful are listed below.

- *EnableViewState*indicates whether the page maintains view state for itself and its controls. You can get or set this property. The default is **true**, view state is maintained.
- *ErrorPage*specifies the error page to which the browser should be redirected in case an unhandled exception occurs.
- *IsPostBack*indicates whether the page is being loaded in response to a postback from the client or is being loaded for the first time.

- *IsValid*indicates whether page validation succeeded.
- *Request*gets the HTTP Request object, which allows you to access data from incoming HTTP requests.
- *Response*gets the HTTP Response object, which allows you to send response data to a browser.
- *Session*gets the current Session object, which is provided by ASP.NET for storing session state.

6.8 ADO.NET CONNECTIVITY

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET is a set of libraries included in the Microsoft .NET Framework that help us to communicate with various data stores from .NET applications. The ADO.NET libraries include classes for connecting to a data source, submitting queries, and processing results. We can also use ADO.NET as a robust, hierarchical, disconnected data cache to work with data offline. The central disconnected object, the DataSet, allows us to sort, search, filter, store pending changes, and navigates through hierarchical data. The DataSet also includes a number of features that bridge the gap between traditional data access and XML development. Developers can now work with XML data through traditional data access interfaces and vice-versa.

Some objects of ADO.NET are:-

- **Connections :** For connection and managing transactions against a database.
- **Commands :** For issuing SQL commands against a database.
- **DataReaders :** For reading a forward-only stream of data records from a SQL Server data source.
- **DataSets :** For storing, remoting and programming against flat data, XML data and relational data.
- **DataAdapters :** For pushing data into a DataSet, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server.NET Data Provider (System.Data.SqlClient) and OLE DB.NET Data Provider (System.Data.OleDb). We will use the SQL Server.NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB.NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

6.8.1 CONNECTIONS

A Connection object represents a connection to data source. For SQL Server we use the name space System Data. SQL Client. Sql Connection and for OLE DB we use the System. Data. OleDb. OleDb Connection. We can specify the type of data source, its location, and other attributes through the various properties of the Connection object. A Connection object is roughly equivalent to an ADO Connection object; we use it to connect to and disconnect from our database. A Connection object acts as a conduit through which other objects, such as a DataAdapter and Command objects, communicate with our database to submit queries and retrieve results.

6.8.2 COMMANDS

The process of interacting with a database means that we must specify the actions we want from database to occur. This is done with a command object. Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as SQLCommand. For SQL Server we use the namespace System.Data.SQLClient.SqlCommand and for OLE DB we use the namespace System.Data.OleDb.OleDbCommand. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. We can also use input and output parameters, and return values as part of our command syntax.

6.8.3 DATAREADERS

The DataReader object is designed to help us retrieve and examine the rows returned by our query as quickly as possible. We can use the DataReader object to examine the results of a query one row at a time. When we move forward to the next row, the contents of the previous row are discarded. The DataReader doesn't support updating. The data returned by the DataReader is read-only. Because the DataReader object supports such a minimal set of features, it's extremely fast and lightweight. The disadvantage of using a DataReader object is that it requires an open database connection and increases network activity. The DataReader provides a non-buffered stream of data that allows procedural logic to efficiently process results from a data source sequentially. The DataReader is a good choice when retrieving large amount of data; only one row of data will be cached in memory at a time.

6.8.4 DATASETS

A DataSet object, as its name indicates, contains a set of data. It is a container for a number of DataTable objects (stored in the DataSet object's Tables collection). ADO.NET was created to help developers build large multi-tiered database applications. At times, we might want to

access a component running on a middle-tier server to retrieve the contents of many tables. Rather than having to repeatedly call the server in order to fetch that data one table at a time, we can package all of the data into a DataSet object and return it in a single call. However, a DataSet object does a great deal more than act as a container for multiple DataTable objects.

The DataSet object has features that allow you to write it to and read it from a file or and area of memory. We can save just the contents of the DataSet object, just the structure of the DataSet object, or both. ADO.NET stores this data as an XML document. Because ADO.NET and XML are so tightly coupled, moving data back and forth between ADO.NET DataSet objects and XML documents is easy.

6.8.5 DATA ADAPTERS

The DataAdapter object represents a new concept for Microsoft data access models. It acts as a bridge between a database and the disconnected objects in the ADO.NET object model. The DataAdapter object's Fill method provides an efficient mechanism to fetch the results of a query into a DataSet or a DataTable so that we can work with our data off-line. We can also use DataAdapter objects to submit the pending changes stored in your DataSet objects to our database.

The ADO.NET DataAdapter object exposes a number of properties that are actually Command objects. For instance, the SelectCommand property contains a Command object that represents the query that we'll use to populate our DataSet object. The DataAdapter object also has UpdateCommand, InsertCommand and DeleteCommand properties that correspond to Command objects used when we submit modified, new, or deleted rows to our database, respectively. We can set the UpdateCommand, InsertCommand, and DeleteCommand properties to call stored procedures or a SQL statement. Then we can simply call the Update method on the DataAdapter object and ADO.NET will use the Command objects which we have created to submit the cached changes in DataSet to our database.

6.9 SUMMARY

ASP.NET is a unified Web development platform that greatly simplifies the implementation of sophisticated Web applications. In this unit we introduced the fundamentals of ASP.NET and Web Forms, which make it easy to develop interactive Web sites. Server controls present the programmer with an event model similar to what is provided by controls in ordinary Windows programming. This high-level programming model rests on a lower-level request/response programming model that is common to earlier approaches to Web programming and is still accessible to the ASP.NET programmer.

The Visual Studio .NET development environment includes a Form Designer, which makes it very easy to visually lay out Web forms, and with a click you can add event handlers. ASP.NET makes it very easy to handle state management. Configuration is based on XML files and is very flexible. There are a great variety of server controls, including wrappers around HTML controls, validation controls, and rich controls such as a Calendar. Data binding makes it easy to display data from a variety of data sources.

JavaScript and VBScript are generally used for Client-Side scripting. Client-Side scripting executes in the browser after the page is loaded. Using Client-Side scripting, we can add cool features to our page. Both, HTML and the script are together in the same file and the script is downloading as part of the page which anyone can view. A Client-Side script runs only on a browser that supports scripting and specifically the scripting language that is used. Since the script is in the file as the HTML and as it executes on the machinery we use, the page may rake longer time to download.

ASP.NET is purely server-side technology. ASP.NET code executes on the server before it is sent to the browser. The code that is sent back to the browser is pure HTML and not ASP.NET code. Like client-side scripting, ASP.NET code is similar in a way that it allows us to write our code alongside HTML.

Unlike client-side scripting, ASP.NET code is executed on the server and not in the browser. The script that we write alongside our HTML is not sent back to the browser and that prevents others from stealing the code we developed.

A Web Form is physically expressed by a file with the extension **.aspx**. Any HTML page could be renamed to have this extension and could be accessed using the new extension with identical results to the original. Thus Web Forms are upwardly compatible with HTML pages.

The way code can be separated from the form is what makes a Web Form special. This code can be either in a separate file (having an extension corresponding to a .NET language, such as **.vb** for VB.NET) or in the **.aspx** file, within a <SCRIPT RUNAT="SERVER"> ... /SCRIPT> block. When your page is run in the Web server, the user interface code runs and dynamically generates the output for the page.

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

Check Your Progress

- Give the meaning of ADO.NET.
- What is page life cycle in Web form architecture?

6.10 TERMINAL QUESTIONS

1. Define the term ASP and its features.
2. What do you understand by ASP.NET challenges? Explain.
3. Write programming fundamental of ASP.NET.
4. What is page life cycle in Web form architecture? Explain.
5. Define .NET framework.
6. Give the advantages of ASP.NET.
7. Write a short note on Web form architecture.
8. What do you understand by ADO.NET? Explain in detail.

Uttar Pradesh Rajarshi Tandon Open University

Bachelor in Computer Application

BCA-E10

Client Server Technology

BLOCK

3

INTRODUCTION TO ASP.NET

UNIT-7

Working With Forms and Controls

UNIT-8

ADO.NET

UNIT-9

ASP.NET State Management

UNIT-10

Configuration

Course Design Committee

Dr. Ashutosh Gupta, **Chairman**
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Prof. R.S. Yadav **Member**
Dept. of Computer Science and Engineering, MNNIT, Allahabad

Ms. Marisha **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Course Preparation Committee

Dr. Krishan Kumar **Author**
Assistant Professor,
Department of Comuter Science Faculty of Technology
Gurukula Kangri Vishwavidyalaya, Haridwar (UK)

Dr. V.K. Saraswat **Editor**
Director (IET, Khandare Campus)
Institute of Engineering and Technology
Dr. B.R. Ambedkar University, Agra-282002

Dr. Ashutosh Gupta,
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

©UPRTOU, Prayagraj-2020
ISBN : 978-93-83328-13-0

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2020.
Printed By : Chandrakala Universal Pvt. 42/7 Jawahar Lal Neharu Road, Prayagraj.

BLOCK INTRODUCTION

Block-3 basically contains four units which are intended with client-server technology and its design applications. The entire block revolves around the knowledge of ASP.NET framework and its related components like from and controls, state management, and configuration. Hence it has been broken into four units.

Unit-7 aims to describe the basics of Web forms and controls. Some controls like button, text field, labels etc. are the common controls. This unit deals with the windows forms and controls. As windows forms and controls are important for any programming language. Theses working control and forms make programming easy and useful. This also gives birth to event driven programming. As we know for every click an event is generated for which you need to write the code of different procedures. Moreover, Forms are used to create (rather primitive) GUIs on Web pages. Usually the main purpose is to ask the user for information. The information is then sent back to the server.

Unit-8 deals with the database connectivity. Controls like ADO and data controls are the important controls used as the basic elements between front end and back end. Connection pooling is another advance concept which is being widely used in many patterns. ADO.NET is a large set of .NET classes that enable us to retrieve and manipulate data, and update data sources, in very many different ways. As an integral part of the .NET framework, it shares many of its features: features such as multi-language support, garbage collection, just-in-time compilation, object-oriented design, and dynamic caching, and is far more than an upgrade of previous versions of ADO. ADO.NET is set to become a core component of any data-driven .NET application or Web Service, and understanding its power will be essential to anyone wishing to utilize .NET data support to maximum effect.

Unit-9 handles the session of different types. Session management is very important not only for time saving as well as for security. Assemblies and its types are explained. As a consequence, Web programmers must be very conscious about state management. Unlike traditional applications, Web applications must be very explicit about any state that is maintained on behalf of a client, and there is no one standard way to maintain that state. Moreover, remember that ASP.NET (and the Web) is stateless and the Web server does not keep track of past client requests. Furthermore, different technologies handle the issue of statement management differently like ASP.NET is somewhat unique in this regard but PHP works similarly.

Unit-10 aims to describe the Windows configuration, .net configuration, caching and its Types, SQL Cache Invalidation. Moreover, the configuration also plays important role for all software. Improper

configuration or installation of the software may lead to errors or sometimes it may be harmful to such extent that it may crash the hardware or system for which it intended. Hence it becomes necessary to know about the proper configuration or installation of the software. Normally when we purchase a computer or any other hardware like router, switch, printer etc; some configuration is automatically done and some need to be done manually. We are intended with the second case i.e. manual configuration. In this unit, Windows and .NET configuration has been discussed however there may me other configurations like Linux configuration etc.

UNIT-7 WORKING WITH FORMS AND CONTROLS

Structure

7.0 INTRODUCTION

This unit deals with the windows forms and controls. As windows forms and controls are important for any programming language. Theses working control and forms make programming easy and useful. This also gives birth to event driven programming. As we know for every click an event is generated for which you need to write the code of different procedures. Moreover, Forms are used to create (rather primitive) GUIs on Web pages. Usually the main purpose is to ask the user for information. The information is then sent back to the server.

A form is an area that can contain form elements. Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc. Other kinds of tags can be mixed in with the form elements. A form usually contains a Submit button to send the information in the form elements to the server. The form's parameters tell JavaScript how to send the information to the server (there are two different ways it could be sent). Forms can be used for other things, such as a GUI for simple programs.

7.1 OBJECVTIVES

This unit aims to achieve the following objectives:

- Definition of forms and its working.

- Multiple tags used in HTML to work with forms.
- Role of different types of controls.
- Life cycle of ASP.Net page
- Life cycle events
- Creation of web form
- Submitting the information through web from
- Creating a web based project
- Knowing the code behind the server

7.2 LIFE CYCLE OF ASP.NET PAGE

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. In other words, you can write your own code to override the default code. The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives, are part of this control tree. You can see the control tree by adding trace= "true" to the page directive. We will cover page directives and tracing under 'directives' and 'event handling'.

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

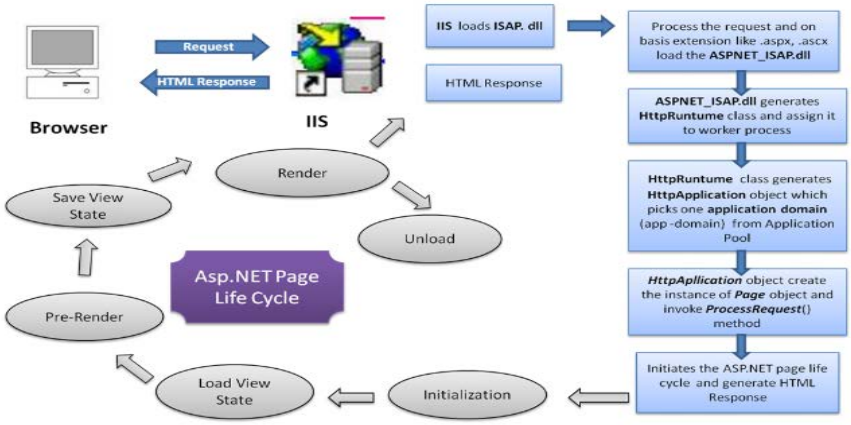


Figure 7.1: ASP.NET Page Life Cycle

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.

Following are the different stages of an ASP.NET page:

- **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.
- **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.
- **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.
- **Page load** - At this stage, control properties are set using the view state and control state values.
- **Validation** - Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
- **Postback event handling** - If the request is a postback (old request), the related event handler is invoked.
- **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.
- **Unload** - The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done

7.1.1 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. The IsCallbackand and IsCrossPagePostBack properties have also been set at this time**.**

- **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. The Init event of individual controls occurs before the Init event of the page.
- **InitComplete** - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- **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.
- **Load** - The 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.

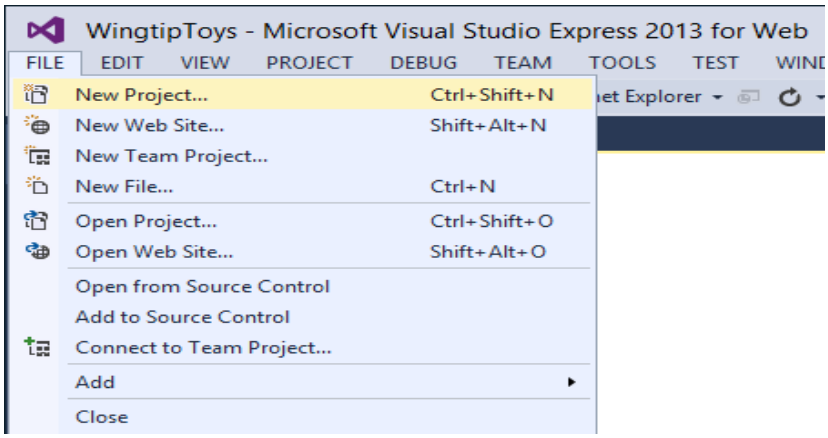
7.3 CREATING AN ASP.NET WEB APPLICATION PROJECT

ASP.NET provides an abstraction layer on top of HTTP on which the web applications are built. It provides high-level entities such as classes and components within an object-oriented paradigm. The key development tool for building ASP.NET applications and front ends is

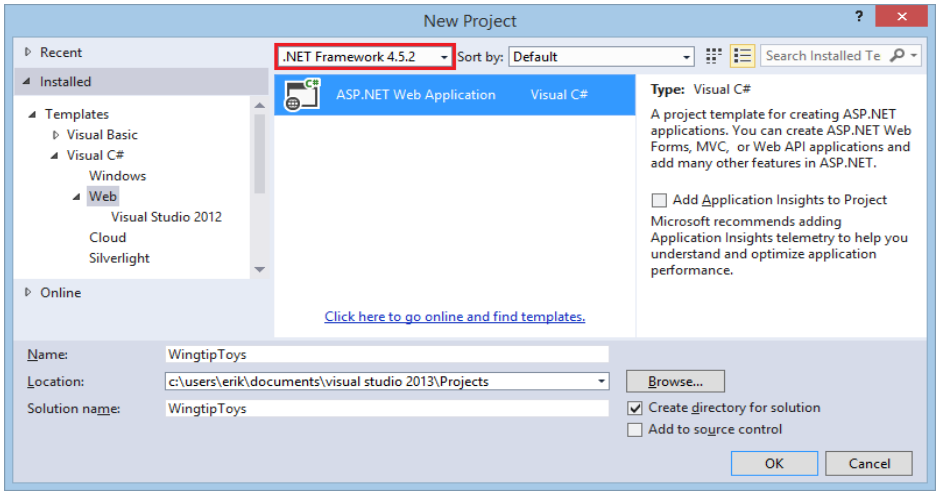
Visual Studio. Visual Studio is an integrated development environment for writing, compiling, and debugging the code. It provides a complete set of development tools for building ASP.NET web applications, web services, desktop applications, and mobile applications.

7.3.1 STEPS FOR CREATING A PROJECT

1. Open Visual Studio.
2. Select **New Project** from the **File** menu in Visual Studio.



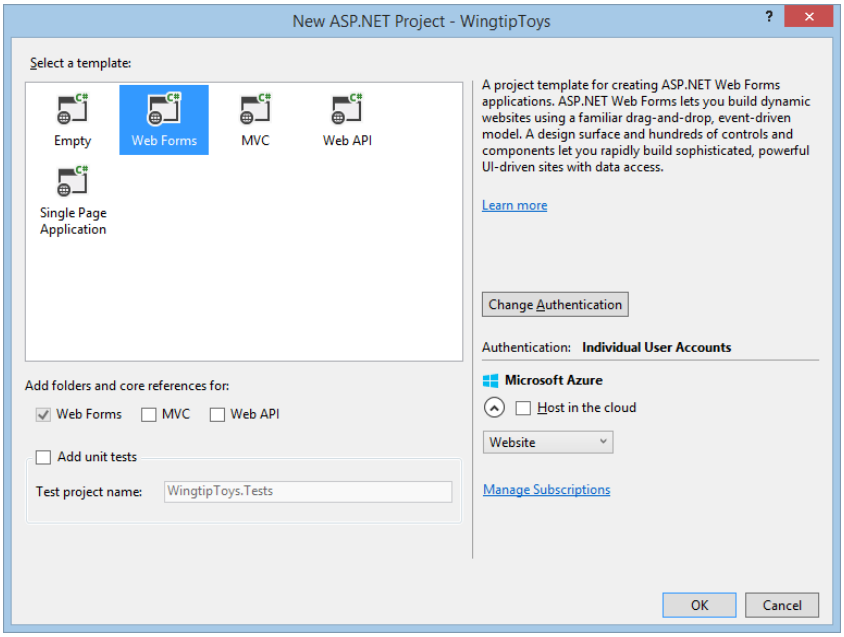
3. Select the **Templates** -> **Visual C#** -> **Web** templates group on the left.
4. Choose the **ASP.NET Web Application** template in the center column.
   This tutorial series is using .NET Framework 4.5.2.
5. Name your project *WingtipToys* and choose the **OK** button.



Note : The name of the project in this tutorial series is **WingtipToys**. It is recommended that you use this *exact* project

name so that the code provided throughout the tutorial series functions as expected.

6. Next, select the **Web Forms** template and choose the **Create Project** button.



The project will take a little time to create. When it's ready, open the **Default.aspx** page.

7.4 CREATING WEB FORMS

ASP.NET Web Forms is a part of the ASP.NET web application framework and is included with Visual Studio. It is one of the four programming models you can use to create ASP.NET web applications, the others are ASP.NET MVC, ASP.NET Web Pages, and ASP.NET Single Page Applications.

Web Forms are pages that your users request using their browser. These pages can be written using a combination of HTML, client-script, server controls, and server code. When users request a page, it is compiled and executed on the server by the framework, and then the framework generates the HTML markup that the browser can render. An ASP.NET Web Forms page presents information to the user in any browser or client device.

Using Visual Studio, you can create ASP.NET Web Forms. The Visual Studio Integrated Development Environment (IDE) lets you drag and drop server controls to lay out your Web Forms page. You can then easily set properties, methods, and events for controls on the page or for the page iteself. These properties, methods, and events are used to define the web page's behavior, look and feel, and so on. To write server code to

handle the logic for the page, you can use a .NET language like Visual Basic or C#.

7.4.1 ASP.NET WEB FORMS

- 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.

Check Your Progress

- What do you understand by a Web form?
- Give the steps to create a Web project.
- Write the important terms of ASP.Net page life cycle.

7.5 USING SERVER CONTROL & WEB SERVER CONTROLS

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, which are:

- 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] />

In addition, visual studio has the following features, to help produce in error-free coding:

- Dragging and dropping of controls in design view
- IntelliSense feature that displays and auto-completes the properties
- The properties window to set the property values directly

7.5.1 PROPERTIES OF THE SERVER CONTROLS

ASP.NET server controls with a visual aspect are derived from the WebControl class and inherit all the properties, events, and methods of this class. The WebControl class itself and some other server controls that are not visually rendered are derived from the System.Web.UI.Control class. For example, PlaceHolder control or XML control. 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.
CausesValidation	Indicates if it causes validation.
ChildControlCreated	It indicates whether the server control's child controls have been created.
ClientID	Control ID for HTML markup.
Context	The HttpContext object associated with the server control.
Controls	Collection of all controls contained within the control.
ControlStyle	The style of the Web server control.
CssClass	CSS class
DataItemContainer	Gets a reference to the naming container if the naming container implements IDataItemContainer.

DataKeysContainer	Gets a reference to the naming container if the naming container implements IDataKeysControl.
DesignMode	It indicates whether the control is being used on a design surface.
DisabledCssClass	Gets or sets the CSS class to apply to the rendered HTML element when the control is disabled.
Enabled	Indicates whether the control is grayed out.
EnableTheming	Indicates whether theming applies to the control.
EnableViewState	Indicates whether the view state of the control is maintained.
Events	Gets a list of event handler delegates for the control.
Font	Font.
Forecolor	Foreground color.
HasAttributes	Indicates whether the control has attributes set.
HasChildViewState	Indicates whether the current server control's child controls have any saved view-state settings.
Height	Height in pixels or %.
ID	Identifier for the control.
IsChildControlStateCleared	Indicates whether controls contained within this control have control state.
IsEnabled	Gets a value indicating whether the control is

	enabled.
IsTrackingViewState	It indicates whether the server control is saving changes to its view state.
IsViewStateEnabled	It indicates whether view state is enabled for this control.
LoadViewStateById	It indicates whether the control participates in loading its view state by ID instead of index.
Page	Page containing the control.
Parent	Parent control.
RenderingCompatibility	It specifies the ASP.NET version that the rendered HTML will be compatible with.
Site	The container that hosts the current control when rendered on a design surface.
SkinID	Gets or sets the skin to apply to the control.
Style	Gets a collection of text attributes that will be rendered as a style attribute on the outer tag of the Web server control.
TabIndex	Gets or sets the tab index of the Web server control.
TagKey	Gets the HtmlTextWriterTag value that corresponds to this Web server control.
TagName	Gets the name of the control tag.
TemplateControl	The template that contains this control.
TemplateSourceDirectory	Gets the virtual directory of the page or control containing this control.
ToolTip	Gets or sets the text displayed when the mouse pointer hovers over the web server

	control.
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

7.6 USING CODE BEHIND PAGES (HTML SEVER)

The HTML server controls are basically the standard HTML controls enhanced to enable server side processing. The HTML controls such as the header tags, anchor tags, and input elements are not processed by the server but are sent to the browser for display. They are specifically converted to a server control by adding the attribute runat="server" and adding an id attribute to make them available for server-side processing.

For example, consider the HTML input control:

<input type="text" size="40">

It could be converted to a server control, by adding the runat and id attribute:

<input type="text" id="testtext" size="40" runat="server">

7.6.1 ADVANTAGES OF USING HTML SERVER CONTROLS

Although ASP.NET server controls can perform every job accomplished by the HTML server controls, the later controls are useful in the following cases:

- Using static tables for layout purposes.
- Converting a HTML page to run under ASP.NET

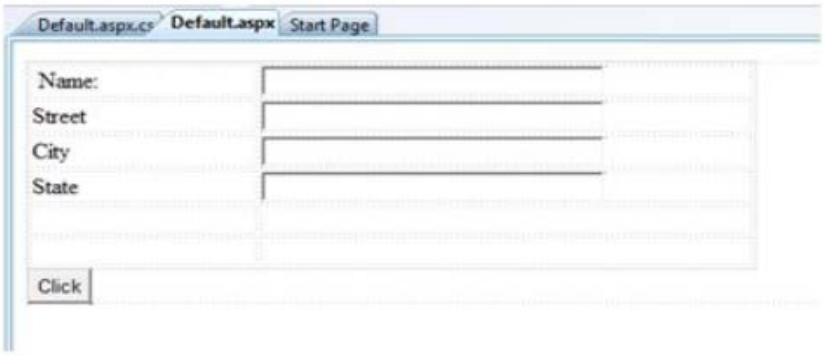
The following table describes the HTML server controls:

Control Name	**HTML tag**
HtmlHead	<head>element
HtmlInputButton	<input type=button|submit|reset>
HtmlInputCheckbox	<input type=checkbox>
HtmlInputFile	<input type = file>
HtmlInputHidden	<input type = hidden>
HtmlInputImage	<input type = image>
HtmlInputPassword	<input type = password>
HtmlInputRadioButton	<input type = radio>
HtmlInputReset	<input type = reset>
HtmlText	<input type = text|password>
HtmlImage	<img> element
HtmlLink	<link> element
HtmlAnchor	<a> element
HtmlButton	<button> element
HtmlButton	<button> element
HtmlForm	<form> element
HtmlTable	<table> element

HtmlTableCell	<td> and <th>
HtmlTableRow	<tr> element
HtmlTitle	<title> element
HtmlSelect	<select&t; element
HtmlGenericControl	All HTML controls not listed

7.6.2 EXAMPLE

The following example uses a basic HTML table for layout. It uses some boxes for getting input from the users such as name, address, city, state etc. It also has a button control, which is clicked to get the user data displayed in the last row of the table. The page should look like this in the design view:



The code for the content page shows the use of the HTML table element for layout.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="htmlserver._Default" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>
```

```
<style type="text/css">
.style1
{
width: 156px;
}
.style2
{
width: 332px;
}
</style>
</head>
<body>
<form id="form1" runat="server">
<div>
<table style="width: 54%;">
<tr>
<td class="style1">Name:</td>
<td class="style2">
<asp:TextBox ID="txtname" runat="server"  style="width:230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class="style1">Street</td>
<td class="style2">
<asp:TextBox ID="txtstreet" runat="server"  style="width:230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class="style1">City</td>
```

```
<td class="style2">
<asp:TextBox ID="txtcity" runat="server"  style="width:230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class="style1">State</td>
<td class="style2">
<asp:TextBox ID="txtstate" runat="server" style="width:230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class="style1"> </td>
<td class="style2"></td>
</tr>
<tr>
<td class="style1"></td>
<td ID="displayrow" runat ="server" class="style2">
</td>
</tr>
</table>
</div>
<asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Click" />
</form>
</body>
</html>
```

The code behind the button control:

```
protected void Button1_Click(object sender, EventArgs e)
{
```

```
string str = "";
str += txtname.Text + "<br />";
str += txtstreet.Text + "<br />";
str += txtcity.Text + "<br />";
str += txtstate.Text + "<br />";
displayrow.InnerHtml = str;
}
```

Note: The standard HTML tags have been used for the page layout. The last row of the HTML table is used for data display. It needed server side processing, so an ID attribute and the runat attribute has been added to it.

7.7 VALIDATION CONTROLS USAGE OF SKINS AND THEMES

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

7.7.1 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.

7.7.2 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>
```

7.7.3 RANGEVALIDATOR CONTROL

The Range Validator 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.

The syntax of the control is as given:

<asp:RangeValidator ID="rvclass" runat="server"
ControlToValidate="txtclass"

ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"

MinimumValue="6" Type="Integer">

</asp:RangeValidator>

7.7.4 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.

The basic syntax of the control is as follows:

<asp:CompareValidator ID="CompareValidator1" runat="server"

ErrorMessage="CompareValidator">

</asp:CompareValidator>

7.7.5 REGULAREXPRESSIONVALIDATOR

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property. The following table summarizes the commonly used syntax constructs for regular expressions:

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>

7.7.6 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>

7.7.7 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:" />

7.7.8 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.

Check Your Progress

- Define the terms skin and themes.
- What is HTML server? Explain

7.8 SUMMARY

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.

A form is an area that can contain form elements. Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc. Other kinds of tags can be mixed in with the form elements. A form usually contains a Submit button to send the information in the form elements to the server.

The different stages of an ASP.NET page are: Page request, Starting of page life cycle, Page initialization, Page load, Validation, Postback event handling, Page rendering, Unload.

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.

The main page life cycle events are: PreInit, Init, InitComplete, LoadViewState, LoadPostData, PreLoad, Load, LoadComplete, PreRender, PreRenderComplete, SaveStateComplete, UnLoad.

The HTML server controls are basically the standard HTML controls enhanced to enable server side processing. The HTML controls such as the header tags, anchor tags, and input elements are not processed by the server but are sent to the browser for display. They are specifically converted to a server control by adding the attribute runat="server" and adding an id attribute to make them available for server-side processing.

7.9 TERMINAL QUESTIONS

1. Define the term Web form.
2. What is ASP.Net page life cycle?

3. Explain the life cycle events.
4. Give the meaning of HTML Server control.
5. Write the advantages of HTML Server control.
6. Write and explain the Steps for Creating a Project using diagram.
7. Explain the properties of the Server Controls.
8. What do you mean by RegularExpressionvalidator?

UNIT-8 ADO.NET

Structure

8.0 INTRODUCTION

ADO.NET is a set of classes (a framework) to interact with data sources such as databases and XML files. ADO is the acronym for ActiveX Data Objects. It allows us to connect to underlying data or databases. It has classes and methods to retrieve and manipulate data. The following are a few of the .NET applications that use ADO.NET to connect to a database, execute commands and retrieve data from the database.

- ASP.NET Web Applications
- Console Applications
- Windows Applications

ADO.NET is a large set of .NET classes that enable us to retrieve and manipulate data, and update data sources, in very many different ways. As an integral part of the .NET framework, it shares many of its features: features such as multi-language support, garbage collection, just-in-time compilation, object-oriented design, and dynamic caching, and is far more than an upgrade of previous versions of ADO.

ADO.NET is set to become a core component of any data-driven .NET application or Web Service, and understanding its power will be

essential to anyone wishing to utilize .NET data support to maximum effect.

This unit is aimed at developers, who already have some experience of developing or experimenting within the .NET framework, with either C# or Visual Basic .NET. We have already discussed about the ASP and ASP.NET. But, we have not covered the basics of C# or Visual Basic .NET, and assume some prior experience of Microsoft data access technologies.

Moreover, In this unit, we're just going to take a fairly quick overview of ADO.NET. This will be fast-paced, and we won't shy away from showing snippets of code, as this really is the best way to get to grips with the concepts. Hopefully this chapter will give you a basic understanding of the basic workings of ADO.NET, and give you a taste of some of its best features. By the end of the chapter, we hope that you'll be convinced of the advantages of ADO.NET, and eager to go further.

ADO.NET is the latest in a long line of data access technologies released by Microsoft. ADO.NET differs somewhat from the previous technologies, however, in that it comes as part of a whole new platform called the .NET Framework. This platform is set to revolutionize every area of development, and ADO.NET is just one aspect of that. We'll therefore start by looking quickly at the main features of .NET.

Furthermore, 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.

This unit is designed for anyone that wants to learn how to use ADO.NET to access data in databases. Throughout this unit you will be introduced to the concepts of the data handling using the Microsoft .NET Framework. Emphasis will be on ADO.NET programming standards and practices. In this unit you will learn the following: - Using the Connection Class - Using the Command Class - Using DataSets - Using DataTables - Using Parameters - Using Stored Procedures. This unit will show how to use ADO.NET to manipulate data in OLE DB and SQL Server databases. After reading this book you will be able to retrieve data, insert, update and delete data from relational databases using ADO.NET.

8.1 OBJECVTIVES

This unit aims to achieve the following objectives:

- Definition of ADO.NET and its working
- advantages of ADO.NET
- Required main features of .NET like CLS, MSIL etc.

- retrieve data, insert, update and delete data from relational databases using ADO.NET
- Brief history of data access
- ADO.NET architecture
- Knowing the code behind the ADO and XML

8.2 THE .NET FRAMEWORK

It's no exaggeration to say that Microsoft's release of its new development and run-time environment, the .NET Framework, will revolutionize all aspects of programming in the Microsoft world. The benefits of this new platform will be felt in all areas of our code and in all types of application we develop. The .NET Framework is in itself a huge topic, and we can't cover every aspect in detail here, but since it's important to understand the basic principles behind .NET before attempting any ADO.NET programming, we'll quickly review the basics here.

8.2.1 THE COMMON LANGUAGE RUNTIME

The foundation on which the .NET Framework is built is the Common Language Runtime (CLR). The CLR is the execution environment that manages .NET code at run time. In some ways, it is comparable to the Java Virtual Machine (JVM), or to the Visual Basic 6 runtime (msvbvm60.dll). Like these, the .NET Framework needs to be installed on any machine where .NET programs will be run. Unlike these, however, the CLR was designed specifically to support code written in many different languages. It's true that many different languages have been written that target the JVM (at present more than there are for .NET), but multiple language support wasn't one of the primary design considerations of the JVM. In the case of the CLR, this really was one of the most important considerations.

In order to achieve cross-language support, all .NET programs are compiled prior to deployment into a low-level language called Intermediate Language (IL). Microsoft's implementation of this language is called Microsoft Intermediate Language, or MSIL. This IL code is then just-in-time compiled into native code at run time. This means that, whatever the original language of the source code, .NET executables and DLLs are always deployed in IL, so there are no differences between components originally written in C# and those written in VB .NET. This aids cross-language interoperability (such as the ability to derive a class written in one language from one written in any of the other .NET languages). However, it also allows applications to be deployed without modifications onto any supported platform (currently Windows 9x/ME, Windows NT4, Windows 2000, or Windows XP) – the JIT compiler handles optimizations for the processor/OS of the deployment machine.

Microsoft provides compilers for four .NET languages:

- C# - a new C-based language designed specifically for the .NET Framework.
- Visual Basic.NET – a version of the Visual Basic language updated for .NET (for example, with full object-oriented features, structured exception handling, and many of the other things VB developers have been demanding for years!).
- JScript.NET – Microsoft's implementation of the JavaScript scripting language, updated for .NET.
- Managed C++ – C++ with "managed extensions" to support .NET features that couldn't be implemented using the existing features of the language. Unlike the other three languages, the C++ compiler doesn't come free with the .NET Framework SDK, but is shipped with Visual Studio.NET.
- J# – essentially Visual J++ (including Microsoft extensions to Java such as COM support) for the .NET Framework.

Garbage Collection

One of the most important services provided by the CLR is garbage collection. In C and C++, if an object is instantiated, the memory it uses needs to be released before it can be reused. Failure to do this result in a "memory leak" – unused memory that can't be reclaimed by the system. As the amount of leaked memory increases, the performance of the application obviously deteriorates. However, because the error isn't obvious and only takes effect over time, these errors are notoriously difficult to trace. The CLR solves this problem by implementing a garbage collector. At periodic intervals (when there is no more room on the heap), the garbage collector will check all object references, and release the memory held by objects that have run out of scope and can no longer be accessed by the application. This exempts the programmer from having to destroy the objects explicitly, and solves the problem of memory leaks. There are a couple of points to remember here: firstly, we can't predict exactly when the garbage collector will run (although we can force a collection), so objects can remain in memory for some time after we've finished with them; secondly, the CLR won't clear up unmanaged resources – we need to do that ourselves. The usual way of doing this is to expose a method named Dispose, which will release all external resources and which can be called when we've finished with the object.

The Common Language Infrastructure

Although the .NET Framework is currently only available for Windows platforms, Microsoft has submitted a subset of .NET (the Common Language Infrastructure, or CLI) to the European Computer Manufacturers' Association (ECMA) for acceptance as an open standard.

Versions of the CLI are in development for the FreeBSD and Linux operating systems. Similarly, specifications for C# and IL (the latter termed Common Intermediate Language, or CIL) have also been submitted to ECMA, and non-Microsoft implementations of the CLI will also implement these.

8.2.2 ASSEMBLIES

.NET code is deployed as an assembly. Assemblies consist of compiled IL code, and must contain one primary file (except in the case of dynamic assemblies, which are stored entirely in memory). This can be an executable (.exe) file, a DLL, or a compiled ASP.NET web application or Web Service. As well as the primary file, an assembly can contain resource files (such as images or icons) and other code modules. Most importantly, however, assemblies contain metadata. This metadata consists of two parts: the type metadata includes information about all the exported types and their methods defined in the assembly. As well as IL code, .NET assemblies contain a section known as the manifest. This section contains the assembly metadata, or information about the assembly itself, such as the version and build numbers.

This metadata allows assemblies to be completely self-describing: the assembly itself contains all the information necessary to install and run the application. There's no longer any need for type libraries or registry entries. Installation can be as simple as copying the assembly onto the target machine. Better still, because the assembly contains version information, multiple versions of the same component can be installed side-by-side on the same machine. This ends the problem known as "DLL Hell", where an application installing a new version of an existing component would break programs that used the old version.

8.2.3 THE COMMON TYPE SYSTEM

The foundation on which the CLR's cross-language features are built is the Common Type System (CTS). In order for classes defined in different languages to be able to communicate with each other, they need a common way to represent data – a common set of data types. All the predefined types that are available in IL are defined in the CTS. This means that all data in .NET code is ultimately stored in the same data types, because all .NET code compiles to IL.

The CTS distinguishes between two fundamental categories of data types – value types and reference types. Value types (including most of the built-in types, as well as structs and enumerations) contain their data directly. For example, a variable of an integer type stores the integer directly on the program's stack. Reference types (including String and Object, as well as arrays and most user-defined types such as classes and interfaces) store only a reference to their data on the stack – the data itself is stored in a different area of memory known as the heap. The difference between these types is particularly evident when passing parameters to

methods. All method parameters are by default passed by value, not by reference. However, in the case of reference types, the value is nothing more than a reference to the location on the heap where the data is stored. As a result, reference-type parameters behave very much as we would expect arguments passed by reference to behave – changing the value of the variable within the body of the method will affect the original variable, too. This is an important point to remember if you pass ADO.NET objects into a method.

The Common Language Specification

One important point to note about the CTS is that not all features are exposed by all languages. For example, C# has a signed byte data type (sbyte), which isn't available in Visual Basic.NET. This could cause problems with language interoperability, so the Common Language Specification (CLS) defines a subset of the CTS, which all compilers must support. It's perfectly possible to expose features that aren't included in the CLS (for example, a C# class with a public property of type sbyte). However, it's important to remember that such features can't be guaranteed to be accessible from other languages – in this example, we wouldn't be able to access the class from VB.NET.

8.2.4 NET CLASS LIBRARIES

Finally, we come to perhaps the most important feature of all – a vast set of class libraries to accomplish just about any programming task conceivable. Classes and other types within the .NET Framework are organized into namespaces, similar to Java packages. These namespaces can be nested within other namespaces, and allow us to identify our classes and distinguish them from third-party classes with the same name.

Together with the .NET Framework, Microsoft provides a huge set of classes and other types, mostly within the System namespace, or one of the many nested namespaces. This includes the primitive types such as integers – the C# int and VB.NET Integer types are just aliases for the System.Int32 type. However, it also includes classes used for Windows applications, web applications, directory services, file access, and many others – including, of course, data access. These data access classes are collectively known as ADO.NET. In fact, .NET programming is effectively programming with the .NET class libraries – it's impossible to write any program in C# or VB NET that doesn't use these libraries.

8.3 BRIEF HISTORY OF DATA ACCESS

At first, programmatic access to databases was performed by native libraries, such as DBLib for SQL Server, and the Oracle Call Interface (OCI) for Oracle. This allowed for fast database access because no extra layer was involved – we simply wrote code that accessed the database directly. However, it also meant that developers had to learn a

different set of APIs for every database system they ever needed to access, and if the application had to be updated to run against a different database system, all the data access code would have to be changed.

8.3.1 ODBC

As a solution to this, in the early 1990s Microsoft and other companies developed Open Database Connectivity, or ODBC. This provided a common data access layer, which could be used to access almost any relational database management system (RDBMS). ODBC uses an RDBMS -specific driver to communicate with the data source. The drivers (sometimes no more than a wrapper around native API calls) are loaded and managed by the ODBC Driver Manager. This also provides features such as connection pooling – the ability to reuse connections, rather than destroying connections when they are closed and creating a new connection every time the database is accessed. The application communicates with the Driver Manager through a standard API, so (in theory) if we wanted to update the application to connect to a different RDBMS, we only needed to change the connection details (in practice, there were often differences in the SQL dialect supported). Perhaps the most important feature of ODBC, however, was the fact that it was an open standard, widely adopted even by the Open Source community. As a result, ODBC drivers have been developed for many database systems that can't be accessed directly by later data access technologies. As we'll see shortly, this means ODBC still has a role to play in conjunction with ADO.NET.

8.3.2 DAO

One of the problems with ODBC is that it was designed to be used from low-level languages such as C++. As the importance of Visual Basic grew, there was a need for a data access technology that could be used more naturally from VB. This need was met in VB 3 with Data Access Objects (DAO). DAO provided a simple object model for talking to Jet, the database engine behind Microsoft's Access desktop database. As DAO was optimized for Access (although it can also be used to connect to ODBC data sources), it is very fast – in fact, still the fastest way of talking to Access from VB 6.

8.3.3 RDO

Due to its optimization for Access, DAO was very slow when used with ODBC data sources. To get round this, Microsoft introduced Remote Data Objects (RDO) with the Enterprise Edition of VB 4 (32-bit version only). RDO provides a simple object model, similar to that of DAO, designed specifically for access to ODBC data sources. RDO is essentially a thin wrapper over the ODBC API.

8.3.4 OLE DB

The next big shake-up in the world of data access technologies came with the release of OLE DB. Architecturally, OLE DB bears some resemblance to ODBC: communication with the data source takes place through OLE DB providers (similar in concept to ODBC drivers), which are designed for each supported type of data source. OLE DB providers implement a set of COM interfaces, which allow access to the data in a standard row/column format. An application that makes use of this data is known as an OLE DB consumer. As well as these standard data providers, which extract data from a data source and make it available through the OLE DB interfaces, OLE DB also has a number of service providers. These form a "middle tier" of the OLE DB architecture, providing services that are used with the data provider. These services include connection pooling, transaction enlistment (the ability to register MTS/COM+ components automatically within an MTS/COM+ transaction), data persistence, client-side data manipulation (the Client Cursor Engine, or CCE), hierarchical recordsets (data shaping), and data remoting (the ability to instantiate an OLE DB data provider on a remote machine).

The real innovation behind OLE DB was Microsoft's strategy for Universal Data Access (UDA). The thinking behind UDA is that data is stored in many places – e-mails, Excel spreadsheets, web pages, and so on, as well as traditional databases – and that we should be able to access all this data programmatically, through a single unified data access technology. OLE DB is the base for Microsoft's implementation of this strategy. The number of OLE DB providers has been gradually rising to cover both relational database systems (even the opensource MySQL database now has an OLE DB provider), and non-relational data sources such as the Exchange 2000 Web Store, Project 2000 files, and IIS virtual directories. However, even before these providers became available, Microsoft ensured wide-ranging support for OLE DB by supplying an OLE DB provider for ODBC drivers, This meant that right from the start OLE DB could be used to access any data source that had an ODBC driver. As we shall see, this successful tactic has been adopted again for ADO.NET.

8.3.5 ADO

ActiveX Data Objects (ADO) is the technology that gave its name to ADO.NET (although in reality the differences are far greater than the similarities). ADO is merely an OLE DB consumer – a thin layer allowing users of high-level languages such as VB and script languages to access OLE DB data sources through a simple object model; ADO is to OLE DB more or less what RDO was to ODBC. Its popularity lay in the fact it gave the vast number of Visual Basic, ASP, and Visual J++ developers easy access to data in many different locations. If OLE DB was the foundation on which UDA was built, ADO was the guise in which it appeared to the majority of developers. And, in certain scenarios, ADO still represents a

valid choice for developers on the .NET Framework. Moreover, because many of the classes and concepts are similar, knowledge of ADO is a big advantage when learning ADO.NET. We will look at the relationship between ADO and ADO.NET in more detail later on in the chapter.

8.4 INTRODUCTION TO ADO.NET

Although we've presented it as something of an inevitability that .NET would bring a new data access API, we haven't yet really said why. After all, it's perfectly possible to carry on using ADO in .NET applications through COM interoperability. However, there are some very good reasons why ADO wasn't really suited to the new programming environment. We'll look quickly at some of the ways in which ADO.NET improves upon ADO called from .NET, before looking at the ADO.NET architecture in more detail.

8.4.1 ADVANTAGES OF USING MANAGED CLASSES

Firstly, and most obviously, if we're using .NET then COM interoperability adds overhead to our application. .NET communicates with COM components via proxies called Runtime Callable Wrappers, and method calls need to be marshaled from the proxy to the COM object. In addition, COM components can't take advantage of the benefits of the CLR such as JIT compilation and the managed execution environment – they need to be compiled to native code prior to installation. This makes it essential to have a genuine .NET class library for data access.

8.4.2 CROSS-LANGUAGE SUPPORT

Another factor is the fact that ADO wasn't really designed for cross-language use; it was aimed primarily at VB programmers. As a result, ADO makes much use of optional method parameters, which are supported by VB and VB.NET, but not by C-based languages such as C#. This means that if you use ADO from C#, you will need to specify all parameters in method calls; for example, if you call the Connection.Open method and don't want to specify any options, you will need to include the adConnectUnspecified parameter! This makes ADO programming under .NET considerably more time-consuming.

8.4.3 CLEANER ARCHITECTURE

As we noted above, ADO is no more than a thin layer over OLE DB. This makes the ADO architecture slightly cumbersome, as extra layers are introduced between the application and the data source. While much ADO.NET code will still use OLE DB for the immediate future, this will decrease as more native .NET data providers become available. Where a native provider exists, ADO.NET can be much faster than ADO, as the providers communicate directly with the data source.

8.4.4 XML SUPPORT

One of the key features of the .NET Framework is its support for XML. XML is the standard transport and persistence format throughout the .NET Framework. While ADO had some support for XML from version 2.1 onwards, this was very limited, and required XML documents to be in exactly the right format.

8.4.5 OPTIMIZED OBJECT MODEL

Finally, it's important to remember that the .NET Framework is aimed squarely at developing distributed applications, and particularly Internet-enabled applications. In this context, it's clear that certain types of connection are better than others. In an Internet application, we don't want to hold a connection open for a long time, as this could create a bottleneck as the number of open connections to the data source increase, and hence destroy scalability. ADO didn't encourage disconnected recordsets, whereas ADO.NET has different classes for connected and disconnected access, and doesn't permit updateable connected recordsets. We'll look at this issue in more detail later in the chapter.

Check Your Progress

- What do you mean by data access
- Give the advantages using the managed classes

8.5 HIGH DEFINITION OF ADO.NET

If you have a background in Microsoft's previous COM-based data access model (Active Data Objects, or ADO), understand that ADO.NET has very little to do with ADO beyond the letters "A," "D," and "O." While it is true that there is some relationship between the two systems (e.g., each has the concept of connection and command objects), some familiar ADO types (e.g., the Recordset) no longer exist. Furthermore, there are a number of new ADO.NET types that have no direct equivalent under classic ADO (e.g., the data adapter). Unlike classic ADO, which was primarily designed for tightly coupled client/server systems, ADO.NET was built with the disconnected world in mind, using DataSets. This type represents a local copy of any number of related data tables, each of which contain a collection of rows and column. Using the DataSet, the calling assembly (such as a web page or desktop executable) is able to manipulate and update a DataSet's contents while disconnected from the data source, and send any modified data back for processing using a related data adapter.

Another major difference between classic ADO and ADO.NET is that ADO.NET has deep support for XML data representation. In fact, the data obtained from a data store is serialized (by default) as XML. Given that XML is often transported between layers using standard HTTP, ADO.NET is not limited by firewall constraints. Perhaps the most fundamental difference between classic ADO and ADO.NET is that ADO.NET is a managed library of code, therefore it plays by the same rules as any managed library. The types that make up ADO.NET use the CLR memory management protocol, adhere to the same type system (classes, interfaces, enums, structures, and delegates), and can be accessed by any .NET language.

From a programmatic point of view, the bulk of ADO.NET is represented by a core assembly named System.Data.dll. Within this binary, you may find a good number of namespaces (Figure 8.1), many of which represent the types of a particular ADO.NET data provider.

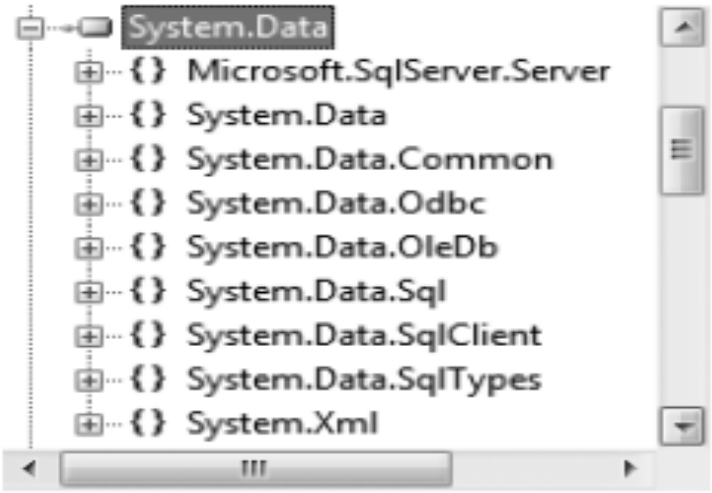


Figure 8.1: Core ADO.NET assembly

The ADO.NET libraries can be used in two conceptually unique manners: connected or disconnected. When you make use of the *connected layer*, your code base will explicitly connect to and disconnect from the underlying data store. When you are using ADO.NET in this manner, you typically interact with the data store using connection objects, command objects, and data reader objects. The disconnected layer, allows you to manipulate a set of DataTable objects (contained within a DataSet) that functions as a client-side copy of the external data. When you obtain a DataSet using a related data adapter object, the connection is automatically opened and closed on your behalf. As you would guess, this approach helps quickly free up connections for other callers and goes a long way to increasing the scalability of the systems.

Moreover, once a caller receives a DataSet, it is able to traverse and manipulate the contents without incurring the cost of network traffic.

As well, if the caller wishes to submit the changes back to the data store, the data adapter (in conjunction with a set of SQL statements) is used once again to update the data source, at which point the connection is closed immediately.

8.6 ARCHITECTURAL OVERVIEW OF ADO.NET

The ADO.NET object model consists of two fundamental components: the **DataSet**, which is disconnected from the data source and doesn't need to know where the data it holds came from; and the **.NET data provider**. The .NET data providers allow us to connect to the data source, and to execute SQL commands against it.

8.6.1 .NET DATA PROVIDERS

At the time of writing, there are three .NET data providers available: for SQL Server, for OLE DB data sources, and for ODBC-compliant data sources. Each provider exists in a namespace within the *System.Data* namespace, and consists of a number of classes.

Data Provider Components

Each .NET data provider consists of four main components:

- **Connection** – used to connect to the data source
- **Command**– used to execute a command against the data source and retrieve a DataReader or

  DataSet, or to execute an INSERT, UPDATE, or DELETE command against the data source
- **DataReader**– a forward-only, read-only connected resultset
- **DataAdapter** – used to populate a DataSet with data from the data source, and to update the data Source

The Connection Classes

The connection classes are very similar to the ADO Connection object, and like that, they are used to represent a connection to a specific data source. The connection classes store the information that ADO.NET needs to connect to a data source in the form of a familiar connection string (just as in ADO). The *IDbConnection* interface's *ConnectionString* property holds information such as the username and password of the user, the name and location of the data source to connect to, and so on. In addition, the connection classes also have methods for opening and closing connections, and for beginning a transaction, and properties for setting the

timeout period of the connection and for returning the current state (open or closed) of the connection.

The Command Classes

The command classes expose the *IDbCommand* interface and are similar to the ADO Command object – they are used to execute SQL statements or stored procedures in the data source. Also, like the ADO Command object, the command classes have a *CommandText* property, which contains the text of the command to be executed against the data source, and a *CommandType* property, which indicates whether the command is a SQL statement, the name of a stored procedure, or the name of a table.

There are three distinct execute methods: ExecuteReader, which returns a DataReader; ExecuteScalar, which returns a single value; and ExecuteNonQuery, for use when no data will be returned from the query (for example, for a SQL UPDATE statement).

Again like their ADO equivalent, the command classes have a Parameters collection – a collection of objects to represent the parameters to be passed into a stored procedure. These objects expose the IDataParameter interface, and form part of the .NET provider. That is, each provider has a separate implementation of the IDataParameter (and IDataParameterCollection) interfaces.

The Data Reader

The DataReader is ADO.NET's answer to the connected recordset in ADO. However, the DataReader is forward-only and read-only – you can't navigate through it at random, and you can't use it to update the data source. It therefore allows extremely fast access to data that we just want to iterate through once, and it is recommended to use the DataReader (rather than the DataSet) wherever possible. A DataReader can only be returned from a call to the ExecuteReader method of a command object; we can't instantiate it directly. This forces us to instantiate a command object explicitly, unlike in ADO, where we could retrieve a Recordset object without ever explicitly creating a Command object. This makes the ADO.NET object model more transparent than the "flat" hierarchy of ADO.

The Data Adapter

The last main component of the .NET data provider is the DataAdapter. The DataAdapter acts as a bridge between the disconnected DataSet and the data source. It exposes two interfaces; the first of these, IDataAdapter, defines methods for populating a DataSet with data from the data source, and for updating the data source with changes made to the DataSet on the client. The second interface, IDbDataAdapter, defines four properties, each of type IDbCommand. These properties each set or return a command object specifying the command to be executed when the data source is to be queried or updated:

8.6.2 EXISTING DATA PROVIDERS

Three .NET data providers are currently available; these allow us to access any type of data source that we could access using ADO 2.1. The reason we've said ADO 2.1 rather than 2.5 (or later) is that the OLE DB 2.5 interfaces –IRow, IStream, etc. (exposed by the ADO Record and Stream objects) – are not supported by the OleDb provider. This means that we'll still need to use "classic" ADO with data sources such as web directories and the Exchange 2000 Web Store, until such time as ADO.NET equivalents for the Internet Publishing (MSDAIPP) and Exchange 2000 (ExOLEDB) OLE DB providers become available.

The SqlClient Provider

The SqlClient provider ships with ADO.NET and resides in the System.Data.SqlClient namespace. It can be used to access SQL Server 7.0 or later databases, or MSDE databases. The SqlClient provider can't be used with SQL Server 6.5 or earlier databases, so you will need to use the OleDb .NET provider with the OLE DB provider for SQL Server (SQLOLEDB) if you want to access an earlier version of SQL Server. However, if you can use the SqlClient provider, it is strongly recommended that you do so – using the OleDb provider adds an extra layer to your data access code, and uses COM interoperability behind the scenes (OLE DB is COM-based).

The OleDb Provider

If you're not using SQL Server 7.0 or later, it's almost certain that your best bet will be to use the OleDb provider, at least until more .NET providers are released. There are a couple of exceptions to this rule – if your data source has an ODBC driver, but not an OLE DB provider, then you will need to use the Odbc .NET provider. Support for MSDASQL (the OLE DB provider for ODBC drivers) was withdrawn from the OleDb provider somewhere between Beta 1 and Beta 2 of the .NET Framework, so there really is no alternative to this. This was probably done to prevent the use of ODBC Data Source Names (DSNs) with ADO.NET, except where ODBC really is required. Even in ADO, using DSNs involved a substantial performance penalty (particularly when an OLE DB provider was available), but the extra layers would be intolerable under .NET.

Think of the architecture involved: ADO.NET – COM interop – (optional) OLE DB services – OLE DB provider – ODBC driver – data source!

The Odbc Provider

Unlike the other two .NET providers, the Odbc provider isn't shipped with the .NET Framework. The current beta version can be downloaded as a single .exe file of 503KB from the MSDN site and simply run this executable to install the classes – this program will install the assembly into the Global Assembly Cache, so the classes will automatically be globally available on the local machine. However, you will need to add a

reference to the assembly (System.Data.Odbc.dll) to your projects to use the provider. The Odbc provider should be used whenever you need to access a data source with no OLE DB provider (such as PostgreSQL or older databases such as Paradox or dBase), or if you need to use an ODBC driver for functionality that isn't available with the OLE DB provider. Architecturally, the Odbc provider is similar to the OleDb provider – it acts as a .NET wrapper around the ODBC API, and allows ADO.NET to access a data source through an ODBC driver.

8.6.3 THE DATASET

The other major component of ADO.NET is the DataSet; this corresponds very roughly to the ADO recordset. It differs, however, in two important respects. The first of these is that the DataSet is always disconnected, and as a consequence doesn't care where the data comes from – the DataSet can be used in exactly the same way to manipulate data from a traditional data source or from an XML document. In order to connect a DataSet to a data source, we need to use the DataAdapter as an intermediary between the DataSet and the .NET data provider.

After opening the connection just as we did before, there are three steps involved to populating the DataSet:

- **Instantiate a new DataAdapter object**: Before we fill the DataSet, we'll obviously need to specify the connection information and the data we want to fill it with. There are a number of ways of doing that, but probably the easiest is to pass the command text for the SQL query and either a connection string or an open connection into the DataAdapter's constructor, as we do above.

- **Create the new DataSet:**For doing this we have to create object of DataSet class by giving a call to the constructor of DataSet calss by using the following syntax.

  DataSet ds=new DataSet(), this line creates a new data set object whose memory location will be pointed by variable ds.

- **Call the DataAdapter's Fill method:** We pass the DataSet we want to populate as a parameter to this method, and also the name of the table within the DataSet we want to fill. If we call the Fill method against a closed connection, the connection will automatically be opened, and then reclosed when the DataSet has been filled.

- In order to do so you need to create object of SqlDataAdapter class. It is predefined class of C# class library to create its object we need to call its constructor by passing two parameters; first one is a string which represents sql query to be executed in sql environment and second is connection string contains information: information about data source, information about database and

information about sql credentials. Following we are giving a sample of object creation of SqlDataAdapter class.

SqlDataAdapter da=new SqlDataAdapter (“select * from table_name, “Data Source=.,Initial Catalog=database_name, User Id=sa, password=1”);

The DataTable Class

This last parameter gives a clue to the second important difference between a DataSet and an ADO recordset – the DataSet can contain more than one table of data. True, something similar was available in ADO with data shaping, but the tables in a DataSet can even be taken from different data sources. And, better still, we don't have the horrible SHAPE syntax to deal with. To achieve this, ADO.NET also has a DataTable class, which represents a single table within a DataSet. The DataSet has a Tables property, which returns a collection of these objects (a DataTableCollection). The DataTable represents data in the usual tabular format, and has collections of DataColumn and DataRow objects representing each column and row in the table.

8.7 ADO.NET AND XML

Perhaps the single most impressive new feature of ADO.NET is its built-in support for XML. In fact, XML is now the standard persistence format for ADO.NET DataSets. While we've been able to save recordsets in XML format since ADO 2.1, the default format remained the proprietary Advanced Data TableGram (ADTG) format, and XML support was always limited. We couldn't, for example, load an arbitrary XML document into an ADO recordset – the document had to be in exactly the right format. XML support in ADO.NET is far more complete – XML is absolutely integral to ADO.NET, and not just an add-on. XML is the format used to serialize and transport DataSets. Serializing a DataSet as an XML document (to a file, a stream, or a TextWriter object) is a trivial matter:

```
// Save the DataSet as an XML file

ds.WriteXml(@"C:\CSharp\Employees.xml");
```

The format of the generated XML document is far more readable than the ADO equivalent – the columns are represented by elements, not attributes, and there aren't a lot of unnecessary XML namespaces:

```
<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<Employees>

<EmployeeID>1</EmployeeID>

<FirstName>Nancy</FirstName>
```

<LastName>Davolio</LastName>

</Employees>

<Employees>

<EmployeeID>2</EmployeeID>

<FirstName>Andrew</FirstName>

<LastName>Fuller</LastName>

</Employees>

<!-- and so on... -->

</NewDataSet>

In addition, we can load any well-formed XML document into a DataSet, without having to use a predefined structure (although we might lose content if the structure of the document is not basically tabular).

Check Your Progress

- What do you mean by .NET data provider
- Compare ADO.NET and XML

8.10 SUMMARY

ADO.NET is the data access component of Microsoft's new .NET Framework. Microsoft bills ADO.NET as "an evolutionary improvement" over previous versions of ADO, a claim that has been hotly debated since its announcement. It is certainly true that the ADO.NET object model bears very little relationship to earlier versions of ADO.

In fact, whether you decide to love it or hate it, one fact about the .NET Framework seems undeniable: it levels the playing ground. Whether you've been at this computer game longer than you care to talk about or you're still sorting out your heaps and stacks, learning the .NET Framework will require a major investment. We're all beginners now. So welcome to Microsoft ADO.NET Step by Step. Through the exercises in this book, I will introduce you to the ADO.NET object model, and you'll learn how to use that model in developing data-bound Windows Forms and Web Forms. In later topics, we'll look at how ADO.NET interacts with XML and how to access older versions of ADO from the .NET environment. Since we're all beginners, an exhaustive treatment would be, well, exhausting, so this book is necessarily limited in scope. My goal is to provide you with an understanding of the ADO.NET objects—what they are and how they work together. So fair warning: this book will not make you an expert in ADO.NET. (How I wish it were that simple!) What this book will give you is a road map, a fundamental understanding of the

environment, from which you will be able to build expertise. You'll know what you need to do to start building data applications. The rest will come with time and experience. This book is a place to start. Although I've pointed out language differences where they might be confusing, in order to keep the book within manageable proportions I've assumed that you are already familiar with Visual Basic .NET or Visual C# .NET. If you're completely new to the .NET environment, you might want to start with Microsoft Visual Basic .NET Step by Step by Michael Halvorson (Microsoft Press, 2002) or Microsoft Visual C# .NET Step by Step by John Sharp and Jon Jagger (Microsoft Press, 2002), depending on your language of choice.

The exercises that include programming are provided in both Microsoft Visual Basic and Microsoft C#. The two versions are identical (except for the difference between the languages), so simply choose the exercise in the language of your choice and skip the other version.

8.10 TERMINAL QUESTIONS

1. What do you understand by ADO.NET?
2. Give the evolution of data access.
3. What are the ADO.NET components?
4. Write a short note on DataSet.
5. How can you define the DataSet structure?
6. What is the difference Between DataReader and DataSet?
7. Explain relation of ADO.NET with XML.

UNIT-9 ASP.NET STATE MANAGEMENT

Structure

9.0 INTRODUCTION

Before the discussion of state management in ASP.NET, let's get one thing straight: Attempting to manage state in Web applications goes against the fundamental design principles of the Web. One of the primary goals of the Web and its underlying protocol, HTTP, is to provide a scalable medium for sharing information. Adding user state inherently reduces scalability because the pages shown to a particular user will be different from those shown to another user and thus cannot be reused or cached.

In spite of this fact, many applications deployed on the Web require user-specific state to function properly. Applications ranging from e-commerce shopping sites to local company intranet sites depend on the ability to track individual requests from distinct users and store state on behalf of each client, whether it's items in a shopping cart or which days were selected on a calendar as requested vacation days. Although maintaining client-specific state is not officially part of the HTTP protocol, there is a proposal in place for adding state management to HTTP. RFC 210914 defines a proposed standard for state management for HTTP also known as cookies. Although it is only a proposed standard and not yet an official part of the HTTP specification, cookies are in

widespread use today in almost all browsers, and many Web sites rely on cookies for their functionality.

As a consequence, Web programmers must be very conscious about state management. Unlike traditional applications, Web applications must be very explicit about any state that is maintained on behalf of a client, and there is no one standard way to maintain that state. Moreover, remember that ASP.NET (and the Web) is stateless and the Web server does not keep track of past client requests. Furthermore, different technologies handle the issue of statement management differently like ASP.NET is somewhat unique in this regard but PHP works similarly.

Moreover, there some issues like client state management consumes bandwidth and introduces security risks because sensitive data is passed back and forth with each page post-back. Preserving state on a server can overburden servers: we also must consider Web farms and Web gardens.

9.1 OBJECVTIVES

At the end of this unit you would be familiar with the following topics:

- Definition of state management
- Types of state management
- Session management
- Client state management
- Cookies
- Issues in Cookies
- Server side state management
- Managing site
- Session variables

9.2 TYPES OF STATE

One of the most important decisions you face when designing a Web application is where to store your state. ASP.NET provides four types of state: application state, session state, cookie state, and view state. In this unit, we explore each type of state, when it is most applicable, and any disadvantages you should be aware of if you decide to make use of it. ASP.NET, like its predecessor, ASP, provides a pair of objects for managing application-level state and session-level state. Application state is where information that is global to the application may be stored. For efficiency, this state is typically stored once and then read from many times. Session state is maintained on a per-client basis. When a client first

accesses any page in an application, an ASP.NET generated session ID is created. That session ID is then transmitted between the server and the client via HTTP either using client-side cookies or encoded in a mangled version of the URL (URL mangling is discussed in detail later in this chapter). On subsequent accesses by the same client, state associated with that session ID may be viewed and modified. Cookies provide the ability to store small amounts of data on a client's machine. Once a cookie is set, all subsequent pages accessed by the same client will transmit the cookie and its value.

Finally, view state is a yet another way of storing state on behalf of a client by saving and restoring values from a hidden field when a form is posted. Although this technique for retaining state has been used by Web developers in the past, ASP.NET provides a simplified mechanism for taking advantage of it. As we have seen, it is possible to place items into the ViewState property bag available in every Page-derived class. When that page issues a POST request to itself, the values placed in the ViewState property bag can then be retrieved, the key restriction being that view state works only when a page posts to itself. Table 9.1 summarizes the advantages and disadvantages of each of the four types of state available in ASP.NET.

Table 9.1: State Type Comparison in ASP.NET

Type of State	**Scope of State**	**Advantages**	**Disadvantages**
Application	Global to the application	Shared across all clients	o Overuse limits scalability o Not shared across multiple machines in a Web farm or processors in a Web garden o Primary purpose subsumed by data cache in ASP.NET
Session	Per client	Can configure to be shared across machines in a Web farm and processors in a	o Requires cookies or URL mangling to manage client association o Off-host storage can be inefficient

		Web garden	
Cookie	Per client	o Works regardless of server configuration o State stored on Client o State can live beyond current session	o Limited memory (~4KB) o Clients may not support cookies or may explicitly disable them o State is sent back and forth with each request
View	Across POST requests to the same page	o Works regardless of server configuration	o State is retained only with POST request made to the same page o State is sent back and forth with each request

9.2.1 APPLICATION STATE

Application state is something that should be used with care, and in most cases, avoided altogether. Although it is a convenient repository for global data in a Web application, its use can severely limit the scalability of an application, especially if it is used to store shared, updateable state. It is also an unreliable place to store data, because it is replicated with each application instance and is not saved if the application is recycled. With this warning in mind, let's explore how it works. Application state is accessed through the Application property of the HttpApplication class, which returns an instance of class HttpApplicationState. This class is a named object collection, which means that it can hold data of any type as part of a key/value pair. Listing 9.1 shows a typical use of application state. As soon as the application is started, it loads the data from the database. Subsequent data accesses will not need to go to the database but will instead access the application state object's cached version. Data that is pre-fetched in this way must be static, because it will not be unloaded from the application until the application is recycled or otherwise stopped and restarted.

The one feature of application state that cannot be replaced by the data cache is the ability to have shared updateable state. Arguably, however, this type of state should not be used at all in a Web application, because it inherently limits scalability and is unreliable as a mechanism

for storing meaningful data. In the previous example, we were using application state to save statistics on browser type access. This information is maintained only as long as the application is running, and it is stored separately in each instance of the application. This means that when the process recycles, the data is lost. It also means that if this application is deployed in a Web farm (or a Web garden), separate browser statistics will be kept for each running instance of the application across different machines (or CPUs). To more reliably collect this type of statistical information, it would make more sense to save the data to a central database and avoid application state altogether.

Example: Use of Application State for Data Prefetching

```
// Inside of global.asax
void Application_Start(object src, EventArgs e)
{
DataSet ds = new DataSet();
// population of dataset from ADO.NET query not shown
// Cache DataSet reference
Application["FooDataSet"] = ds;
}
// In some page within the application
private void Page_Load(object src, EventArgs e)
{
DataSet ds = (DataSet)(Application["FooDataSet"]);
// ...
MyDataGrid.DataSource = ds;
// ...
}
```

9.2.2 Session State

Maintaining state on behalf of each client is often necessary in Web applications, whether it is used to keep track of items in a shopping cart or to note viewing preferences for a particular user. ASP.NET provides three ways of maintaining client-specific state: session state, cookie state, and view state. Each technique has its advantages and disadvantages. Session state is the most flexible and, in general, the most efficient. ASP.NET has enhanced session state to address some of the problems associated with it in previous versions of ASP, including the

abilities to host session state out of process (or in a database) and to track session state without using cookies.

Session state is maintained on behalf of each client within an ASP.NET application. When a new client begins to interact with the application, a new session ID (or session key) is generated and associated with all subsequent requests from that same client (either using a cookie or via URL mangling). By default, the session state is maintained in the same process and AppDomain as your application, so you can store any data type necessary in session state. If you elect to house session state in another process or in a database, however, there are restrictions on what can be stored, as we will discuss shortly. Session state is maintained in an instance of the HttpSessionState class and is accessible through the Session property of both the Page and HttpContext classes. When a request comes in to an application, the Session properties of the Page and HttpContext class used to service that request are initialized to the current instance of HttpSession- State that is associated with that particular client.

Example: HttpSessionState Class

```
public sealed class HttpSessionState : ICollection,
IEnumerable
{
// properties
public int CodePage {get; set;}
public int Count {get;}
public bool IsCookieless {get;}
public bool IsNewSession {get;}
public bool IsReadOnly {get;}
public KeysCollection Keys {get;}
public int LCID {get; set;}
public SessionStateMode Mode {get;}
public string SessionID {get;}
public HttpStaticObjectsCollection StaticObjects {get;}
public int Timeout {get; set;}
// indexers
public object this[string] {get; set;}
public object this[int] {get; set;}
// methods
```

```
public void Abandon();
public void Add(string name, object value);
public void Clear();
public void Remove(string name);
public void RemoveAll();
public void RemoveAt(int index);
//...
}
public class Page : TemplateControl, IHttpHandler
{
public virtual HttpSessionState Session {get;}
//..
}
public sealed class HttpContext : IServiceProvider
{
public HttpSessionState Session {get;}
//...
}
```

9.2.3 COOKIE STATE

Although not part of the HTTP specification (yet), cookies are often used to store user preferences, session variables, or identity. The server issues a Set-Cookie header in its response to a client that contains the value it wants to store. The client is then expected to store the information associated with the URL or domain that issued the cookie. In subsequent requests to that URL or domain, the client should include the cookie information using the Cookie header. Some limitations of cookies include the fact that many browsers limit the amount of data sent through cookies (only 4,096 bytes are guaranteed) and that clients can potentially disable all cookie support in their browser.

ASP.NET provides an HttpCookie class for managing cookie data. Listing 10-16 shows the HttpCookie class definition and the cookie collection properties exposed by the request and response objects. Note that the request and response objects both expose the collection of cookies through the HttpCookieCollection type, which is just a type-safe derivative of the NameObjectCollectionBase class, designed for storing HttpCookie class instances. Each cookie can store multiple name/value pairs, as specified by RFC 2109, which are accessible through the Values

collection of the Http-Cookie class or indirectly through the default indexer provided by the class.

Example: The HttpCookie Class

```
public sealed class HttpCookie
{
public string Domain {get; set;}
public DateTime Expires {get; set;}
public bool HasKeys {get; }
public string this[string key] {get; set;}
public string Name {get; set;}
public string Path {get; set;}
public string Secure {get; set;}
public string Value {get; set;}
public NameValueCollection Values {get; }
//...
}
public sealed class HttpRequest
{
public HttpCookieCollection Cookies {get;}
//...
}
public sealed class HttpResponse
{
public HttpCookieCollection Cookies {get;}
//...
}
```

9.3 ASP.NET - MANAGING STATE

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, and Application State.

9.3.1 VIEW STATE

The view state is the state of the page and all its controls. It is automatically maintained across posts by the 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 following:

- 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.

Table 9.2: StateBag Class Properties

Properties	**Description**
Itemname	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.

Table 9.3 : StateBag Class Methods

Methods	**Description**
Addname,value	Adds an item to the view state collection and existing item is updated.
Clear	Removes all the items from the collection.
EqualsObject	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.
Removename	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.
ToString	Returns a string representing the state bag object.

Example: The concept of storing view state.

Let us keep a counter, which is incremented each time the page is posted back by clicking a button on the page. A label control shows the value in the counter.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="statedemo._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title> Untitled Page </title>

</head>

<body>

    <form>

        <div>

            <h3>View State demo</h3>

            Page Counter:

            <asp:Label ID="lblCounter" runat="server" />

            <asp:Button ID="btnIncrement" runat="server"
            Text="Add Count" onclick="btnIncrement_Click" />

        </div>

    </form>

</body>

</html>
```

Example:

```
public partial class _Default : System.Web.UI.Page

{

    public int counter

    {

        get

        {

            if (ViewState["pcounter"] != null)

            {
```

```
                return ((int)ViewState["pcounter"]);
            }
            else
            {
                return 0;
            }
        }
        set
        {
            ViewState["pcounter"] = value;
    }
    }
    protected void Page_Load(object sender, EventArgs e)
    {
    lblCounter.Text = counter.ToString();      counter++;
    }
}
```

9.3.2 CONTROL STATE

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

9.3.2 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.

Table 9.4: Http SessionState class Properties

Properties	**Description**
SessionID	The unique session identifier.
Itemname	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.

Table 9.5 : HttpSessionState Class Methods

Method	**Description**
Addname,value	Adds an item to the session state collection.
Clear	Removes all the items from session state collection.
Removename	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.

The session state object is a name-value pair to store and retrieve some information from the session state object.

Example :

```
void StoreSessionInfo()
{
String fromuser = TextBox1.Text;   Session["fromuser"] = fromuser;
}
void RetrieveSessionInfo()
{
String fromuser = Session["fromuser"];   Label1.Text = fromuser;
}
```

The above code stores only strings in the Session dictionary object, however, it can store all the primitive data types and arrays composed of primitive data types, as well as the DataSet, DataTable, HashTable, and Image objects, as well as any user-defined class that inherits from the ISerializable object.

9.3.4 APPLICATION STATE

The ASP.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 properties and methods of HttpApplicationState class are shown in table 9.1 & 9.2 respectively.

Table 9.6 : Properties of Http Application State

Properties	**Description**
Itemname	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.

Table 9.7: Methods of HttpApplicationState

Method	Description
Addname,value	Adds an item to the application state collection.
Clear	Removes all the items from application state collection.
Removename	Removes the specified item from the application state collection
RemoveAll	Removes all objects from the application -state 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.

Application state data is generally maintained by writing handlers for the events: Application_Start, Application_End, Application_Error, Session_Start, Session_End.

The following code snippet shows the basic syntax for storing application state information:

```
Void Application_Start(object sender, EventArgs e)
{
Application["startMessage"] = "The application has started.";
}
Void Application_End(object sender, EventArgs e)
{
Application["endtMessage"] = "The application has ended.";
}
```

Check Your Progress

- What do you mean by state management?
- Give the names of different types of state management.

9.4 APPLICATION AND SESSION VARIABLES

The Application and Session objects can be used to store values that are global either to a particular user (the Session) or to all users (the Application). Within the onStart events, we can initialize these variables. We can also store new variables, or change existing values, in the code inside any other ASP page. Initializing variables is very important, especially with a language like VBScript that uses Variants. Imagine the following code in a page:

```
Response.Write("The current value is: " & Session("MyValue"))
```

This places the contents of the **Session** variable **MyValue** in the page. The only problem with this code is if the variable has not been initialized.

The current value is: Any **Variant** (the only data type available in VBScript) that has not been assigned a value is said to be **Empty**. Because we are dumping the variable as its default type, we get nothing. The best way to solve this type of problem is either assign a default value to it, or examine the variable using the **IsEmpty()** function.

Here's how we could use

Is Empty():

```
varTheValue = Session("MyValue")

If IsEmpty(varTheValue) Then varTheValue = "* Undefined *"

Response.Write("The current value is: " & varTheValue)
```

Alternatively, we can set any default value we like in the **Session_onStart** event, so that we have a value ready for access in that session:

```
Sub Session_OnStart

Session("MyValue") = 42

End Sub
```

9.5 COUNTING SESSIONS

An immediately obvious use of this technique is to count how many sessions have occurred during the current application. All we do is use a variable stored in the **Application** object, which is then available to all sessions:

```
Sub Application_OnStart

Application("NumVisitors") = 0

End Sub
```

Now, in **Session_onStart**, we can increment the value for each new session:

```
Sub Session_OnStart

Application.Lock

Application("NumVisitors") = Application("NumVisitors") + 1

Application.Unlock

End Sub
```

Them we can drop it into the 'welcome' page with a few lines of code:

```
<% Application.Lock %>

<H3> Your are visitor number <% = Application("NumVisitors")
%></H3>

<% Application.Unlock %>
```

9.6 WHAT IS COOKIE

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With ASP, you can both create and retrieve cookie values.

9.6.1 HOW TO CREATE A COOKIE?

The "Response.Cookies" command is used to create cookies.

Note : The Response.Cookies command must appear BEFORE the <html> tag.

In the example below, we will create a cookie named "firstname" and assign the value "Alex" to it:

```
<%
Response.Cookies("firstname")="Alex"
%>
```

It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:

```
<%
Response.Cookies("firstname")="Alex"

Response.Cookies("firstname").Expires=#May 10,2012#

%>
```

9.6.2 HOW TO RETRIEVE A COOKIE VALUE

The "Request.Cookies" command is used to retrieve a cookie value. In the example below, we retrieve the value of the cookie named "firstname" and display it on a page :

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)

%>
```

Output: Firstname=Alex

9.6.3 LIMITATIONS OF COOKIES

While simple, cookies have disadvantages too

- A cookie can only be 4096 bytes in size
- Most browsers restrict the total number of cookies per site
- Users can refuse to accept cookies so don't try to use them to store critical information

9.7 STORING VARIABLES IN DATABASE

Following code can been used to store the variables in the database

```
SqlConnection con = new SqlConnection("Initial Catalog=Thumbnail

Data Source='OWNER- PC\\SQLEXPRESS';integrated security=true;"))

{

using (SqlCommand command = new   Sql Command ("Get User Login Details", con))

{

command.Parameters.Add (new Sql Parameter ("@UserName",
```

```
SqlDbType.VarChar)).Value = username;
command.Parameters.Add(new Sql Parameter ("@Password",
SqlDbType.VarChar)).Value = password
con.Open();
if (con.State == ConnectionState.Open)
{
using (SqlDataReader reader = comm.ExecuteReader())
{
if (reader.Read())
{
if (reader["username"] != DBNull.Value)
      {
Session["Username"] = reader["username"].ToString();
       }
return true;
}
else
{
return false;
}
}
}
else
{
throw new Exception("Could not open database");
}
}
}
```

9.8 CLEARING ASP.NET SESSION VARIABLES

ASP.NET Session is one of most common state management technique for any ASP.NET Web Application. If you want to do a quick refresh or want to know something more, please go ahead and read one of my article “Exploring Session in ASP.NET” published at Code Project. ASP.NET provides several methods for removing Session. But which

methods needs to use at what time, is a must known stuff for asp.net developer. In this post I going to talk about bit internals of removing session variables from applications. Why this Post ? I found many people having some confusion around removing / clearing the session variable (Mainly with Session.Clear(), Session.RemoveAll(), Session.Abandon()) , which method needs to use, what is the purpose of particular method etc. ASP.NET Provides below methods to clearing or removing Session information:

- Session.Clear()
- Session.RemoveAll()
- Session.Abandon()
- Session.RemoveAt(index)
- Session.Remove(string)

We will be mainly focusing the first three methods. Let's start with *Session.Clear()* and *Session.RemoveAll()*. Well, you may ask, why I am starting with two methods together. Yes, we are in same track. Both *Session.Clear()* and *Session.RemoveAll()*does the same job. Let's explore it

Let's assume stored below information with in session variable.

```
protected void Page_Load(object sender, EvenArgs e)
{
Session["Item1"] = "My session Info 1";
Session["Item2"] = "My session Info 2";
Session["Item3"] = "My session Info 3";
Session["Item4"] = "My session Info 4";
Session["Item5"] = "My session Info 5";
}
```

now, if you want to remove all the items from session you can use either ***Session.RemoveAll()*** or ***Session.Clear()***.if you check the definition from meta data file you will get below details, where description says the same.

```
//
// Summary
// Removes all keys and values from the session-state collection.
public void Clear ();
//
```

```
// Summary
// Removes all keys and values from the session-state collection.
public void removeAll();
```

9.8 ASSEMBLIES

Assembly is a compiled output of program which are used for easy deployment of an application. They are executables in the form of exe or dll. It also is a collection of resources that were used while building the application and is responsible for all the logical functioning.

9.8.1 TYPES OF ASSEMBLIES

Private Assemblies: are accessible by a single application. They reside within the application folder and are unique by name. They can be directly used by copying and pasting them to the bin folder.

Shared Assemblies: are shared between multiple applications to ensure reusability. They are placed in GAC.

Satellite Assemblies: are the assemblies to provide the support for multiple languages based on different cultures. These are kept in different modules based on the different categories available.

9.8.2 DIFFERENCE BETWEEN PRIVATE AND SHARED ASSEMBLY

The terms 'private' and 'shared' refer to how an assembly is deployed, not any intrinsic attributes of the assembly. A private assembly is normally used by a single application, and is stored in the application's directory, or a sub-directory beneath. A shared assembly is intended to be used by multiple applications, and is normally stored in the global assembly cache (GAC)**,** which is a central repository for assemblies. A shared assembly can also be stored outside the GAC, in which case each application must be pointed to its location via a codebase entry in the application's configuration file. The main advantage of deploying assemblies to the GAC is that the GAC can support multiple versions of the same assembly side-by-side. Assemblies deployed to the GAC must be strong-named. Outside the GAC, strong-naming is optional.

9.8.3 CREATING AND USING STRONG-NAMED ASSEMBLIES

A strong name consists of the assembly's identity—its simple text name, version number, and culture information (if provided)—plus a public key and a digital signature. It is generated from an assembly file using the corresponding private key. (The assembly file contains the

assembly manifest, which contains the names and hashes of all the files that make up the assembly.)

A strong-named assembly can only use types from other strong-named assemblies. Otherwise, the security of the strong-named assembly would be compromised.

The following scenario outlines the process of signing an assembly with a strong name and later referencing it by that name.

1. Assembly A is created with a strong name using one of the following methods:
   - Using a development environment that supports creating strong names, such as Visual Studio 2005.
   - Creating a cryptographic key pair using the Strong Name tool (Sn.exe) and assigning that key pair to the assembly using either a command-line compiler or the Assembly Linker (Al.exe). The Windows Software Development Kit (SDK) provides both Sn.exe and Al.exe.
2. The development environment or tool signs the hash of the file containing the assembly's manifest with the developer's private key. This digital signature is stored in the portable executable (PE) file that contains Assembly A's manifest.
3. Assembly B is a consumer of Assembly A. The reference section of Assembly B's manifest includes a token that represents Assembly A's public key. A token is a portion of the full public key and is used rather than the key itself to save space.
4. The common language runtime verifies the strong name signature when the assembly is placed in the global assembly cache. When binding by strong name at run time, the common language runtime compares the key stored in Assembly B's manifest with the key used to generate the strong name for Assembly A. If the .NET Framework security checks pass and the bind succeeds, Assembly B has a guarantee that Assembly A's bits have not been tampered with and that these bits actually come from the developers of Assembly A.

Check Your Progress

- What is session variable?
- Discuss the limitations of cookies.

9.10 SUMMARY

State management influences almost every aspect of a Web application’s design, and it is important to understand all the options available for state management as well as their implications for usability,

performance, and scalability. ASP.NET provides four types of state, each of which may be the best choice in different parts of your application. State that is global to an application may be stored in the application state bag, although it is typically preferable to use the new data cache instead of application state in ASP.NET. Client-specific state can be stored either in the session state bag, as client-side cookies, or as view state.

Session state is most commonly used for storing data that should not be sent back and forth with each request, either because it is too large or because the information should not be visible on the Internet. Cookie state is useful for small client-specific pieces of information such as preferences, authentication keys, and session keys. View state is a useful alternative to session state for information that needs to be retained across posts back to the same page. Finally, enhancements to the session state model in ASP.NET give developers the flexibility to rely on session state even for applications that are deployed on Web farms or Web gardens through remote session storage.

9.11 TERMINAL QUESTIONS

1. Define the term state management.
2. Write the categories and need of state management.
3. Discuss about the role of state management.
4. What do you understand by session management? Explain.
5. Explain the meaning of application state.
6. Write the role of control state.
7. Discuss about cookies and its limitations.
8. What do you understand by assembly?
9. Explain multiple types of assembly.
10. Write a short note on Creating and Using Strong-Named Assemblies.

UNIT-10 CONFIGURATION

Structure

10.0 INTRODUCTION

It is very clear that computer includes two parts which are hardware and software. Software is the soul of the hardware and termed as very important part for it. Without software, the hardware is nothing or it cannot work properly. This software differs from computer to computer or we can say hardware to hardware. The software is intended with a particular machine e.g. client machines have some kind of software while server computer have other kind of software. Moreover, operating system is very popular software which provides more or less all the things that hardware requires to come into running position or implementation. Operating systems are of many types like single-user, multi-user, distributed, network, hand-held, client-server etc. More or less the functions of all the operating systems are same.

Moreover, the configuration also plays important role for all software. Improper configuration or installation of the software may lead to errors or sometimes it may be harmful to such extent that it may crash the hardware or system for which it intended. Hence it becomes necessary to know about the proper configuration or installation of the software. Normally when we purchase a computer or any other hardware like router, switch, printer etc; some configuration is automatically done and some need to be done manually. We are intended with the second case i.e. manual configuration. In this unit, Windows and .NET configuration has

been discussed however there may me other configurations like Linux configuration etc.

As computer system requires configuration. This is obvious to anyone who has used even the simplest consumer grade machines, which demand that you supply your name and the computer's name when they are first unpacked. Anyone who has worked with system or network administrators knows that full configuration is far more complex, including the ability — and sometimes the need — to set options that most people don't even know exist. What is less obvious, even to many system administrators, is that configuration management is a vital part of system and site security. Not only do security devices themselves require configuration, many rather ordinary aspects of life with computers (e.g., the fact that laptops are sometimes turned off and hence can't be reconfigured) have implications for security. Security configuration has another unique feature: it has strong interactions with business and personnel policy. Many security decisions are concrete instantiations of management decisions.

Security configuration has another unique feature: it has strong interactions with business and personnel policy. Many security decisions are concrete instantiations of management decisions. Incorrect configuration can endanger business relationships or have legal ramifications. Conversely, by definition security involves dealing with enemies. That is, someone will try to counter your moves. Managing security configuration is not simply a matter of designing the right configuration and distributing it, contending only with Murphy's Law; instead, the administrator must contend with active attempts to subvert configurations, evade them, or drive the system into an improbable state not anticipated by the configuration. Note well that "enemies" can include a site's own employees.

10.1 OBJECTIVES

At the end of this unit you would be familiar with the following topics:

- About configuration
- Windows configuration
- .NET configuration
- Issues in configuration
- Caching
- Types of Caching

10.2 WHAT IS CONFIGURATION

Configuration can refer to either hardware or software, or the combination of both. Generally a configuration is the arrangement of the

multiple parts in a machine. In the computer world when, people talk about their computer configuration, basically they refer to the technical specification or the "tech specs" of those computers. These specs typically include processor speed, the amount of RAM, hard Drive Space, and the type of video card in the machine etc. Without configuration a computer is incomplete. Therefore, it is evident that Computer systems require configuration. This is obvious to anyone who has used even the simplest consumer grade machines, which demand that you supply your name and the computer's name when they are first unpacked. Anyone who has worked with system or network administrators knows that full configuration is far more complex, including the ability — and sometimes the need — to set options that most people don't even know exist. What is less obvious, even to many system administrators, is that configuration management is a vital part of system.

And other aspect of configuration is site security. Not only security devices themselves require configuration, many rather ordinary aspects of life with computers (e.g., the fact that laptops are sometimes turned off and hence can't be reconfigured) have implications for security.

Normally configuration of a computer is used in different ways:

- In networks, a configuration often means the Networks' Topology or its physical structre.

- During the Installation of hardware and software configuration is the method or process of defining options that are provided.

- For instance a typical configuration for a PC consist of main memory in MB, a CD/DVD, a hard disk, a modem, CD ROM drive, a VGA monitor and the Windows Operating System.

When you install a new device or program, sometimes you need to configure it, which means to set Various Switches Jumpers (for hardware) and to define value of parameters (for Software).

Security configuration has another unique feature: It has strong interactions with business and personnel policy. Many security decisions are concrete instantiations of management decisions. Incorrect configuration can endanger business relationships or have legal ramifications. Conversely, by definition security involves dealing with enemies. That is, someone will try to counter your moves. Managing security configuration is not simply a matter of designing the right configuration and distributing it, contending only with Murphy's Law; instead, the administrator must contend with active attempts to subvert configurations, evade them, or drive the system into an improbable state not anticipated by the configuration. Note well that "enemies" can include a site's own employees.

Some form of centralized configuration management is necessary, then both for correct operation and security. While there are many ways to accomplish this, a number of requirements must be satisfied:

10.2.1 SECURITY

The configuration management scheme must be secure, in many senses of the word. Its databases must be protected against unauthorized access or modification, the communications with the managed hosts must be authenticated, integrity-protected, and often encrypted; the commands to the management system must be properly authenticated.

10.2.2 DATABASE-DRIVEN

Different machines have different roles; as such, they require different configurations. A desktop machine may have no need for a web server; depending on the user's needs, though, it may need a database server. Other machines — says, mail servers — may need neither, but may need other specialized software.

10.2.3 ROBUSTNESS

There *will* be configuration failures. Machines may be down during attempted updates; other, unmanaged changes may have occurred; changes — specifically including vendor-supplied patches—may fail to install. If the configuration changes are security-relevant, this could have serious consequences.

10.3 NEED OF CONFIGURATION

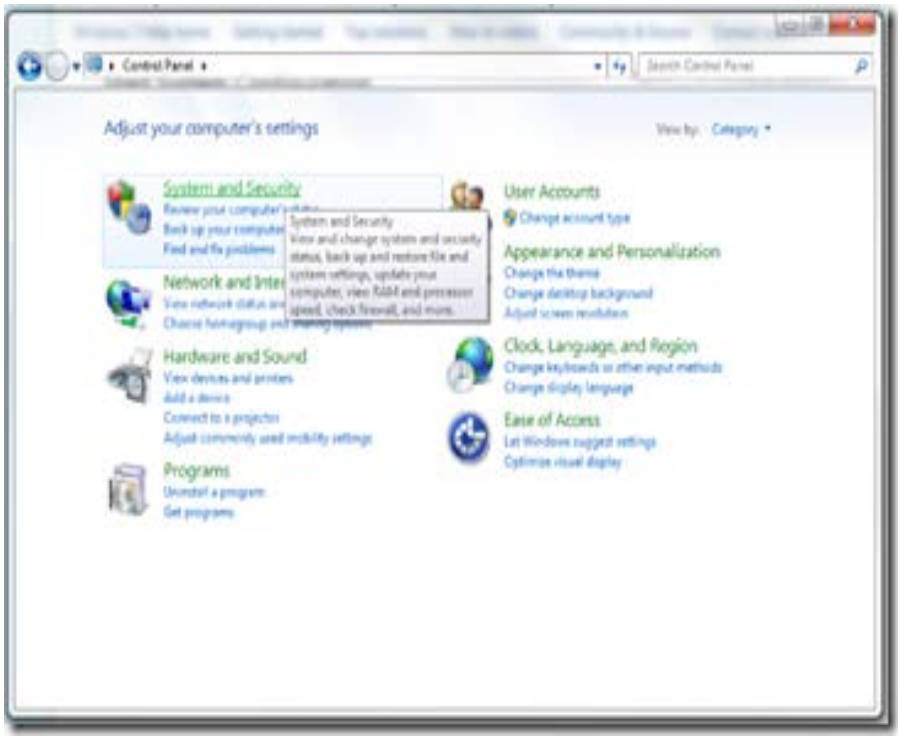
The need of software can be explained using the following:

- So that a hardware work properly and without errors.
- To reduce the probability of hardware crash.
- Efficient and effective use of memory.
- To reduce the idle time of the CPU.
- The correct version of the software module that one has to continue its coding.
- An accurate copy of the last year's version 4.1 of the TMY software package.
- The version of the design document matches the software version we are adapting to a new customer.
- The version of the software system is installed at ABC Industries.

- The changes which have been introduced in the version installed at the Industries' site.
- Changes introduced in the new version of the software.
- The full list of customers that use version of one's software.
- For surety that the version installed at some company must not include undocumented changes.
- Multiple people have to work on software that is changing
- More than one version of the software has to be supported:
  - ✓ Released systems
  - ✓ Custom configured systems (different functionality)
  - ✓ System(s) under development
  - ✓ Software on different machines & operating systems

10.4 WINDOWS CONFIGURATION

Windows configuration is a system utility to troubleshoot the Microsoft windows Startup process it can disable or enable Software, Drivers and windows. Services that run at startup or change boot parameters. Configuration will be applied during either user login or computer startup. It also provides the Status of the applied configuration which is related to System Configuration. Desktop central, in addition to windows configurations, also offers various Desktop Management Capabilities to meet the administration need of a Windows Networks. It is often used for speeding up the Microsoft windows startup process of the machine. According to Microsoft, MSConfig was not meant to be used as a startup management.



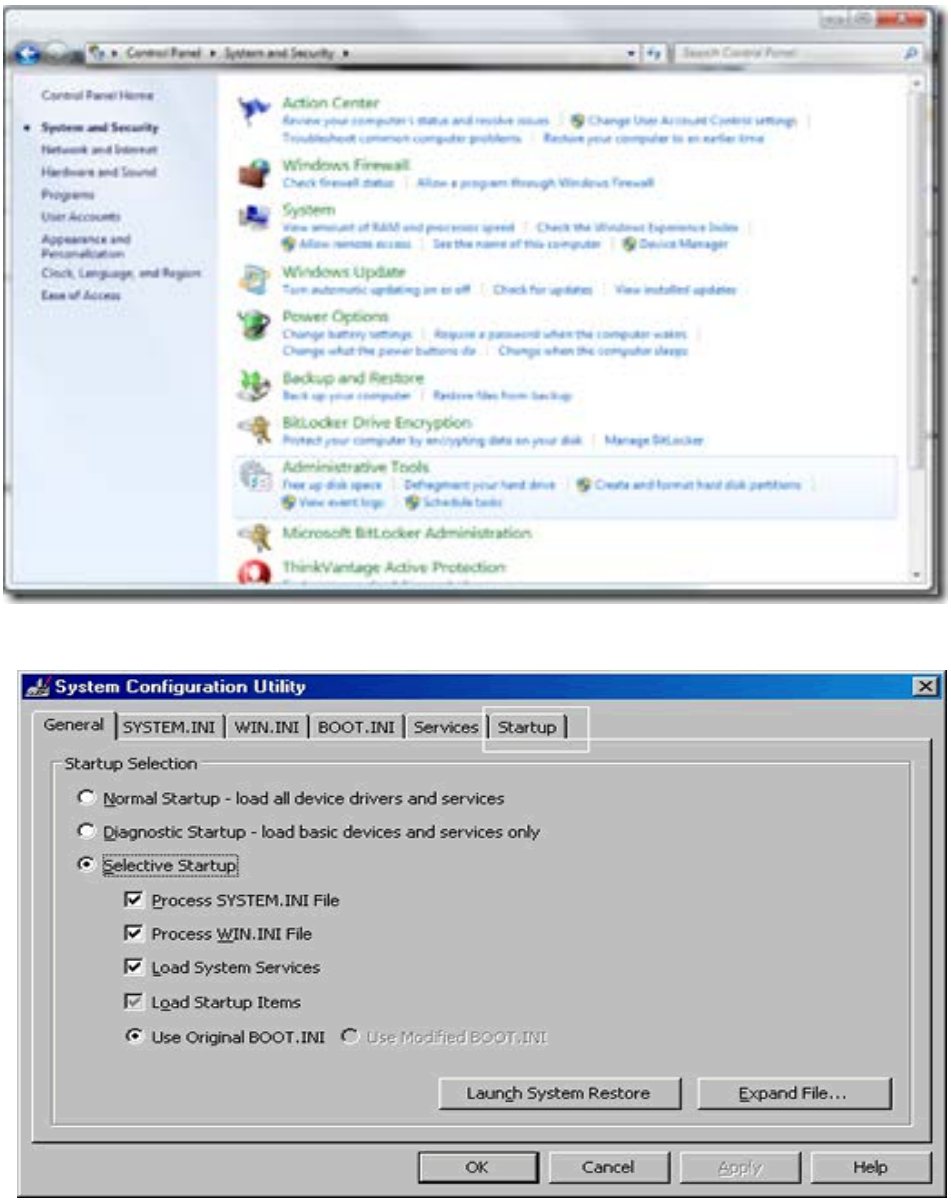


Figure 10.1: Windows Configuration

The Windows configuration falls under three categories: 1) Desktop Configuration 2) Computer Configuration 3) Application configuration

10.4.1 DESKTOP CONFIGURATION

What is available at the desktop level depends on your network configuration and specifically what group policy setting has been applied.

You control the information and the software that a user can access. The user does not need to be involved in this. Setting might also include the data that application need to preserve the user state such as a user's custom dictionary, lost files, and data that control the interface and the behavior of the application. It also includes setting path, environment variables, display properties, driver mapping, managing shortcut, configuring IP / shared printers, displaying message box, launching applications, launching games, and better interface.

10.4.2 COMPUTER CONFIGURATION

Computer configuration includes like managing local users, groups, windows' services, and scheduling application, manipulating registry entries, installing software configuring power schemes and executing custom scripts.

10.4.3 APPLICATION CONFIGURATION

Application configuration files contain settings specific to an application .this file contains configuration settings that the common language runtime reads and settings that the application can read. This type configuration includes configuring windows applications such as MS Office, Matlab, MS Outlook, Internet Explorer etc.

10.4.4 SECURITY CONFIGURATION

The security Configuration is a stand-alone snap-in tool that users can use to import one or more saved configuration to a private security database. Importing configurations builds a machine-specific security database that stores a composite configuration .you can apply this composite configuration to the computer and analyze the current system configuration against the stored composite configuration stored in the database. It includes configuring firewall settings, security policies, displaying legal messages and alerts.

10.4.5 USER CONFIGURATION

User group policy setting are the setting located under the user configuration node in group policy, which affect users and are obtained when a user logs on.

Note: Windows configuration is bundled with all versions of Microsoft Windows operating system since windows 98 except windows 2000 and windows vista.

10.5 .NET CONFIGURATION

Using the features of the .Net configuration system, you can configure all .Net application on an entire server, a single .Net application,

or Individual pages or applications, subdirectories. You can configure features, such as authentication modes, page caching compiler, custom errors, debug and trace option and much more.

The features of .Net configuration system can be applied only to .Net resources. E.g. forms authentication only restricts access to .Net files, not to static files or ASP classic files unless those resources are mapped to .NET files name extensions. Use the configuration features of internet information services (IIS) to configure other .NET resources.

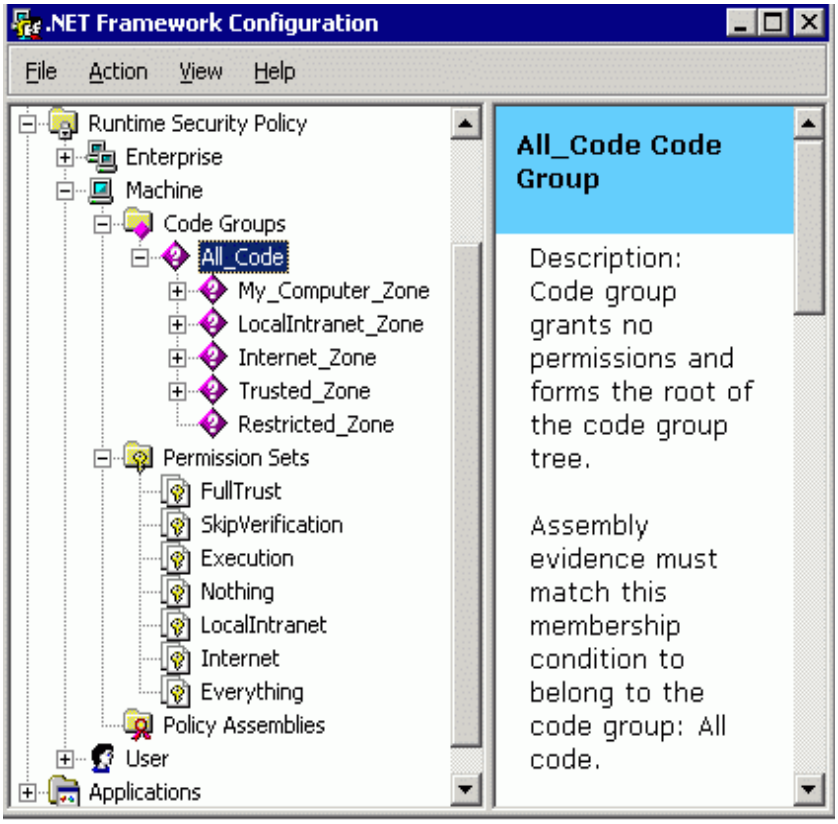


Figure 10.2: .NET Configuration

10.5.1 .NET CONFIGURATION FILES

.Net configuration data is stored in XML text files that are named as Web config. Web config files can appear in multiple directories in .Net Application. These files allow you to easily edit configuration data before during or after application are deployed on server. You can create and edit .NET configuration files by using standard text editors like the .NET MMC snap-in, the web site administration tool, or the .Net configuration API.

.NET configuration files keep application configuration setting separate from application code. Keeping configuration data separate from code makes it easy for you to associate settings with application, change

setting as needed after deploying an application, and extend the configuration schema.

10.5.2 CONFIGURATION FILES HIERARCHY AND INHERITANCE

Each Web.Config file applies configuration setting to the directory that it is in and to all of the child directories below it. Setting in child directories can override or modify setting that is specified in parent directories. Configuration setting in a web config file can optionally be applied to individual files or subdirectories by specifying a path in a location element. The root of the .NET configuration hierarchy is the *systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Web.Config* file, which includes settings that apply to all .NET application that run a specific version of the .NET Framework. As each .NET application inherit default configuration setting from the root Web.Config file, you need to create these config files only for settings that override the default settings.

10.5.3 .NET CONFIGURATION TOOLS

The .NET Framework Configuration tool is a Microsoft Management Console (MMC) snap-in that allows you to manage and configure assemblies in the Global Assembly Cache, adjust code access security policy, and adjust remoting services. Mscorcfg.msc is installed with the .NET Framework SDK.

Starting the .NET Framework Configuration Tool

To run Mscorcfg.msc from the Start menu we can do the following

1. On a computer running Windows 2000 Professional, click **Start**, point to **Settings**, and click **Control Panel**. Double-click **Administrative Tools**. In the **Administrative Tools** window, double-click **Microsoft .NET Framework Configuration**.
2. On a computer running Windows XP Professional, click **Start**, click **Control Panel**, click **Performance and Maintenance**, and click **Administrative Tools**. In the **Administrative Tools** window, double-click **Microsoft .NET Framework Configuration**.
3. On a computer running Windows 2000 Server or Windows Server 2003 family, click **Start**, point to **Programs**, and point to **Administrative Tools**. Click **Microsoft .NET Framework Configuration**.

To run Mscorcfg.msc from the Microsoft Management Console

1. Start the Microsoft Management Console by typing the following at a command prompt: **mmc**.

2. On the **File** menu, click **Add/Remove Snap-in** (or press CTRL+M) to display the Add/Remove Snap-in dialog box.

3. In the **Add/Remove Snap-in** dialog box, click **Add** to display the **Add Standalone Snap-in** dialog box.

4. In the **Add Standalone Snap-in** dialog box, select a version of the .NET Framework Configuration tool, and then click **Add**.

CHECK YOUR PROGRESS

- Define the term configuration.
- What is windows configuration?
- What is .NET configuration?

10.6 CACHING

Other aspect in the configuration is Caching. Cache memory is a small but fast memory meant to hold data for reuse in the near future. Maurice V. Wilkes introduced the concept to fulfill the speed gap between the CPU and main memory. He envisioned that cache memory would bridge that gap by using the principles of spatial and temporal locality. Recent advances in computer systems engineering have pushed cache memory to higher levels in the computer systems hierarchy. On each new level, the implementation details differ (to reflect the concrete requirements of the particular system level), but the essence stays the same (to reflect the chosen methods for using the principles of spatial and temporal locality). So, the principles of spatial and temporal locality help the concept survive and spread into the newly opened layers of the emerging computer system hierarchies.

Caching is the process of storing data in cache. Caching is an area of a computer's memory devoted to temporarily Storing recently used information. The content, which includes HTML pages, images, files, and Web objects, is stored on the local hard drive in order to make is faster for the user to access it . And, it helps in improving the efficiency of the computer and its overall performance e.g. the files you request by looking at the web page are stored on your disk in a cache.

Most caching occurs without the knowing to the user about it. For example when a user returns to a web page which has been recently accessed the browser. That can pull those files from the cache instead of the original server because it has stored the user's activity. The storage of information saves the user time by getting to it faster traffic on the networks.

10.6.1 LAYERS OF CACHING

A careful analysis on various system levels in current systems reveals seven layers of caching. These are as under:

1. CPU (in uniprocessor systems)
2. SMP (in shared memory multiprocessor systems)
3. DSM (in distributed shared memory systems)
5. DFM (in distributed file management and smart disk systems)
6. DPC (in distributed proxy cache systems)
7. WWW (on the World Wide Web level), and
8. IAI (on the Internet application and integration level).

In principle, this number could be higher (if a higher granularity of system analysis is implied) or lower (because different caching layers are highly correlated). We can define the principles of spatial and temporal locality on each layer.

1. CPU (In Uniprocessor Systems)

The traditional definition of spatial and temporal localities comes from the Uniprocessor environment. Spatial locality implies that the *next* data item in the address space is most likely to be used next, while temporal locality implies that the *last* data item used is most likely to be used next. Implementation is typically based on a fast but expensive memory (the price is affordable because, by definition, cache memory is small). Even if we use the same technology for the main memory and cache memory, the cache memory will be faster because smaller memories have a shorter access time. Recent research tries to split the CPU cache into two sub caches: one for spatial locality and one for temporal locality.

2. SMP (in shared memory multiprocessor systems)

On the SMP level, spatial and temporal locality continues to be present on the uniprocessor level. However, on the multiprocessor level, new forms of locality gain importance: processor locality, locality of shared data, and so forth. Implementation also includes mechanisms for maintaining data consistency, on either the hardware or software levels. Recent research concentrates on cache miss and bus traffic reduction by combining conventional and new approaches, such as the prefetch and injection approaches.

3. DSM (in distributed shared memory systems)

Caches on the DSM level also exist onthe CPU and SMP levels (a DSM system often consists of clusters implemented as SMP systems). Misses on the DSM level can be extremely costly; however, the caches on the DSM level have much more difficulty capturing locality.

4. DFM (in distributed file management and smart disk systems)

On the DFM level, caches can help implement several different applications (media servers, file distribution, and so forth). Spatial locality is present much more than temporal locality. Additional types of locality, stemming from the specific internal and external disk structure, can also be defined and used for better system efficiency. Recent research concentrates on the so-called smart disks, using different specialized resources to maximize performance

5. **DPC (in distributed proxy cache systems)**

   The DPC level uses caching in conjunction with protection. In addition to spatial locality (present to a smaller extent) and temporal locality (present to a larger extent), we can define and use many different types of locality: URL, geographical (if it can be defined), user, institutional, and so forth. Often, distinguishing the specifics of the distributed proxy cache is difficult; consequently, strategic errors in proxy cache design are possible.

6. **WWW (on the World Wide Web level)**

   On the WWW level, caches subdivide into client, server, and network caches. The client cache's primary goal is to handle data reusability, which, improves the Web latency. The server cache's primary goal is to reduce the server node workload, and the network cache's primary goal is to help clients benefit from the earlier accesses (to the same data) by other clients sharing the network cache. Types of locality in the three cache subtypes are the subject of ongoing research. Current implementations concentrate on problems in the domain of cache management (for example, replacement protocols) and cache cooperation (cooperation protocols).

7. **IAI (on the Internet application and integration level)**

   On the IAI level, we try to detect reusability and use the principles of locality in the systems responsible for Web oriented application and integration software. The existing types of locality depend on the specific types of software in use. Concrete implementations differ greatly and mainly concentrate on the issues of importance for cache management and cache replacement.

10.7 TYPES OF CACHING

As it has been discussed in previous topic Caching is a technique widely used in computing to increase performance by keeping frequently accessed or expensive data in memory. In context of web application, Caching is used to retain the pages or data across HTTP request and reuse them without the expense of recreating them. There are different types of caching.

10.7.1 PAGE OUTPUT CACHING

Page output caching refer to the ability of the web server to cache a certain webpage after user request in its memory so that further request for the same page will check for the cached page's validity and not result in resource usage and the page will be returned to user from cache. Caching the dynamic output generated by a request .sometimes it is useful to cache the output of a website even for a minute which will Result in a better performance for caching the whole page should have output.

Cache directive=<%@.

<%Output cache duration =”60”.

VaryByParam=”states”%>

10.7.2 FRAGMENT CACHING

Caches the portion of the page generated by the request. Some time it is not practical to cache the entire page , in such cases We can cache a portion of page

<%@.Output cache duration=”120”

varyByParam=”category ID; Selected ID%>

10.7.3 DATA CACHING

Caches the objects programmatically for caching .Net provides a Cache objects for eg; cache [“States”] = ds states.

10.7.4 APPLICATION CACHING

Application data caching is a mechanism for storing the data objects on cache. it has nothing to do with the page caching.

10.8 SQL CACHE INVALIDATION

SQL cache invalidation enables you to make the cache entry dependent On the database, so the cache entry will only be cleared When data in the data base is changed. SQL cache invalidation is two types:

1. Polling based invalidation.
2. Notification-based invalidation.

10.8.1 POLLING BASED INVALIDATION

This mechanism uses polling to check if a table has been updated since the page was cached. To enable table based caching we requires the following technique.

A) Enable notification for the database using the asp net regsql.exe tool

>aspnet_regsql.exe –S".\SQLExpress"-E-d"pubs"-ed

This only needs to be done once for each database.

B) Enable notification for the table(s) you want to have dependencies On using aspnet_regsql.exe tool.

>aspnet_regsql.exe –S".\SQLExpress"-E-d"pubs"-et-t"authors"

C) Register the notification in the configuration for the applications.

<system.web>

<Caching>

<sqlCacheDependency enabled ="true pollTime="1000">

<Database>

<add name="PubsDB" connectionStringName="Pubs"/>

</database>

</sqlCacheDependency>

</caching>

</system.web>

The poll time specifies how often the application checks to see whether the data has changed.

The following example uses output caching for a data-source using a table based notification.

10.7.2 NOTIFICATION-BASED CACHE INVALIDATION

This mechanism uses the query change notification mechanism of sql server 2005 to detect changes to the result of queries. Unlike polling based invalidation for sql server 7.0 and 2000 , notification based invalidation requires much less setup.

1. Unlike polling based validation, no <sqlCacheDependency> needs to be registered in your application's configuration furthermore, no special configuration using the aspnet_regsql.exe tool is needed.

2. A notification based dependency is configured on the OutputCache directive using the string command notification. This value indicates to ASP.NET that a notification based dependency should be created for the page or data source control.

   On a page

   <%@OutputCache Duration=”999999”SqlDependency=”CommandNotification”Vary ByParam”%>

   On a data-source control

   <asp: Sql Data Source

   EnableCaching=”ture”SqlCacheDependency=”CommandNotificati on”CacheDuration=”infinite…/>

3. System.Data.sqlClient.SqlDependency.Start () method must be called somewhere in the application before the first SQL Query is executed. This method could be placed in application Start () event in global.asax file.

Whenever a command is issued to sql server 2005, ASP.NET and ADO.NET will automatically create a cache dependency that listens to change notification sent from the sql server. As data is changed in sql server these notification will cause the cached queries to be invalidated on the web server.

CHECK YOUR PROGRESS

- What are types of caching?
- What is SQL Cache Invalidation?

10.9 SUMMARY

The fallout of the unit is to know the basic concepts of configuration. Configuration is nothing but step by step process for the successful implementation of hardware or software or both. Hardware and software, two are important categories. It is clear that configuration, and hence configuration management, are crucial to security. We assert that it is equally clear that even in moderate-size environments, it should not be done by hand. Too much can be wrong if done manually. Configuration is essential for software as well as hardware. Software configuration deals operating systems, application software, system software, and many more.

Moreover, you can say that new software may be at client-side as well as server side it plays a very important role.

Three elements are necessary for successful management of security configuration: clear policies; accurate knowledge of all computers and network elements; and proper management software. To be sure, all of these are needed for other types of configuration management. However, in a security setting, a failure can have consequences far beyond the failure of one device or system. Furthermore, the way a system is set up, or the assortment of components that make up the system. Configuration can refer to either hardware or software, or the combination of both. For instance, a typical configuration for a PC consists of 32MB (megabytes) main memory, a floppy drive, a hard disk, a modem, a CD-ROM drive, a VGA monitor, and the Windows operating system.

Many software products require that the computer have a certain *minimum configuration.* For example, the software might require a graphics display monitor and a video adapter, a particular microprocessor, and a minimum amount of main memory. When you install a new device or program, you sometimes need to configure it, which means to set various switches and jumpers (for hardware) and to define values of parameters (for software). For example, the device or program may need to know what type of video adapter you have and what type of printer is connected to the computer. Thanks to new technologies, such as plug-and-play, much of this configuration is performed automatically.

First, we must understand what configurations should be like, and how they should be set. This is not just a question of file contents and the like; the human element — how people understand and specify configurations — is at least as important.

Second, we need to be able to abstract and parameterize configurations. That is, we need to be able to create meta configurations, and merge these with the knowledge of the machines that actually exist. The difficulty of doing these raises a third issue: can we create configuration mechanisms that are more amenable to such specification? Today's systems were designed for hand configuration, with few nods to automation. Can we do better?

Fourth, we need effective ways to understand exactly what our networks really consist of. Note that this needs to be done in ways that protect privacy, not so much for business reasons as because often, consumer-grade machines are used for business purposes. It may be acceptable for corporate machines to respond to "who's out there, and what are your capabilities" messages; it certainly is not acceptable in a consumer environment.

10.10 TERMINALQUESTIONS

1. What do you mean by configuration? Explain.

2. Write the main advantages and disadvantages of configuration.
3. Give the types of configurations.
4. Explain .NET configuration.
5. Describe windows configuration.
6. What do you understand by caching and its types? Explain.

Uttar Pradesh Rajarshi Tandon Open University

Bachelor in Computer Application

BCA-E10

Client Server Technology

BLOCK

4

CLIENT SIDE AND SERVER SIDE LOGIN SERVICES

UNIT-11

HTML and JAVA Script

UNIT-12

ASP.NET Web Services

UNIT-13

AJAX

UNIT-14

Developing a Small Application Using ASP.NET

Course Design Committee

Dr. Ashutosh Gupta, **Chairman**
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Prof. R.S. Yadav **Member**
Dept. of Computer Science and Engineering, MNNIT, Allahabad

Ms. Marisha **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Member**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

Course Preparation Committee

Dr. Krishan Kumar **Author**
Assistant Professor,
Department of Comuter Science Faculty of Technology
Gurukula Kangri Vishwavidyalaya, Haridwar (UK)

Dr. V.K. Saraswat **Editor**
Director (IET, Khandare Campus)
Institute of Engineering and Technology
Dr. B.R. Ambedkar University, Agra-282002

Dr. Ashutosh Gupta,
Director (In-charge)
School of Computer and Information Science, UPRTOU, Allahabad

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (Computer Science)
School of Science, UPRTOU, Allahabad

©UPRTOU, Prayagraj-2020
ISBN : 978-93-83328-13-0

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2020.
Printed By : Chandrakala Universal Pvt. 42/7 Jawahar Lal Neharu Road, Prayagraj.

BLOCK INTRODUCTION

Block-4 basically contains four units which are related with ASP.NET web services, HTML and JavaScript, DHTML, AJAX, and a small application.

Unit-11 covers the basics of HTML and JavaScript. It explores HTML's role and change in the today's changing environment of Internet applications. HTML is basically a markup language used to develop static client-side programs. It also aims understanding the concept of HTML and JavaScript. It also gives the understanding of HTML Form Tag and elements within it. Moreover, this unit describes the themes to Customize a Site Web based security and ASP.NET authentication service. This unit gives the information about HTML, JavaScript, CSS, web based security, authorization services. As HTML has become crucial component for every web based application. With the increasing demand of client side validations JavaScript has also played a very important role in all kind of web application irrespective of the different technologies like Java, Dot Net, PHP etc.

DHTML or dynamic hypertext markup language is another advance important feature added recently which provided a new direction and robustness for the new web sites. CSS, JavaScript, and XML are three key elements of DHTL which makes HTML a powerful tool.

Unit-12 introduce web services, creating web services, invoking web services.Today, web services have become the crucial part of any field for better services. Almost, all organizations offer information on Internet. Interaction between user and programs is also necessary for all the function in web services. This unit briefly reviews web services as an application integration technology. Moreover, it defines the term web service and describes the web services model. In a normal application you need to write the Business logic repeatedly for the same requirements so due to the requirements you can write a single web service for multiple applications that allow an access method on any platform used.

Unit-13 explains the fundamental of AJAX and its relation with .NET. Ajax, which stands for *Asynchronous JavaScript and XML*, is a set of techniques for creating highly interactive web sites and web applications. The idea is to make what's on the Web appear to be local by giving you a rich user experience, offering you features that usually only appear in desktop applications. The emphasis in Ajax applications is to update the web page, using data fetched from the Internet, without refreshing the web page in the browser. You saw an example of that with Google Suggest, where a drop-down list appears in the browser without a page refresh.

Unit-14 discusses an application using ASP.NET for a real life example. project titled "Bluetooth Based Attendance Management System" is a small project has been developed using ASP.NET as a front-end and MS

Access as a back-end. It is an attendance management system which can be used School, College, University etc. This project aims to train the basic concepts of ASP.NET for the students so that they cloud be familiar with the environment of the industry. The project has been developed using the traditional waterfall model following all the required steps of software engineering like SRS, feasibility study, design, coding and implementation, testing, and finally maintenance.

UNIT-11 HTML & JAVASCRIPT

Structure

11.0 INTRODUCTION

This unit gives the information about HTML, JavaScript, CSS, web based security, authorization services. As HTML has become crucial component for every web based application It is also necessary to know about its basic characteristics, syntax, advantages and its usefulness. With the increasing demand of client side validations JavaScript has also played a very important role in all kind of web application irrespective of the different technologies like Java, Dot Net, PHP etc.

DHTML or dynamic hypertext markup language is another advance important feature added recently which provided a new direction and robustness for the new web sites. CSS, JavaScript, and XML are three key elements of DHTL which makes HTML a powerful tool.

Further theses technologies are important in another sense that these are common for all. Like XML is being used for data exchange and JavaScript is used for client-side validations, popup menus, windows etc.

JavaScript has many predefined functions which are used for different works like printing etc.

HTML is a tag based language and it uses many tags. On the other hand XML also uses user defined tags. Form tag is also very important to submit the information from client to server. It has many attributes like to create different types of buttons. Furthermore, with the popularity of Internet security has become a crucial issue for the web applications. Hence web security is another area which must be carefully handled.

Last but not the least Login controls are also necessary to understand so theses are discussed in this unit. Without the knowledge of login controls it not possible to develop a good web application. Also the session handling is done using theses controls.

11.1 OBJECTIVES

At the end of this unit you would come know about the following things:

- Hypertext Markup Language
- The tags of HTML like form, head, body, table etc
- Creation of user defined functions using JavaScript
- Issues in HTML and JavaScript
- Working with CSS
- Themes to Customize a website
- Security in Internet based applications
- Login Controls
- Authorizing Users

11.2 FUNDAMENTAL HTML

Html stands for Hyper Text Markup Language. Html is mainly used for designing the client side web pages, this is a static page it means you can only view Html page not give request and not get response from server. Using Html pages browser get user information through form (This is an Html element). It also aims to develop static web pages in which it changes are not possible. Html provides so many elements/tags like <p>, <img>, <h1>, <form>, <table> etc. to design a web page. On the other hand hypertext is the text which contains various links to other texts. The hypertext pages are interconnected by hyperlinks. When mouse is clicked on these links a new web page appears.

Moreover any tag based language known as markup language, for example gml, sgml, html, xml etc. All the markup languages have several tags. But XML (Extensible markup language) has user defined tags also. Like other language Html also has different versions starting from 1.0, 2.0, 3.0, 4.0, 5.0 etc. The version 5.0 is the present version. As a beginning it was incorporated in Netscape navigator browser. Tim Berners-Lee is known as father of Html. Earlier It was known as SGML. The first publicly available description of HTML was a document called "HTML Tags", first described, on the Internet by Berners-Lee in late 1991.

11.2.1 FEATURES FOR HTML

The important features of Html are:

- Html is used to develop a static page.
- Html is not a case sensitive language.
- It is an error free language.
- It is interpreted not compiled.
- Html is simple to understand and easy language.
- It provides facilities to add audio, video, image on web pages.
- Html is the platform independent language, it can be run on any platform like window, linux, Mac.
- Each and every elements of html should be enclosed within the angular brackets (<>).
- Html programs are executed by the interpreter of the browser software.
- Html program are saved with .htm or .html extension.
- The current version of Html is Html 5.0

11.2.2 HTML TAG

Tags in any language are the main building blocks. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

Syntax:

<tag> content </tag>

11.2.3 HTML TAG EXAMPLES

HTML is not case sensitive language so you can write Html tags in lowercase or uppercase letters. Some Html tags are given below:

Table 11.1: Paired Tags

Name of tag	Meaning
<html> ... </html>	Delimit the beginning and end of the entire Hypertext Markup Language (HTML) document.
<body> ... </body>	Delimit the beginning and end of the document body.
<head>	Used for title, java script, css code etc
<form>	Used to submit data
<h1>	Used for heading
<image>	Used to insert image
<a >... 	Create a hyperlink anchor (href attribute) or fragment identifier (id attribute).
<table>	Used to insert table
<abbr> ... </abbr>	The enclosed text is an abbreviation.
<caption> ... </caption>	Define a caption for a table.
...	Center the enclosed text.
<frame> ... </frame>	Define a frame within a frameset.

Table 11.2: Single Tags

Name of tag	Meaning
<hr>	Break the current text flow and insert a horizontal rule.
<p>	Used to keep the texts in form of the paragraph
 	Break the current text flow, resuming at the beginning of the next line.

<embed>	Embed an application in a document.
<Font Color = </Font>	Used to change the properties of a font
<image>	Used to insert image

11.3 FORM TAG

This element is used to create a form on html page. Which is a logical grouping of controls available on the page. Form contains controls such as text fields, email field, password fields, checkboxes, button, radio buttons, submit button, menus etc.

Place a form anywhere inside the body of a document, with its elements enclosed by the <form> tag and its respective end tag (</form>). You can, and we recommend you often do, include regular body content inside a form to specially label user-input fields and to provide directions.

Browsers flow the special form elements into the containing paragraphs as though they were small images embedded into the text. There aren't any special layout rules for form elements, so you need to use other elements, such as tables and stylesheets, to control the placement of elements within the text flow.

You must define at least two specialform attributes, which provide the name of the form's processing server and the method by which the parameters are to be sent to the server. A third, optional attribute lets you change how the parameters get encoded for secure transmission over the network.

The HTML **<form>**tag is used to create an HTML form and it has following syntax:

Syntax:

```
<form>
//input controls e.g. textfield, checkbox, button, radiobutton
</form>
```

11.3.1 FORM ATTRIBUTES

Attribute of form tag are: accept, action, charset, class, dir, enctype, id, lang, method, name, onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onReset, onSubmit, style, target, title. Apart from common attributes, following is a list of the most frequently used form attributes:

Table-11.3: Form Attributes

Attribute	Description
Name	are used for provide a name to the form
action method	Backend script ready to process your passed data. Method to be used to upload data. The most frequently used are GET and POST methods.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
enctype	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are: application/x-www-form-urlencoded - This is the standard method most forms use in simple scenarios. mutlipart/form-data - This is used when you want to upload binary data in the form of files like image, word file etc.

11.3.2 HTML FORM CONTROLS

There are different types of form controls that you can use to collect data using HTML form:

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

11.3.3 TEXT INPUT CONTROLS

There are three types of text input used on forms:

Single-line text input controls - This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>**tag.

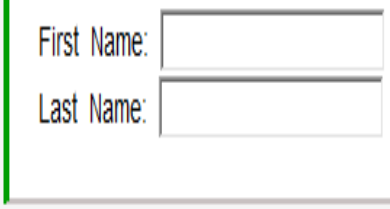
Password input controls - This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.

Multi-line text input controls - This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>**tag.

Example

```
<html>
<head>
 <form>
   First Name: <input type="text" name="firstname"/> <br/>
   Last Name:  <input type="text" name="lastname"/>
 </form>
</body>
</html>
```

Result



Check Your Progress

- What do you understand by HTML
- Write the syntax and working of some commonly used HTML tags.

11.4 HTML LISTS

Two primary lists are available in HTML, ordered and unordered.

11.4.1 UNORDERED LISTS

The unordered list is used when the order of item is not significant. Rather, numbering of the items is usually marked with bullets like shaded disk, circle or square. Default bullet is shaded disk. The beginning and ending tags for unordered list are <UL> and </UL> respectively. Each item is identified by a list item tag <LI>.

Eg:

<UL>

<LI> pick up mail

<LI> walk the dog

<LI> water the plant

<LI> clean the room

</UL>

Output:

pick up mail

walk the dog

water the plant

clean the room

11.4.2 ORDERED LISTS

In this category, order of the items does matter. The elements are prefixed by a symbol that denotes their order in the list. By default Arabic numbers are automatically made by the browser. The beginning and ending tags for an ordered lists are <OL> and </OL>. The beginning tag for each list item is <LI>. By default each list starts counting at 1.

Attributes of the Ordered List

<OL start= integer> Start tells the browser what number to start counting at.

<OL type= 1|A|a|I|i> Sets the type of numbering to use.

Example:

<OL Start=5 >

<LI> tap gently with a spoon

<LI> lift mold off of gelatin

<LI> turn left at the first light

Output:

5. tap gently with a spoon

6. lift mold off of gelatin

7. turn left at the first light

11.5 JAVASCRIPT

11.5.1 INTRODUCTION

JavaScript is a object-based scripting language and it is light weighted. It is first implemented by Netscape (with help from Sun Microsystems). JavaScript was created by Brendan Eich at Netscape in 1995 for the purpose of allowing code in web-pages (performing logical operation on client side). It is not compiled but translated. JavaScript Translator is responsible to translate the JavaScript code which is embedded in browser.

Netscape first introduced a JavaScript interpreter in Navigator 2. The interpreter was an extra software component in the browser that was capable of interpreting JavaSript source code inside an HTML document. This means that the web page developers do not need software other than a text editor to develop any web page. The basic characteristics are:

- JavaScript is a lightweight, interpreted programming language
- Designed for creating network-centric applications
- Complementary to and integrated with Java
- Complementary to and integrated with HTML
- Open and cross-platform

11.5.2 WHY WE USE JAVASCRIPT?

Using HTML you can only design a web page but you cannot run any logic on web browser like addition of two numbers, checking any condition, looping statements (for, while), decision making statement (if-else) at client side. All these are not possible using HTML. Therefore, to perform all such tasks at client side you need a scripting language called JavaScript. JavaScript lets developers to develop client-side as well as server- side programs. This languge is a very powerful tool to put the client-side validations. Some features are:

- JavaScript is very easy to implement. All you need to do is put your code in the HTML document and tell the browser that it is JavaScript.
- JavaScript works on web users' computers — even when they are offline!

- JavaScript allows you to create highly responsive interfaces that improve the user experience and provide dynamic functionality, without having to wait for the server to react and show another page.
- JavaScript can load content into the document if and when the user needs it, without reloading the entire page — this is commonly referred to as Ajax.
- JavaScript can test for what is possible in your browser and react accordingly — this is called Principles of unobtrusive JavaScript or sometimes defensive scripting.
- JavaScript can help fix browser problems or patch holes in browser support — for example fixing CSS layout issues in certain browsers.

11.5.3 WHERE IT IS USED?

It is used to create interactive websites. It is mainly used for:

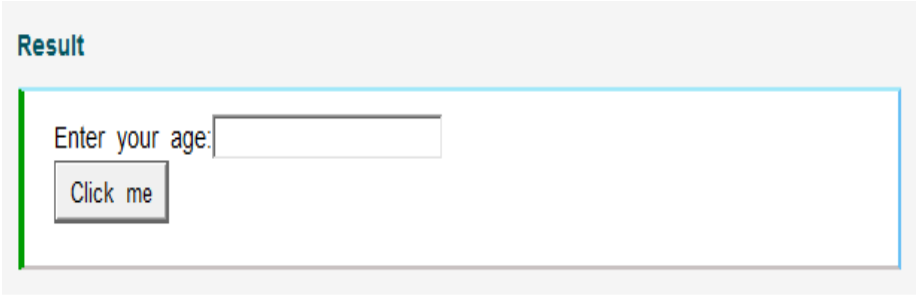
- Client-side validation
- Dynamic drop-down menus
- Displaying data and time
- Build small but complete client side programs
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

11.5.4 JAVASCRIPT SAMPLE PROGRAM

Example: Verify age of any person, if age is greater than 18 show message adult otherwise show under 18

Example

```
<html>
<head>
<script>
function verify(){
var no;
no=Number(document.getElementById("age").value);
if(no<18)
{
alert("Under 18");
}
else
{
alert("You are Adult");
}
}
</script>
</head>
<body>
Enter your age:<input id="age"><br />
<button onclick="verify()">Click me</button>
</body>
</html>
```



11.5.5 BENEFITS OF JAVASCRIPT

1. Associative arrays.
2. Loosely typed variables.
3. Regular expression.
4. Objects and classes.
5. Highly evolved date, math and string libraries.
6. W3C DOM support in the JavaScript.

11.5.6 DISADVANTAGES OF JAVASCRIPT

1. Developer depends on the browser support for the JavaScript.
2. There is no way to hide the JavaScript code in case of commercial application.

11.6 WORKING WITH CSS

CSS stands for Cascading Style Sheets. Styles define how to display the HTML elements. Cascading Style Sheets (CSS) is a rule based language that applies styling to HTML elements. We write CSS rules in Html elements (<p>, <img>), and modify properties of those elements such as color, background color, width, border thickness, font size, etc. CSS rule, is made up of two parts: Selector & Declaration. These are also shown in figure11.1.

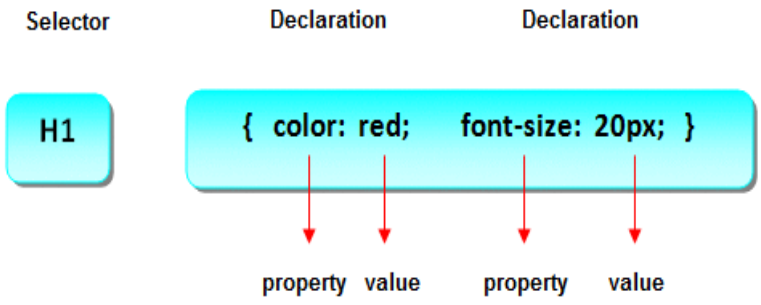


Figure 11.1: Selector & Declaration

Conceptually, the Selector part of CSS Identifies the HTML elements that the rule will be applied to, identified by the actual element name, e.g. <body>, <p>, <h1>. And the Declaration part contains property and value. For example; suppose that we want size of our text 10px then it would be declared as font-size: 10px. Here font-size is properties and 10px is its value, and all this declaration is called declaration. The declaration is also split into two parts, they are separated by a colon " : ".

Syntax

font-size:15px;

In the above code properties (font-size) and value (15px) is separated by colon ":". All the CSS syntax is combination of these fives things:

- Selector
- property/value
- declaration
- declaration block
- curly braces

11.6.1 SELECTOR

Identifies the HTML elements that the rule will be applied to, identified by the actual element name, e.g. <body>, <p>, <h1>

Example

```
H1{
color: red;
font-size:10px;
}
```

11.6.2 DECLARATION

Declaration part contains property and value. Example: suppose that we want size of our text 10px then it declare as font-size:10px. Here font-size is properties and 10px is there value, and all this declaration is called declaration. Declaration Block is multiple declaration lines including the curly braces. The declaration is also split into two parts, they are separated by a colon " : "

11.6.3 PROPERTY & VALUE

The property is the style attribute you want to change and value is value of attribute. Example: suppose that we want to change size of our text 10px and color is red then it declared as:

font-size:10px; color: red.

Here font-size is property and 10px is the value and color is also property and red is there value.

Note : If there is only one property - value pair in the declaration, we do not need to end it with a semicolon. However, because a declaration can consist of several property - value pairs, and each property - value pair within a rule must be separated by a semicolon.

11.6.4 ELEMENTS OF STYLE

At the simplest level, a style is nothing more than a rule the browser follows to render a particular HTML or XHTML tag's contents. Each tag has a number of style properties associated with it, whose values define how that tag is rendered by the browser. A rule defines a specific value for one or more properties of a tag. For example, most tags can have a color property, the value of which defines the color in which the modern GUI browser should display the contents of the tag. Other properties include fonts, line spacing, margins, borders, sound volume, and voice etc.

There are three ways to attach a style to a tag:

- Inline style
- Document level style
- External style sheet

Inline Style: The style attribute

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property. The following syntax shows how to change the colour and the left margin of a <h1> element.

```
<h1 style="color:blue; margin-left:30px;">This is a heading</h1>
```

Internal style

An internal style sheet may be used if one single page has a unique style. Internal styles are defined within the <style> element, inside the <head> section of an HTML page.

Example:

```
<head>
<style>
body {
background-color: linen;
}
h1{
```

```
color: maroon;
margin-left: 40px;
}
</style>
</head>
```

External style

With an external style sheet, you can change the look of an entire website by changing just one file! Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section.

Example

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
body {
background-color: lightblue;
}
h1 {
color: navy;
margin-left: 20px;
}
```

11.7 THEMES TO CUSTOMIZE A WEBSITE

Website is one place where people judge by the cover, and are mostly right. You can customize it further by choosing your own background image if you want. That is why a good looking business website has a significant effect on the business. *Zoho Sites* had already made it easy for anyone to build such business websites easily. Today, we are making it even friendlier on the design front. There are many ways to do it like a visual editor.

Till now, theme-customization could only be done with a bit of CSS. But not anymore. You can now customize visual properties of a theme, such as background image, menu, color, size and font face of text without a HTML/CSS, of course. You can preview the changes as and when you make them. What's exciting about this visual editor is, using it, you can create many different versions of the same theme. The themes in our gallery are ready-to-use, but you can re-paint them, so that they look

just how you want, in every way. The customization of themes, together with flexible layouts and drag-drop builder, makes Zoho Sites the easiest way to create a business website, all by you.

Moreover, the new website builder has a selection of themes and we can edit the HTML of a page, but there does not seem to be a way to alter the header or footer. How can this be done? Can we create a new theme? I found following way to alter the style sheet of an existing template (e.g. the online-shop).

- On the Website go to: Customize -> HTML-Editor
- Choose 'Main layout' from the selection box
- Add your custom style sheet just before the closing </head> tag

Also you can change the footer if you select 'Automatic footer' and 'Footer Copyright' from the selection box.

Note: Take a look here http://www.slideshare.net/openobject/odoo-create-themes-for-website. You basically create a new theme in your custom add-ons folder with the structure explained there and select it later.

Check Your Progress

- What do you understand by JavaScript?
- Write a small script for login and password validation using JavaScript.
- How the theme of a web site can be changed?

11.8 WEB BASED SECURITY

Web application security is a branch of Information Security that deals specifically with security of websites, web applications and web services. At a high level, Web application security draws on the principles of application security but applies them specifically to Internet and Web systems. With the inception of Web 2.0, increased information sharing through social networking and increasing business adoption of the Web as a means of doing business and delivering service, websites are often attacked directly. Hackers either seek to compromise the corporate network or the end-users accessing the website by subjecting them to drive-by downloading. As a result, industry is paying great attention to the security of the web applications themselves in addition to the security of the underlying computer network and operating systems.

Basically, security is fundamentally based on people and processes; there are a number of technical solutions to consider when designing, building and testing secure web applications. At a high level, these solutions include:

- Black box testing tools such as Web application security scannersvulnerability scanners and penetration testing software.
- White box testing tools such as static source code analyzers.
- FuzzingTools used for input testing.
- Web application security scanner (vulnerability scanner).
- Web application firewalls (WAF) used to provide firewall-type protection at the web application layer.
- Password cracking tools for testing password strength and implementation.

11.9 ASP.NET AUTHENTICATION SERVICE

ASP.NET implements additional authentication schemes using authentication providers, which are separate from and apply only after the IIS authentication schemes. ASP.NET supports the following authentication providers:

- Windows (default)
- Forms
- Passport
- None

To enable an authentication provider for an ASP.NET application, use the authentication element in either machine.config or Web.config as follows:

```
<system.web>
<!-- mode=[Windows|Forms|Passport|None] -->
<authentication mode="Windows" />
</system.web>
```

Each ASP.NET authentication provider supports an OnAuthenticate event that occurs during the authentication process, which you can use to implement a custom authorization scheme. The primary purpose of this event is to attach a custom object that implements the IPrincipal Interfaceto the context. Which ASP.NET authentication provider you use typically depends upon which IIS authentication scheme you choose. If you are using any of the IIS authentication schemes other than Anonymous, you will likely use the Windows authentication provider. Otherwise, you will use Forms, Passport, or None.

11.10 ASP.NET LOGIN CONTROLS

ASP.NET provides robust login (authentication) functionality for ASP.NET Web applications without requiring programming. The default

Visual Studio project templates for Web applications and for Web sites include prebuilt pages that let users register a new account, log in, and change their passwords For information about how to use the built-in login page templates, see Walkthrough: Creating an ASP.NET Web Site with Basic User Login.

You can also create your own pages that you can add ASP.NET login controls to in order to add login functionality. To use the login controls, you create a Web pages and then add the login controls to them from the **Toolbox**.

Typically, you restrict access to ASP.NET pages by putting them into a protected folder. You then configure the folder to deny access to anonymous users (users who are not logged in) and to grant access to authenticated (logged-in) users. (The default project template for Web projects includes a folder named Accounts that is already configured to allow access only to logged-in users.) Optionally, you can assign users to roles and then authorize users to access specific Web pages by role.

By default, login controls integrate with ASP.NET membership and ASP.NET forms authentication to help automate user authentication for a Web site. For information about how to use ASP.NET membership with forms authentication, see Introduction to Membership.

By default, the ASP.NET login controls work in plain text over HTTP. If you are concerned about security, use HTTPS with SSL encryption. For more information about SSL, see Configuring SSL on a Web Server or a Web Site in the IIS documentation.

11.11 AUTHORIZING USERS

Authorization determines whether an identity should be granted access to a specific resource. In ASP.NET, there are two ways to authorize access to a given resource.

11.11.1 FILE AUTHORIZATION

It is performed by the FileAuthorizationModule. It checks the access control list (ACL) of the .aspx or .asmx handler file to determine whether a user should have access to the file. ACL permissions are verified for the user's Windows identity (if Windows authentication is enabled) or for the Windows identity of the ASP.NET process. For more information, see ASP.NET Impersonation.

11.11.2 URL AUTHORIZATION

It is performed by the UrlAuthorizationModule, which maps users and roles to URLs in ASP.NET applications. This module can be used to selectively allow or deny access to arbitrary parts of an application (typically directories) for specific users or roles.

11.11.3 USING URL AUTHORIZATION

With URL authorization, you explicitly allow or deny access to a particular directory by user name or role. To do so, you create an authorizationsection in the configuration file for that directory. To enable URL authorization, you specify a list of users or roles in the allow or deny elements of the authorization section of a configuration file. The permissions established for a directory also apply to its subdirectories, unless configuration files in a subdirectory override them.

The following shows the syntax for the authorization section:

<authorization>

<[allow|deny] users roles verbs />

</authorization>

The allow or deny element is required. You must specify either the users or the roles attribute. Both can be included, but both are not required.The verbs attribute is optional. Theallow and deny elements grant and revoke access, respectively. Each element supports the attributes shown in the following table:

Table 11.4: Attribute and Their Meaning

Attribute	**Description**
Users	Identities the targeted identities (user accounts) for this element. Anonymous users are identified using a question mark (?). All authenticated users are specified using an asterisk (*).
Roles	Identities a role (a RolePrincipal object) for the current request that is allowed or denied access to the resource.
Verbs	Defines the HTTP verbs to which the action applies, such as GET, HEAD, and POST. The default is “*”, which specifies all verbs.

11.11.4 RULES APPLIED

Rules contained in application-level configuration files take precedence over inherited rules. The system determines which rule takes precedence by constructing a merged list of all rules for a URL, with the most recent rules (those nearest in the hierarchy) at the head of the list.

Given a set of merged rules for an application, ASP.NET starts at the head of the list and checks rules until the first match is found. The default configuration for ASP.NET contains an <allow users="*"> element, which authorizes all users. (By default, this rule is applied last.) If no other authorization rules match, the request is allowed. If a match is found and the match is a deny element, the request is returned with the 401 HTTP status code. If an allow element matches, the module allows the request to be processed further.

In a configuration file, you can also create a location element to specify a particular file or directory to which settings in that the location element should apply.

Check Your Progress

- Define the term web based security.
- Write the significance of Login controls.

11.12 SUMMARY

HTML is a tag based language and it uses many tags. On the other hand XML also uses user defined tags. Form tag is also very important to submit the information from client to server. It has many attributes like to create different types of buttons. Furthermore, with the popularity of Internet security has become a crucial issue for the web applications. Hence web security is another area which must be carefully handled.

Using Html pages browser get user information through form (This is an Html element). It also aims to develop static web pages in which it changes are not possible. Html provides so many elements/tags like <p>, <img>, <h1>, <form>, <table> etc. to design a web page.

The form tag i.e. <form> is used to create a form on html page. Which is a logical grouping of controls available on the page. Form contains controls such as text fields, email field, password fields, checkboxes, button, radio buttons, submit button, menus etc. There are different types of form controls that you can use to collect data using HTML form: Text Input, Checkboxes, Radio Box, Select Box, File Select boxes, Hidden, Clickable Buttons, Submit and Reset Button.

Java Script is a object-based scripting language and it is light weighted. It is first implemented by Netscape (with help from Sun Microsystems). JavaScript was created by Brendan Eich at Netscape in 1995 for the purpose of allowing code in web-pages (performing logical operation on client side). It is not compiled but translated. JavaScript Translator is responsible to translate the JavaScript code which is embedded in browser.

CSS stands for Cascading Style Sheets. Styles define how to display the HTML elements. Cascading Style Sheets (CSS) is a rule based language that applies styling to HTML elements.

Authorization determines whether an identity should be granted access to a specific resource. In ASP.NET, there are two ways to authorize access to a given resource.

11.13 TERMINALQUESTIONS

1. What is HTML? Why we Use HTML?
2. Explain the difference between radio button and checkbox in HTML?
3. What is form Tag in HTML? Give example using with form Tag.
4. What is JavaScript? Where JavaScript is used?
5. JavaScript is which type of language? Name those languages in which JavaScript is used.
6. How do we insert JavaScript code onto a web page?
7. What is CSS? Why we use CSS?
8. Is CSS is case Sensitive? Compare margin and padding.
9. What is web based Security? Name some technologies which are used in web Security.
10. What is ASP.NET Authentication Service?
11. What are ASP.NET login controls?
12. Write a short note on Authorization.

UNIT-12 ASP.NET WEB SERVICES

Structure

12.0 INTRODUCTION

Today, web services have become the crucial part of any field for better services. Almost, all organizations offer information on Internet. Interaction between user and programs is also necessary for all the function in web services. This unit briefly reviews web services as an application integration technology. Moreover, it defines the term web service and describes the web services model. In a normal application you need to write the Business logic repeatedly for the same requirements so due to the requirements you can write a single web service for multiple applications that allow an access method on any platform used.

Usually, web services are ready to move for program to-program interactions. Web Services allow companies to reduce the cost of doing e-business, to deploy solutions faster and to open up new opportunities. The key to reaching this new horizon is a common program-to-program communications model, built on existing and emerging standards such as Hypertext Markup language (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI).

Web Services allow applications to be integrated more rapidly, easily and less expensively than ever before. Integration occurs at a higher level in the protocol stack, based on messages centered more on service semantics and less on network protocol semantics, thus enabling loose integration of business functions. These characteristics are ideal for connecting business functions across the web, both between enterprises and within enterprises. They provide a unifying programming model so that application integration inside and outside the enterprise can be done with a common approach, leveraging a common infrastructure. The integration and application of Web Services can be done in an incremental manner, using existing languages and platforms and by adopting existing legacy applications.

Furthermore, Web Services compliment Java-2 platform, Enterprise Edition (J2EE), Common Object Request Broker Architecture (CORBA) and other standards for integration with more tightly coupled distributed and non distributed applications. Web Services are a technology for deploying and providing access to business functions over the Web; J2EE, CORBA and other standards are technologies for implementing Web Services.

Although early use of Web Services is peer-wise and ad hoc, it still addresses the complete problem of program-to-program communications including describing, publishing and finding interfaces. And, as the use of Web Services grows and the industry matures, more dynamic models of application integration will develop. Eventually, systems integration through Web Services will happen dynamically at runtime. Just-in-time integration will herald a new era of business-to-business integration over the Internet.

12.1 OBJECTIVES

At the end of this unit you would come know about the following:

- Web Services
- Web services Model
- Roles in Web Services Architecture
- Artifacts of a Web Service
- Web Services Development Life Cycle
- Architecture Overview
- Web Service Flow

12.2 DEFINITION OF WEB SERVICES

Basically, a web service is a kind of software which is available over Internet. This uses extensible markup language i.e. XML. XML is

used to encode for every kind of communication in web service. As we know that XML is used to exchange the data over the Internet hence, for a web service XML plays a very important role and makes software programs very friendly and useful for the users. For example, a web service client makes an XML request and waits for an XML response. For the request a user sends a XML message. Similarly it waits for an XML message in response. In other words, a web service is an interface that describes a collection of operations that are network accessible through standardized XML messaging system. Web services can convert easily the existing applications in web applications.

Moreover, same application and platform meanthat one can create a web service in any language, such as Java or other languages and that the language web service can be used in a .Net based application and also a .Net web service or in another application to exchange the information. The method in web services always start with [webMethod] attributes, it means that it is a web method that is accessible anywhere, the same as my web application.

A Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location. The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written. This allows and encourages Web Services-based applications to be loosely coupled, component-oriented, cross-technology implementations. Web Services fulfill a specific task or a set of tasks. They can be used alone or with other Web Services to carry out a complex aggregation or a business transaction.

12.3 THE WEB SERVICES MODEL

The Web Services architecture is based upon the interactions among three roles:

- service provider
- service registry
- service requestor

The interactions involve the publishing, finding and binding operations. Together, these roles and operations act upon the Web Services artifacts: the Web service software module and its description. In a typical scenario, a service provider hosts a network-accessible software module (an implementation of a Web service). The service provider defines a service description for the Web service and publishes it to a service requestor or service registry. The service requestor uses a find operation to retrieve the service description locally or from the service

registry and uses the service description to bind with the service provider and invoke or interact with the Web service implementation. Service provider and service requestor roles are logical constructs and a service can exhibit characteristics of both. Figure 12.1 illustrates these operations, the components providing them and their interactions.

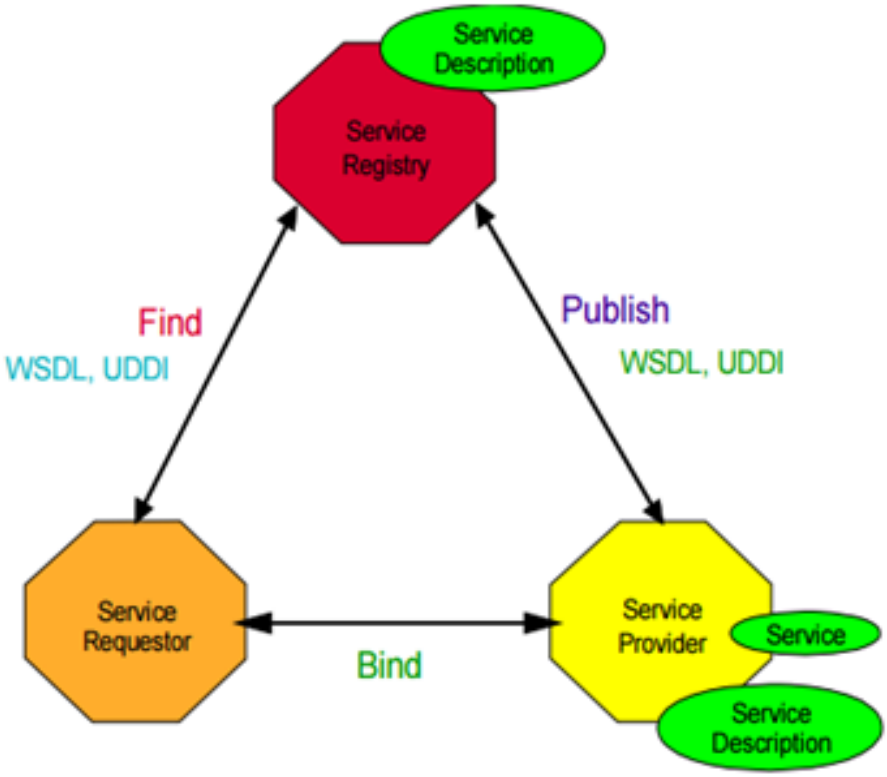


Figure 12.1: Web services roles, operations and artifacts

Check Your Progress

- Define the term web service.
- Write the names of three roles on which the web service architecture is based.

12.4 ROLES IN WEB SERVICE ARCHITECTURE

12.4.1 SERVICE PROVIDER

From a business perspective, this is the owner of the service. From an Architectural perspective, this is the platform that hosts access to the service.

12.4.2 SERVICE REQUESTOR

From a business perspective, this is the business that requires certain functions to be satisfied. From an architectural perspective, this is the application that is looking for and invoking or initiating an interaction with a service. The service requestor role can be played by a browser driven by a person or a program without a user interface, for example another Web service.

12.4.3 SERVICE REGISTRY

This is a searchable registry of service descriptions where service providers publish their service descriptions. Service requestors find services and obtain binding information (in the service descriptions) for services during development for static binding or during execution for dynamic binding. For statically bound service requestors, the service registry is an optional role in the architecture, because a service provider can send the description directly to service requestors. Likewise, service requestors can obtain a service description from other sources besides a service registry, such as a local file, FTP site, Web site, Advertisement and Discovery of Services (ADS) or Discovery of Web Services (DISCO).

12.5 OPERATIONS IN WEB SERVICE ARCHITECTURE

For an application to take advantage of Web Services, three behaviors must take place: publication of service descriptions, lookup or finding of service descriptions, and binding or invoking of services based on the service description. These behaviors can occur singly or iteratively. In detail, these operations are:

12.5.1 PUBLISH

To be accessible, a service description needs to be published so that the service requestor can find it. Where it is published can vary depending upon the requirements of the application (see “Service Publication” for more details).

12.5.2 FIND

In the find operation, the service requestor retrieves a service description directly or queries the service registry for the type of service required (see “Service Discovery” for more details). The find operation can be involved in two different lifecycle phases for the service requestor: at design time to retrieve the services’ interface description for program development, and at runtime to retrieve the service’s binding and location description for invocation.

12.5.3 BIND

Eventually, a service needs to be invoked. In the bind operation the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact and invoke the service.

12.6 ARTIFACTS OF A WEB SERVICE

Where a Web service is an interface described by a service description and its implementation. A service is a software module deployed on network accessible platforms provided by the service provider. It exists to be invoked by or to interact with a service requestor. It can also function as a requestor, using other Web Services in its implementation.

12.6.1 SERVICE DESCRIPTION

The service description contains the details of the interface and implementation of the service. This includes its data types, operations, binding information and network location. It could also include categorization and other metadata to facilitate discovery and utilization by service requestors. The service description might be published to a service requestor or to a service registry. The Web Services architecture explains how to instantiate the elements and implement the operations in an interoperable manner.

12.6.2 SERVICE DESCRIPTION: FROM XML MESSAGING TO WEB SERVICES

It is through the service description that the service provider communicates all the specifications for invoking the Web service to the service requestor. The service description is key to making the Web Services architecture loosely coupled and reducing the amount of required shared understanding and custom programming and integration between the service provider and the service requestor. For example, neither the requestor nor the provider must be aware of the other's underlying platform, programming language, or distributed object model (if any). The service description combined with the underlying SOAP infrastructure sufficiently encapsulates this detail away from the service requestor's application and the service provider’s Web service.

12.6.3 THE BASIC SERVICE DESCRIPTION

The IBM Web Services architecture uses WSDL for base-level service description. WSDL has been submitted to the W3C for standardization. WSDL is an XML document for describing web services

as a set of endpoints operating on messages containing either document-oriented or procedure-oriented (RPC) messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

Related concrete endpoints are combined into abstract endpoints or services. WSDL is extensible to allow description of endpoints and their messages, regardless of what message formats or network protocols are used to communicate. However, the only currently described bindings are for SOAP 1.1, HTTP POST, and Multipurpose Internet Mail Extensions (MIME).

The use of WSDL in the IBM Web Services architecture conventionally divides the basic service description into two parts: the service interface and the service implementation. This enables each part to be defined separately and independently, and reused by other parts.

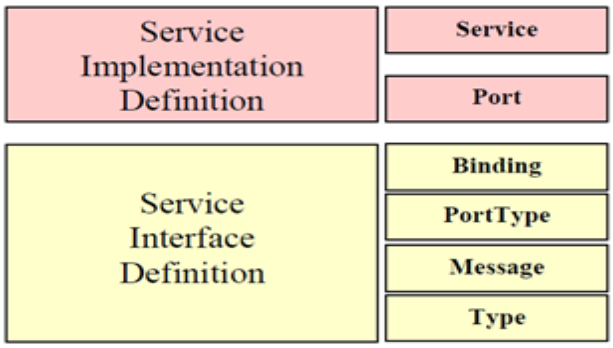


Figure 12.2 : Basic Service Description

A service interface definition is an abstract or reusable service definition that can be instantiated and referenced by multiple service implementation definitions. Think of a service interface definition as an Interface Definition Language (IDL), Java interface or Web service type. This allows common industry-standard service types to be defined and implemented by multiple service implementers. This is analogous to defining an abstract interface in a programming language and having multiple concrete implementations. Service interfaces can be defined by industry standards organizations such as RosettaNet, or HL7 for the health industry.

The service interface contains WSDL elements that comprise the reusable portion of the service description: WSDL:binding, WSDL:portType, WSDL:message and WSDL:type elements as depicted in Figure 5. In the WSDL:portType element, the operations of the Web service are defined. The operations define what XML messages can appear in the input and output data flows. Think of an operation as a method signature in a programming language. The WSDL:message element specifies which XML data types constitute various parts of a message. WSDL:message element is used to define the input and output parameters of an operation. The use of complex data types within the

message is described in the WSDL:types element. The WSDL:binding element describes the protocol, data format, security and other attributes for a particular service interface (WSDL:portType).

The service implementation definition is a WSDL document that describes how a particular service interface is implemented by a given service provider. A Web service is modeled as a WSDL:service element. A service element contains a collection (usually one) of WSDL:port elements. A port associates an endpoint (for example, a network address location or URL) with a WSDL:binding element from a service interface definition.

To illustrate the allocation of responsibility, the Open Applications Group (OAG) might define a service interface definition for the Open Applications Group Integration Specification (OAGIS) purchase-order standard. This service interface definition would define WSDL:type, WSDL:message, WSDL:portType and WSDL:binding.

A service provider can choose to develop a Web service that implements the OAGIS purchase order service interface. The service provider would develop a service implementation definition document that describes the WSDL service, port and address location elements that describe the network address of the provider's Web service and other implementation-specific details. The service interface definition together with the service implementation definition makes up a complete WSDL definition of the service. This pair contains sufficient information to describe to the service requestor how to invoke and interact with the Web service. The service requestor can require other information about the service provider's endpoint.

Check Your Progress

- What is the Basic Service Description?
- Give some idea about artifacts of web service.

12.7 WEB SERVICES DEVELOPMENT LIFE CYCLE

Where does Web Services come in picture? Well, Web services is the mechanism that ASP.NET framework provides to make it easy for us to write code to facilitate connectivity between applications. As ASP.NET developer, If you need an application that will be used by many other applications then you can simply decide to write a web service for it and ASP.NET framework will take care of doing the low level SOAP and XML work for us. The Web Services development lifecycle includes the design, deployment, and runtime requirements for each of the roles: service registry, service provider and service requestor. Each role has

specific requirements for each element of the development lifecycle. The development lifecycle can have four phases. Theses phases are explained below.

12.7.1 BUILD

The build phase of the lifecycle includes development and testing of the Web service implementation, the definition of the service interface description and the definition of the service implementation description. Web service implementations can be provided by creating new Web Services, transforming existing applications into Web Services, and composing new Web Services from other Web Services and applications.

12.7.2 DEPLOY

The deploy phase includes the publication of the service interface and service implementation definition to a service requestor or service registry and deployment of the executables for the Web service into an execution environment (typically, a Web application server).

12.7.3 RUN

During the run phase, the Web service is available for invocation. At this point, the Web service is fully deployed, operational and network-accessible from the service provider. Now the service requestor can perform the find and bind operations.

12.7.4 MANAGE

The manage phase covers ongoing management and administration of the Web service application. Security, availability, performance, quality of service and business processes must all be addressed.

12.8 ARCHITECTURE OVERVIEW

The architecture of the Web Services normally includes six layers. On the other hand, we can examine the IBM Web Services architecture in several layers. First, we will look at a conceptual stack for Web Services and the stack details. Then we will discuss the criteria for choosing the network protocol. We will also review basic XML-based messaging distributed computing. We extend basic XML messaging with service description, which is explained in terms of a service description stack. Following this, we discuss the role of service description in the Web Services architecture, illustrating the range of service publication techniques supporting static and dynamic Web Services applications. Related to service publication, we discuss the role of service discovery. Finally, we describe extensions of the basic Web Services architecture required to make Web Services viable for e-business.

12.8.1 THE WEB SERVICES STACK

To perform the three operations of publish, find and bind in an interoperable manner, there must be a Web Services stack that embraces standards at each level. Figure 12.2 shows a conceptual Web Services stack. The upper layers build upon the capabilities provided by the lower layers. The vertical towers represent requirements that must be addressed at every level of the stack. The text on the left represents standard technologies that apply at that layer of the stack.

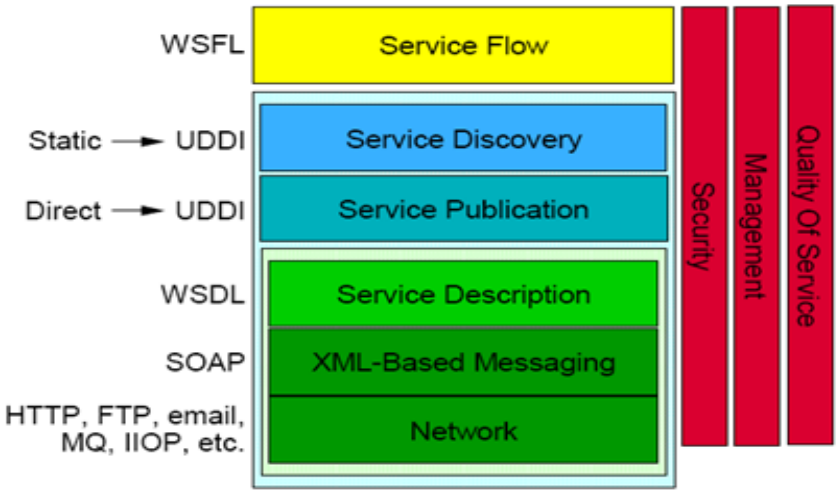


Figure 12.3 : Web Services Conceptual Stack

The foundation of the Web Services stack is the network. Web Services must be network accessible to be invoked by a service requestor. Web Services that are publicly available on the Internet use commonly deployed network protocols. Because of its ubiquity, HTTP is the de facto standard network protocol for Internet-available Web Services. Other Internet protocols can also be supported, including SMTP and FTP. Intranet domains can use reliable messaging and call infrastructures like MQSeries, CORBA, and so on.

The next layer, XML-based messaging, represents the use of XML as the basis for the messaging protocol. SOAP is the chosen XML messaging protocol for many reasons:

- It is the standardized enveloping mechanism for communicating document-centric messages and remote procedure calls using XML.
- It is simple; it is basically an HTTP POST with an XML envelope as payload.
- It is preferred over simple HTTP POST of XML because it defines a standard mechanism to incorporate orthogonal extensions to the

message using SOAP headers and a standard encoding of operation or function.

- SOAP messages support the publishing, finding and binding operations in the Web Services architecture. The section "XML Messaging-Based Distributed Computing" describes this layer in more detail.

The service description layer is actually a stack of description documents. First, WSDL is the de facto standard for XML-based service description. This is the minimum standard service description necessary to support interoperable Web Services. WSDL defines the interface and mechanics of service interaction. Additional description is necessary to specify the business context, qualities of service and service-to-service relationships. The WSDL document can be complemented by other service description documents to describe these higher level aspects of the Web service. For example, business context is described using UDDI data structures in addition to the WSDL document. Service composition and flow are described in a Web Services Flow Language (WSFL) document. The section "Service Description: From XML Messaging to Web Services" describes this layer in more detail.

Because a Web service is defined as being network-accessible via SOAP and represented by a service description, the first three layers of this stack are required to provide or use any Web service. The simplest stack would consist of HTTP for the network layer, the SOAP protocol for the XML messaging layer and WSDL for the service description layer. This is the interoperable base stack that all inter-enterprise, or public, Web Services should support. Web Services, especially intra-enterprise, or private, Web Services, can support other network protocols and distributed computing technologies. Figure 12.3 depicts the interoperable base stack.

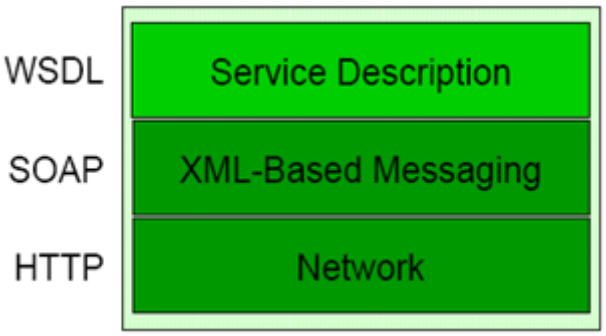


Figure 12.4 : Interoperable Base Web Services Stack

The stack depicted in Figure 12.3 provides for interoperability and enables Web Services to leverage the existing Internet infrastructure. This creates a low cost of entry to a ubiquitous environment. Flexibility is not compromised by the interoperability requirement, because additional

support can be provided for alternative and value-add technologies. For example, SOAP over HTTP must be supported, but SOAP over MQ can be supported as well. While the bottom three layers of the stack identify technologies for compliance and interoperability, the next two layers-service publication and service discovery-can be implemented with a range of solutions.

Any action that makes a WSDL document available to a service requestor, at any stage of the service requestor's lifecycle, qualifies as service publication. The simplest, most static example at this layer is the service provider sending a WSDL document directly to a service requestor. This is called direct publication. E-mail is one vehicle for direct publication. Direct publication is useful for statically bound applications. Alternatively, the service provider can publish the WSDL document describing the service to a host local WSDL registry, private UDDI registry or the UDDI operator node. The variety of service publication mechanisms is discussed in more detail in the section "Service Publication."

Because a Web service cannot be discovered if it has not been published, service discovery depends upon service publication. The variety of discovery mechanisms at this layer parallels the set of publication mechanisms. Any mechanism that allows the service requestor to gain access to the service description and make it available to the application at runtime qualifies as service discovery. The simplest, most static example of discovery is static discovery wherein the service requestor retrieves a WSDL document from a local file. This is usually the WSDL document obtained through a direct publish or the results of a previous find operation. Alternatively, the service can be discovered at design time or runtime using a local WSDL registry, a private UDDI registry or the UDDI operator node. The variety of service discovery mechanisms is discussed in more detail in the section "Service Discovery."

Because a Web service's implementation is a software module, it is natural to produce Web Services by composing Web Services. A composition of Web Services could play one of several roles. Intra-enterprise Web Services might collaborate to present a single Web service interface to the public, or the Web Services from different enterprises might collaborate to perform machine-to-machine, business-to-business transactions. Alternatively, a workflow manager might call each Web service as it participates in a business process. The topmost layer, service flow, describes how service-to-service communications, collaborations, and flows are performed. WSFL is used to describe these interactions. The topic of Web Services flows is covered in its own section "Business Processes, Workflows and Web Services."

For a Web Services application to meet the stringent demands of today's e-businesses, enterprise-class infrastructure must be supplied, including security, management and quality of service. These vertical

towers must be addressed at each layer of the stack. The solutions at each layer can be independent of each other. More of these vertical towers will emerge as the Web Services paradigm is adopted and evolved. We discuss these vertical towers in more detail in the section "Web Services for Real e-business."

The bottom layers of this stack, representing the base Web Services stack, are relatively mature and more standardized than the layers higher in the stack. The maturation and adoption of Web Services will drive the development and standardization of the higher levels of the stack and the vertical towers.

12.8.2 THE NETWORK

At the base of the Web Services stack is the network. This layer can represent any number of network protocols: HTTP, FTP, SMTP, Message Queuing (MQ), Remote Method Invocation (RMI) over Internet Inter ORB Protocol (IIOP), e-mail, and so on. The network protocol used in any given situation depends on application requirements.

For Web Services accessible from the Internet, the network technology choices will favor ubiquitously deployed protocols such as HTTP. For Web Services being provided and consumed within an Intranet, there is the opportunity to agree upon the use of alternative network technologies. The network technology can be chosen based on other requirements, including security, availability, performance and reliability. This allows Web Services to capitalize on existing higher-function networking infrastructures and message-oriented middleware, such as MQSeries. Within an enterprise with multiple types of network infrastructures, HTTP can be used to bridge between them.

One of the benefits of Web Services is that it provides a unified programming model for the development and usage of private Intranet and public Internet services. As a result, the choice of network technology will be transparent to the developer of the service.

12.8.3 XML MESSAGING-BASED DISTRIBUTED COMPUTING

The most fundamental underpinnings of the IBM Web Services architecture is XML messaging. The current industry standard for XML messaging is SOAP. IBM, Microsoft and others submitted SOAP to the W3C as the basis of the XML Protocol Working Group. The XML protocol will replace SOAP as the industry-standard XML messaging protocol. When the W3C has released a draft standard for the XML protocol, the IBM Web Services architecture will migrate from SOAP to the XML protocol.

SOAP is a simple and lightweight XML-based mechanism for exchanging structured data between network applications. SOAP consists

of three parts: an envelope that defines a framework for describing what is in a message, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls (RPCs) and responses. SOAP can be used in combination with or re-enveloped by a variety of network protocols such as HTTP, SMTP, FTP, RMI over IIOP or MQ.

While it is important to understand this foundation, most Web service developers will not have to deal with this infrastructure directly. Most Web Services will use optimized programming language-specific bindings generated from WSDL. This optimization can be especially valuable when a service provider and requestor are both executing in similar environments.

Figure 12.4 shows how XML messaging (that is, SOAP) and network protocols forms the basis of the IBM Web Services architecture.

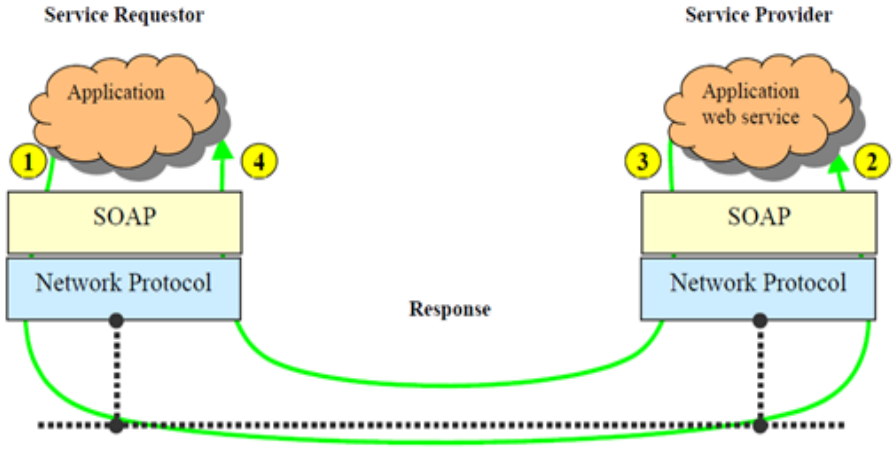


Figure 12.5 : XML Messaging Using SOAP

The basic requirements for a network node to play the role of requestor or provider in XML messaging-based distributed computing are the ability to build, parse a SOAP message, or both, and the ability to communicate over a network (receive, send messages, or both).

Typically, a SOAP server running in a Web application server performs these functions. Alternatively, a programming language-specific runtime library can be used that encapsulates these functions within an API. Application integration with SOAP can be achieved by using four basic steps:

1. In Figure 4, at (1) a service requestor's application creates a SOAP message. This SOAP message is the request that invokes the Web service operation provided by the service provider. The XML document in the body of the message can be a SOAP RPC request or a document-centric message as indicated in the service

description. The service requestor presents this message together with the network address of the service provider to the SOAP infrastructure (for example, a SOAP client runtime). The SOAP client runtime interacts with an underlying network protocol (for example HTTP) to send the SOAP message out over the network.

2. At (2) the network infrastructure delivers the message to the service provider's SOAP runtime (for example a SOAP server). The SOAP server routes the request message to the service provider's Web service. The SOAP runtime is responsible for converting the XML message into programming language-specific objects if required by the application. This conversion is governed by the encoding schemes found within the message.

3. The Web service is responsible for processing the request message and formulating a response. The response is also a SOAP message. At (3) the response SOAP message is presented to the SOAP runtime with the service requestor as the destination. In the case of synchronous request/response over HTTP, the underlying request/response nature of the networking protocol is used to implement the request/response nature of the messaging. The SOAP runtime sends the SOAP message response to the service request or over the network.

4. At (4) the response message is received by the networking infrastructure on the service requestor's node. The message is routed through the SOAP infrastructure; potentially converting the XML message into objects in a target programming language. The response message is then presented to the application.

This example uses the request/response transmission primitive that is quite common in most distributed computing environments. The request/response exchange can be synchronous or asynchronous. Other transmission primitives such as one-way messaging (no response), notification (push-style response), publish/subscribe are possible using SOAP.

12.9 A SIMPLE WEB SERVICES WORKFLOW

Figure 12.6 illustrates a simple workflow involving Web Services. In Figure 6, a buyer service (which might be a simple client) is ordering goods from a seller service. The seller service is a Web service whose interface is defined using WSDL. The buyer service is invoking the order method on the seller service using SOAP and the WSDL definition for the seller service. The buyer service knows what to expect in the SOAP reply message because this is defined in the WSDL definition for the seller service.

Figure 6 shows in and out boxes for each Web service involved in a workflow. In general, the in and out boxes are defined using WSDL, as described previously in this paper. The tool that encapsulates code to

create the buyer service derives the format of the outbox or API for the seller service by analyzing and transforming the WSDL description of the seller service, and also knows what to expect from the seller service by analyzing the WSDL description. The in and out boxes in Figure 6 are symbolic representations of this analysis activity.

Figure 12.6 : Simple workflow

Check Your Progress

- What is the Basic Service Description?
- Define XML Messaging Using SOAP.

12.10 SUMMARY

In this unit you learnt many things like web services, association of web service with ASP.NET, its architecture etc. You have also learnt about web service using the scenario when our applications often require code to determine the number of days, such as how long the customer is associated with us, also to convert from a date of present days into days or years and so on. ASP.NET is a very good and popular environment for the development of web based applications.

Web service is the mechanism that ASP.NET framework provides to make it easy for us to write code to facilitate connectivity between applications. As ASP.NET developer, If you need an application that will be used by many other applications then you can simply decide to write a web service for it and ASP.NET framework will take care of doing the low level SOAP and XML work for us. More precisely, "web service is the communication platform between two different or same platform applications that allows to use their web method." The method in web services always start with [webMethod] attributes, it means that it is a web method that is accessible anywhere, the same as my web application.

Today, most of software engineers and developers are using this model. Efficient and effective Internet applications can be developed within the required deadline along with all the flexibility features. Almost, all organizations offer information on Internet today. Interaction between user and programs is also necessary for all the function in web services.

Basically, a web service is a kind of software which is available over Internet. This uses extensible markup language i.e. XML. XML is used to encode for every kind of communication in web service. As we know that XML is used to exchange the data over the Internet hence, for a web service XML plays a very important role and makes software programs very friendly and useful for the users.

The most fundamental underpinnings of the IBM Web Services architecture is XML messaging. The current industry standard for XML messaging is SOAP. SOAP is a simple and lightweight XML-based mechanism for exchanging structured data between network applications. SOAP consists of three parts: an envelope that defines a framework for describing what is in a message, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls (RPCs) and responses. SOAP can be used in combination with or re-enveloped by a variety of network protocols such as HTTP, SMTP, FTP, RMI over IIOP or MQ.

A Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location.

The seller service is a Web service whose interface is defined using WSDL. The buyer service is invoking the order method on the seller service using SOAP and the WSDL definition for the seller service. The buyer service knows what to expect in the SOAP reply message because this is defined in the WSDL definition for the seller service.

12.11 TERMINAL QUESTIONS

1. What is web service?
2. What does different or same application and platform mean?
3. What do you understand by XML Messaging Using SOAP?
4. Explain web services model.
5. Write a short note on roles in web service architecture.
6. What do you understand by artifacts of web service?
7. Discuss web service development life cycle.
8. Explain the web service architecture.
9. Give the meaning of a simple web service workflow.
10. Write a short note on roles in web service architecture.

UNIT-13 AJAX

Structure

13.0 INTRODUCTION TO AJAX

Ajax, which stands for *Asynchronous JavaScript and XML*, is a set of techniques for creating highly interactive web sites and web applications. The idea is to make what's on the Web appear to be local by giving you a rich user experience, offering you features that usually only appear in desktop applications. The emphasis in Ajax applications is to update the web page, using data fetched from the Internet, without refreshing the web page in the browser. You saw an example of that with Google Suggest, where a drop-down list appears in the browser without a page refresh.

Figure 13.1 : Simple Cookies Details of Ajax

The term “Ajax” was created by Jesse James Garrett, president of Adaptive Path, in a February 18, 2005 article collecting the technologies that already existed, and which make up Ajax, under one umbrella term. That article, “Ajax: A New Approach to Web Applications,” the most important one in the annals of Ajax, appears at www.adaptivepath.com/ideas/essays/archives/000385.php.

Basically AJAX provides client-side scripting that allows communication between the server and the browser without reloading the page. **AJAX** stands for "Asynchronous JavaScript and XML". The word **Asynchronous** means that the user need not wait until the server replies. AJAX allows sending only important information to the server not the entire page. So, only valuable data from the client side is routed to the server side. It makes one’s application interactive and faster. Moreover, AJAX allows sending and receiving data asynchronously without reloading the web page. So it is very fast.

Speaking of change, Ajax is a bit of a change from the earlier types of web pages, be they static HTML or Dynamic HTML/DHTML. The interesting thing is that all types of web pages rely upon essentially the same ingredients: HTML, JavaScript, CSS, and sometimes XML. In this Unit, to consider the only two additional factors that can affect the end result: the browser and the web server.

As, AJAX allows sending only important information to the server not the entire page, so only valuable data from the client side is routed to the server side which makes an application interactive and faster. AJAX is not a programming or scripting language, no new invention and no separate Web service, module or plug-in. But, It is a group of inter-related technologies like javascript, dom, xml, html, css etc. It is an algorithm with 'old' technologies similar to the Dynamic Html or DHTML. It also allows creating server connections in the background while a user is interacting with a web-page at front-end. These connections can be created asynchronously, which means that the user need not wait until the server replies.

13.1 OBJECTIVES

At end of this Unit you would be able to know about the following

- AJAX Concepts
- How to use JavaScript to make AJAX requests
- How AJAX works and how you can use JavaScript to communicate with a web server.
- Implementation of AJAX features
- Advantages of AJAX
- Use of get and post method

- Disadvantages of AJAX
- Main issues of AJAX
- Technologies used in AJAX

13.2 WHY TO USE AJAX

As far as the software side is concerned, you need a browser that can run JavaScript, such as Internet Explorer or Firefox. Ajax revolves around browsers, so you need to have access to an Internet browser. AJAX allows you to send only important information to the server not the entire page. Intuitive and natural user interaction i.e. No clicking required and Mouse movement is a sufficient event trigger. "Partial screen update" replaces the "click, wait, and refresh" user interaction model. Only user interface elements that contain new information are updated asynchronously (no interruption to user operation). The rest of the user interface remains displayed without interruption (no loss of operational context).

So only valuable data from the client side is routed towards the server side. It makes your application interactive and faster. Now, In absence of AJAX, imagine the look at a typical desktop application (Spreadsheet app, etc.). The program responses intuitively and quickly. The program gives to user meaningful feedbacks instantly. A cell in a spreadsheet changes colour when you hover your mouse over it. Icons light up as mouse hovers them. Things happen naturally. No need to click a button or a link to trigger an event.

13.3 WHERE IT IS USED

There are too many web applications running on the web that are using ajax technology like **gmail**, **facebook**,**twitter**, **google map**, **youtube** etc. ASP.NET AJAX server controls mainly provide functionality for having partial page updates, update progress indication, and frequent updates based on a timer. Also, it takes care of generating all the JavaScript that is required to perform these functionalities. So with these controls, the developer doesn't have to write any JavaScript to implement AJAX.

The controls provided by ASP.NET for having AJAX functionality are:

1. ScriptManager
2. ScriptManagerProxy
3. UpdatePanel
4. UpdateProgress
5. Timer

13.2.1 SCRIPTMANAGER

- The ScriptManager control is a non visual component on the page. This control is required on each page that needs to have AJAX features implemented on it. The main functionality of a ScriptManager control is to push Microsoft AJAX framework code to the client side when the page is being rendered. This control can be thought of as the agent which would write the JavaScript required on the client side to facilitate AJAX functionality.
- There should be only one ScriptManager control on the page that needs AJAX functionality. Let us create a webpage and add a ScriptManager control to it:

  <asp:ScriptManagerID="ScriptManager1"runat="server"/>
- 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.

13.2.2 SCRIPTMANAGERPROXY

We have seen that the ScriptManager control is required on the page that needs AJAX functionality. We also know that there should be only one ScriptManager control on the page. Now consider a situation where there is a master page and content page and both need AJAX functionalities. There is one more scenario, let's say we have a UserControl that needs AJAX and it has to be added on a page where AJAX is already implemented. Since there could be only one ScriptManager on the page, adding a ScriptManager control in these scenarios will result in two ScriptManager controls on the page. So to handle such conditions, the ScriptManagerProxy control can be used.

ScriptManagerProxy should be used on content pages that have master pages containing a ScriptManager control. It can also be used inside UserControls when the page containing the UserControl already has the ScriptManager control.

13.2.3 UPDATEPANEL

This is a container control that contains other ASP.NET controls. This control defines a region that is capable of making partial page

updates. We can add various server controls inside an UpdatePanel and this controls inside the UpdatePanel communicates to the server irrespective of the page's postback.

Let us add an UpdatePanel on the page and some server controls inside it. We will try to do some arithmetic operations inside this UpdatePanel and try to get the results without a postback. Once the controls are added, the design view of the page will look like:

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.

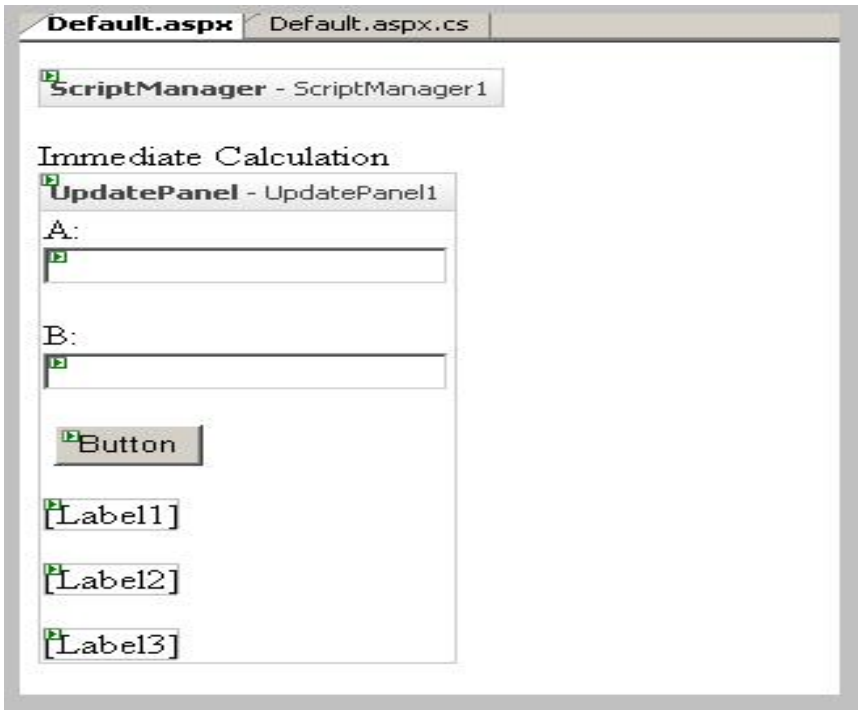


Figure 13.2 : UpdatePanel output

Now let us handle the button click event and perform the arithmetic operations on that:

Example

Hide Copy Code

```
protectedvoid btnCalculate_Click(object sender, EventArgs e)
{
```

```
try
    {
int a = Convert.ToInt32(txtA.Text);
int b = Convert.ToInt32(txtB.Text);
int sum = a + b;
int difference = a - b;
int multiplication = a * b;
 Label1.Text = string.Format("Sum = {0}", sum);
 Label2.Text = string.Format("Difference = {0}", difference);
 Label3.Text= string.Format("Multiplication = {0}", multiplication);
    }
catch (Exception ex)
    {
//pokemon exception handling
    }
}
```

Now since all the controls are inside the UpdatePanel control, clicking the button will not result in a postback but it will asynchronously call the server-side function and give us the results. When we run the page in the browser:

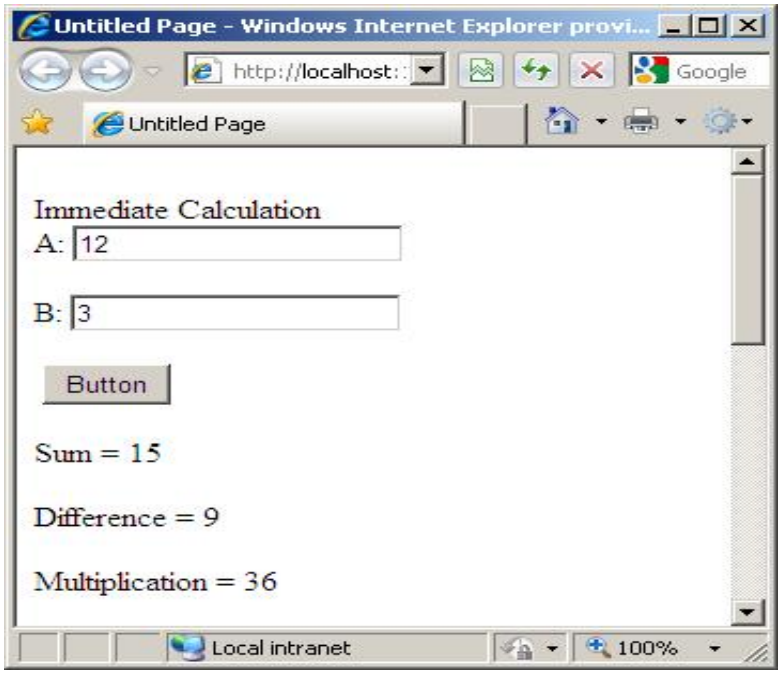


Figure 13.3 : Arithmetic Operations

Notice that clicking on the button does not cause the postback but gives us the result asynchronously. We can control partial page updates using the UpdateMode property of the UpdatePanel and setting Trigger.

Table 13.1 : Properties of the UpdatePanel 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.

Table 13.2: Methods of the UpdatePanel 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 which has been shown in the following 13.3 table.

Table 13.3 : Methods of the UpdatePanel Control

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.

13.2.4 UPDATEPROGRESS

The scenario we just handled gave us the results instantly, but imagine a scenario where the server side processing for the asynchronous event takes some time. If the operation is time consuming then we can provide the user feedback by using the UpdateProgress control inside the

UpdatePanel. Let us have one more UpdatePanel on the page doing the same task, but this time we will make the server side functionality take more time than required (by using sleep). We will add a simple UpdateProgress control to make the user aware of the fact that some processing is being done by the page right now. Let us look at the Design view of this UpdatePanel and UpdateProgress control now.

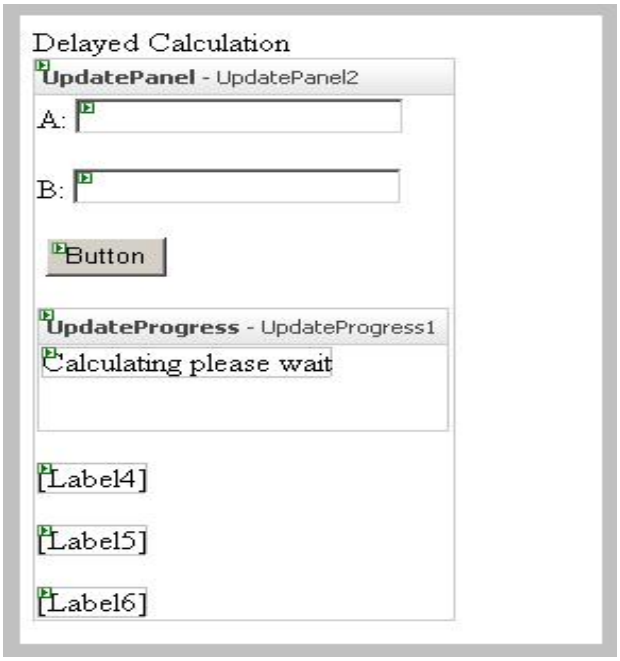


Figure 13.4 : UpdateProgress

Let us handle the server side event for button click but this time let's add a sleep for some time here.

Example

Hide Copy Code

```
protectedvoid btnCalculate2_Click(object sender, EventArgs e)
{
try
    {
//Lets pretend that we are doiing something time consuming
System.Threading.Thread.Sleep(3000);
int a = Convert.ToInt32(txtA2.Text);
int b = Convert.ToInt32(txtB2.Text);
int sum = a + b;
```

```
int difference = a - b;
 int multiplication = a * b;
 Label4.Text = string.Format("Sum = {0}", sum);
 Label5.Text = string.Format("Difference = {0}", difference);
 Label6.Text = string.Format("Multiplication = {0}", multiplication);
    }
 catch (Exception ex)
    {
 //pokemon exception handling
    }
 }
```

Now when we run the page and click the button, the updateProgress message will be shown.

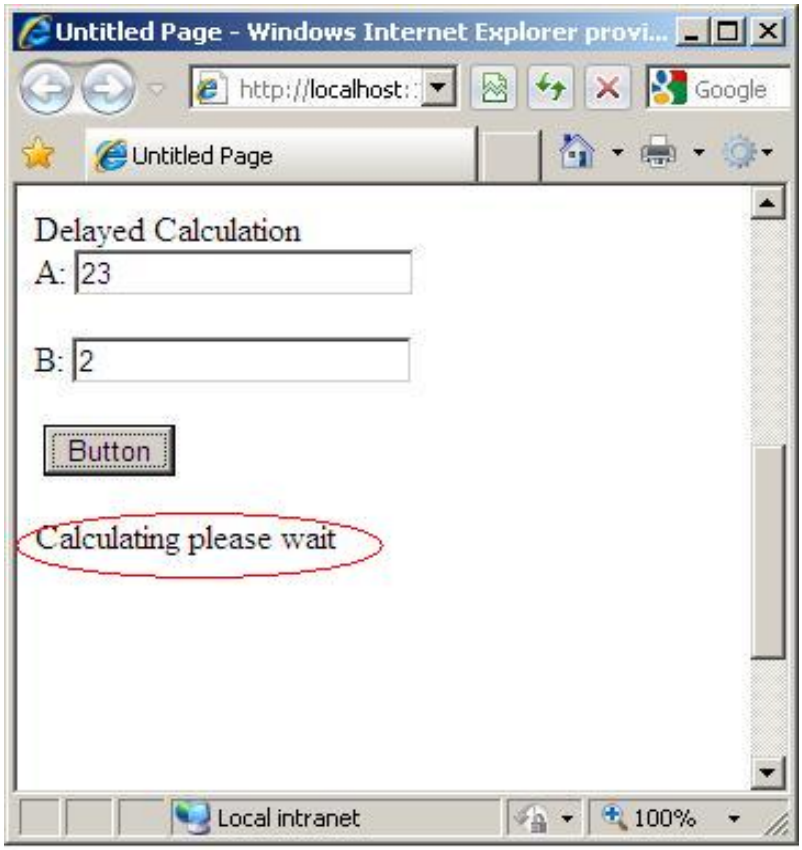


Figure 13.5 : UpdateProgress with sleep

We can also have images and animated GIFs inside the updateProgress control to provide more user friendly feedback.

Table 13.4: Methods of the UpdatePanel Control Properties of the UpdateProgress 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.

Table 13.5 : Methods of the UpdateProgress 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.

13.2.1 TIMER

There might be some scenarios where we want to update a particular portion of the page after some time duration irrespective of user

action. To achieve this, we can use the Timer control. Let us add a timer control to our page and display the server time after every 5 seconds. The design view of the page will look like:

Figure 13.6 : Timer

Let us now handle the timer_tick event. Since the control is inside the UpdatePanel the time will be updated after every 5 seconds without causing any postback. Let us look at the server side code for the timer event and then run the page and see the time changing every 5 seconds.

Example

Hide Copy Code

```
protectedvoid Timer1_Tick(object sender, EventArgs e)
{
   Label8.Text = DateTime.Now.ToString();
}
```

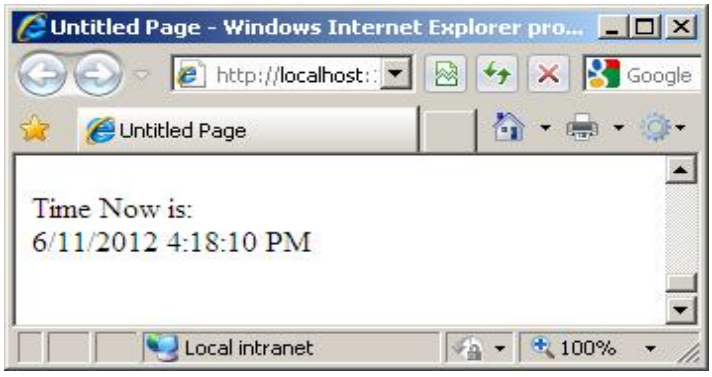


Figure 13.6 : Timer tick event

13.4 TECHNOLOGIES USED IN AJAX

Basically Ajax is not a technology but group of inter-related technologies**.** Following technologies are used in Ajax:

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation.

DOM

It is used for dynamic display and interaction with data.

XML or JSON

For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

ASP or JSP

Server side

XMLHttpRequest

For asynchronous communication between client and server.For more visit next page.

JavaScript

It is used to bring above technologies together. It used for Client-side validation and validate user input in an HTML form before sending the data to a server.

Check Your Progress

- How you can define AJAX and its benefits?
- Write names of technologies used in AJAX.

13.5 BROWSER SUPPORT

Without a web browser, though, web pages are rather useless. The majority of people wandering around the Internet wouldn't fully appreciate them. Yes, there is the indentation, but without a browser, there is no scripting or pictures. A lot can be said about web browsers; after all, they color our web browsing experience nearly as much as the pages we visit. The decision to use a specific web browser probably says a great deal about who each of us is as an individual. All the available browsers cannot support AJAX. Here is a list of major browsers that support AJAX.

- Mozilla Firefox 1.0 and above.
- Netscape version 7.1 and above.
- Apple Safari 1.2 and above.
- Microsoft Internet Explorer 5 and above.
- Konqueror.

- Opera 7.6 and above.

When you write your next application, do consider the browsers that do not support AJAX.

NOTE : When we say that a browser does not support AJAX, it simply means that the browser does not support the creation of Javascript object – XMLHttpRequest object.

13.5.1 WRITING BROWSER SPECIFIC CODE

The simplest way to make your source code compatible with a browser is to use *try...catch*blocks in your JavaScript.

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
function ajaxFunction(){
var ajaxRequest; // The variable that makes Ajax possible!
try
{
// Opera 8.0+, Firefox, Safari
ajaxRequest = new XMLHttpRequest();
}
catch (e){
// Internet Explorer Browsers
try{
ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e) {
try
{
ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
}catch (e){
```

```
// Something went wrong
alert("Your browser broke!");
return false;
}
}
}
}
//-->
</script>
<form name='myForm'>
Name: <input type='text' name='username' /><br />
Time: <input type='text' name='time' />
</form>
</body>
</html>
```

In the above JavaScript code, we try three times to make our XMLHttpRequest object. Our first attempt:

```
ajaxRequest = new XMLHttpRequest();
```

It is for Opera 8.0+, Firefox, and Safari browsers. If it fails, we try two more times to make the correct object for an Internet Explorer browser with:

```
ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
```

If it doesn't work, then we can use a very outdated browser that doesn't support XMLHttpRequest, which also means it doesn't support AJAX. Most likely though, our variable ajaxRequest will now be set to whatever *XMLHttpRequest* standard the browser uses and we can start sending data to the server. The step-wise AJAX workflow is explained in the next chapter.

13.6 AJAX REQUEST

XMLHttpRequest Object, following methods are allow to interact with the server.

Property	Description
open(method, url, boolean)	Specifies the type of method, URL and Boolean(if true than handle asynchronous or false than handle synchronous) • method: type of request, GET or POST • url: the location of the file o the server (with path) • boolean: true (asynchronous) / false (synchronous) • optionally: You wish to login and password may be added to arguments
send("string")	String: use only POST method request.

13.6.1 GET OR POST METHOD

- GET is simpler and faster than POST, so mostly use a GET.
- POST request use when sending a large amount of data to the server, update contain on the server also POST is secure and robust method than GET.
- POST request use to send the data to send to the server.

GET Method

Syntax

```
xmlhttp.open("GET", url,true)// xmlhttp is variable name
xmlhttp.send()
```

Example

```
req.open("GET","ajax_demo.txt",true);// req is variable name
req.send(null);
```

Post Method

Syntax

```
xmlhttp.open("POST", url,true)// xmlhttp is variable name
xmlhttp.send(String)
```

Example

```
req.open("POST","ajax_demo.txt",true);// req is variable name
req.send("hitesh");
```

13.7 AJAX RESPONSE

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

Property	Description
responseText	get the response data as a string.
responseXML	get the response data as XML data.

13.7.1 THE RESPONSETEXT PROPERTY

If the response from the server is not XML, use the responseText property. The responseText property returns the response as a string, and you can use it accordingly:

13.7.2 THE RESPONSEXML PROPERTY

If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property.

Check Your Progress

- Discuss get and post method of AJAX?
- What is AJAX response?

13.8 ADVANTAGE AND DISADVANTAGE OF AJAX

13.8.1 ADVANTAGE OF AJAX

- Independent of server technology.
- Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because Javascript is used.

- Using ajax you can develop faster and more interactive web applications.
- Ajax based application use less server bandwidth, because no need to reload complete page.

13.8.2 DISADVANTAGE OF AJAX

- Possible network latency problems. People should be given feedback about the processing.
- Does not run on all browsers.
- Search Engine like Google can't index Ajax pages.
- Security is less in AJAX application. Anyone can view the source code written for Ajax.
- The back button problem. People think that when they press back button, they will return to the last change they made, but in AJAX this doesn't hold.

13.9 CURRENT ISSUES OF AJAX

AJAX is growing very fast and that is the reason that it contains many issues with it. We hope with the passes of time, they will be resolved and AJAX will become ideal for web applications. We are listing down a few issues that AJAX currently suffers from.

13.9.1 COMPLEXITY IS INCREASED

- Server-side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic.
- Page developers must have JavaScript technology skills.

13.9.2 AJAX-BASED APPLICATIONS CAN BE DIFFICULT TO DEBUG, TEST, AND MAINTAIN

- JavaScript is hard to test - automatic testing is hard.
- Weak modularity in JavaScript.
- Lack of design patterns or best practice guidelines yet.

13.9.3 TOOLKITS/FRAMEWORKS ARE NOT MATURE YET

- Most of them are in beta phase.

13.9.4 NO STANDARDIZATION OF THE XMLHTTPREQUEST YET

- Future version of IE will address this.

13.9.5 NO SUPPORT OF XMLHTTPREQUEST IN OLD BROWSERS

- Iframe will help.

13.9.6 JAVASCRIPT TECHNOLOGY DEPENDENCY AND INCOMPATIBILITY

- Must be enabled for applications to function.
- Still some browser incompatibilities exist.

13.9.7 JAVASCRIPT CODE IS VISIBLE TO A HACKER

- Poorly designed JavaScript code can invite security problems.

Check Your Progress

- Is there any disadvantage of AJAX?
- Discuss current issues in AJAX.

13.10 SUMMARY

What we have tried to do in this unit is to understand AJAX that what is it and how it might be useful for programmers. There are various ways of implementing AJAX in an ASP.NET application. We have only touched the very basics of the ASP.NET AJAX server controls in this article. There are a lot of configuration options and properties associated with each of these server controls which we have not discussed. Understanding these controls gives us one way of having AJAX features in website.

GET is simpler and faster than POST, so mostly use a GET. POST request use when sending a large amount of data to the server, update contain on the server also POST is secure and robust method than GET. POST request use to send the data to send to the server.

Ajax is group of inter-related technologies such as: HTML/XHTML and CSS, DOM, XML or JSON (Javascript Object Notation) is like XML but short and faster than XML, ASP or JSP,

XMLHttpRequest (used for asynchronous communication between client and server), JavaScript.

All the available browsers cannot support AJAX. Here is a list of major browsers that support AJAX: Mozilla Firefox 1.0 and above, Netscape version 7.1 and above, Apple Safari 1.2 and above, Microsoft Internet Explorer 5 and above, Konqueror, Opera 7.6 and above.

The main aadvantages of AJAX: 1) Independent of server technology. 2) Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because JavaScript is used. 3) Using AJAX you can develop faster and more interactive web applications. 4) Ajax based application use less server bandwidth, because no need to reload complete page.

And the disadvantages of AJAX are: 1) Possible network latency problems. People should be given feedback about the processing. 2) Does not run on all browsers. 3) Search Engine like Google can't index Ajax pages. 4) Security is less in AJAX application. Anyone can view the source code written for Ajax. 5) The back button problem. People think that when they press back button, they will return to the last change they made, but in AJAX this doesn't hold.

13.11 TERMINAL QUESTIONS

1. What do you mean by AJAX?
2. Discuss the requirement of AJAX.
3. Explain the current issues of AJAX.
4. Discuss get and post method of AJAX?
5. What is AJAX request and response?
6. Write a program for arithmetic operations.
7. Discuss the advantages and disadvantages of AJAX.
8. Write a small program using AJAX to display the time.
9. Discuss the browser support.
10. Compare AJAX with other technologies.

UNIT-14 DEVELOPING A SMALL APPLICATION USING ASP.NET

Structure

14.0 INTRODUCTION

The project titled "**Bluetooth Based Attandance Management System**" is a small project has been developed using ASP.NET as a front-end and MS Access as a back-end. It is an attendance management system which can be used School, College, University etc. This project aims to train the basic concepts of ASP.NET for the students so that they clould be familiar with the environment of the indusrey. The project has been developed using the traditional waterfall model following allthe required steps of software engineering like SRS, feasibility study, design, coding and implementation, testing, and finally maintenance. It is tryst which couldfulfil the primary knowledge of a project and hence explained from a very basic point of view.

The use of the attendance card is not new rather widely popular for many years. Today, OTR cards or punch cards have been used for clocking in working hours. These are paper cards that are inserted in a machine which will then record the exact time when the employee has arrived. The paper cards have eventually been replaced by sturdier cards that are sized just like the bank card or ID. In fact, some ID cards issued by companies can also be used for time keeping and are inserted into digital time recorders. An issue with the attendance card is that some workers will actually ask co-workers to time in for them. Some have

attempted to remedy this dilemma through the use of signature logs that are attached next to the attendance recorder.

It is also very important and necessary to know that what is a software project or product. A software project is entirely different fromthe other projects i.e. a software product is different from the other products like, car, soap, clothes, shoes etc. The software industry entirely depends on the requirement of clients which changes time-to-time. A client, who asks for something before staring the project, demands different requirenments after some time during the running projects. This creates the chaos for the developers but also shows the skill and expertise of software industry professionals so that they could filfil the needs at any time.

A software development project is a complex undertaking by two or more persons within the boundaries of time, budget, and staff resources that produces new or enhanced computer code that adds significant business value to a new or existing business process. Although this is a restrictive definition, it does define the types of software development projects that are addressed in this book. The criteria for these projects are that they have the potential of adding significant business value and are not trivial undertakings. These development projects will have significant business value, be highly visible, be of moderate to high complexity, and were needed yesterday.

Moreover, project management software is software used for project planning, scheduling, resource allocation and change management. It allows project managers (PMs), stakeholders and users to control costs and manage budgeting, quality management and documentation and also may be used as an administration system. Project management software is also used for collaboration and communication between project stakeholders. A project is a temporary endeavor, having a defined beginning and end (usually constrained by date, but can be by funding or deliverables), undertaken to meet unique goals and objectives, usually to bring about beneficial change or added value.

However this brings me to another problem: using this definition in orthodox way would make many ventures, well, non-projects. Consider Google search as an example - its development isn't constrained by date or funding or specific deliverables, yet my guess is no one would deny it is a project. Also, many R&D projects don't suit the definition well. Same with many startup projects which change rapidly along with their constraints.

14.1 OBJECTIVES

At end of this Unit you would be able to know about the following

- Software engineering
- What is a software project
- Software Requirement Specification

- Feasibility Study
- Coding and Implementation
- DFD and ER Diagram
- Testing
- Maintenance

14.2 WHAT IS SOFTWARE ENGINEERING

When I'm speaking as a project manager, a "project" is a formalized process with a defined goal and an attendant methodology. One can argue about how well-defined the goal should be, how formal the process ought to be, or how rigorous the applied methodology may be, but you can cut through a lot of fog by saying that any project isn't being *actively managed* to complete a finite goal is simply not a project from a PM perspective. For example, ongoing technical support would not generally be considered a discrete project. On the other hand, forming a team tasked with delivering an embiggening feature for your therblig generator by the third quarter of next year fits the PM-oriented definition of a project rather well.

The above scenario indicates us to know about the software engineering. Without understanding the basics of software engineering concepts one is unable to know about is versatile scope and its subdomains. Hence, first we would understand that what software engineering actually has. Let us first understand what software engineering stands for. The term is made of two words, software and engineering.

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product. Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

14.2.1 DEFINITIONS

IEEE defines software engineering as:

(1) The application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in the above statement. Fritz Bauer, a German computer scientist, defines software engineering as: Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

14.2.2 SOFTWARE EVOLUTION

The process of developing a software product using software engineering principles and methods is referred to as **software evolution.** This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

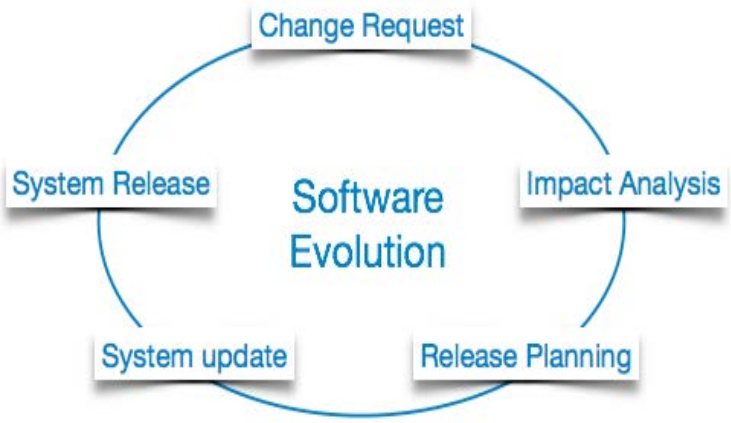


Figure 14.1 : Software Evolution

Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

14.2.3 SOFTWARE EVOLUTION LAWS

Lehman has given laws for software evolution. He divided the software into three different categories:

- **S-type (static-type) -** This is a software, which works strictly according to defined specifications and solutions. The solution and

the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.

- **P-type (practical-type) -** This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly. For example, gaming software.
- **E-type (embedded-type) -** This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, Online trading software.

14.2.4 E-TYPE SOFTWARE EVOLUTION

Lehman has given eight laws for E-Type software evolution -

- **Continuing change -** An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful.
- **Increasing complexity -** As an E-type software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.
- **Conservation of familiarity -** The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system.
- **Continuing growth-** In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
- **Reducing quality -** An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
- **Feedback systems-** The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
- **Self-regulation -** E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
- **Organizational stability -** The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

14.2.5 SOFTWARE PARADIGMS

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:

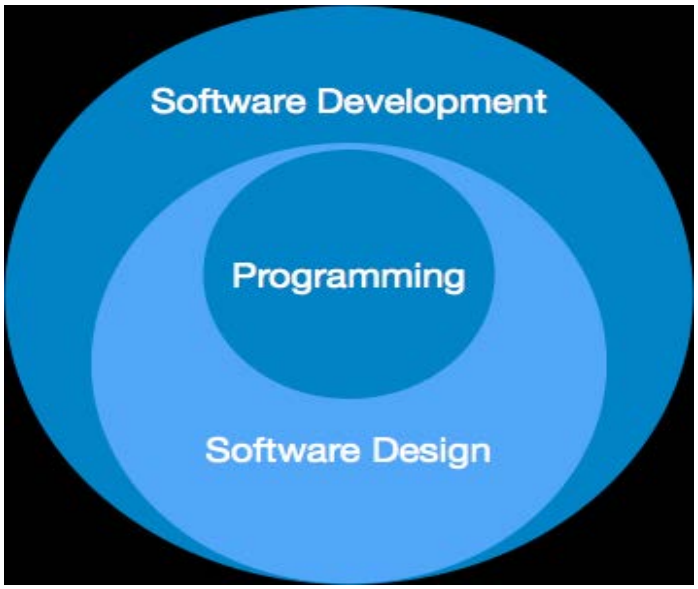


Figure 14.2 : Software Paradigms

Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development Paradigm

This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build. It consists of: *Requirement gathering, Software design, and Programming*.

Software Design Paradigm

This paradigm is a part of Software Development and includes: *Design, Maintenance, and Programming.*

Programming Paradigm

This paradigm is related closely to programming aspect of software development. This includes: Coding, Testing, Integration.

14.2.5 NEED OF SOFTWARE ENGINEERING

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- **Large software -** It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
- **Scalability-** If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
- **Cost-** As hardware industry has shown its skills and huge manufacturing has lower down he price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- **Dynamic Nature-** The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- **Quality Management-** Better process of software development provides better and quality software product.

14.2.5 CHARACTERISTICS OF GOOD SOFTWARE

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds: *Operational, Transitional,* and *Maintenance*. Well-engineered and crafted software is expected to have the following characteristics:

Operational

This tells us how well software works in operations. It can be measured on: *Budget, Usability, Efficiency, Correctness, Functionality, Dependability, Security, Safety.*

Transitional

This aspect is important when the software is moved from one platform to another: *Portability, Interoperability, Reusability, Adaptability.*

Maintenance

This aspect briefs about how well software has the capabilities to maintain itself in the ever-changing environment:

- Modularity
- Maintainability
- Flexibility

- Scalability

14.3 FUNDAMENTAL OF PROJECT & PROBLEM DEFINITION

Nowadays, instructors in universities and colleges take the attendance manually either by calling out individual's name or by passing around an attendance sheet for student's signature to confirm his/her presence. Using these methods is both cumbersome and time-consuming. Therefore a method of taking attendance using instructor's mobile telephone has been presented in this paper which is paperless, quick, and accurate. Application software installed in the instructor's mobile telephone enables it to query students' mobile telephone via Bluetooth connection and, through transfer of students' mobile telephones' Media Access Control (MAC) addresses to the instructor's mobile telephone; and hence the presence of the student can be confirmed. Moreover, detailed record of a student's attendance can also be generated for printing and filing, if needed.

Student Attendance System is based on Bluetooth and RFID reader application.This project has been developed to take learner attendance during class or lab. The RFID reader gets the student’s information through student matrix card. After getting the student information, it will be send to the computer. Later the in-charge whoever may connect with PC using Bluetooth to see his/her presence or absence.

Therefore this system could be used to avoid cheating about their presence or absence. At the same time, this system will also send an e-mail about the attendance details after class dismiss. Bluetooth based new wireless applications can add comfort and security by automation of the tasks which were controlled manually earlier. Advantages of low cost, low power and robustness of Bluetooth have been exploited to propose and execute two new consumer systems in the form of an electronic attendance record system. The reason of the development of biometric system is to take student attendance more efficiently. This method uses the student’s matrix card to track student’s attendance and sent information to the computer and the computer will send data to a mobile phone lecturer. The listing of students will be automatic, quicker and more security intensive than current methods of registration.

This project is derived from a topic suggested by Mr. H.R. Gerber for the development of an automated class attendance recording device. The device must positively identify students and provide reliable class attendance logs for the benefit of students and lecturers. Attendance logs must be stored on a centralized database in order to generate reports and statistics. Therefore, the device must be able to communicate with a central database server. Students should be able to access information; and personalized reports must be generated by the system for effective self-assessment and keeping up-to-date. Higher authorities should also be able

to view attendance information and be able to make changes time-to-time in the system.

14.3.1 AIMS & OBJECTIVES

Attendance Management System is the easiest way to keep track of attendance for community organizations such as school clubs, scouting units, church groups, athletics, or volunteer groups. Attendance Management System covers the requirements of the Personnel Department in terms of Manpower Analysis, day-to-day monitoring of the Attendance, Maintaining Statutory Registers, Monitoring of Leave Records, Calculation of Overtime and transfer of relevant information to the Payroll System.

14.3.2 LARGE-SCALE COMPANIES

If you want something that is more precise and unlikely to be tampered with by the naughty employee, the fingerprint based attendance system is the choice for you. These systems make use of fingerprint readers, or little glass panels attached to the attendance machine.The employees would simply put their fingerprints on the reader which will then scan the print and identify the employee. The fingerprint readers will then automatically login the employee on an electronic database.

There are other variants too to the fingerprint reader, such as the iris scanner. Like the fingerprint, no two people have the same eyes. A scanner will scan the eyes and automatically log the employee in. Remember, however, these high technology systems are much more expensive than the usual card reader. You would also need to create a fingerprint or iris database from all of your employees so that the scanners would be able to make comparisons. The theoretical study and the experiments show that the Iris recognition mechanism is the most accurate and reliable recognition system.

14.3.3 E-COMMERCE

Performance based systems. Finally, there is performance based attendance keeping systems. These are increasingly being utilized to ensure not only employee attendance, but their productivity and efficiency as well. an example are the computer companies and online companies that will log in the employee based on factors such as when the employee logs in to the company web site or computer, and other activities such as mouse clicks, and keyboard taps. This is still in the experimental phase, however, but is widely being studied by many companies. One major limitation, however, is that these attendance keeping systems will not work for companies and business that do not require the emplouees to make high use of the computer.

14.4 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software are refined by establishing a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other information pertinent to requirements.

Requirements collection is crucial to the development of successful information systems. To achieve a high level of IS quality, it is essential that the SRS be developed in a systematic and comprehensive way. If this is done, the system meet the user's needs, and will lead to user satisfaction. If it is not done, the software is likely to not meet the user's requirements, even if the software conforms with the specification and has few defects.

An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated. Methods of defining an SRS are described by the IEEE (Institute of Electrical and Electronics Engineers) specification 830-1998.

14.4.1 SRS FEATURES

SRS should come up with following features:

- User Requirements are expressed in natural language.
- Technical requirements are expressed in structured language, which is used inside the organization.
- Design description should be written in Pseudo code.
- Format of Forms and GUI screen prints.
- Conditional and mathematical notations for DFDs etc.

14.4.2 SOFTWARE REQUIREMENT VALIDATION

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal,

impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions -

- If they can be practically implemented
- If they are valid and as per functionality and domain of software
- If there are any ambiguities
- If they are complete
- If they can be demonstrated

14.4.3 SOFTWARE REQUIREMENT CHARACTERISTICS

Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.

A complete Software Requirement Specifications must be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source

Check Your Progress

- What do you mean by a software project?
- Discuss SRS.
- Give the names of SRS features.

14.5 FEASIBILITY STUDY

Feasibility study is the process of determination of whether or not a project is worth doing. Feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. The contents and recommendations of this feasibility study helped

us as a sound basis for deciding how to precede the project. It helped in taking decisions such as which software to use, hardware combinations, etc.

The following is the process diagram for feasibility analysis. In the diagram, the feasibility analysis starts with the user set of requirements. With this, the existing system is also observed. The next step is to check for the deficiencies in the existing system. By evaluating the above points a fresh idea is conceived to define and quantify the required goals. The user consent is very important for the new plan. Along with, for implementing the new system, the ability of the organization is also checked. Besides that, a set of alternatives and their feasibility is also considered in case of any failure in the proposed system. Thus, feasibility study is an important part in software development.

In the SDLC (Systems Development Life Cycle) of our project we maintained a number of feasibility checkpoints between the two phases of the SDLC. These checkpoints indicate that the management decision to be made after a phase is complete. The feasibility checkpoints in our project were as follows:

- Survey phase checkpoint
- Study phase checkpoint
- Selection phase checkpoint
- Acquisition phase checkpoint
- Design phase checkpoint

We conducted three tests for Project feasibility namely, Technical, Economical, and Operational feasibilities. Which are described below.

14.5.1 TECHNICAL FEASIBILITY

Technical feasibility determines whether the work for the project can be done with the existing equipment, software technology and available personnel. Technical feasibility is concerned with specifying equipment and software that will satisfy the user requirement. This project is feasible on technical remarks also, as the proposed system is more beneficiary in terms of having a sound proof system with new technical components installed on the system. The proposed system can run on any machines supporting Windows and Internet services and works on the best software and hardware that had been used while designing the system so it would be feasible in all technical terms of feasibility. Technical Feasibility addresses three major issues:

- **Is the proposed Technology or Solution Practical?**

  The technologies used are matured enough so that they can be applied to our problems. The practicality of the solution we have developed is proved with the use of the technologies we have

chosen. The technologies such as ASP, IIS, VC# and the compatible H/Ws are so familiar with the today's knowledge based industry that anyone can easily be compatible to the proposed environment.

- **Do we currently posses the necessary technology?**

  We first make sure that whether the required technologies are available to us or not. If they are available then we must ask if we have the capacity. For instance, "Will our current Printer be able to handle the new reports and forms required of a new system?

- **Do we possess the necessary Technical Expertise and is the schedule reasonable?**

  This consideration of technical feasibility is often forgotten during feasibility analysis. We may have the technology, but that doesn't mean we have the skills required to properly apply that technology.

14.5.2 ECONOMICAL FEASIBILITY

Economical feasibility determines whether there are sufficient benefits in sufficient benefits in creating to make the cost acceptable, or is the cost of the system too high. This signifies cost-benefit analysis and savings. On behalf of the cost-benefit analysis, the proposed system is feasible and is economical regarding its pre-assumed cost for making a system. Economical feasibility has great importance as it can outweigh other feasibilities because costs affect organization decisions. The concept of Economic Feasibility deals with the fact that a system can be developed and will be used on installation must be profitable for the organization. The cost to conduct a full system investigation, the cost of hardware and software, the benefits in the form of reduced expenditure are all discussed during the economic feasibility.

During the economical feasibility test we maintained the balance between the Operational and Economical feasibilities, as the two were the conflicting. For example the solution that provides the best operational impact for the end-users may also be the most expensive and, therefore, the least economically feasible. We classified the costs of our Social Networking site according to the phase in which they occur. As we know that the system development costs are usually one-time costs that will not recur after the project has been completed. For calculating the Development costs we evaluated certain cost categories:

- Personal costs
- Computer Usage
- Training
- Supply and equipment costs

- Cost of any computer equipments and software.

In order to test whether the Proposed System is cost-effective or not we evaluated it through three techniques viz.

- Payback analysis
- Return on Investment:
- Net Present value

14.5.3 OPERATIONAL FEASIBILITY

Operation feasibility is a measure of how people feel about the system. Operational Feasibility criteria measure the urgency of the problem or the acceptability of a solution. Operational Feasibility is dependent upon determining human resources for the project. It refers to projecting whether the system will operate and be used once it is installed. If the ultimate users are comfortable with the present system and they see no problem with its continuance, then resistance to its operation will be zero. Behaviorally also the proposed system is feasible. A particular application may be technically and but may fail to produce the forecasted benefits, because the company is not able to get it to work.

For the system, it is not necessary that the user must be a computer expert, but any computer operator given a little bit of knowledge and training can easily operate. Our Project is operationally feasible since there is no need for special training of staff member and whatever little instructing on this system is required can be done so quite easily and quickly as it is essentially This project is being developed keeping in mind the general people who one have very little knowledge of computer operation, but can easily access their required database and other related information. The redundancies can be decreased to a large extent as the system will be fully automated.

14.6 REQUIREMENT SPECIFICATIONS

14.6.1 HARDWARE REQUIREMENT

Minimum hardware at clinet as well as server side configuration required:

1. **RAM:-** 512 MB or More
2. **CPU:-** Pentium 2.0 GHz or higher
3. **Hard disk:- 2** GB or more

14.6.2 SOFTWARE REQUIREMENT

Minimum software at front-end as well as back-end configuration required:

1. **Font End :** - ASP.Net C# (MICROSOFT VISUAL Studio 2010)
2. **Back End :** - SQL SERVER 2005

14.6.3 MEMORY REQUIREMENT

1. 2GB RAM for Server
2. 20 GB External memory storage (Hard Disk Drive) for Server

14.6.4 OPERATIONS REQUIREMENT

1. User friendly interface
2. Required knowledge of all processes involved
3. Easy to navigate
4. Login level for security
5. Functionality control
6. Constraint for proper inputs are applied

14.6.5 SITE ADAPTATION REQUIREMENT

1. Seating area for operator to made with ergonomics in view.
2. A place for computer if desired is required

14.6.6 PRODUCT FUNCTION

Product shall perform following functions: -

User Characteristics

User is basic computer savvy but need to be adapted to computerized environment across the organisation.

Constrains

Time and transition is constraint. At no point of time manual processes can be stopped, which shall challenge updated data to be transited into computer based system.

Assumptions

User shall not do much change in structure of manual records that shall form basis of database structure. Neither user at any level of organisation is reluctant to go for computer based technology.

Dependencies

There is straight Dependency on user providing information about processes and structures and reports.

14.6.7 EXTERNAL INTERFACES

1. User Friendly.
2. Color scheme as per the color scheme of organisation.
3. Menu Driven
4. Authorization level.
5. Data is viewable in User customized form

14.6.8 FUNCTIONS

1. Login Management.
2. Addition of suppliers, printers, customers, employee, catalogue (design) records.

14.6.9 PERFORMANCE

1. Should be robust.
2. Should be recoverable.
3. Fast processing of functions.
4. User flexibility

14.6.10 LOGICAL DATABASE

DFDs have to be drawn to understand logical database and relationship between tables and cascading extents of modification in database.

14.6.11 SOFTWARE SYSTEM ATTRIBUTES

1. Reliability
2. Availability
3. Security
4. Maintainability

CHECK YOUR PROGRESS

- Discuss feasibility.
- What are software system attributes?

14.7 SYSTEM ANALYSIS AND DESIGN

System Analysis by definition is a process of systematic investigation for the purpose of gathering data, interpreting the facts, diagnosing the problem and using this information to either build a completely new system or to recommend the improvements to the existing system. A satisfactory system analysis involves the process of examining a business situation with the intent of improving it through better methods and procedures. In its core sense, the analysis phase defines the requirements of the system and the problems which user is trying to solve irrespective of how the requirements would be accomplished. There are two methods to perform System Requirement Analysis: Structured analysis and object oriented analysis.

14.7.1 STRUCTURED ANALYSIS

Structured Analysis is an analysis method that provides a basis for developing a model of software to be developed. The objective of structured analysis is to identify the customer requirements and establish a basis to create a software model

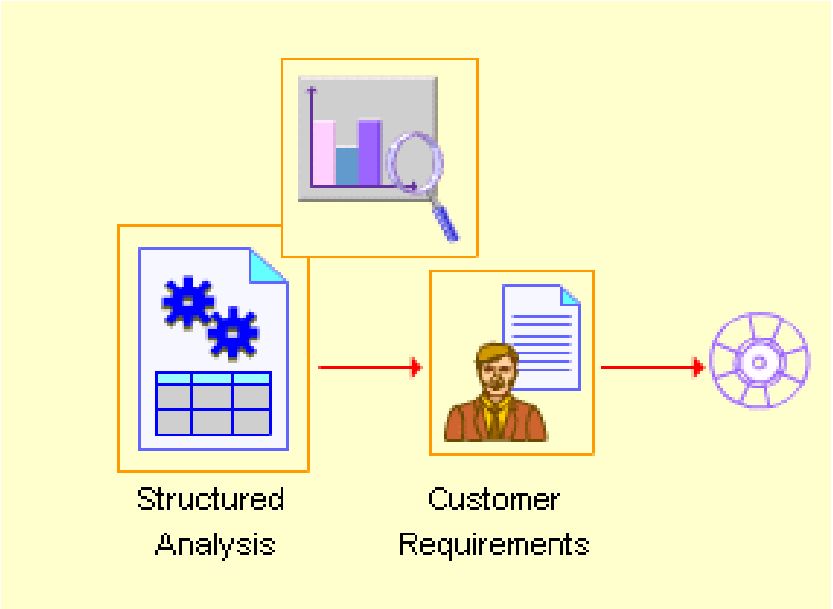


Figure 14.3 : System Analysis

The main components of a Structured Analysis are

- Data Dictionary
- Entity Relationship Diagram
- Data Flow Diagram
- Process Specification
- Control Specification

14.7.2 OBJECT ORIENTED ANALYSIS

It refers to a detailed study of the various objects involved in a system and the relationship of these objects with each other. While performing an object oriented analysis, the focus of the system analyst is on the availability of the objects that are relevant to software development.

14.7.3 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

14.7.4 E-R DIAGRAM

Entity-relationship (E-R) diagram is detailed logical representation of data for an organization. It is data oriented model of a system whereas DFD is a process oriented model. The ER diagram represents data at rest while DFD tracks motions of data. ERD does not provide any information regarding functionality of data. It has three main components – data entities, their relationship and their associated attributes.

Entity: It is most elementary thing of an organization about which data is to be maintained. Every entity has unique identity. It is represented by rectangular box with the name of entity written inside.

Relationship: Entities are connected to each other by relationships. It indicates how two entities are associated. A diamond notation with name of relationship represents as written inside. Entity types that participate in relationship is called degree of relationship. It can be one to one, one to many or many to many.

Attributes: Attribute is a property or characteristic of an entity that is of interest to the organization. It is represented by oval shaped box with name of attribute written inside it.

14.7.5 MODULAR DESIGN CONCEPTS

Functional Independence

The concept of functional independence is a direct outgrowth of modularity and the concepts of abstraction and information hiding. The

principle of information hiding suggests that modules be "characterized by design decisions that (each) hides from all others". In other words modules should be specified and designed so that information (procedure and data) contained within a module is inaccessible to other modules that have no need for such information. Hiding implies that effective modularity can be achieved by defining a set of independent modules that communicate with one another only that information necessary to achieve software function. Abstraction helps to define the procedure entities that make up the software. As data and procedure are hidden from other parts of the software inadvertent errors introduced during modification are less likely to propagate to other locations within the software. Functional independence is achieved by developing modules with "single-minded" function and an "aversion" to excessive interaction with other modules.

Advantages

Independent modules are easier to maintain (and test) because secondary effects caused by design or code modification are limited, error propagation is reduced, and reusable modules are possible. Thus with taking utmost care of this concept we have maintained functional independence in our project at some extent that required interaction among different modules is maintained.

Cohesion

Cohesion of a module represents how tightly bound the internal elements of the module are to one another. Cohesion of a module gives the designer an idea about whether the different elements of a module belong together in the same module.

Coupling

Coupling is a measure of interconnection among modules in a software structure. Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface. In software design, we strive for lowest possible coupling. Simple connectivity among modules results in software that is easier to understand and less prone to a "ripple effect" when errors occur at one location and propagate through a system.

- *Data coupling:*Data coupling means simple argument list (data) is passed and a one to one correspondence exists. A variation of data coupling is found when a portion of a data structure rather than simple arguments is passed via a module interface.
- *Control coupling:* When a "control flag" (a variable that controls decisions in a subordinate or super ordinate module) is passed between modules.
- *External coupling:* It is a relatively high level of coupling occurs when modules are tied to an environment external to software.

- *Common coupling:* When a number of modules reference a global data area. In Job Portal we have maintained the use of global data but restricted ourselves against the common consequences of this coupling.
- *Content coupling:* The highest degree of coupling, content coupling occurs when one module makes use of data or control information maintained within the boundary of another module. Secondarily, content coupling occurs when branches are made into the middle of a module. As this type of coupling makes software complex so in Job Portal we have tried our best to avoid such coupling.

Note: Cohesion and coupling are clearly related. Usually the greater the cohesion of each module in a system, the lower the coupling between modules is. So we have maintained a balance between these two engineering concepts.

14.7.6 DATABASE DESIGN

Admin Table

Fieldname	Datatype	Constraints	Description
Name	Varchar(50)	Not null	Name of Admin
UID	Varchar(100)	Primary Key	Email Id or User ID of Admin
PWD	Varchar(50)	Not null	Password of Admin

Faculty Registration_Table

Field Name	Data type	Constrain	Description
Teacher ID	Varchar(50)	Primary Key	Name of the Faculty
Teacher Name	Varchar(20).	Not null	Name OF The faculty
Address	Varchar(100)	Not Null	Email id of the faculty will also used as primary key to identify the user
Department	Varchar(100)	Not null	Department Of Faculty

Branch	Varchar(10)		Branch Of The Faculty
Qualification	Varchar(25)		Degree Name
Subject	Varchar(20)		Subject To be Teach
Position	Varchar(20)		University Position
Email ID	Varchar(50)		Email Id of The faculty
Password	Varchar(20)		Password in encrypted Form
Experiance	Varchar(50)		Experience Of The faculty
Salary	Varchar(20)		Salary of the Faculty

Student_Login_Table

Fieldname	**Datatype**	**Constraints**	**Description**
BlueTooth ID	Varchar(10)	Primary Key	Bluetooth ID
Student ID	Varchar(10)		Student ID for the Uniqueness
Student Name	Varchar(25)		Name of the student, it can be same for student to student
Father's name	Varchar(25)	Not null	Name of the student father's, it can be same for student to student
Mother's name	Varchar(25)	Not null	Name of the student mother's, it can be same for student to student
City	Varchar(25)		City of the student
Address	Varchar(200)		Address of Employers Company

14.8 CODING AND OUTPUT/INTERFACE

Coding is basically the computer language used to develop apps, websites and software. Without it, we'd have none of the major technology we've come to rely on such as Facebook, our smart phones, the browser we choose to view our favourite blogs or even the blogs themselves. It all runs on code.

To put it very simply, the code is what tells your computer what to do. To go a bit deeper, computers don't understand words. They only understand the concepts of on and off. The capabilities of a computer are guided by on and off switches or transistors. Binary code represents these on and off transistors as the digits 1 and 0. An infinite number of combinations of these codes make your computer work. In order to make binary code manageable, computer programming languages were formed. These languages each serve different purposes, but they all allow programmers to translate important commands into binary code.

The benefits of learning to code are actually quite vast. No longer do we live in a time when only tech professionals are using this useful language. Being able to utilize the commands of code yourself will enable you to have more control of the technology you depend upon. Being able to understand basic code would allow you to make tweaks to the design of your site without having to pay a webmaster to do it for you or to wait for someone from IT to take care of the ticket you submitted ages ago. A knowledge of code can take you even further if you decide to pursue it.

14.8.1 MASTER PAGE.ASPX

<%@MasterLanguage="C#"AutoEventWireup="true"CodeFile="MainMasterPage.master.cs"Inherits="MainMasterPage"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">

<headrunat="server">

<title></title>

<asp:ContentPlaceHolderid="head"runat="server">

</asp:ContentPlaceHolder>

<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>

<metaname="keywords"content="pink shop, store template, ecommerce, online shopping, CSS, HTML"/>

<metaname="description"content="Pink Shop is a free ecommerce template provided by templatemo.com"/>

<linkhref="templatemo_style.css"rel="stylesheet"type="text/css"/>

```
<linkrel="stylesheet"type="text/css"href="stylesheet/styles.css"/>
<scriptlanguage="javascript"type="text/javascript">
function clearText(field) {
if (field.defaultValue == field.value) field.value = '';
elseif (field.value == '') field.value = field.defaultValue;
    }
</script>
<scriptlanguage="javascript"type="text/javascript"src="scripts/mootools-
1.2.1-core.js"></script>
<scriptlanguage="javascript"type="text/javascript"src="scripts/mootools-
1.2-more.js"></script>
<scriptlanguage="javascript"type="text/javascript"src="scripts/slideitmoo
-1.1.js"></script>
<scriptlanguage="javascript"type="text/javascript">
window.addEvents({
'domready': function () {
/* thumbnails example , div containers */
new SlideItMoo({
overallContainer: 'SlideItMoo_outer',
elementScrolled: 'SlideItMoo_inner',
thumbsContainer: 'SlideItMoo_items',
itemsVisible: 5,
elemsSlide: 3,
duration: 200,
itemsSelector: '.SlideItMoo_element',
itemWidth: 140,
showControls: 1
            });
        },
    });
</script>
</head>
```

```
<body>
<formid="form1"runat="server">
<divid="templatemo_wrapper">
        <divid="templatemo_menu">
<%-- <ul>
<li><a href="#" class="current"><span>.01</span>Home</a></li>
<li><a href="http://www.templatemo.com/page/1"
target="_parent"><span>.02</span>Templates</a></li>
<li><a href="http://www.flashmo.com/page/1"
target="_parent"><span>.03</span>Flash</a></li>
<li><a href="http://www.koflash.com"
target="_parent"><span>.04</span>Gallery</a></li>
<li><a href="#"><span>.05</span>Company</a></li>
<li><a href="#"><span>.06</span>Contact</a></li>
</ul>--%>
</div><!-- end of templatemo_menu -->
<divid="templatemo_header_bar">
<divid="header"><divclass="right"></div>
<h1><ahref="http://www.templatemo.com"target="_parent">
<imgsrc="images1/n19.jpg"alt="Gurukula Kangri University"/>
<marquee><span>Gurukula Kangri University.....</span></marquee>
</a></h1>
</div>
<divid="search_box">
<formaction="#"method="get">
<inputtype="text"value="Haridwar....."name="q"size="10"id="searchfiel
d"title="searchfield"onfocus="clearText(this)"onblur="clearText(this)"/>
<inputtype="submit"name="Search"value=""alt="Search"id="searchbutto
n"title="Search"/>
</form>
</div>
</div><!-- end of templatemo_header_bar -->
<divclass="cleaner"></div>
```

```
<divid="sidebar"><divclass="sidebar_top"></div><divclass="sidebar_bot
tom"></div>

<divclass="sidebar_section">

<asp:ContentPlaceHolderID="ContentPlaceHolder1"runat="server"></asp
:ContentPlaceHolder>

                <divclass="cleaner"></div></div>

<divclass="sidebar_section">

        <h2>Courses</h2>

<marqueealign="verticle">

<ulclass="categories_list">

<li><ahref="#">B.Sc</a></li>

<li><ahref="#">B.B.A</a></li>

<li><ahref="#">B Pharma</a></li>

<li><ahref="#">B.P.Ed</a></li>

<li><ahref="#">B.A</a></li>

<li><ahref="#">M.B.A</a></li>

<li><ahref="#">M.Sc</a></li>

<li><ahref="#">M Pharma</a></li>

<li><ahref="#">M.C.A</a></li>

<li><ahref="#">P.Hd</a></li>

</ul>

</marquee>

</div>

<divclass="sidebar_section">

<h2>Gate Of Happiness</h2>

<divclass="image_wrapper"><ahref="http://www.templatemo.com/page/7
"target="_parent"><imgsrc="images1/n12.jpg"alt="product"/></a></div>

<divclass="discount"><span></span> | <ahref="#">Read more</a></div>

</div>

</div><!-- end of sidebar -->

<divid="templatmeo_content">

        <divid="latest_product_gallery">
```

```
<h2>Gallery Of Gurukula Kangri.....</h2>
<divid="SlideItMoo_outer">
<divid="SlideItMoo_inner">
<divid="SlideItMoo_items">
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n15.jpg"alt="gurukula"width="500px"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n17.jpg"alt="gurukula"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n18.jpg"alt="gurukula"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n2.jpg"alt="gurukula"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n25.jpg"alt="gurukula"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n20.jpg"alt="gurukula"/></a>
</div>
<divclass="SlideItMoo_element">
<ahref="#">
<imgsrc="images1/n21"alt="gurukula"/></a>
```

```
</div>

<divclass="SlideItMoo_element">

<ahref="#">

<imgsrc="images1/n24.jpg"alt="gurukula"/></a>

</div>

</div>

</div>

</div>

        </div><!-- end of latest_content_gallery -->

<divclass="content_section">

        <h2>Welcome to Gurukula Kangri University</h2>

<p><ahref="http://www.templatemo.com"target="_parent">Gurukula
Kangri University Is a well stablished Institute </a>that provided a moral
education to our new generation
<ahref="http://www.templatemo.com"target="_parent">for there future
developement</a>and make them enough strong that they can fight in this
cruel world. <ahref="http://validator.w3.org/check?uri=referer">this
institute is established in
1902</a>&amp;amp;<ahref="http://jigsaw.w3.org/css-
validator/check/referer">by Swami Dayanand</a>. to joined the student
from there root <ahref="http://www.photovaco.com"target="_blank">it is
a vast institute situated near the bank of river holy Ganga</a> in this
institute we feel like home and its also provided the facility to the students
that they never think anything else and try to become a good person.</p>

</div>

<divclass="content_section">

        <h2>Our Products</h2>

<divclass="product_box margin_r35">

        <h3>B.Tech Building</h3>

<divclass="image_wrapper"><ahref="http://www.templatemo.com/page/1
"target="_parent"><imgsrc="images1/n25.jpg"alt="gurukula"/></a></div
>

<pclass="price"></p>

<ahref="#"></a> | <ahref="#"></a>

</div>

<%-- <div class="product_box margin_r35">
```

<h3></h3>

<a
href="http://www.templatemo.com/page/2" target="_parent"><img
src="images1/n3.jpg.jpg" alt="gurukula" />

 |

--%>

<divclass="product_box">

<h3>Ved Mandir Of Gurukula</h3>

<divclass="image_wrapper"><ahref="http://www.templatemo.com/page/3
"target="_parent"><imgsrc="images1/n15.jpg"alt="gurukula"/></div
>

<pclass="price">

<ahref="#"> | <ahref="#">

<divclass="cleaner">

<divclass="product_box margin_r35">

<h3>Main Campus Of Gurukula</h3>

<divclass="image_wrapper"><ahref="http://www.templatemo.com/page/4
"target="_parent"><imgsrc="images1/n2.jpg"alt="gurukula"/>

<pclass="price">

<ahref="#"> | <ahref="#">

<divclass="product_box margin_r35">

<h3>Gurukula Library</h3>

<divclass="image_wrapper"><ahref="http://www.templatemo.com/page/5
"target="_parent"><imgsrc="images1/n21.jpg"alt="gurukula"/></div
>

<pclass="price">

<ahref="#"> | <ahref="#">

<%--

<h3> Vivamus at justo</h3>

```
<div class="image_wrapper"><a
href="http://www.templatemo.com/page/6" target="_parent"><img
src="images1/24.jpg" alt="gurukula" /></a></div>

<p class="price"></p>

<a href="#">Detail</a> | <a href="#">Buy Now</a>

</div>--%>

<divclass="cleaner"></div>

<divclass="button_01"><ahref="#">View All</a></div>

</div>

</div><!-- end of templatmeo_content -->

</div><!-- end of templatemo_wrapper -->

<divid="templatemo_footer_wrapper">

        <divid="templatemo_footer">

        <ulclass="footer_menu">

<li><ahref="#">Home</a></li>

<li><ahref="http://www.gkv.in.com/page/1"target="_parent">CSS
Templates</a></li>

<li><ahref="http://www.gurukulakangri.com/page/1"target="_parent">Fl
ash Resources</a></li>

<li><ahref="#">Gallery</a></li>

<li><ahref="#">Company</a></li>

<liclass="last_menu"><ahref="#">Contact</a></li>

</ul>

    Copyright © 2048 <ahref="#">Your Company Name</a> |

<ahref="http://www.iwebsitetemplate.com"target="_parent">Website
Templates</a> by
<ahref="http://www.templatemo.com"target="_parent">Free CSS
Template</a></div>

        <!-- end of footer -->

</div><!-- end of footer_wrapper -->

<divalign=center>This Is Gurukula kangri University <ahref='http://all-
free-download.com/free-website-templates/'>Haridwar.....</a></div>

</form>

</body>

</html>
```

14.8.2 MASTER PAGE2.ASPX

```
<%@MasterLanguage="C#"AutoEventWireup="true"CodeFile="PrincipalMasterPage.master.cs"Inherits="Principal_PrincipalMasterPage"%>


<!DOCTYPEhtml>
<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
<asp:ContentPlaceHolderid="head"runat="server">
</asp:ContentPlaceHolder>
<styletype="text/css">
.auto-style1 {
width: 100%;
      }
.auto-style2 {
width: 258px;
      }
.auto-style3 {
width: 258px;
height: 29px;
      }
.auto-style5 {
height: 29px;
      }
.auto-style6 {
width: 258px;
         }
.auto-style7 {
      }
</style>
</head>
```

```
<body>
<formid="form1"runat="server">
<tableclass="auto-style1">
<tr>
<tdclass="auto-style2">
<asp:ImageID="Image1"runat="server"Height="150px"ImageUrl="~/ima
ges/n1.jpg"Width="258px"/>
</td>
<tdcolspan="10">
<asp:AdRotatorID="AdRotator1"runat="server"AdvertisementFile="~/Ad
vertisements.xml"Height="150px"Width="950px"/>
</td>
</tr>
<tr>
<tdclass="auto-style3"style="font-family: Rockwell; font-size: x-large;
color: #000000">
<asp:MenuID="Menu1"runat="server">
<Items>
<asp:MenuItemText="Home"Value="Home"></asp:MenuItem>
</Items>
</asp:Menu>
</td>
<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">
</td>
<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">
 </td>
<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large;">
</td>
<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">
</td>
<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">
</td>
```

<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">

</td>

<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">

 </td>

<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">

 </td>

<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">

 </td>

<tdclass="auto-style5"style="font-family: Rockwell; font-size: x-large">

<asp:MenuID="Menu7"runat="server"OnMenuItemClick="Menu7_Menu ItemClick">

<Items>

<asp:MenuItemText="Logout"Value="Logout"></asp:MenuItem>

</Items>

</asp:Menu>

</td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large">

<asp:MenuID="Menu2"runat="server">

<Items>

<asp:MenuItemText="Account"Value="Account">

<asp:MenuItemText="Change Password"Value="Change Password"></asp:MenuItem>

<asp:MenuItemText="Update Profile"Value="Update Profile"></asp:MenuItem>

<asp:MenuItemText="View Profile"Value="View Profile"></asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

<asp:MenuID="Menu3"runat="server">

<Items>

<asp:MenuItemText="Student Information"Value="Student Information">

<asp:MenuItemText="Add Student"Value="Add Student"NavigateUrl="CreateStudent.aspx"></asp:MenuItem>

<asp:MenuItemText="Delete Student"Value="Delete Student"></asp:MenuItem>

<asp:MenuItemText="Update Student"Value="Update Student"></asp:MenuItem>

<asp:MenuItemText="View All Student"Value="View All Student"></asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

<asp:MenuID="Menu4"runat="server">

<Items>

<asp:MenuItemText="Faculty Information"Value="Faculty Information">

<asp:MenuItemText="Add Faculty"Value="Add Faculty">

</asp:MenuItem>

<asp:MenuItemText="Delete Faculty"Value="Delete Faculty"></asp:MenuItem>

<asp:MenuItemText="Update Faculty"Value="Update Faculty"></asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

<asp:MenuID="Menu5"runat="server">

<Items>

<asp:MenuItemText="Search"Value="Search">

<asp:MenuItemText="Faculty"Value="Faculty"></asp:MenuItem>

<asp:MenuItemText="student"Value="student"></asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

<asp:MenuID="Menu6"runat="server">

<Items>

<asp:MenuItemText="Check Attandance"Value="Check Attandance">

<asp:MenuItemText="Student"Value="Student">

<asp:MenuItemText="All Students"Value="All Students"NavigateUrl="AllStudentAtten.aspx"></asp:MenuItem>

<asp:MenuItemText="Particular Student"Value="Particular Student"NavigateUrl="ParticularStudentAttendance.aspx"></asp:MenuItem>

</asp:MenuItem>

<asp:MenuItemText="Faculty"Value="Faculty">

<asp:MenuItemText="All"Value="All"></asp:MenuItem>

<asp:MenuItemText="Particular"Value="Particular"></asp:MenuItem>

</asp:MenuItem>

</asp:MenuItem>

</Items>

</asp:Menu>

</td>

<tdcolspan="6"class="auto-style7">

</td>

<tdcolspan="4"rowspan="9"><asp:ContentPlaceHolderID="ContentPlaceHolder2"runat="server">

</asp:ContentPlaceHolder></td>

<td></td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

<tr>

<tdclass="auto-style6"style="font-size: large"> </td>

<tdcolspan="6"class="auto-style7">

 </td>

<td> </td>

</tr>

</table>

</form>

</body>

</html>

14.8.3 MASTER PAGE3.ASPX

<%@MasterLanguage="C#"AutoEventWireup="true"CodeFile="Faculty MasterPage.master.cs"Inherits="FacultyMember_FacultyMasterPage"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">

<headrunat="server">

```
<title></title>
<asp:ContentPlaceHolderid="head"runat="server">
</asp:ContentPlaceHolder>
<styletype="text/css">
.auto-style1 {
width: 100%;
	}
.auto-style2 {
width: 14px;
	}
.auto-style4 {
width: 100px;
	}
</style>
</head>
<bodystyle="background-image: url('/images/n9.jpg')">
<formid="form1"runat="server">
<div>
</div>
<tableclass="auto-style1">
<tr>
<tdclass="auto-style2">
<asp:ImageID="Image1"runat="server"Height="150px"ImageUrl="~/ima
ges/n2.jpg"Width="250px"/>
</td>
<td>
<asp:AdRotatorID="AdRotator1"runat="server"AdvertisementFile="~/Ad
vertisements.xml"Height="150px"Width="950px"/>
</td>
</tr>
</table>
<tableclass="auto-style1">
```

<tr>

<td>

<asp:ButtonID="Button1"runat="server"Text="Home"OnClick="Button1
_Click"/>

</td>

<td>

<asp:ButtonID="Button2"runat="server"OnClick="Button2_Click"Text="
Save Attandance"/>

</td>

<td>

<asp:ButtonID="Button3"runat="server"Text="Save Lab Attandance"/>

</td>

<td>

<asp:ButtonID="Button4"runat="server"Text="Search
Student"OnClick="Button4_Click"/>

</td>

<td>

<asp:ButtonID="Button5"runat="server"Text="Check
Attandance"OnClick="Button5_Click"/>

</td>

</tr>

</table>

<tableclass="auto-style1">

<tr>

<tdclass="auto-style4">

<asp:ButtonID="Button6"runat="server"Text="View
profile"OnClick="Button6_Click"/>

</td>

<tdrowspan="3">

<asp:ContentPlaceHolderid="ContentPlaceHolder1"runat="server">

</asp:ContentPlaceHolder>

</td>

</tr>

<tr>

<tdclass="auto-style4"> </td>

</tr>

<tr>

<tdclass="auto-style4">

<asp:ButtonID="Button7"runat="server"Text="Update
Profile"Width="99px"/>

</td>

</tr>

<tr>

<tdclass="auto-style4"> </td>

<tdrowspan="3"> </td>

</tr>

<tr>

<tdclass="auto-style4">

<asp:ButtonID="Button8"runat="server"Text="Change
Password"Width="91px"OnClick="Button8_Click"/>

</td>

</tr>

<tr>

<tdclass="auto-style4"> </td>

</tr>

</table>

</form>

</body>

</html>

14.8.4 HOME.ASPX

<%@PageTitle=""Language="C#"MasterPageFile="~/MainMasterPage.m
aster"AutoEventWireup="true"CodeFile="Home.aspx.cs"Inherits="Home
"%>

<asp:ContentID="Content1"ContentPlaceHolderID="head"runat="Server"
>

```
</asp:Content>
<asp:ContentID="Content2"ContentPlaceHolderID="ContentPlaceHolder
1"runat="Server">
<table>
<tr>
<tdcolspan="2">
<h2>Members</h2>
</td>
</tr>
<tr>
<td>
<label>Username </label>
</td>
<td>
<asp:TextBoxID="TextBox1"runat="server"></asp:TextBox></td>
</tr>
<tr>
<td>
<label>
                Password</label></td>
<td>
<asp:TextBoxID="TextBox2"runat="server"></asp:TextBox></td>
</tr>
<tr>
<td>
<label>User type </label>
</td>
<td>
<asp:RadioButtonID="rd1"runat="server"Text="Student"GroupName="s"
/>
<br/>
```

<asp:RadioButtonID="rd2"runat="server"Text="Faculty
Member"GroupName="s"/></td>

</tr>

<tr>

<td></td>

<td>

<asp:ButtonID="Button1"runat="server"Text="Login"OnClientClick="co
nfirm('Are u sure u want to login yes then click ok else
cancel!')"OnClick="Button1_Click"/></td>

</tr>

</table>

</asp:Content>

14.8.5 HOME.ASPX.CS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

publicpartialclassHome : System.Web.UI.Page

{

protectedvoid Page_Load(object sender, EventArgs e)

{

}

protectedvoid Button1_Click(object sender, EventArgs e)

{

SqlConnection con =
newSqlConnection(ConfigurationManager.ConnectionStrings[1].Connecti
onString);

```
if (rd2.Checked)
      {
SqlDataAdapter adp = newSqlDataAdapter("select position from
FacultyInfo where tid='" + TextBox1.Text + "' and pass='" +
TextBox2.Text + "'", con);
DataSet ds = newDataSet();
adp.Fill(ds);
if (ds.Tables[0].Rows.Count > 0)
          {
if (ds.Tables[0].Rows[0][0].ToString() =="hod")
              {
Response.Write("<script>alert('ok')</script>");
Server.Transfer("~/Principal/Home.aspx");
              }
else
              {
Response.Write("<script>alert('.....WELCOME.....')</script>");
Session["ltid"] = TextBox1.Text;
//  Server.Transfer("~/FacultyMember/Home.aspx");
Response.Redirect("~/FacultyMember/Home.aspx");
              }
          }
else
          {
Response.Write("<script>alert('Sorry ! Invalid User')</script>");
          }
      }
  }
protectedvoid TextBox2_TextChanged(object sender, EventArgs e)
  {
  }
```

14.8.6 ATTANDANCE.ASPX

<%@PageTitle=""Language="C#"
MasterPageFile="~/FacultyMember/FacultyMasterPage.master"AutoEvent
tWireup="true"CodeFile="Attendence.aspx.cs"Inherits="FacultyMember_
Attendence"%>

<asp:ContentID="Content1"ContentPlaceHolderID="head"Runat="Server
">

</asp:Content>

<asp:ContentID="Content2"ContentPlaceHolderID="ContentPlaceHolder
1"Runat="Server">

<asp:GridViewID="GridView1"runat="server"AutoGenerateColumns="fa
lse"BackColor="White"BorderColor="#336666"BorderStyle="Double"Bo
rderWidth="3px"CellPadding="4"GridLines="Horizontal">

<FooterStyleBackColor="White"ForeColor="#333333"/>

<HeaderStyleBackColor="#336666"Font-
Bold="True"ForeColor="White"/>

<PagerStyleBackColor="#336666"ForeColor="White"HorizontalAlign="
Center"/>

<RowStyleBackColor="White"ForeColor="#333333"/>

<SelectedRowStyleBackColor="#339966"Font-
Bold="True"ForeColor="White"/>

<SortedAscendingCellStyleBackColor="#F7F7F7"/>

<SortedAscendingHeaderStyleBackColor="#487575"/>

<SortedDescendingCellStyleBackColor="#E5E5E5"/>

<SortedDescendingHeaderStyleBackColor="#275353"/>

<Columns>

<asp:BoundFieldDataField="student_id"HeaderText="Student ID"/>

<asp:BoundFieldDataField="student_name"HeaderText="Student
Name"/>

<asp:BoundFieldDataField="father_name"HeaderText="Father Name"/>

<asp:BoundFieldDataField="Course"HeaderText="Class"/>

<asp:BoundFieldDataField="sem"HeaderText="Semester"/>

<asp:TemplateFieldHeaderText="Attendence">

<ItemTemplate>

<asp:RadioButtonID="rd1"runat="server"Text="Present"GroupName="a"
></asp:RadioButton>

<asp:RadioButtonID="rd2"runat="server"Text="Absent"GroupName="a"
></asp:RadioButton>

<asp:RadioButtonID="rd3"runat="server"Text="Leave"GroupName="a">
</asp:RadioButton>

</ItemTemplate>

</asp:TemplateField>

</Columns>

</asp:GridView>

<asp:ButtonID="Button1"runat="server"Text="Save"OnClick="Save"></
asp:Button>

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

</asp:Content>

14.8.7 ATTANDANCE.ASPX.CS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

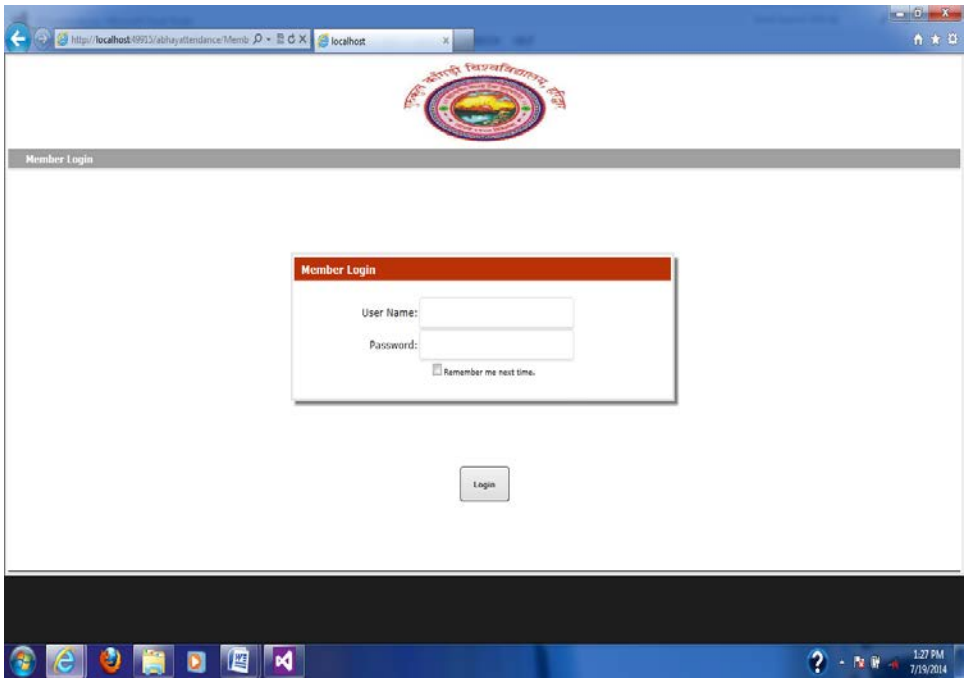
using System.Data.SqlClient;

```
using System.Configuration;
publicpartialclassFacultyMember_Attendence : System.Web.UI.Page
{
SqlConnection con;
SqlDataAdapteradp;
DataSet ds;
SqlCommand cmd;
protectedvoid Page_Load(object sender, EventArgs e)
   {
con =
newSqlConnection(ConfigurationManager.ConnectionStrings[1].Connecti
onString);
if (!IsPostBack)
      {
FillGrid();
      }
   }
publicvoid FillGrid()
   {
int sn1 = Convert.ToInt32(Request.QueryString["sn"].ToString());
string sn = sn1.ToString();
adp = newSqlDataAdapter("select
student_id,student_name,father_name,course ,sem from StudentInfo
where Course='M.C.A' and sem='" + sn + "'", con);
ds = newDataSet();
adp.Fill(ds);
      GridView1.DataSource = ds;
GridView1.DataBind();
   }
publicvoid Save(object sender, EventArgs e)
   {
string str=null;
GridViewRow gr;
```

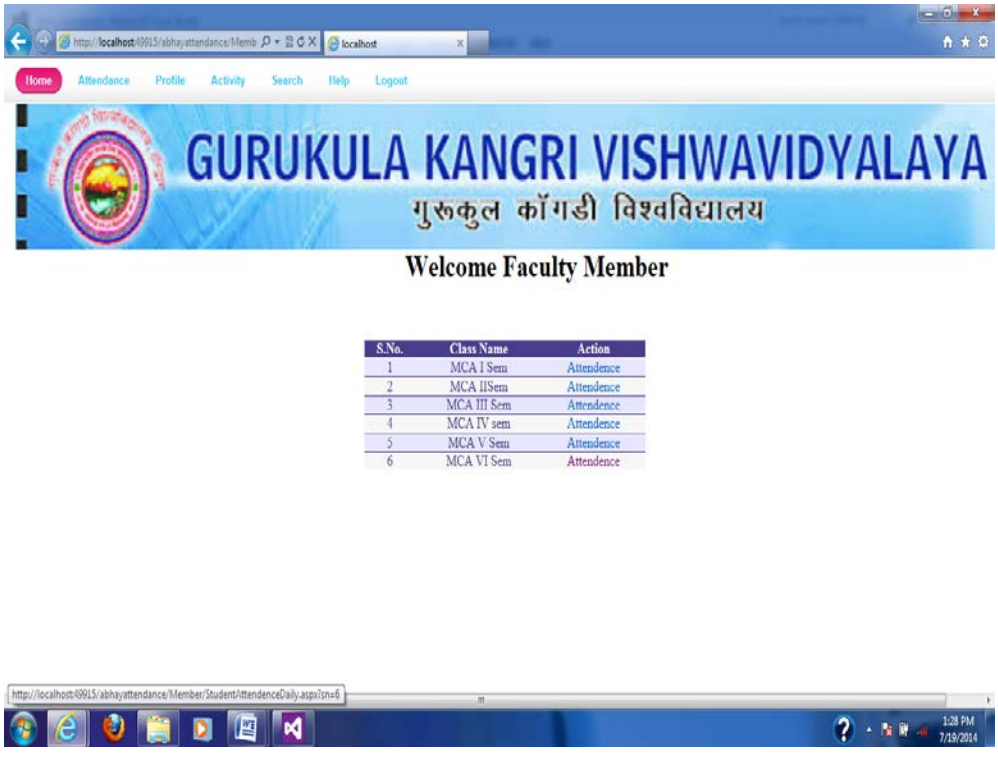
```
int day = DateTime.Now.Day;
int month = DateTime.Now.Month;
int year = DateTime.Now.Year;
string tid = "1";
string stime = "12:40";
adp = newSqlDataAdapter("select branch,subject from FacultyInfo where
tid='" + tid + "'", con);
ds = newDataSet();
adp.Fill(ds);
string br = ds.Tables[0].Rows[0][0].ToString();
string sub = ds.Tables[0].Rows[0][1].ToString();
string etime = DateTime.Now.ToShortTimeString();
for (int i = 0; i < GridView1.Rows.Count; i++)
      {
gr=GridView1.Rows[i];
RadioButton rb1 = (RadioButton)gr.FindControl("rd1");
RadioButton rb2 = (RadioButton)gr.FindControl("rd2");
RadioButton rb3 = (RadioButton)gr.FindControl("rd3");
if (rb1.Checked==true)
          {
str = "P";
             Label1.Text = rb1.Checked.ToString();
          }
elseif (rb2.Checked==true)
          {
str = "A";
          }
else
          {
str = "L";
          }
```

```
string sid = gr.Cells[0].Text;
string sname = gr.Cells[1].Text;
string course = gr.Cells[3].Text;
string sem = gr.Cells[4].Text;
        cmd = newSqlCommand("insert into Attandance
values(@a,@b,@c,@d,@e,@f,@g,@h,@i,@j,@k,@l,@m)", con);
cmd.Parameters.AddWithValue("@a", sid);
cmd.Parameters.AddWithValue("@b", sname);
cmd.Parameters.AddWithValue("@c", course);
cmd.Parameters.AddWithValue("@d", sem);
cmd.Parameters.AddWithValue("@e", br);
cmd.Parameters.AddWithValue("@f", day);
cmd.Parameters.AddWithValue("@g", month);
cmd.Parameters.AddWithValue("@h", year);
cmd.Parameters.AddWithValue("@i", tid);
cmd.Parameters.AddWithValue("@j", stime);
cmd.Parameters.AddWithValue("@k", etime);
cmd.Parameters.AddWithValue("@l", sub);
cmd.Parameters.AddWithValue("@m", str);
con.Open();
cmd.ExecuteNonQuery();
con.Close();
      }
Response.Write("<script>alert('Attendence Saved !')</script>");
   }
}
```

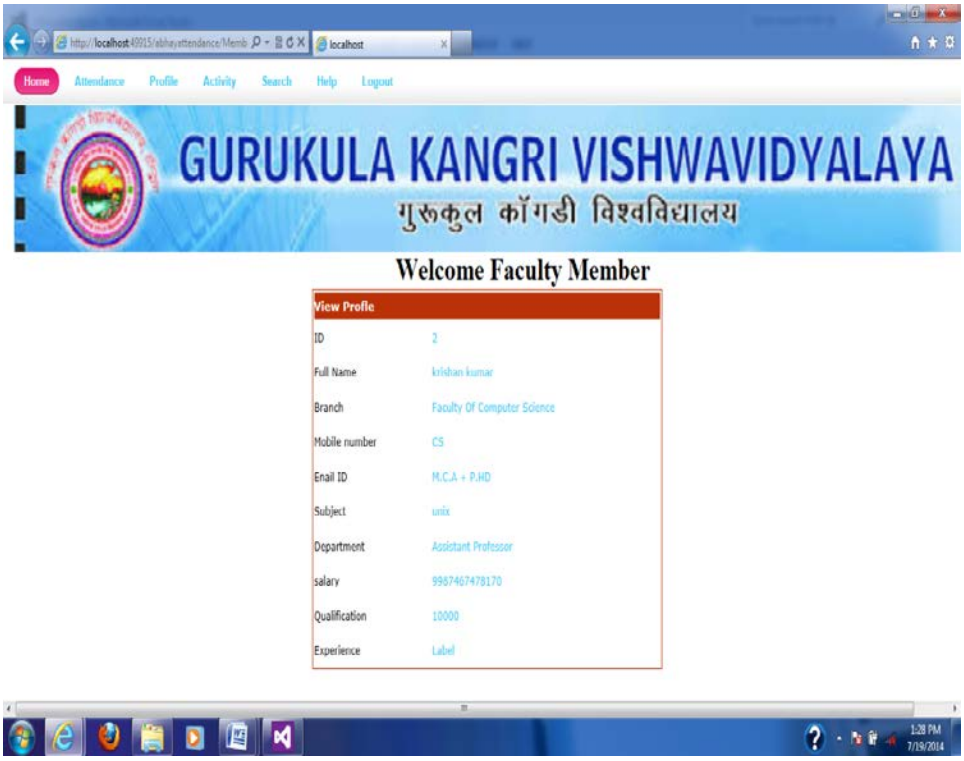
Output-1: Sample Layout



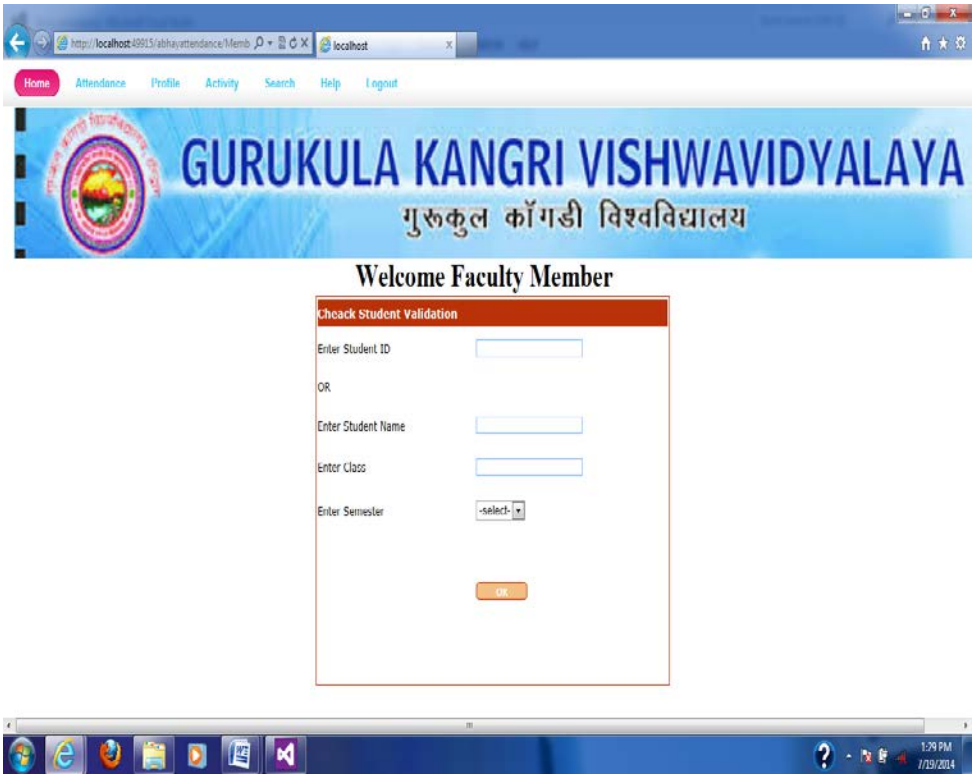
Output-2: Login Page



Output-3 : Attendance Page



Output-4: View Profile Page



14.9 SUMMARY

The last unit i.e. fourteen is basically intended with the hand-on-practice based on ASP.NET environment. We must also be familiar with the basic concepts of the software development. To develop a software project whether small, medium or big software professionals must have basic idea of different phases of software development life cycle.

Softwareis more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. The computer software serves as the basic for modern scientific investigations and engineering problem solving. The computer software is in present time is used in all fields. As we move into 21st century, it will become the driver for new advances in every from elementary education to other latest trends.

Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

The process of developing a software product using software engineering principles and methods is referred to as *software evolution***.** This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

Waterfall model is a traditional popular software development model and other models are more or less similar to the waterfall model. Every software project contains at least requirement analysis, design, coding, testing, and maintenance.

Cohesion of a module represents how tightly bound the internal elements of the module are to one another. Cohesion of a module gives the designer an idea about whether the different elements of a module belong together in the same module.

Coupling is a measure of interconnection among modules in a software structure. Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface. In software design, we strive for lowest possible coupling.

System Analysis by definition is a process of systematic investigation for the purpose of gathering data, interpreting the facts, diagnosing the problem and using this information to either build a completely new system or to recommend the improvements to the existing system. A satisfactory system analysis involves the process of examining a business situation with the intent of improving it through better methods and procedures. In its core sense, the analysis phase defines the requirements of the system and the problems which user is trying to solve

irrespective of how the requirements would be accomplished. There are two methods to perform System Requirement Analysis: Structured analysis and object oriented analysis.

Check Your Progress

- What is entity and relationship in ER diagram?
- What is system analysis?

14.10 TERMINAL QUESTIONS

1. What do you understand by a project?
2. Explain the meaning of software.
3. Give the meaning of software engineering.
4. Explain the goals of software engineering.
5. Write short note on feasibility study.
6. Discuss software requirement specification (SRS).
7. Compare ER diagram and Data flow diagram.
8. What is the difference between cohesion and coupling?

ROUGH WORK