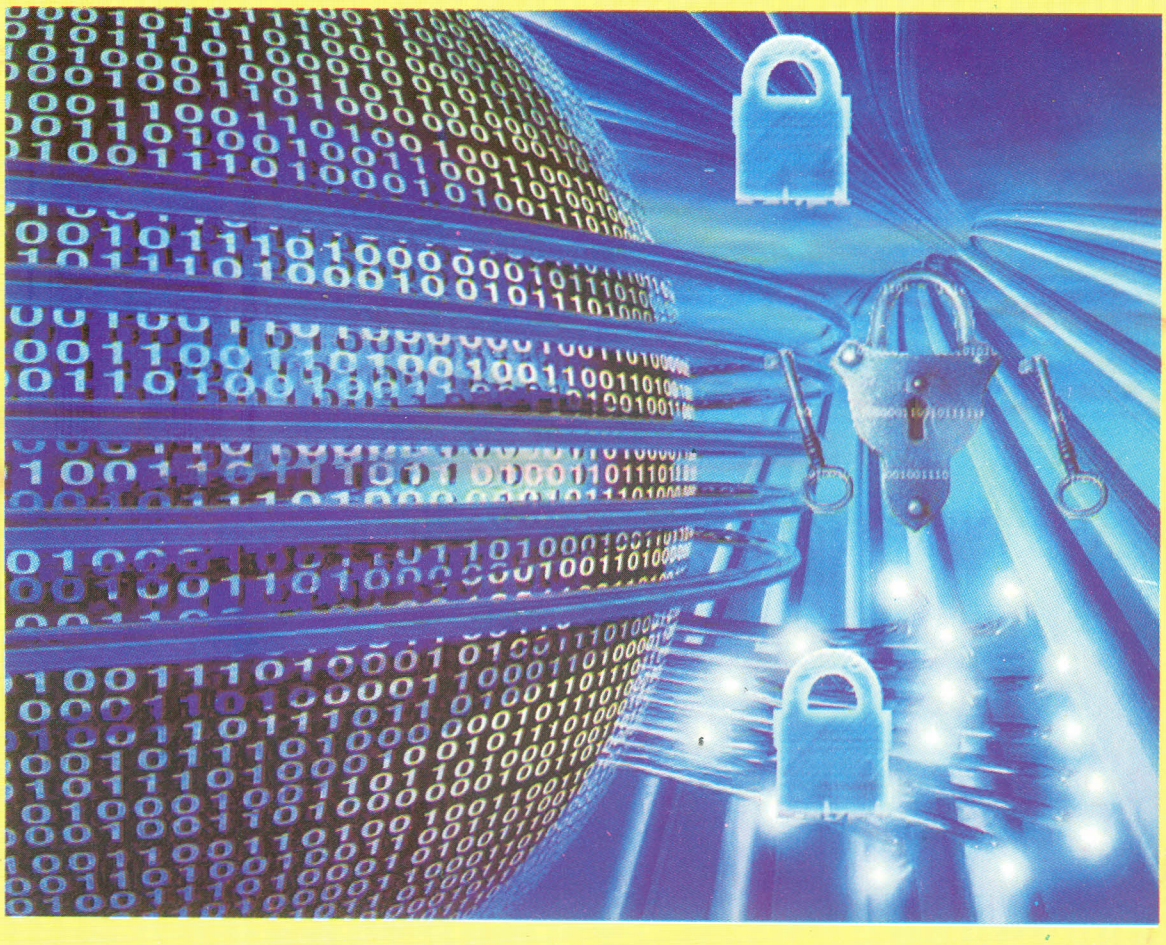
ignou
THE PEOPLE'S
UNIVERSITY

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-023
CYBER SECURITY



Web Technology

3

"शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्रा की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।"

- इन्दिरा गाँधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

- Indira Gandhi

ignou
THE PEOPLE'S
UNIVERSITY
Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-023
Cyber Security

Block

3

WEB TECHNOLOGY

Programme Expert/Design Committee of Post Graduate Diploma in Information Security (PGDIS)

Prof. K.R. Srivathsan
Pro Vice-Chancellor, IGNOU

Mr. B.J. Srinath, Sr. Director & Scientist 'G',CERT-In, Department of Information Technology, Ministry of Communication and Information Technology, Govt of India

Mr. A.S.A Krishnan, Director, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Mr. S. Balasubramony, Dy. Superintendent of Police, CBI, Cyber Crime Investigation Cell Delhi

Mr. B.V.C. Rao, Technical Director, National Informatics Centre, Ministry of Communication and Information Technology

Prof. M.N. Doja, Professor, Department of Computer Engineering, Jamia Milia Islamia New Delhi

Dr. D.K. Lobiyal, Associate Professor, School of Computer and Systems Sciences, JNU New Delhi

Mr. Omveer Singh, Scientist, CERT-In, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology Govt of India

Dr. Vivek Mudgil, Director, Eninov Systems Noida

Mr. V.V.Subrahmanyam, Assistant Professor School of Computer and Information Science IGNOU

Mr. Anup Girdhar, CEO, Sedulity Solutions & Technologies, New Delhi

Prof. A.K. Saini, Professor, University School of Management Studies, Guru Gobind Singh Indraprastha University, Delhi

Mr. C.S. Rao, Technical Director in Cyber Security Division, National Informatics Centre Ministry of Communication and Information Technology

Prof. C.G. Naidu, Director, School of Vocational Education & Training, IGNOU

Prof. Manohar Lal, Director, School of Computer and Information Science, IGNOU

Prof. K. Subramanian, Director, ACIIL, IGNOU Former Deputy Director General, National Informatics Centre, Ministry of Communication and Information Technology, Govt of India

Prof. K. Elumalai, Director, School of Law IGNOU

Dr. A. Murali M Rao, Joint Director, Computer Division, IGNOU

Mr. P.V. Suresh, Sr. Assistant Professor School of Computer and Information Science IGNOU

Ms. Mansi Sharma, Assistant Professor, School of Law, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU
Programme Coordinator

Block Preparation

Unit Writer

Mr. Subhajit Chakrabarty
Director (IT & IS), Apparel Export Promotion Council (AEPC), Gurgaon
(Unit 1, 2, 3 & 4)

Block Editors

Prof. K.R. Srivathsan
Pro Vice-Chancellor
IGNOU

Ms. Urshla Kant
Assistant Professor
School of Vocational Education & Training, IGNOU

Proof Reading

Ms. Urshla Kant
Assistant Professor
School of Vocational Education & Training
IGNOU

Production

Mr. B. Natrajan
Dy. Registrar (Pub.)
MPDD, IGNOU, New Delhi

Mr. Jitender Sethi
Asstt. Registrar (Pub.)
MPDD, IGNOU, New Delhi

Mr. Hemant Parida
Proof Reader
MPDD, IGNOU, New Delhi

August, 2011

ISBN: 978-81-266-5617-2

Further information about the School of Vocational Education and Training and the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110068. or the website of IGNOU www.ignou.ac.in

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi, by the Registrar, MPDD

Laser typeset by Mctronics Printographics, 27/3 Ward No. 1, Opp. Mother Dairy, Mehrauli, New Delhi-30

Printed at A-One Offset Printers 5/34, Kirti Nagar Indl. Area, New Delhi-110 015

BLOCK INTRODUCTION

Web Technology covers the different aspects of web. The web or the internet is a fascinating place which has provided a tremendous opportunity to communicate globally all this could be made possible only through a well structured architecture which rapidly evolved with the times. Present day web architecture is vastly different from that during the formative days of the web. The evolution of the web architecture has also been largely participative, democratic, transparent yet futuristic. The World Wide Web Consortium (W3C) is an international community that develops standards to ensure the long-term growth of the Web. This block comprises of four units and is designed in the following way;

The **Unit one** deals with the introduction to web architecture. This architecture is multi tiered and several protocols and technologies are identified with it. Key components of the architecture are HTML, URI and HTTP. Other key protocols are FTP, SMTP, MIME, SSL/TLS. Key technologies are CGI, Java technologies, VBScript, PHP and ASP.NET.

The **Unit two** describes client side scripts. There are several client side scripting languages but they have several similarities. They are embedded in an HTML document and have several programmatic features such as control structures, conditional statements etc.

The **Unit three** covers about server side scripts. The generic web server architecture contains several sub-systems with the basic purpose of responding to the client request. This is required because the client can not be exposed to all the resources in the web server. Accordingly server side scripts are so designed so as to hide the internal working of the server and also provide scripting embedded within the HTML markup. JSP and ASP are popular server side scripting technologies and ASP.NET provides the functionality of .NET Framework.

Unit four explains attacks on web application. Nothing is absolutely secure in the internet. The various threats to web applications are analyzed and classified into groups. SQL Injection is one such important threat. The attacks can be detected statically or dynamically. Many preventive measures and solutions have emerged.

Hope you benefit from this block.

ACKNOWLEDGEMENT

The material we have used is purely for educational purposes. Every effort has been made to trace the copyright holders of material reproduced in this book. Should any infringement have occurred, the publishers and editors apologize and will be pleased to make the necessary corrections in future editions of this book.

UNIT 1 INTRODUCTION TO WEB ARCHITECTURE

Structure

1.0 INTRODUCTION

The web or the internet is a fascinating place which has provided a tremendous opportunity to communicate globally all this could be made possible only through a well structured architecture which rapidly evolved with the times. Present day web architecture is vastly different from that during the formative days of the web. The evolution of the web architecture has also been largely participative, democratic, transparent yet futuristic. The World Wide Web Consortium (W3C) is an international community that develops standards to ensure the long-term growth of the Web.

1.1 OBJECTIVES

After studying this unit, you should be able to:

- identify the basic purposes and principles of web architecture;
- understand the responsibilities and working of the Internet architecture board;
- understand the basic web architecture;
- recognize related transfer protocols of the web;
- understand the various extensions of the basic web architecture; and
- note some important recent attempts on extensibility of web architecture.

1.2 PURPOSE OF WEB ARCHITECTURE

As no building can be build without a structural plan borne in the mind first, such a vast thing as the internet requires not only an elaborate structural frame work but also with well documented architecture and well designed evolution processes.

1.2.1 Basic Purposes

The basic purpose of web architecture is to provide a structure for the internet, based on ease of use. Internet is a highly successful instrument for human communication primarily because it is so easy to communicate over internet, whether through display of information from websites or through exchanging e-mails and real-time chats.

An important purpose is to plan for its growth in scale organically, i.e. to grow on its own. This is possible because the purpose web architecture is to make it as free as possible. This avoids international socio-political complications.

A purpose of the web architecture is to provide a means of sharing any information whether it be music, messages, poetry, business transactions and so on. With e-business and e-commerce the web architecture has also seeks to provide for security in transactions and privacy of the individual.

1.2.2 Basic Principles

The most important principle of web architecture is that the rules must be unanimously accepted and followed. Therefore, the governing body for web architecture makes generic rules which are applicable to all. This is important because the scope is global and pervasive.

Another principle is that updations and framing of new rules take place only through a process of change requests coming in a free manner and deliberations in a transparent manner so as to avoid the possibilities of binding to specific technologies.

Freedom of expression is a principle strongly advocated in the web architecture. There are examples of nations who have attempted to shut down internet infrastructure for socio-political reasons and also of groups/rivals putting up alternative internet infrastructure so as to communicate globally. Further, blocking of websites is possible at various levels. But then so are alternative and new sites possible.

Cerf and Kahn (1974) gave the following technical principles for internetworkin

- minimalism, autonomy – no internal changes required to interconnect networ

- best effort service model
- stateless routers
- decentralized control.

1.2.3 Unique Features of the Architecture

The first unique feature of the Internet which was the major cause of its success was packet switching. For example, telephone connections of those days required circuit switching indicating the necessity of a continuous circuit connection. Packet switching made transmission of data possible in specific "bursts", remaining silent for the remaining period so that the network could be used for other connections. Also, the packets could now travel through alternate routes intelligently in case some routes were busy.

Another unique feature was the gateway (router) which evolved. These devices could connect various networks with different data rates, packet sizes and specifications.

The global addressing scheme (Internet address) was another unique feature. Thus, computers across the world could be uniquely addressed. A network address was given a unique name of domain (e.g. .com, .edu, .in) and host name/sub-domain.

The history of the Internet depicts these developments:

Mid 1960: Papers on "Packet Switching" emerge.

End 1969s: ARPA sponsors the development of a packet-switching network, called the ARPANET.

1974: The TCP/IP protocols and model are being proposed by Cerf/Kahn.

1980: IPv4 is introduced.

1983: ARPANET adopts TCP/IP.

1984: NSF funds a TCP/IP based backbone network which becomes the NSFNET.

1995: NSF stops funding of NSFNET. The Internet becomes completely commercial.

1.3 GOVERNING THE WEB ARCHITECTURE

The Internet Society (ISOC) is the overall body governing the Internet. The Internet Engineering Task Force (IETF) is the principal body for Internet standards and designs. The IAB is a committee of the IETF and is an advisory body of the ISOC. The IAB was originally called the Internet Activities Board, set up in 1983. The IAB consists of thirteen members. Of these, six are nominated each year from the IETF for a two year term. The thirteen members is the IETF chair. The Internet Research Task Force (IRTF) chair is an ex-officio non-voting member and there is a volunteer Executive Director.

1.3.1 Role of IETF, IAB and IESG in Web Architecture

IAB members are basically generalists but with good knowledge of Internet architecture. The detailed work of the process of Internet Standards is done by the Internet Engineering Steering Group (IESG). The IESG consists of specialists from various technical areas and these experts are drawn from the IETF. The principal body for the development of new standards is, of course, the IETF. The IAB studies the "big picture" of the Internet.

The primary responsibility of the IAB is long-range planning and co-ordination among various areas of IETF activities. The IAB reviews the architectural consistency for the formation of any new IETF working group. The IAB sponsors and organizes the Internet Research Task Force (IRTF) and convenes workshop and reviews.

Other responsibilities of IAB are appellate board against Internet Engineering Steering Group (IESG) actions, advisory body to the ISOC and approval of appointments.

1.3.2 Process of Internet Standards

The process of Standards starts with a Request For Comments (RFC). The steps for publishing and IETF Standard is given below:

a) Publish the RFC document as an Internet-Draft (I-D).

b) Receive comments on the draft.

c) Edit the draft and submitted to IESG.

d) Receive reviews from various review teams.

e) Edit as necessary.

f) Published by the RFC Editor.

Check Your Progress 1

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) Why Web Architecture?

.....

.....

.....

.....

2) Mention three basic principles of Web Architecture.

.....

.....

.....

.....

3) Distinguish between the main roles of IESG and IAB.

.....

.....

.....

.....

4) Fill up the blanks:

a) _____ is the appellate board for actions of the IESG. (IRTF/Internet Arbitrator/IAB).

b) _____ publishes the IETF standard.

1.4 BASIC WEB ARCHITECTURE

The most elementary web architecture is two-tiered; it comprises of a web server and a web client. The Common Gateway Interface (CGI) extends the two-tier architecture to a three-tier architecture by adding a back-end server. Various languages and platforms exist to exploit the two-tier architecture or three-tier architecture.

1.4.1 Two-tier and Three-tier Architecture

The first of the two-tier architecture is the web client which displays the information. Many commonly used web clients are internet browsers such as Internet Explorer, Mozilla, Opera, Netscape and Google Chrome.

The other part of the two-tier architecture is the web server which provides information to the client. The commonly used web servers are IIS for Microsoft technologies and Apache/Tomcat for Java technologies.

This information may be stored with the web server or storage connected to it, directly or indirectly. At the client end small information such as cookies related to session information, user information or temporary transaction information may be stored.

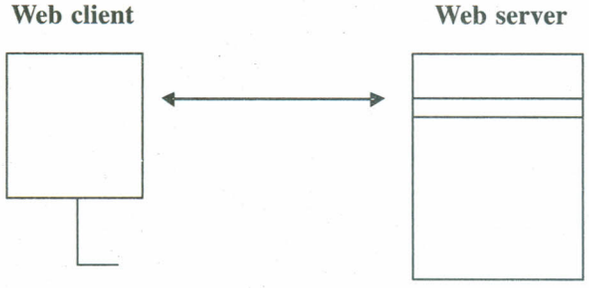


Fig. 1: Basic Two tier architecture

The third tier is the Common Gateway Interface (CGI) which is a set of standards that defines how a dynamic document is written, how data are input to the program and how the output result is used. CGI is not a new language but allows programmers to use any of several languages such as C,C++, Bourne shell, C Shell, Tcl or Perl. The web server interacts with the CGI to provide dynamic content.

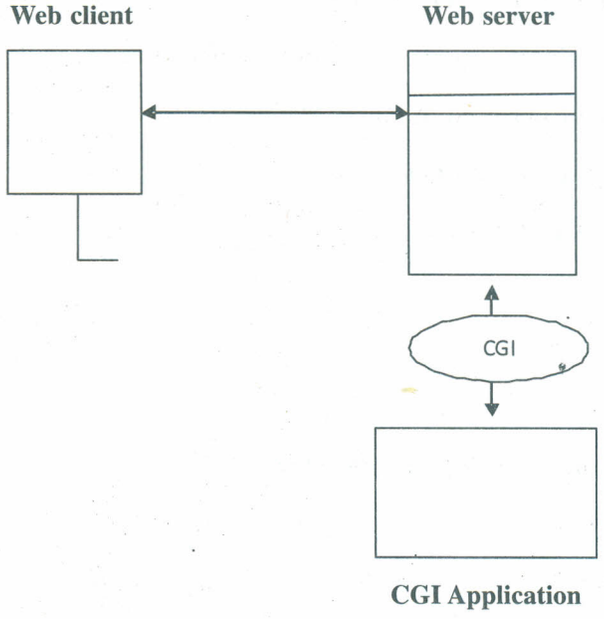


Fig. 2: Basic Three tier architecture

The basic web architecture is associated with three key standards – HyperText Markup Language (HTML), Uniform Resource Identifier (URI) and HyperText Transfer Protocol (HTTP).

1.4.2 HyperText Markup Language

HTML is a language for encoding document content. HTML has evolved out of Standard Generalized Markup Language (SGML). SGML was approved in 1986 as a standard which specifies a meta-language for defining document markup systems through an SGML Document Type Definition (DTD) which specifies valid tag names and element attributes. The tags are hierarchical and structured. HTML was approved as a standard in 1995 with its HTML 2.0 specification. HTML 3.0 was published as a W3C recommendation in 1997 while HTML 4.0 was also published in the same year. HTML 5.0 was published as a working draft by the W3C in 2008.

The HTML tags normally coming pairs like <head> and </head>. The browser does not display the HTML tag but interprets the content within the tag based on the tag. Browsers can refer to Cascading Style Sheets (CSS) to interpret the appearance and layout of the content within the tags.

A sample HTML program with Document Type Declaration (DTD) of HTML 5 is given below.

```
<!DOCTYPE html>
<html>
<head>
<title>Hello HTML</title>
</head>
<body>
<p>Hello World</p>
</body>
</html>
```

The general form of paired HTML tags is:-

```
<tag attribute1="value1" attribute2="value2">content</tag>
```

The general form of unpaired or empty HTML tag is:

```
<tag attribute1="value1" attribute2="value2">
```

Header of the HTML document is given as:

```
<head>
<title>The title</title>
</head>
```

HTML headings are with <h1> to <h6> tags:

```
<h1>Headings1</h1>
<h2>Headings2</h2>
<h3>Headings3</h3>
```

<h4>Headings4</h4>

<h5>Headings5</h5>

<h6>Headings6</h6>

Paragraphs are represented as:

paragraph1

Line breaks are indicated by :-
.

Commented is indicated as:

<!--- this is a comment --- >

Markup for presentation can be indicated as follows:-

bold face

strong text

italic

However, many presentational markup tags are not permitted in HTML5.

Markup for hyper linking to other documents can be indicated as follows:

IGNOU

In case of image as a hyperlink, the <img src= … > tag can be added:

<img src="ignou-symbol.gif"" alt="Click here to go to IGNOU" width="30" height="30" border="5">

HTML 4.0 defines 252 character entity references and 1114050 numeric character references. For example the double-quote character ("), when used to quote an attribute value, must be specified as " or " or " when it appears within the attribute value.

Till HTML 4.0, HTML document start with a doctype such as:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" http://www.w3.org/TR/html4/strict.dtd>

However, HTML5 does not define a DTD, so the doctype declaration is HTML5 is simply:

<!doctype html>

Writing HTML code requires knowledge of HTML syntax. But there are some What You See Is What You Get (WYSIWYG) editors in which the user simply uses Graphical User Interface (GUI) tools to lay out a document and the WYSIWYG editor automatically renders this as an HTML document; the HTML code is automatically generated. Some examples are Frontpage and Dreamweaver editors. However, they often produce verbose and useless code.

1.4.3 Uniform Resource Identifier

A client that wants to access any web page on the internet needs the web address. To access a document over the internet, HTTP uses locators. The Uniform Resource Locator (URL) is standard for specifying the web address, based on network location. Uniform Resource Locator (URL) is subset of Uniform Resource Identifier (URI). The identifier may specify either the location of resource (as a URL) or may specify its name (as a URN, i.e. Uniform Resource Namespace) independent of location of the resource. Therefore a URI could be URL or a URN

These days the term URL is more commonly used in place of URI. The URL defines four things: protocol, host computer, port and path. The protocol is the client/server program used to retrieve the document. Common protocols which can retrieve a document are FTP and HTTP. The host is the computer on which the information is located. The URL can optionally contain the port number of the server; it is inserted between the host and the path and it is separated by a colon. Path is the pathname of the file where the information is located. The path can itself contain directories, subdirectories and files.

1.4.4 HyperText Transfer Protocol

While HTML is used to encode document content, HyperText Transfer Protocol (HTTP) is used to transmit or access data over the web. The HTTP protocol functions as a combination of FTP and SMTP. The HTTP uses only one TCP connection (without separate control connection) on port 80. HTTP messages are read and interpreted by the HTTP server and HTTP client (browser). The client sends an HTTP request to the server while the server sends an HTTP response to the client. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol, i.e. each HTTP request is unrelated to any previous HTTP request as the server is not required to retain session information.

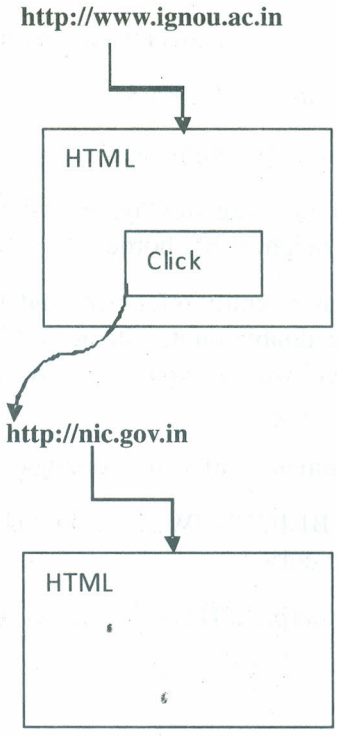


Fig. 3: Stateless HTTP sessions

The HTTP methods are as follows:

Method	Action
GET	Requests a document from the server
HEAD	Requests a information about a document but not the document itself
POST	Sends some information from client to the server
PUT	Sends a documents from server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

The HTTP status codes are as follows:

Code	Phrase	Description
		Informational
100	Continue	The initial part of the requests has been received and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
		Success
200	Ok	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted but it is not immediately acted upon.
204	No content	There is no content in the body.
		Redirection
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
		Client error
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format request is not acceptable.
		Server error
500	Internal server error	There is an error, such as a crash at the server side.
501	Not implemented	The action requested can not be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Check Your Progress 2

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What are components of the three-tier web architecture?

.....

.....

.....

.....

2) How is a hyperlink to another document made in HTML?

.....

.....

.....

.....

3) What does URL define?

.....

.....

.....

.....

4) What is a stateless protocol?

.....

.....

.....

.....

1.5 RELATED PROTOCOLS FOR TRANSFER

1.5.1 FTP

File Transfer Protocol (FTP) is a mechanism provided by TCP/IP for transferring a file from one host to another. FTP differs from other client/server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses), making FTP more efficient. FTP uses TCP port 21 for the control connection and port 20 for the data connection. The control connection opens when a user starts an FTP session. In this situation in which the control connection is open, data connections can be opened and closed multiple times for multiple file transfers sequentially.

FTP can transfer a file using one of the following three transmission modes:

- Stream mode
- Block mode
- Compressed mode

Stream mode is the default mode data delivered as a continuous stream of bytes. If the data are simply a stream of bytes end of file is the closing of the data connection by the sender. If the data are divided into records, each record will have a 1-byte end of record character and the end of the file will have a one byte end-of file character.

In block mode, data can be delivered in blocks. Each block is preceded by a 3 byte header, the block descriptor; the next two bytes define the size of the blocks in bytes.

In the compressed mode, the data can be compressed and transferred.

1.5.2 SMTP

Simple Mail Transfer Protocol (SMTP) is an Internet standard for e-mail transmission. SMTP uses TCP port 25. Electronic mail servers use SMTP for outgoing and incoming mail transport while user level client applications use SMTP only for sending messages to a mail server. Client mail applications typically use Post Office Protocol (POP), Internet Message Access Protocol (IMAP), Microsoft Exchange, Lotus Notes etc for receiving mail messages.

The Mail User Agent (MUA) or the mail client submits e-mail to the Mail Submission Agent (MSA) using SMTP. The MSA delivers the mail to the Mail Transfer Agent (MTA). The MSA and the MTA are often on the same machine. The MTA locates the target by using the Domain Name System (DNS) and delivers it to a Mail Delivery Agent (MDA) which saves the message in mail box format. This mail can be retrieved by the Mail User Agent (MUA).

Following are typical SMTP keywords:

Keyword	**Arguments**
HELLO	Sender's host name
MAIL FROM	Sender of the message
RCPT TO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RESET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name
SENT FROM	Intended recipient of the message
SMOL FROM	Intended recipient of the message
SMAL FROM	Intended recipient of the message
Following are typical SMTP responses:	
Code	**Description**
	Positive Completion Reply
211	Systems status or health reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	Using not local; the message will be forwarded

	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implement
550	Command is not executed; mail box unavailable
551	User not local
552	Requested action aborted; exceeded shortage location
553	Requested action not taken; mail box name not allowed
554	Transaction failed

1.5.3 MIME

Electronic mail has some limitations. It can send messages only in NVT 7-bit ASCII format. For example, it is can not be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Chinese) and it can not be used to send binary files or video/audio.

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail by transforming them to ASCII data. The message at the receiving end is transformed back to the original data.

MIME defines five headers that can be added to the original e-mail header section:

1) MIME version
2) Content-type
3) Content-transfer-encoding
4) Content-ID
5) Content-Description

1.5.4 SSL/TLS

Secure Socket Layer (SSL) is designed to provide communication security over the internet. The data received from the application are compressed (optionally), signed and encrypted. Transport Layer Security (TLS) protocol is the successor to SSL; they are cryptographic protocols.

First, a client requests for a secure connection from a server. The server checks the best cipher and hash function from the client and informs the same to the client. The server identifies itself to the client by sending its digital certificates containing the certificate authority. The client encrypts a random number with the

server's public key and sends the result to the server. The server decrypts it with its private key. This completes the handshaking procedure and begins the secured connection. Communication is encrypted during transmission using the secured connection.

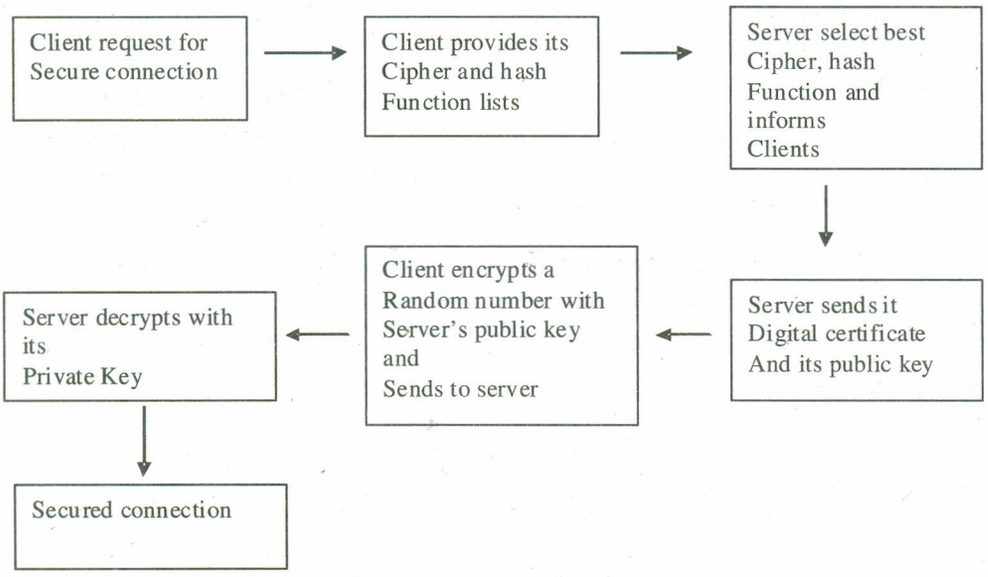


Fig. 4: Handshaking in TLS

1.6 EXTENSIBILITY OF THE BASIC ARCHITECTURE

As mentioned earlier the basic architecture is three-tiered with CGI. Various technologies have added to the dimension of the basic architecture. Recent attempts have been made to improve efficiency also apart from the functionality.

1.6.1 Common Gateway Interface

The term Gateway means that a CGI program can be used to access other resources such as databases, graphical packages etc. the term interface means that there is a set of predefined terms, variables, calls etc that can be used in any CGI program.

After the client makes a request to the server, the server uses the registered CGI program which executes and returns MIME messages back to the server. Generally, for security precautions, CGI programs must be run from a specified directory (e.g. /cgi-bin) under control of the administrator. The most common CGI applications handle HTML <FORM> and <ISINDEX> commands. Information passed from client to server through environment variables encoded in URLs may provide the arguments to the CGI program.

1.6.2 Java Technologies

Java is an elaborate programming language which is similar to C++ in having Object Oriented Programming (OOP) features but having a simpler object model and fewer low level features. Java applications are compiled to class files that run on a Java Virtual Machine (JVM), irrespective of computer architecture. Therefore java is portable. End users may use a Java Runtime Environment (JRE) for java applications or a web browser for java applets.

Java applets are programs that are embedded in other applications, such as web page. The JApplet class provides the framework for the applet. The applet is also placed in the HTML document using the <applet> HTML element, in order to view the applet in the web browser.

Java Servlets are server-side java components that run like an applet on the server side. The GenericServlet class provides the framework for the servlet. Servlets respond to HTTP requests from clients by throwing HTML pages from server side.

JavaServer Pages (JSP) embed java code in a HTML page by using special delimiters <% and %> in HTML. The JSP is compiled to a java servlet where it is accessed for the first time.

Swing is a Graphical User Interface (GUI) for java. The JFrame class provides the framework for Swing. Swing is light weight and built on the Abstract Windowing Toolkit (AWT) which is heavy weight.

The Java Database Connectivity (JDBC) is an Application Programming Interface (API) of java that provides for accessing a database. The Class.forName (String) method loads the JDBC driver class from some JDBC vendor.

1.6.3 PHP

PHP is a scripting language originally designed to produce dynamic web pages. The focus is on server-side scripting. PHP is part of the popular LAMP architecture where L is Linux, A is Apache, M is MySQL and P can be PHP, Python or Perl. PHP is also part of WAMP (Windows/Apache/MySQL/PHP) and MAMP (Mac OS X/Apache/MySQL/PHP).

PHP can be deployed on most web servers, most platforms and most databases. PHP can take input from file or string (say containing text) and provide output in another stream (commonly as HTML). Compilers and OOP features are available.

1.6.4 VBScript

Visual Basic Scripting Edition (VBScript) is a scripting language provided by Microsoft and syntactically similar to Visual Basic. VB Script is installed by default in Microsoft Windows desktops and servers. VBScript can be executed by Internet Explorer (IE), Internet Information Services (IIS) and Windows Script Host (WSH). VBScript is mostly used for client-side scripting. It is similar to JavaScript. When Active Server Pages (ASP) is used for server-side scripting, VBScripting is embedded in an ASP page within <% and %> switches.

1.6.5 ASP.NET

ASP.NET is a web application framework provided by Microsoft to develop web applications, dynamic web sites and web services. It is the successor to ASP and is built on the Common Language Runtime (CLR) to write code using any supported .NET language.

ASP.NET web pages have ".aspx" extension, typically containing static pages. The dynamic code (or the "code-behind") can be contained in an ".aspx.vb", ".aspx.cs", ".aspx.fx" files etc.

Event based code is written such as loading of a page, refreshing of a page, clicking on a control etc. A web control such as a button or a combo box can be assigned properties and response to specified events can be coded.

1.6.6 HTTP Extension, HTTP-NG and SMUX

Working groups where established in W3C to explore the feature development of the HTTP protocol. HTTP-NG was proposed to improve the performance of HTTP 1.1.

HTTP exposes three interfaces, GET, PUT and POST, for retrieving and manipulating data. The HTTP Extension framework was proposed to provide a way for parameters that are visible to HTTP.

The SMUX was proposed as a multiplexing transport mechanism allowing multiple protocols over the same TCP connections. Sharing of congestion information in TCP control blocks may improve TCP performance and thereby improve HTTP transactions.

Check Your Progress 3

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What are the connections in FTP?

.....

.....

.....

.....

.....

2) How does SMTP work with electronic mail servers?

.....

.....

.....

.....

.....

3) Explain the TLS handshake.

.....

.....

.....

.....

.....

4) Write the main advancement of ASP.NET over VBScript.

.....

.....

.....

.....

.....

1.7 LET US SUM UP

This unit is an introduction to the architecture of the web. This architecture is multi tiered and several protocols and technologies are identified with it. Key components of the architecture are HTML, URI and HTTP. Other key protocols are FTP, SMTP, MIME, SSL/TLS. Key technologies are CGI, Java technologies, VBScript, PHP and ASP.NET.

1.8 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) a) To provide a structure for the Internet.

   b) To plan for its growth in scale organically.

   c) To provide a means of sharing information of various types.

2) a) The rules must be unanimously accepted and followed.

   b) Updations and framing of new rules take place only through a process of change requests coming in a free manner and deliberations in a transparent manner.

   c) Freedom of expression.

   Cerf and Kahn (1974) gave the following technical principles for internetworking:

   - minimalism, autonomy – no internal changes required to interconnect networks
   - best effort service model
   - stateless routers
   - decentralized control.

3) IAB members are basically generalists but with good knowledge of Internet architecture. The IESG consists of specialists from various technical areas and these experts are drawn from the IETF. The IAB studies the "big picture" of the Internet. The detailed work of the process of Internet Standards is done by the Internet Engineering Steering Group (IESG).

4) a) IAB

   b) RFC Editor

Check Your Progress 2

1) a) Web client

   b) Web server

   c) CGI

2) IGNOU

3) URL defines the web address, based on network location. It contains protocol, host computer, port and path.

4) Stateless protocol is one in which the server is not required to retain session information, e.g. HTTP is stateless; each HTTP request is unrelated to any previous HTTP request.

Check Your Progress 3

1) There are two TCP connections in FTP: one connection is used for data transfer (port 21), the other connection is for control information (port 20).

2) Refer to Sub-section 1.5.2

3) Refer to Sub-section 1.5.4

4) a) VBScript is mostly used for client-side programming.

b) ASP.NET is built on CLR and can be written using any .NET supported languages.

c) Dynamic code can be written elegantly in a separate file in ASP.NET, with static page written in another file.

1.9 SUGGESTED READINGS

- Chris Bates (2002), *Web Programming Building Internet Applications, 2 Ed*, Wiley-India.
- Ethan Zuckerman & Andrew McLaughlin (2003), *Introduction to Internet Architecture and Institution, http://cyber.law.harvard.edu/ digitaldemocracy/ internetarchitecture.html.*
- http://www.iab.org.
- http://www.ietf.org.
- http://www.w3.org.
- RFC 1958 (1996), "*Architectural Principles of the Internet*", http:// www.ietf.org/rfc/rfc1958.txt.

UNIT 2 CLIENT SIDE SCRIPTS

Structure

2.0 INTRODUCTION

The client side scripts are the programming at client end through scripting languages. Typically, code is written within the HTML document and the code is interpreted when the browser is run. The client side script can make a request to the web server and the server side script may then make a response by returning an HTML page to the client. Several technologies exist for client side scripting. VBScript, ASP and ASP.NET have emerged from Microsoft. JavaScript, JSP are competing technologies to Microsoft. We first examine the internal working of the client browser. Than we understand the language of VBScript and JavaScript.

2.1 OBJECTIVES

After studying this unit, you should be able to:

- get introduced to the internal working of the client browser;
- understand basic VBScript; and
- understand basic JavaScript.

2.2 INTERNAL WORKING OF THE CLIENT BROWSER

The browser has been a revolutionary application. It serves as a gateway to the World Wide Web (WWW) and can throw stateless HTML pages to the client in a cross-platform manner and a wide variety of hardware ranging from cell phones to tablet PCs.

2.2.1 Brief History

Tim Berners-Lee invented the first web browser in 1991. It was text-only and had an HTML editor. Thereafter, another text-only browser called Lynx was made for use with the WWW. In 1993, a graphical browser called Mosaic was invented. Netscape was built on Mosaic soon after. The World Wide Web Consortium (W3C) was founded to promote interoperability among web technologies, quite early in 1994. The Internet Explorer (IE) was released by Microsoft in 1995 and has been dominating the market since with its versions. Another closed-source product which emerged was Opera. Netscape open-sourced their browser under the name Mozilla in 1998. Firefox is a stand-alone alternative of Mozilla with more integrated features. Another open source browser which emerged was the Konqueror. Apple wrote their OS X web browser, Safari. There have been significant code re-use in Konqueror, Safari, IE, Firefox and Netscape. In Netscape 8, based on Firefox, the user may switch between IE-based rendering and Mozilla-based rendering on the fly.

2.2.2 Architecture of the Browser

The architectures of browsers differ but they have generic features. The browsers can be generally considered to be comprised of eight major sub-systems: (1) the User Interface, (2) the Browser Engine, (3) the Rendering Engine, (4) the Networking subsystem, (5) the Scripting Interpreter, (6) the XML Parser, (7) the Display Backend and (8) the Data Persistence subsystem.

The *User Interface* is the basic interface available to the user. Going deeper, the *Browser Engine* provides a high-level interface for querying and manipulating the *Rendering Engine*. The *Rendering Engine* performs parsing and layout for HTML documents with styling in CSS. The *Networking subsystem* provides the features for connecting over the WWW e.g. through HTTP requests. The *Scripting Interpreter* is to interpret JavaScript or other scripting languages. The *XML Parser* is used for parsing so that it can create a parse tree which is ultimately needed to paint/display the information through the user interface. The *Display Backend* provides drawing and windowing primitives, user interface widgets and fonts. The *Data Persistence* subsystem stores various data and session information, in disk or RAM, including bookmarks, cookies and cache.

The above generic architecture is shown diagrammatically below:

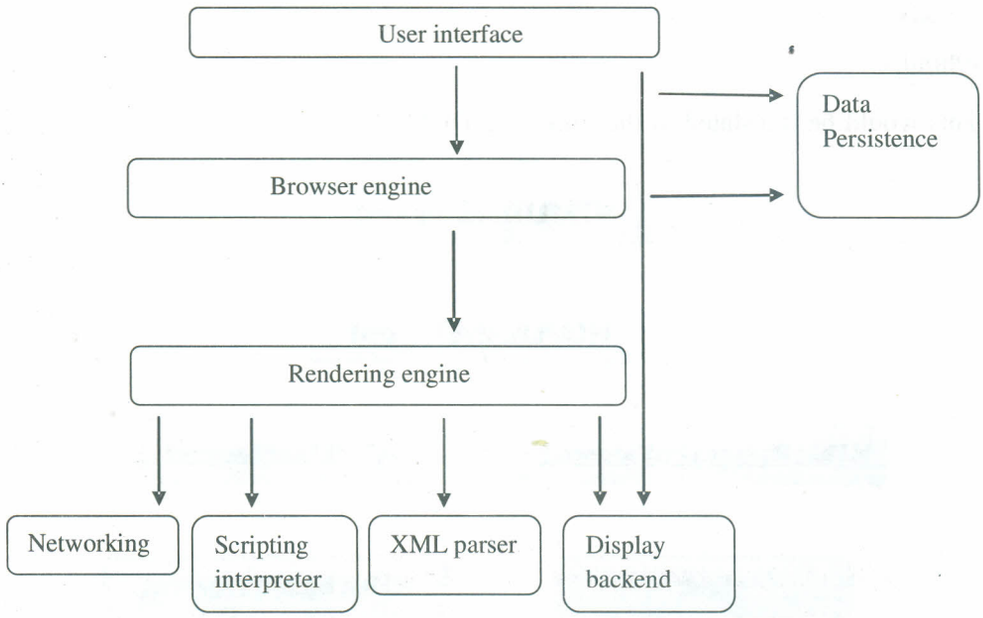


Fig. 1: Generic architecture of a web browser

Each type of browser has its own fine-tuning of the generic architecture. For example, in Mozilla, the *User Interface* is split over two subsystems (to add a User Interface Toolkit). All data persistence is provided by Mozilla's profile mechanism, which stores both high-level data such as bookmarks and low-level data such as a page cache. This is shown below:-

2.2.3 Working of the Internal Sub-systems

The browser's main work is to present the web resource (the web page identified by the address), by requesting it from the server and painting/displaying it on the browser window. The resource format is usually HTML but can be PDF, image etc. The way the browser interprets and displays HTML files is specified in the HTML and CSS specifications. These specifications are maintained by the W3C (World Wide Web Consortium). There is no formal specification for the browser's user interface.

The most important sub-system is the rendering engine which executes the primary function of displaying HTML, XML documents and images. The engine can display other types (like PDF) through browser extensions/plug-ins.

The rendering engine gets the contents of the requested document from the networking layer. This is usually done in 8K chunks. The HTML document is parsed (translated to a structure such as a tree) into a content tree containing the HTML tags as nodes. This tree is called the Document Object Model (DOM) tree. The styling information (such as colour and dimensions) is added to the tree to create the Render tree. The exact co-ordinates of the layout are then added to each node. Finally, a paint program invokes traversing of each node and utilizing the Display Backend to display on the window.

Let us take the example of the following HTML document:

```
<html>
<body>
<p>
Hello World
</p>
<div> <img src="example.png"/></div>
</body>
</html>
```

This would be translated to the following DOM tree:

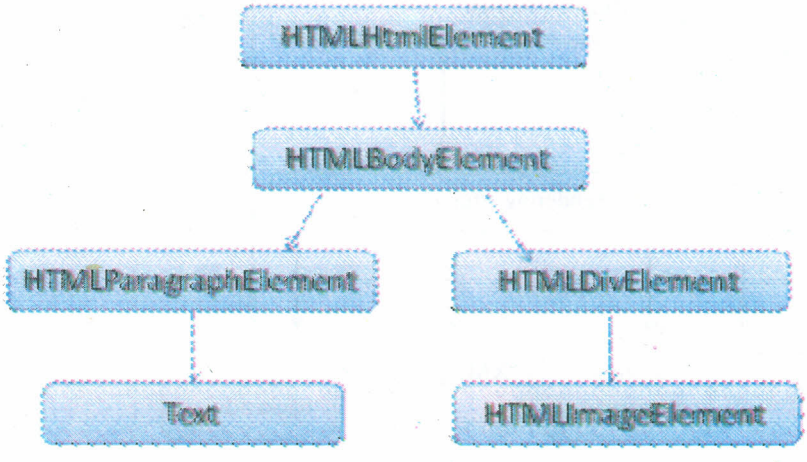


Fig. 2: DOM tree example

The flow of the rendering engine is summarised below:-

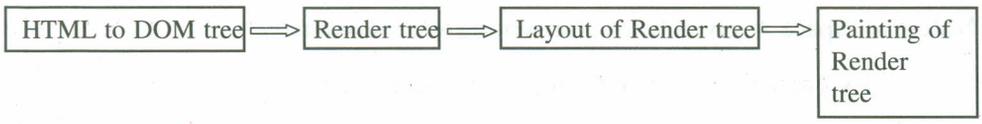


Fig. 3: The basic flow of the Rendering engine

In Firefox, there are two extra trees for adding styling – the rule tree and style context tree.

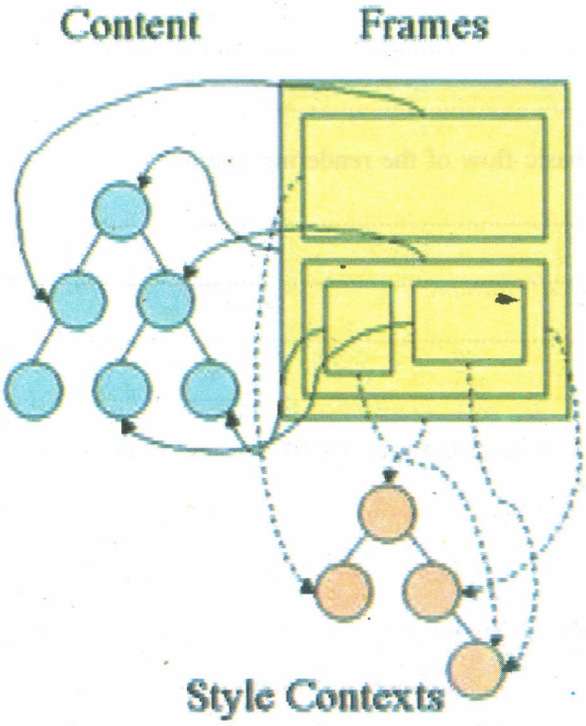


Fig. 4: Rule tree and Style context tree

The flow of Mozilla's Gecko rendering engine is shown below as an example:

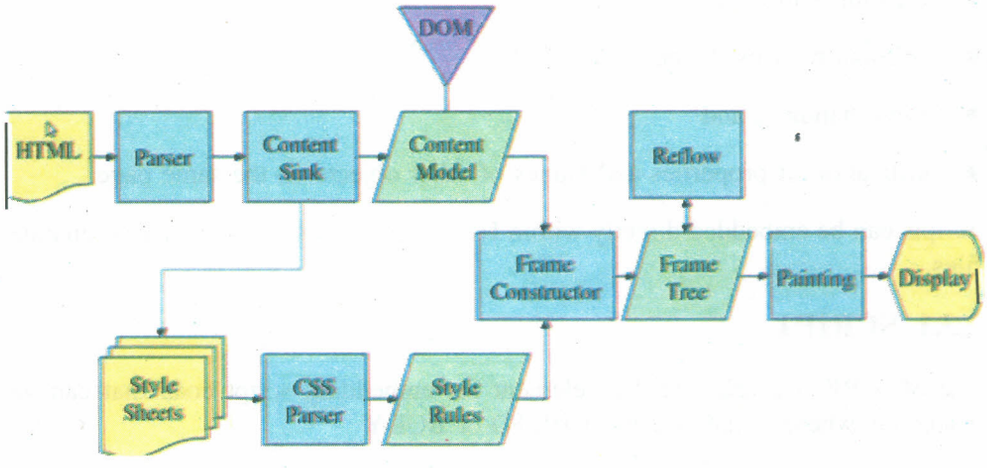


Fig. 5: Flow of Mozilla's Gecko rendering engine

Gecko has an extra layer, the "content sink", between the HTML and the DOM tree. The Content Sink produces the elements of the DOM. Each element is a frame. The Cascading Style Sheet (CSS) Parser produces the Style Rules tree (Rule and Style context). Gecko calls the Render tree as the "Frame tree" and calls the Layout (for placing elements) as "Reflow".

Check Your Progress 1

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What are the major sub-systems of the web browser?

.....

.....

.....

.....

2) What is the basic flow of the rendering engine?

.....

.....

.....

.....

3) What is the difference between DOM tree and Render tree?

.....

.....

.....

.....

2.3 HOW TO WRITE CLIENT SIDE SCRIPTS?

Scripts are generally written for the following purposes:-

- data input in forms,
- validation check of the input data,
- event handling and
- utilization of properties and values of other objects on the same page.

Scripts can be embedded directly within HTML documents or supplied in separate files.

2.3.1 SCRIPT

The SCRIPT is a character-like element for embedding script code that can be placed anywhere in the document HEAD or BODY

The technical specification of the SCRIPT element is given below:

<!ELEMENT script - - CDATA>

<!ATTLIST script

type CDATA #IMPLIED -- media type for script language --

language CDATA #IMPLIED -- predefined script language name --

src %URL #IMPLIED -- URL for an external script -

defer (defer) #IMPLIED --- user may defer execution of script --

>

The explanation of this specification is rather simple; ATTLIST is the list of attributes: type, language, src and defer, all of which are optional.

TYPE specifies the scripting language, for instance: type="text/javascript" or type="text/vbscript".

LANGUAGE specifies the scripting language, for instance "JavaScript" or "VBScript". This attribute is deprecated (not favoured) in favour of the TYPE attribute.

SRC gives a URL for an external script. If a SRC attribute is present, the content of the SCRIPT element should be ignored.

DEFER is an option to defer the execution of the script.

The SCRIPT element is defined as CDATA, in which only one delimiter is recognized: the end tag open (ETAGO) delimiter (i.e. the string "</") immediately following a character ([a-zA-Z]). For example, the following code will not work due to the "</XYZ>" characters found inside of the SCRIPT element:

<SCRIPT type="text/javascript">

document.write ("<XYZ>This won't work</XYZ>")

</SCRIPT>

This code can be expressed legally as follows:-

<SCRIPT type="text/javascript">

document.write ("<XYZ>This will work<\/XYZ>")

</SCRIPT>

In VBScript, it can be expressed legally with the Chr() function, e.g.

"<XYZ>This will work<\" & Chr(47) + "XYZ>"

HTML documents can include multiple SCRIPT elements, which can be placed in the document HEAD or BODY. The script statements for a form can be placed near to the corresponding FORM element.

2.3.2 META and Default Scripting Language

The default scripting language in the absence of TYPE or LANGUAGE attributes can be specified by a META element in the document HEAD, for example:

<META HTTP-EQUIV="Content-Script-Type" CONTENT="text/vbscript">

If there are several such META elements the last one determines the scripting language.

In the absence of either a META element or an HTTP header, the default scripting language is assumed to be JavaScript in most browsers.

2.3.3 NOSCRIPT

What happens if the browser does not support running of scripts?

The **NOSCRIPT element** is used for the purpose:

<!ELEMENT noscript - - (%body.content)>

The content of this element is rendered only in this case. For example, the user may be invited to upgrade to a newer browser or to enable scripts.

<NOSCRIPT>

<P>This document works best with script enabled browsers

</NOSCRIPT>

Check Your Progress 2

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) Give the specification of the SCRIPT element.

.....

.....

.....

.....

.....

2) How can we set the default scripting language at the client end?

.....

.....

.....

.....

.....

2.4 LANGUAGE

We now proceed with the language examples of VBScript and JavaScript. One important point to be seen is that these are "scripting" languages and not pure languages like C, C++ or Java. Therefore, one does not find any main function. These scripting languages cannot run on their own but need to be embedded within the document to be scripted.

2.4.1 Examples in VBScript

The various data types in VBScript are given below: -

Subtype	Description
Empty	Variant is uninitialized. Value is 0 for numeric variables or a zero-length string ("") for string variables.
Null	Variant intentionally contains no valid data.
Boolean	Contains either True or False.
Byte	Contains integer in the range 0 to 255.
Integer	Contains integer in the range -32,768 to 32,767.
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.

Long	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Single	Contains a single-precision, floating-point number in the range -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
String	Contains a variable-length string that can be up to approximately 2 billion characters in length.
Object	Contains an object.
Error	Contains an error number.

The keywords in VBScript are very similar to Visual Basic. For example Dim is used for declaration, If ... Then ... Else is used in loop/control flow, CInt is used to convert Character to Integer, On Error is used for error handling and so on. The various categories of keywords in VBScript are given below:

Category	**Keywords**
Array handling	Array, Dim, Erase, IsArray, LBound, Private, Public, ReDim, UBound
Assignments	Set
Comments	Comments using ' or Rem
Constants/Literals	Empty, False, Nothing, Null, True
Control flow	Do...Loop, For...Next, For Each...Next, If...Then...Else, Select Case, While...Wend, With
Conversions	Abs, Asc, AscB, AscW, CBool, CByte, CCur, CDate, CDbl, Chr, ChrB, ChrW, CInt, CLng, CSng, CStr, DateSerial, DateValue, Fix, Hex, Int, Oct, Sgn, TimeSerial, TimeValue
Dates/Times	Date, DateAdd, DateDiff, DatePart, DateSerial, DateValue, Day, Hour, Minute, Month, MonthName, Now, Second, Time, TimeSerial, TimeValue, Weekday, WeekdayName, Year
Declarations	Class, Const, Dim, Function, Private, Property Get, Property Let, Property Set, Public, ReDim, Sub
Error Handling	Err, On Error
Expressions	Eval, Execute, RegExp, Replace, Test
Formatting Strings	FormatCurrency, FormatDateTime, FormatNumber, FormatPercent
Input/Output	InputBox, LoadPicture, MsgBox
Literals	Empty, False, Nothing, Null, True

Math	Atn, Cos, Exp, Log, Randomize, Rnd, Sin, Sqr, Tan
Miscellaneous	Eval Function, Execute Statement, RGB Function
Objects	CreateObject, Err Object, GetObject, RegExp
Operators	Addition (+) and, Division (/), Equality (=), Eqv, Exponentiation (^), Greater Than (>), Greater Than or Equal To (>=), Imp, Inequality (<>), Integer Division (\), Is, Less Than (<), Less Than or Equal To (<=), Modulus arithmetic (Mod), Multiplication (*), Negation (-) or, String concatenation (&), Subtraction (-), Xor
Options	Option Explicit
Procedures	Call, Function, Property Get, Property Let, Property Set, Sub
Rounding	Abs, Fix, Int, Round, Sgn
Script Engine ID	ScriptEngine, ScriptEngineBuildVersion, ScriptEngine MajorVersion, ScriptEngineMinorVersion
Strings	Asc, AscB, AscW, Chr, ChrB, ChrW, Filter, InStr, InStrB, InStrRev, Join, LCase, Len, LenB, Left, LeftB, LTrim, Mid, MidB, Replace, Right, RightB, RTrim, Space, Split, StrComp, String, StrReverse, Trim, UCase
Variants	IsArray, IsDate, IsEmpty, IsNull, IsNumeric, IsObject, TypeName, VarType

A single-dimension array containing 6 elements can be declared as follows:

Dim A(5)

Beginning at zero and ending at 5, data can be assigned to the elements of an array as follows:

A(0) = 100

A(1) = 500

. . .

A(5) = 400

You can have as many as 60 dimensions. For example, the MyTable variable is a two-dimensional array consisting of 10 rows and 5 columns:

Dim MyTable(9, 4)

You can also declare an array whose size changes when the script is running. This type of array is called a dynamic array. The array is initially declared without mentioning any size or number of dimensions inside the parenthesis using either the Dim statement or using the ReDim statement as follows:

Dim A()

ReDim B()

To use the dynamic array, ReDim is used subsequently such as:

ReDim A(25)

The Preserve keyword to preserve the contents of the array as the resizing takes place.

```
ReDim Preserve A(30)
```

User-defined constants can be created in VBScript using the Const statement. For example:

```
Const MyString = "This is my string."

Const MyAge = 49

Const CutoffDate = #06/18/2008#
```

This scripting example captures the difference in dates:

```
<HTML>

<HEAD>

<TITLE>Place Your Order</TITLE>

<SCRIPT LANGUAGE="VBScript">

<!--

Function TimeSpan(Dt)

TimeSpan = (Now() – CDate(Dt))

End Function

-->

</SCRIPT>

</HEAD>

<BODY></BODY>

</HTML>
```

The following shows the code for If … Then … ElseIf …. Else, as in Visual Basic:

```
Sub ReportValue(value)

If value = 2009 Then

MsgBox value

ElseIf value = 2010 Then

MsgBox value

ElseIf value = 2011 then

Msgbox value

Else

Msgbox "Choose between 2009, 2010 and 2011"

End If
```

The following example causes a procedure called MyLoop to execute 100 times.

```
Sub DoLoop()

Dim x

MsgBox "Start"

For x = 1 To 100

MyLoop
```

```
Next
Msgbox "The last counter is" & x
End Sub
```

In the following example, the fCelsius function calculates degrees Celsius from degrees Fahrenheit. When the function is called from the ConvertFahrenheit ToCelsius Sub procedure, a variable containing the argument value is passed to the function. The Function returns a value (in VBScript the return type is variant) which is displayed in a message box.

```
Sub ConvertFahrenheitToCelsius()
temp = InputBox("Enter temperature in degrees Fahrenheit", 1)
MsgBox "The temperature is" & fCelsius(temp) & "degrees Celsius."
End Sub
Function fCelsius(fDegrees)
fCelsius = (fDegrees – 32) * 5/9
End Function
```

Events associated with the objects in the HTML page can be handled with scripting.

In VBScript, an example is:

```
<INPUT NAME=txtBox1 size=50>
<SCRIPT TYPE="text/vbscript">
Sub txtBox1_changed()
If txtBox1.value = "India" Then
buttonSubmit.enabled = True
Else
buttonSubmit.enabled = False
End If
End Sub
</SCRIPT>
```

The following scripting example shows the message box when a button is clicked with a mouse:

```
<HTML>
<HEAD>
<TITLE>This is an example</TITLE>
</HEAD>
<BODY>
<FORM NAME="frmButtonExample">
<INPUT TYPE="Button" NAME="btnExample" VALUE="Click">
<SCRIPT FOR="btnExample" EVENT="onClick" LANGUAGE="VBScript">
MsgBox "Button Pressed!"
```

</SCRIPT>

</FORM>

</BODY>

</HTML>

2.4.2 Examples in JavaScript

Core JavaScript contains a set of core objects such as Array and Date and a set of core language elements such as operators, control structures and statements. Client side JavaScript is extended from the core language.

The syntax for JavaScript language is similar to that of C. It uses prototypes instead of classes. You can define an object by creating a constructor function. JavaScript is a loosely typed language, which means one does not declare the data types of variables explicitly. In many cases JavaScript performs conversions automatically when they are needed. For example, if you add a number to an item that consists of text (a string), the number is converted to text.

As in C, the language of JavaScript has do ... while loop, for loop, if ... else conditional statement, try ... catch error handling etc. The important statements in JavaScript are given below:

Description	Language Element
Terminates the current loop, or if in conjunction with a label, terminates the associated statement.	break Statement
Contains statements to execute when an error occurs in code within the try block.	catch Statement
Activates conditional compilation support.	@cc_on Statement
Causes comments to be ignored by the JavaScript parser.	Comment Statements
Stops the current iteration of a loop, and starts a new iteration.	continue Statement
Starts the debugger.	debugger Statement
Executes a statement block once, and then repeats execution of the loop until a condition expression evaluates to **false**.	do...while Statement
Executes a block of statements for as long as a specified condition is **true**.	for Statement
Executes one or more statements for each element of an object or array.	for...in Statement
Declares a new function.	function Statement
Conditionally executes a group of statements, depending on the value of an expression.	@if Statement
Conditionally executes a group of statements, depending on the value of an expression.	if...else Statement
Provides an identifier for a statement.	Labeled Statement

Exits from the current function and returns a value from that function.	return Statement
Creates variables used with conditional compilation statements.	@set Statement
Enables the execution of one or more statements when a specified expression's value matches a label.	switch Statement
Refers to the current object.	this Statement
Generates an error condition that can be handled by a **try...catch** statement.	throw Statement
Implements error handling for JavaScript.	try Statement
Declares a variable.	var Statement
Executes a statement until a specified condition is **false**.	while Statement
Establishes the default object for a statement.	with Statement

Events associated with objects can be handled in JavaScript extrinsically as well as intrinsically.

For example, the following code invokes a function, named say example_onclick(), when the button is clicked:

```
<script language=JavaScript>

function example_onclick() {

 . . .

}

document.form.button.onclick = example_onclick

</script>
```

In intrinsic event handling, the event can be handled using attributes placed on the HTML elements associated with the object generating the event.

```
<INPUT NAME="txtBoxUserName" onBlur="validUserName(this.value)">
```

In the above example of JavaScript, the input text box is named "txtBoxUserName". A blur event occurs when a form field loses the input focus. So after the user enters the data and the focus is shifted to another object, the function "validUserName" checks the validity of the entered data.

Some of important intrinsic events are given below:

onLoad: when the browser finishes loading a window or all frames.

onUnload: on exit of a document.

onClick: when clicked on buttons, checkboxes, radio buttons, hypertext links, reset, submit buttons etc.

onMouseOver: when the mouse is moved onto an anchor.

onMouseOut: when the mouse is moved out of an anchor.

onFocus: when a field gains the input focus by tabbing or clicking with the mouse.

onBlur: when a form field loses the input focus.

onSubmit: when a user submits a form.

onSelect: when a user selects some of the text within a single or multi-line text field.

onChange: when a form field loses the input focus and its value has been modified.

The following example randomizes the document background color:

```
<BODY BGCOLOR='&{randomrbg()};'>
```

This example dims the background for evening hours ...

```
<BACKGROUND SRC='&{if(Date.getHours > 18)...};'>
```

This example sets the size of an image based upon document properties:

```
<IMG SRC=bar.gif WIDTH='&{document.banner.width/2};' HEIGHT='50%'>
```

Here is an interesting example where the password can be captured by a client-site script!

```
<html>
<head>
<meta  http-equiv="Content-Type"      content="text/html;
charset=UTF-8">
<SCRIPT LANGUAGE=JAVASCRIPT>
function showpassword(form1)
{
pd=form1.elements[1].value;
alert('You entered password: '+pd);
}
</SCRIPT>
</head>
<body>
<FORM Name="Login"> NAME:
<INPUT Type="Text" Name="UserID" Value=""><br><br>
Password:
<INPUT Type="Password"      Name="Password" Value=""><br><br>
<INPUT Type="Button" Name="SubmitButton" Value="Submit"
onClick="showpassword(form)">
</Form>
</body>
</html>
```

The list of Functions, Methods and Objects supported are given at the end for reference.

AJAX

AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-site scripts. It can send as well as receive information in a variety of formats, including XML, HTML and text files. AJAX's most appealing characteristic is that it can do all of this without having to refresh the page. This makes it possible to update portions of a page based upon user events, dynamically. Thus the application becomes faster and more responsive to user actions.

The two enabling features are:

- It is possible to make requests to the server without reloading the page
- It is possible to receive and work with data from the server without reloading.

Functions, Methods and Objects in JavaScript

JavaScript Function	Description	JavaScript object
abs Function	Returns the absolute value of a number.	Math
acos Function	Returns the arccosine of a number.	Math
asin Function	Returns the arcsine of a number.	Math
atan Function	Returns the arctangent of a number.	Math
atan2 Function	Returns the angle (in radians) from the X axis to a point (y,x).	Math
ceil Function	Returns the smallest integer greater than or equal to its numeric argument.	Math
cos Function	Returns the cosine of a number.	Math
create Function	Creates an object that has a specified prototype, and that optionally contains specified properties.	Object
decodeURI Function	Returns the unencoded version of an encoded Uniform Resource Identifier.	Global
decodeURIComponent Function	Returns the unencoded version of an encoded component of a Uniform Resource Identifier.	Global
defineProperties Function	Adds one or more properties to an object, and/or modifies attributes of existing properties.	Object
defineProperty Function	Adds a property to an object, or modifies attributes of an existing property.	Object
encodeURI Function	Encodes a text string as a valid Uniform Resource Identifier.	Global
encodeURIComponent Function	Encodes a text string as a valid component of a Uniform Resource Identifier.	Global
escape Function	Encodes String objects so they can be	Global

eval Function	read on all computers.	Global
exp Function	Evaluates JavaScript code and executes it.	Math
floor Function	Returns e (the base of natural logarithms) raised to a power.	Math
freeze Function	Returns the greatest integer less than or equal to its numeric argument.	Object
fromCharCode Function	Prevents the modification of existing property attributes and values, and prevents the addition of new properties.	String
GetObject Function	Returns a string from a number of Unicode character values.	Global
getOwnProperty Descriptor Function	Returns a reference to an Automation object from a file.	Object
getOwnProperty Names Function	Returns the definition of a data property or an accessor property.	Object
getPrototypeOf Function	Returns the names of the properties and methods of an object.	Object
isArray Function	Returns the prototype of an object.	Array
isExtensible Function	Returns a Boolean value that indicates whether an object is an array.	Object
isFinite Function	Returns a value that indicates whether new properties can be added to an object.	Global
isFrozen Function	Returns a Boolean value that indicates if a supplied number is finite.	Object
isNaN Function	Returns true if existing property attributes and values cannot be modified in an object and new properties cannot be added to the object.	Global
isSealed Function	Returns a Boolean value that indicates whether a value is the reserved value NaN (not a number).	Object
keys Function	Returns true if existing property attributes cannot be modified in an object and new properties cannot be added to the object.	Object
log Function	Returns the names of the enumerable properties and methods of an object. Returns the natural logarithm of a number.	Math
max Function	Returns the greater of two supplied numeric expressions.	Math
min Function	Returns the lesser of two supplied numbers.	Math
now Function	Returns the number of milliseconds between January 1, 1970, and the current date and time.	Date

parse Function (Date)	Parses a string containing a date, and returns the number of milliseconds between that date and midnight, January 1, 1970.	Date
parse Function (JSON)	De-serializes JSON text to produce an in-memory object or array.	JSON
parseFloat Function	Returns a floating-point number converted from a string.	Global
parseInt Function	Returns an integer converted from a string.	Global
pow Function	Returns the value of a base expression raised to a specified power.	Math
preventExtensions Function	Prevents the addition of new properties to an object.	Object
random Function	Returns a pseudorandom number between 0 and 1.	Math
round Function	Returns a specified numeric expression rounded to the nearest integer.	Math
ScriptEngine Function	Returns a string representing the scripting language in use.	Global
ScriptEngineBuild Version Function	Returns the build version number of the scripting engine in use.	Global
ScriptEngineMajor Version Function	Returns the major version number of the scripting engine in use.	Global
ScriptEngineMinor Version Function	Returns the minor version number of the scripting engine in use.	Global
seal Function	Prevents the modification of attributes of existing properties, and prevents the addition of new properties.	Object
sin Function	Returns the sine of a number.	Math
sqrt Function	Returns the square root of a number.	Math
stringify Function	Serializes an object or array into JavaScript Object Notation (JSON) text.	JSON
tan Function	Returns the tangent of a number.	Math
unescape Function	Decodes String objects encoded with the escape method.	Global
UTC Function	Returns the number of milliseconds between midnight, January 1, 1970 Universal Coordinated Time (UTC) (or GMT) and the supplied date.	Date
write Function	Sends strings to the script debugger.	Debug
writeln Function	Sends strings to the script debugger, followed with a newline character.	Debug

JavaScript Method	Description	JavaScript object
anchor method	Places an HTML anchor that has a NAME attribute around text.	String
apply method	Applies a method of an object, substituting another object for the current object.	Function
atEnd method	Returns a Boolean value indicating if the enumerator is at the end of the collection.	Enumerator
big method	Places HTML <BIG> tags around text.	String
bind method	Creates a function that is associated with a specified object, and that can have specific initial parameters.	Function
blink method	Places HTML <BLINK> tags around text.	String
bold method	Places HTML <B> tags around text.	String
call method	Calls a method of an object, substituting another object for the current object.	Function
charAt method	Returns the character at the specified index.	String
charCodeAt method	Returns the Unicode encoding of the specified character.	String
compile method	Compiles a regular expression into an internal format.	Regular Expression
concat method (Array)	Returns a new array consisting of a combination of two arrays.	Array
concat method (String)	Returns a String object containing the concatenation of two supplied strings.	String
dimensions method	Returns the number of dimensions in a VBArray.	VBArray
every method	Checks whether a defined callback function returns true for all elements in an array.	Array
exec method	Executes a search for a match in a specified string.	Regular Expression
filter method	Calls a defined callback function on each element of an array, and returns an array of values for which the callback function returns true.	Array
fixed method	Places HTML <TT> tags around text.	String
fontcolor method	Places HTML <FONT> tags with a COLOR attribute around text.	String
fontsize method	Places HTML <FONT> tags with a SIZE attribute around text.	String
forEach method	Calls a defined callback function for	Array

JavaScript Method	Description	JavaScript object
anchor method	Places an HTML anchor that has a NAME attribute around text.	String
apply method	Applies a method of an object, substituting another object for the current object.	Function
atEnd method	Returns a Boolean value indicating if the enumerator is at the end of the collection.	Enumerator
big method	Places HTML <BIG> tags around text.	String
bind method	Creates a function that is associated with a specified object, and that can have specific initial parameters.	Function
blink method	Places HTML <BLINK> tags around text.	String
bold method	Places HTML <B> tags around text.	String
call method	Calls a method of an object, substituting another object for the current object.	Function
charAt method	Returns the character at the specified index.	String
charCodeAt method	Returns the Unicode encoding of the specified character.	String
compile method	Compiles a regular expression into an internal format.	Regular Expression
concat method (Array)	Returns a new array consisting of a combination of two arrays.	Array
concat method (String)	Returns a String object containing the concatenation of two supplied strings.	String
dimensions method	Returns the number of dimensions in a VBArray.	VBArray
every method	Checks whether a defined callback function returns true for all elements in an array.	Array
exec method	Executes a search for a match in a specified string.	Regular Expression
filter method	Calls a defined callback function on each element of an array, and returns an array of values for which the callback function returns true.	Array
fixed method	Places HTML <TT> tags around text.	String
fontcolor method	Places HTML <FONT> tags with a COLOR attribute around text.	String
fontsize method	Places HTML <FONT> tags with a SIZE attribute around text.	String

forEach method	Calls a defined callback function for each element in an array.	Array
getDate method	Returns the day-of-the-month value using local time.	Date
getDay method	Returns the day-of-the-week value using local time.	Date
getFullYear method	Returns the year value using local time.	Date
getHours method	Returns the hours value using local time.	Date
getItem method	Returns the item at the specified location.	VBArray
getMilliseconds method	Returns the milliseconds value using local time.	Date
getMinutes method	Returns the minutes value using local time.	Date
getMonth method	Returns the month value using local time.	Date
getSeconds method	Returns seconds value using local time.	Date
getTime method	Returns the time value in a Date Object as the number of milliseconds since midnight January 1, 1970.	Date
getTimezoneOffset method	Returns the difference in minutes between the time on the host computer and Universal Coordinated Time (UTC).	Date
getUTCDate method	Returns the day-of-the-month value using UTC.	Date
getUTCDay method	Returns the day-of-the-week value using UTC.	Date
getUTCFullYear method	Returns the year value using UTC.	Date
getUTCHours method	Returns the hours value using UTC.	Date
getUTCMilliseconds method	Returns the milliseconds value using UTC.	Date
getUTCMinutes method	Returns the minutes value using UTC.	Date
getUTCMonth method	Returns the month value using UTC.	Date
getUTCSeconds method	Returns the seconds value using UTC.	Date
getVarDate method	Returns the VT_DATE value in a Date object.	Date
getYear method	Returns the year value .	Date
hasOwnProperty method	Returns a Boolean value that indicates whether an object has a property with the specified name.	Multiple
indexOf method (Array)	Returns the index of the first occurrence of a value in an array.	Array

indexOf method (String)	Returns the character position where the first occurrence of a substring occurs within a String object.	String
isPrototypeOf method	Returns a Boolean value that indicates whether an object exists in another object's prototype chain.	Multiple
italics method	Places HTML <I> tags around text.	String
item method	Returns the current item in the collection.	
join method	Returns a String object consisting of all the elements of an array concatenated together.	Enumerator
lastIndexOf method (Array)	Returns the index of the last occurrence of a specified value in an array.	Array
lastIndexOf method (String)	Returns the last occurrence of a substring within a String object.	Array
lbound method	Returns the lowest index value used in the specified dimension of a VBArray.	String
link method	Places an HTML anchor that has an HREF attribute around text.	VBArray
localeCompare method	Returns a value indicating whether two strings are equivalent in the current locale.	String
map method	Calls a defined callback function on each element of an array, and returns an array that contains the results.	String
match method	Returns, as an array, the results of a search on a string using a supplied Regular Expression object.	Array
moveFirst method	Resets the current item in the collection to the first item.	String
moveNext method	Moves the current item to the next item in the collection.	Enumerator
pop method	Removes the last element from an array and returns it.	Enumerator
propertyIsEnumerable method	Returns a Boolean value that indicates whether a specified property is part of an object and whether it is enumerable.	Array
push method	Appends new elements to an array, and returns the new length of the array.	Multiple
reduce method	Accumulates a single result by calling a defined callback function for all elements in an array. The return value of the callback function is the accumulated result, and is provided as an argument in the next call to the callback function.	Array

reduceRight method	Accumulates a single result by calling a defined callback function for all elements in an array, in descending order. The return value of the callback function is the accumulated result, and is provided as an argument in the next call to the callback function.	Array
replace method	Returns a copy of a string with text replaced using a regular expression.	String
reverse method	Returns an Array object with the elements reversed.	Array
search method	Returns the position of the first substring match in a regular expression search.	String
setDate method	Sets the numeric day of the month using local time.	Date
setFullYear method	Sets the year value using local time.	Date
setHours method	Sets the hours value using local time.	Date
setMilliseconds method	Sets the milliseconds value using local time.	Date
setMinutes method	Sets the minutes value using local time.	Date
setMonth method	Sets the month value using local time.	Date
setSeconds method	Sets the seconds value using local time.	Date
setTime method	Sets the date and time value in the Date object.	Date
setUTCDate method	Sets the numeric day of the month using UTC.	Date
setUTCFullYear method	Sets the year value using UTC.	Date
setUTCHours method	Sets the hours value using UTC.	Date
setUTCMilliseconds method	Sets the milliseconds value using UTC.	Date
setUTCMinutes method	Sets the minutes value using UTC.	Date
setUTCMonth method	Sets the month value using UTC.	Date
setUTCSeconds method	Sets the seconds value using UTC.	Date
setYear method	Sets the year value using local time.	Date
shift method	Removes the first element from an array and returns it.	Array
slice method (Array)	Returns a section of an array.	Array
slice method (String)	Returns a section of a string.	String
small method	Places HTML <SMALL> tags around text.	String

some method	Checks whether a defined callback function returns true for any element of an array.	Array
sort method	Returns an Array object with the elements sorted.	Array
splice method	Removes elements from an array and, if necessary, inserts new elements in their place, returning the deleted elements.	Array
split method	Returns the array of strings that results when a string is separated into substrings.	String
strike method	Places HTML <STRIKE> tags around text.	String
sub method	Places HTML <SUB> tags around text.	String
substr method	Returns a substring beginning at a specified location and having a specified length.	String
substring method	Returns the substring at a specified location within a String object.	String
sup method	Places HTML <SUP> tags around text.	String
test method	Returns a Boolean value that indicates whether or not a pattern exists in a searched string.	Regular Expression
toArray method	Returns a standard JavaScript array converted from a VBArray.	VBArray
toDateString method	Returns a date as a string value.	Date
toExponential method	Returns a string containing a number represented in exponential notation.	Number
toFixed method	Returns a string representing a number in fixed-point notation.	Number
toGMTString method	Returns a date converted to a string using Greenwich Mean Time (GMT).	Date
toISOString method	Returns a date as a string value in ISO format.	Date
toJSON method	Used to transform data of an object type before the JSON serialization.	Date
toLocaleDateString method	Returns a date as a string value appropriate to the host environment's current locale.	Date
toLocaleLowerCase method	Returns a string where all alphabetic characters have been converted to lowercase, taking into account the host environment's current locale.	String
toLocaleString method	Returns an object converted to a string using the current locale.	Multiple

toLocaleTimeString method	Returns a time as a string value appropriate to the host environment's current locale.	Date
toLocaleUpperCase method	Returns a string where all alphabetic characters have been converted to uppercase, taking into account the host environment's current locale.	String
toLowerCase method	Returns a string where all alphabetic characters have been converted to lowercase.	String
toPrecision method	Returns a string containing a number represented either in exponential or fixed-point notation with a specified number of digits.	Number
toString method	Returns a string representation of an object.	Multiple
toTimeString method	Returns a time as a string value.	Date
toUpperCase method	Returns a string where all alphabetic characters have been converted to uppercase.	String
toUTCString method	Returns a date converted to a string using UTC.	Date
trim method	Returns a string with leading and trailing white space and line terminator characters removed.	String
ubound method	Returns the highest index value used in the specified dimension of the VBArray.	VBArray
unshift method	Inserts new elements at the start of an array.	Array
valueOf method	Returns the primitive value of the specified object.	Multiple

JavaScript Object Description	**Language Element**
Enables and returns a reference to an Automation object.	ActiveXObject Object
Provides support for creation of arrays of any data type.	Array Object
An object representing the arguments to the currently executing function, and the functions that called it.	arguments Object
Creates a new Boolean value.	Boolean Object
Enables basic storage and retrieval of dates and times.	Date Object
An intrinsic object that can send output to a script debugger.	Debug Object
Enables enumeration of items in a collection.	Enumerator Object
An object that contains information about errors that occur while JavaScript code is running.	Error Object

Check Your Progress 3

Creates a new function.	Function Object
An intrinsic object whose purpose is to collect global methods into one object.	Global Object
An intrinsic object that provides two methods to convert to and from the JavaScript Object Notation (JSON) format.	JSON Object
An intrinsic object that provides basic mathematics functionality and constants.	Math Object
An object representation of the number data type and placeholder for numeric constants.	Number Object
Provides functionality common to all JavaScript objects.	Object Object
Stores information on regular expression pattern searches.	RegExp Object
Contains a regular expression pattern.	Regular Expression Object
Allows manipulation and formatting of text strings and determination and location of substrings within strings.	String Object
Provides access to Visual Basic safe arrays.	VBArray Object

Check Your Progress 3

Notes: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of this Unit.

1) Write code in JavaScript to randomly change the background colour.

.....

.....

.....

.....

2.5 LET US SUM UP

Though various web browsers exist, there are some common architectural features among them. Basically the browser understands the HTML syntax and converts into a tree, which is then added with styling features and display co-ordinates, to finally display on the window.

There are several client side scripting languages but they have several similarities. They are embedded in an HTML document and have several programmatic features such as control structures, conditional statements etc.

2.6 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) (a) the User Interface, (b) the Browser Engine, (c) the Rendering Engine, (d) the Networking subsystem, (e) the Scripting Interpreter, (f) the XML Parser, (g) the Display Backend and (h) the Data Persistence subsystem.

2) DOM tree – Render tree – Layout of Render tree – Painting of Render tree

3) The styling information (such as colour and dimensions) is added to the DOM tree to create the Render tree.

Check Your Progress 2

1) <!ELEMENT script - - CDATA>

<!ATTLIST script

type CDATA #IMPLIED -- media type for script language --

language CDATA #IMPLIED -- predefined script language name --

src %URL #IMPLIED -- URL for an external script -

defer (defer) #IMPLIED --- user may defer execution of script --

>

2) Refer to Sub-Section 2.3.2

Check Your Progress 3

1) <BODY BGCOLOR='&{randomrbg()};'>

2.7 SUGGESTED READINGS

- Aho, Sethi, Ullman, Compilers (1986), *Principles, Techniques and Tools (aka the "Dragon book")*, Addison-Wesley.
- Chris Bates (2002), *Web Programming Building Internet Applications*, 2 Ed, Wiley-India.
- Chris Waterson, *Gecko Overview*, http://www.mozilla.org/newlayout/doc/gecko-overview.htm.
- Grosskurth, Alan, *A Reference Architecture for Web Browsers*, http://grosskurth.ca/papers/browser-refarch.pdf.
- http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/index.html
- http://webkit.org/.
- http://www.microsoft.com/vbscript/default.htm.
- http://www.w3.org/pub/WWW/TR/WD-script-970314.
- http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/idl-definitions.html.

UNIT 3 SERVER SIDE SCRIPTS

Structure

3.0 INTRODUCTION

We have seen in Unit 1 of this block that the basic web architecture comprises of a client and a server. In Unit 2 we have studied the client side. In this Unit we will understand the server side. As in Unit 2, we will understand the architecture as well as the scripting aspect at the server side.

3.1 OBJECTIVES

After studying this unit, you should be able to:

- understand the basic functioning of a web server;
- understand to configure a web server;
- understand the internal working of a web server;
- understand JavaServer Pages;
- understand ASP; and
- understand ASP.NET.

3.2 HOW A WEB SERVER WORKS?

A web server responds to a request from a client by providing the requested document. Various types of web servers have emerged. This doesn't make a difference at the client-end because the requested documents are provided through over standard protocols such as HTTP, FTP etc. in a standard format such as HTTP, XML etc. Some important web servers are Apache (HTTP Server and Tomcat), Internet Information Services (IIS), Jetty, IBM HTTP Server, Oracle (HTTP Server, iPlanet and Weblogic) etc.

3.2.1 Internal Working of a Web Server

The basic architecture of the web server can see seen as comprising of seven sub-systems:

a) The Reception sub-system

b) The Request Analyzer sub-system

c) The Access Control sub-system

d) The Resource Handler sub-system

e) The Transaction Log sub-system

f) The Utility sub-system and

g) The Operating System Abstraction Layer.

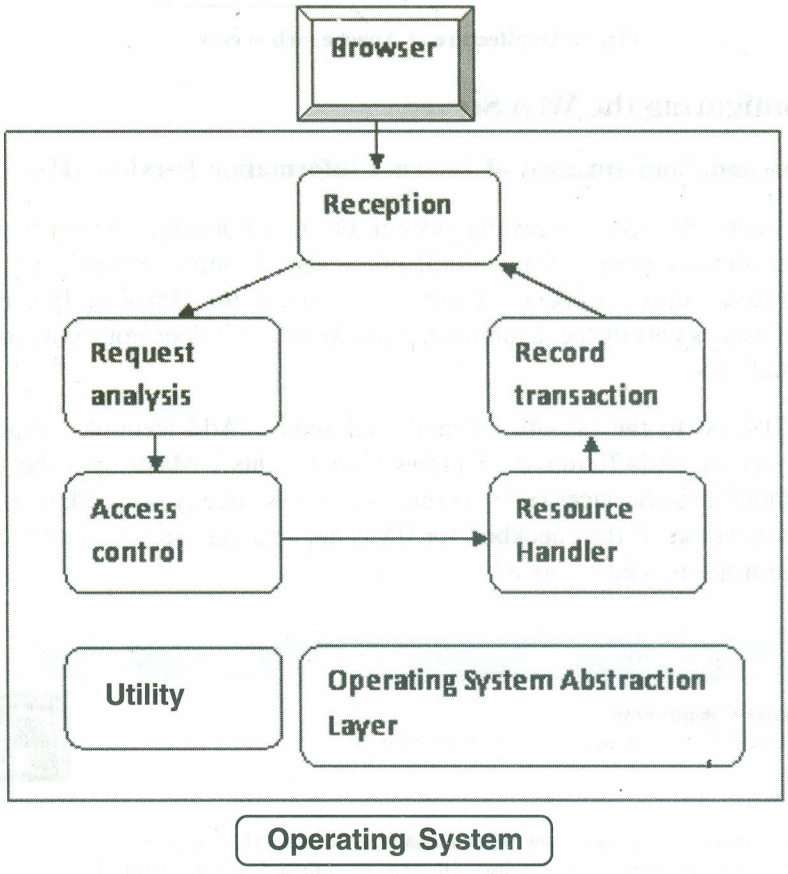


Fig. 1: Generic architecture of a web server

The **Reception** sub-system receives the HTTP request, understands the browser requirements, handles multiple requests and makes an initial interpretation of the request. The **Request Analyzer** sub-system builds an internal representation of the request and translates the address to the specific local address in the server machine. The **Access Control** sub-system checks the authentication (user ID, password) and authorization (grants, privileges) for the user with respect to the requested resource. The **Resource Handler** sub-system handles the request with respect to the type of resource requested, i.e. whether the resource is a file or a program to be executed for the response. The **Transaction Log** sub-system writes the log of all transactions, including the requests and the responses. The **Utility** sub-system contains functions required by the various sub-systems and the **Operating System Abstraction Layer** is useful in case of porting the server to different operating systems/platforms.

Each web server will have its own architecture based on the above generic architecture but with its own variants. Let us take Apache, the most popular web server. Its Reception sub-system is called "Core". Its Request Analyzer sub-subsystem is called "Translation". "MIME Type" and "Response" constitute the Resource Handler sub-system. The architecture is shown diagrammatically below:

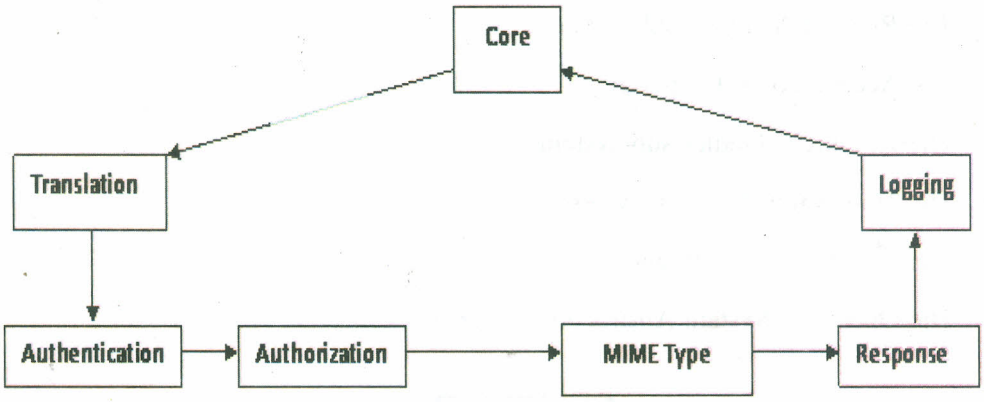


Fig. 2: Architecture of Apache web server

3.2.2 Configuring the Web Server

Installation and Configuration of Internet Information Services (IIS)

IIS is part of the Windows operating system, but it is not selected by default. To find if you already have IIS installed, go to the "Control Panel" and select "Administrative Tools". Check if there is a shortcut for "Internet Information Services" exists as part of the "Administrative Tools". If it does not exist, you will need to install IIS.

To install IIS, go to the "Control Panel" and select "Add Remove Programs". Select the option "Add/Remove Windows Components". Make sure the option "Internet Information Services (IIS)" is selected. If it is already selected, that means you already have IIS. If the checkbox for IIS is not checked, select it and press the "Next" button to proceed to install.

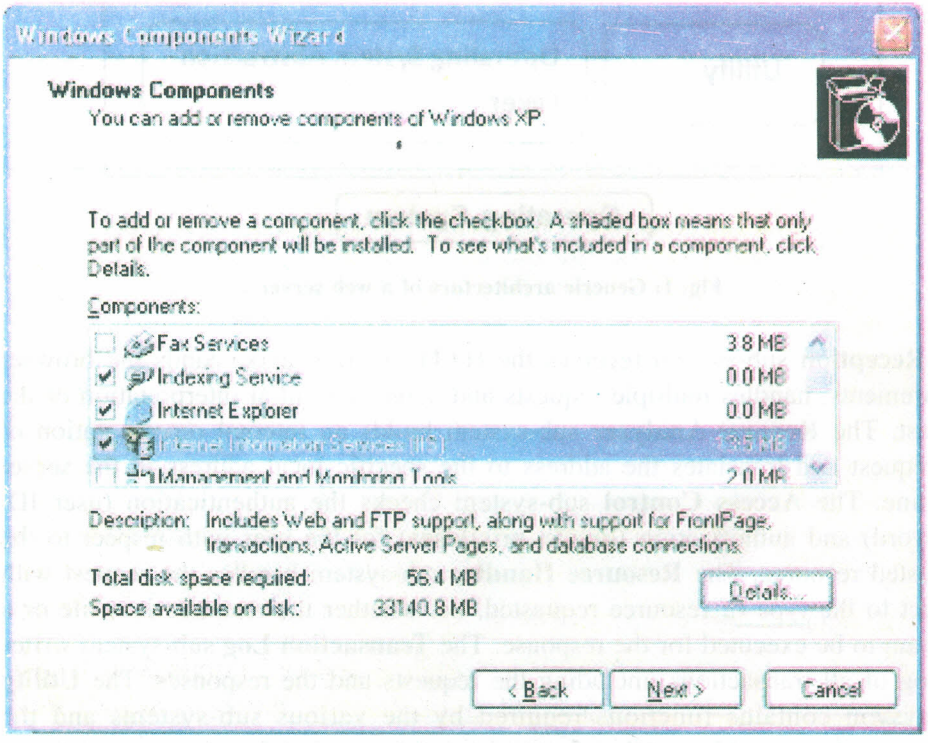


Fig. 3

The Windows Components Installation Wizard would prompt to insert the CD for Windows.

Installing IIS 7.0 on Windows Server 2008

1) Click Start, point to **Administrative Tools** and then click **Server Manager**.

2) In the left pane, select **Roles** and then in the right pane, click **Add Roles**. The Add Roles Wizard is displayed.

3) On the **Before You Begin** page, click **Next**.

4) On the Select **Server Roles page**, from the list of roles displayed, select **Web Server (IIS)** and then click **Next**.

5) On the **Web Server (IIS) page**, click **Next**.

6) On the Select **Role Services page**, a list of IIS 7.0 features is displayed. Ensure that the **IIS 6 Metabase Compatibility** feature is selected. Select any other features if required and then click **Next**.

7) On the **Confirm Installation Selections page**, click **Install**. The Installation Progress page displays the status of actions performed by the Add Roles Wizard.

8) Review the information displayed on the **Installation Results** page and then click **Close**.

Configuring IIS 7.0 and ASP.NET 2.0

1) Click **Start,** point to **Administrative Tools** and then click **Internet Information Services (IIS) Manager**.

2) In the Internet Information Services (IIS) Manager console, from the left pane, select the local computer.

   IIS-related features installed on the computer are displayed in the right pane.

3) In the list of IIS features, double-click **ISAPI** and **CGI Restrictions** and verify if ASP.NET 2.0 is installed and in an allowed state. If ASP.NET 2.0 is installed and is in an allowed state, IIS and ASP.NET have already been configured and you may skip the remaining steps in this procedure.

4) In a Command Prompt window, browse to the ASP.NET 2.0 Framework folder. This directory is generally %WinDir%\Microsoft.NET\Framework\v2.0.50727.

5) To register ASP.NET 2.0 Framework with IIS, run the command:

   aspnet_regiis -i -enable

6) In **Administrative Tools**, click **Services**, select **IIS Admin Service** and then restart the service.

Registering .NET Framework Script Mappings with IIS 7.0

1) Click Start, point to **Administrative Tools** and then click **Internet Information Services (IIS) Manager**.

2) In the Internet Information Services (IIS) Manager console, from the left pane, select the local computer.

   IIS-related features installed on the computer are displayed in the right pane.

3) In the list of IIS features, double-click **Handler Mappings** and verify if mappings for *.svc are installed and enabled. If the mappings are installed and enabled, IIS and ASP.NET have already been configured and you may skip the remaining steps in this procedure.

4) In a Command Prompt window, browse to the directory. %WinDir %\Microsoft. NET\Framework\v3.0\Windows Communication Foundation.

5) To register ASP.NET 3.0 mappings with IIS 7.0, run the command:

ServiceModelReg.exe -r

6) In the **Handler Mappings** pane, verify if the mappings have been added and then **close** the **Internet Information Services (IIS) Manager console**.

To create a new Web site

1) In IIS Manager, expand the local computer, right-click the **Web Sites folder**, point to **New** and then click **Web Site**. The Web Site Creation Wizard appears.

2) Click **Next**.

3) In the **Description box**, type the name of your site (for example, XYZWebsite) and then click **Next**.

4) Type or click the IP address (the default is All Unassigned), TCP port and host header (for example, www.mysite.com) for your site and then click **Next**. Specify an unused TCP port and add the port to the list of exceptions in Windows Firewall.

5) In the **Path** box, type or browse to the directory %systemdrive%\inetpub, create a new directory (for example, XYZ) and set the path to the new directory (for example, %systemdrive%\inetpub\XYZ). Select **Allow anonymous access to this website** and then click Next.

6) Select the check boxes for the Web site access permissions. Select **Read, Run scripts, Execute and Browse** and then click **Next**.

7) Click **Finish**.

A new Web site has been created.

8) Click the **Web Sites node** and then note the Identifier of the new Web site. The Identifier is a unique number assigned to each Web site and is displayed in the right pane of IIS Manager.

Installing and Configuring Tomcat

- **Install the Java JDK**. Download from http://java.sun.com/javase/downloads. If you plan to execute Tomcat manually, you also need to set your PATH environment variable.
- **Install Tomcat**

  1) **Download the software**. Go to *http://tomcat.apache.org/download-60.cgi* and download and unpack the zip file for the current release build of Tomcat.

  2) **Set the JAVA_HOME variable**. Set it to refer to the base JDK directory, not the bin subdirectory.

  3) **Change the port to 80**. Edit *install_dir/conf/**server**.xml* and change the port attribute of the Connector element from 8080 to 80.

  4) **Turn on servlet reloading**. Edit *install_dir/conf/**context**.xml* and change <Context> to <Context reloadable="true" privileged="true">.

  5) **Enable the invoker servlet**. Go to *install_dir/conf/**web**.xml* and uncomment the servlet and servlet-mapping elements that map the invoker servlet to/servlet*.

6) **Turn on directory listings**. Go to *install_dir/conf/**web**.xml*, find the init-param entry for listings and change the value from false to true.

- **Test the server**

  1) Verify that you can start the server. Double-click *install_dir/bin/startup.bat* and *try accessing http://localhost/*.

  2) Check that you can access your own HTML & JSP pages. Drop some simple HTML and JSP pages into *install_dir/webapps/ROOT* and access them with *http://localhost/filename*.

- **Set up your development environment**

  1) Create a development directory. Put it anywhere except within the Tomcat installation hierarchy.

  2) Make shortcuts to the Tomcat startup & shutdown Scripts. Put shortcuts to *install_dir/bin/startup.bat* and *install_dir/bin/shutdown.bat* in your development directory and/or on your desktop.

  3) Set your class path. Use the CLASSPATH environment variable. Include the current directory ("."), the servlet/JSP JAR files (*install_dir/lib/servlet-api.jar and install_dir/lib/jsp-api.jar* and *install_dir/lib/el-api.jar*) and your main development directory from Step 1.

  4) Bookmark the servlet & JSP javadocs. Add the servlet 2.5 API and the JSP 2.1 API to your bookmarks/favorites list.

If you are using Microsoft Windows, you can download a Tomcat installer .exe.

- **It will prompt for the desired port**. It will ask you what port it should run on and make the changes in *server.xml*. But you will still need to manually edit *context.xml* and *web.xml*.
- **It will set JAVA_HOME**.
- **It will set up Start Menu entries**. Instead of using startup.bat and shutdown.bat, you can go to the Start Menu, select Apache Tomcat 6.0, select Monitor Tomcat and select Start or Stop, as shown below.

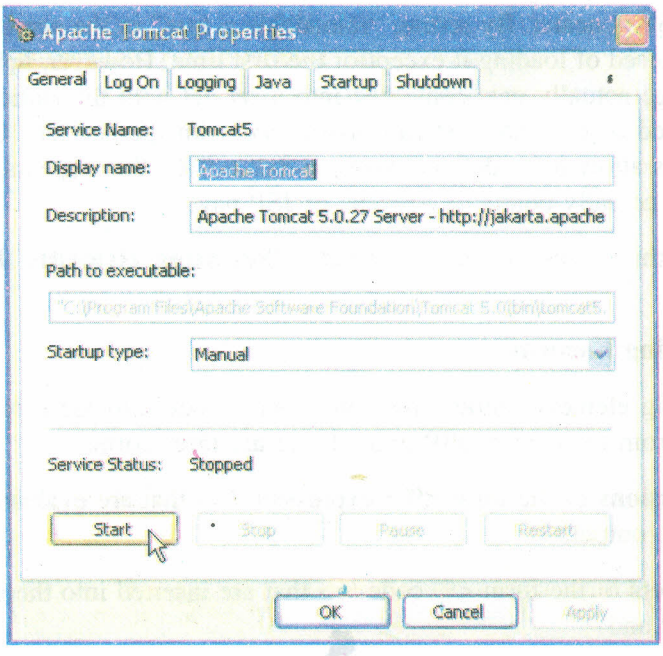


Fig. 4

Check Your Progress 1

Note: a) Space is given below for writing your answers.

b) Compare your answer with the one given at the end of this Unit.

1) What are the generic sub-systems of a web server?

.....

.....

.....

.....

2) What is the main server configuration file in Tomcat?

.....

.....

.....

.....

3.3 JAVASERVER PAGES

JavaServer Pages (JSP) was designed to work as a server side scripting language improving upon the earlier design of Servlets and CGI programs. JSP has since been a very popular server side scripting language for its ease and functionality.

3.3.1 Overview

JSP can separate the dynamic part of web page from the static HTML. One can write the HTML in the usual manner and then enclose the code for the dynamic parts in special tags, most of which start with "<%" and end with "%>". The normal file extension is .jsp and installed normally where the HTML Web page is placed.

JSP is different from a servlet in that the servlet writes long code in separate document for throwing the HTML whereas the JSP is embedded within the HTML code. So JSP is neater. By setting "reloadable = true" in the server configuration, there is no need of loading it exceptfor the first time. However, behind the scenes, the JSP page actually gets converted into a servlet with the static HTML simply being printed to the output stream associated with the servlet's service method. Web servers allow us to define aliases so that a URL that appears to reference an HTML file actually points to a servlet or JSP page.

There are three types of JSP constructs: **Scripting elements, Directives and Actions**.

JSP Scripting Elements

JSP scripting elements allow insertion of Java code into the servlet that will be generated from the current JSP page. There are three forms:

1) **Expressions** of the form <%= expression %> that are evaluated and inserted into the output,

2) **Scriptlets** of the form <% code %> that are inserted into the servlet's service method and

3) **Declarations** of the form <%! code %> that are inserted into the body of the servlet class, outside of any existing methods.

JSP Directives

A JSP directive affects the overall structure of the servlet class. It usually has the following form:

<%@ directive attribute="value" %>

However, one can also combine multiple attribute settings for a single directive, as follows:

<%@ directive attribute1='value1"

attribute2="value2"

...

attributeN="valueN" %>

There are two main types of directives:

1) **Page**, to import classes, customize the servlet superclass etc and
2) **Include**, to insert a file into the servlet class at the time the JSP file is translated into a servlet.

Actions

JSP actions work to control the behavior of the servlet engine. It makes it possible to dynamically insert a file, reuse JavaBeans components, forward to another page or generate HTML for Java plugin. Types of actions are:

- **jsp:include** – Include a file.
- **jsp:useBean** – Find or instantiate a JavaBean.
- **jsp:setProperty** – Set the property of a JavaBean.
- **jsp:getProperty** – Insert the property of a JavaBean into the output.
- **jsp:forward** – Forward to a new page.
- **jsp:plugin** – Generate browser-specific code to make an OBJECT or EMBED tag for the Java plugin.

JSP Element	Syntax	Interpretation	Notes
JSP Expression	<%= expression %>	Expression is evaluated and placed in output.	XML equivalent is <jsp:expression> expression </jsp:expression>. Predefined variables are request, response, out, session, application, config, and pageContext (available in scriptlets also).
JSP Scriptlet	<% code %>	Code is inserted in service method.	XML equivalent is <jsp:scriptlet> code </jsp:scriptlet>.
JSP Declaration	<%! code %>	Code is inserted in body of servlet class, outside of service method.	XML equivalent is <jsp:declaration> code </jsp:declaration>.

JSP page Directive	<%@ page att="val" %>	Directions to the servlet engine about general setup.	XML equivalent is <jsp:directive.page att="val"\>. Legal attributes, with default values in bold, are: • import="*package.class*" • content Type="*MIME-Type*" • isThread Safe="**true**|false" • session="**true**|false • buffer="*size*kb| none" • autoflush="**true**| false" • extends="*package.class*" • info="*message*" • errorPage="*url*" • isErrorPage="true| **false**" • language="*java*"
JSP include Directive	<%@ include file="url" %>	A file on the local system to be included when the JSP page is translated into a servlet.	XML equivalent is <jsp:directive.include file="url"\>. The URL must be a relative one. Use the jsp:include action to include a file at request time instead of translation time.
JSP Comment	<%-- comment --%>	Comment; ignored when JSP page is translated into servlet.	If you want a comment in the resultant HTML, use regular HTML comment syntax of <-- comment -->.
The jsp:include Action	<jsp:include page="*relative URL*" flush="true"/>	Includes a file at the time the page is requested.	If you want to include the file at the time the page is translated, use the page directive with the include attribute instead. **Warning:** on some servers, the included file must be an HTML file or JSP file, as determined by the server (usually based on the file extension).
The jsp:useBean Action	<jsp:useBean att=val*/> or <jsp:useBean att=val*> ... </jsp:useBean>	Find or build a Java Bean.	Possible attributes are: • id="*name*" • scope ="**page**| request|session |application"

The jsp:setProperty Action	<jsp:setProperty att=val*/>	Set bean properties, either explicitly or by designating that value comes from a request parameter.	• class="*package.class*" • type="*package.class*" • beanName ="*package.class*" Legal attributes are • name="*beanName*" • property="*propertyName*|*" • param="*parameterName*" • value="*val*"
The jsp:getProperty Action	<jsp:getProperty name ="*propertyName*" value="*val*"/>	Retrieve and output bean properties.	
The jsp:forward Action	<jsp:forward page="*relative URL*"/>	Forwards request to another page.	
The jsp:plugin Action	<jsp:plugin attribute ="*value*"*> ... </jsp:plugin>	Generates OBJECT or EMBED tags, as appropriate to the browser type, asking that an applet be run using the Java Plugin.	

3.3.2 Language

JSP Expressions

A JSP *expression* is used to insert Java values directly into the output. It has the following form:

<%= Java Expression %>

The Java expression is evaluated, converted to a string and inserted in the page at run-time. This gives the current time:-

Current time: <%= new java.util.Date() %>

The implicit objects (predefined variables) are:

- request, the HttpServletRequest;
- response, the HttpServletResponse;
- session, the HttpSession;
- out, the PrintWriter;
- application, the ServletContext;
- config, the ServletConfig;
- pageContext, the PageContext; and
- page.

For example:

Your hostname: <%= request.getRemoteHost() %>

To write in XML, which is case-sensitive:-

<jsp:expression>

Some Java Expression

</jsp:expression>

JSP Scriptlets

Scriptlets have the following form:

<% Java Code %>

Using pre-defined variables as expressions:

<%

String queryData = request.getQueryString();

out.println("Attached GET data: " + queryData);

%>

JSP Declarations

They have the form:

<%! Java Code %>

Since declarations do not generate any output, they are normally used in conjunction with JSP expressions or scriptlets. For example, to print the number of times the current page has been requested since the server booted:

<%! private int bootCount = 0; %>

Accesses to page since server reboot:

<%= ++bootCount %>

JSP page Directive

The page directive lets you define following attributes (among others):-

- import="*package.class*" or import="*package.class1,...,package.classN*". For example:-

  <%@ page import="java.util.*" %>

- contentType="*MIME-Type*" or contentType="*MIME-Type*; charset=*Character-Set*"

  The default is text/html. For example:

  <%@ page contentType="text/plain" %>

- isThreadSafe="true|false".

  A value of true (the default) indicates normal servlet processing, wherein multiple requests can be processed simultaneously with a single servlet instance. A value of false indicates that requests are either delivered serially or simultaneous requests given separate servlet instances.

- session="truelfalse".

  A value of true (the default) indicates that the predefined variable session (of type HttpSession) would be used or a new session be created. A value of false indicates that no session variable will be used. •

- errorPage="*url*".

  This specifies a JSP page that would process any error Throwables thrown but not caught in the current page.

The JSP include Directive

The syntax is:

<%@ include file="relative url" %>

The jsp:include Action

This can insert files into the page being generated. The syntax is:

<jsp:include page="relative URL" flush="true" />

The jsp:useBean Action

This action is a powerful tool for reusability of Java classes. The syntax for specifying that a bean is:

<jsp:useBean id="*name*" class="*package.class*" />

One can modify its properties via jsp:setProperty or by using a scriptlet and calling a method explicitly on the object with the variable name specified earlier via the id attribute.

You can read the existing properties in a JSP expression or scriptlet by using the jsp:getProperty action.

The jsp:setProperty Action

The syntax is:

<jsp:useBean id="myName" ... />

...

<jsp:setProperty name="myName"

property="someProperty" ... />

The jsp:getProperty Action

The syntax is:

<jsp:useBean id="studentBean" ... />

...

<UL>

<LI>Regn Number:

<jsp:getProperty name="studentBean" property="regnNo" />

<LI>Permanent Address:

<jsp:getProperty name="studentBean" property="addressPerm" />

</UL>

The jsp:forward Action

This forwards the request to another page.

<jsp:forward page="/utils/errorReporter.jsp"/>

<jsp:forward page="<%= someExpression %>"/>

Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>JSP Example</TITLE>
<META NAME="author" CONTENT="mpdd@ignou.ac.in">
<META NAME="keywords"
CONTENT="JSP, JavaServer Pages, Example">
<META NAME="description"
CONTENT="An example of JSP">
<LINK REL=STYLESHEET
HREF="ExampleStyle.css"
TYPE="text/css">
</HEAD>
<BODY BGCOLOR="#FDF5E6" TEXT="#000000" LINK="#0000EE"
VLINK="#551A8B" ALINK="#FF0000">
<CENTER>
<TABLE BORDER=5 BGCOLOR="#EF8429">
<TR><TH CLASS="TITLE">
Using JSP</TABLE>
</CENTER>
<P>
Dynamic content:
<UL>
<LI><B>Expression.</B><BR>
Your hostname: <%= request.getRemoteHost() %>.
<LI><B>Scriptlet.</B><BR>
<% out.println("Attached GET data: " + request.getQueryString()); %>
<LI><B>Declaration (plus expression).</B><BR>
<%! private int bootCount = 0; %>
Accesses to page since server reboot: <%= ++bootCount %>
```

```
<LI><B>Directive (plus expression).</B><BR>
<%@ page import = "java.util.*" %>
Current date: <%= new Date() %>
</UL>
</BODY>
</HTML>
```

Check Your Progress 2

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What are the types of JSP constructs?

.....

.....

.....

.....

2) What are the forms of JSP Scripting elements?

.....

.....

.....

.....

3) What is the main purpose of Java bean?

.....

.....

.....

.....

3.4 ASP

ASP has been derived from Microsoft VBScript. It has extension .asp and is a server side script which has the familiar ease of VBScript .

3.4.1 Brief Overview

A quick way to create an .asp file is to simply rename the HTML with an .asp extension. The ASP functionality can be added later. To publish an .asp file on the Web, save the new file in a virtual directory on your Web site (directory having Script or Execute permission enabled). Then request the file with your browser by typing in the file's URL. The user will receive standard HTML from the server.

ASP uses the delimiters <% and %> to enclose script commands.

```
<HTML>
<BODY>
```

This page was last refreshed on <%= Now() %>.

</BODY>

</HTML>

The VBScript function **Now()** returns the current date and time.

By default, the primary scripting language is VBScript, but a different language can also be used. The HTML <SCRIPT> tag is used to enclose script commands and expressions. This tag can be used to define procedures in multiple languages within an .asp file.

The conditional **If...Then...Else** statement that appears below is a common VBScript statement:

```
<%
Dim dHour
dHour = Hour(Now())
If dHour < 12 Then
strGreeting = "Good Morning!"
Else
strGreeting = "Hello!"
End If
%>
<%= strGreeting %>
```

So this script greets "Good Morning!" or "Hello!" depending upon whether the current server time is before 12 noon or not.

Another way of writing the above is:

```
<%
Dim dHour
dHour = Hour(Now())
If dHour < 12 Then
%>
Good Morning!
<% Else %>
Hello!
<% End If %>
```

Note the wrapping of text and script.

Another way of writing the same (this time, using an implicit object) is using Response.Write:

```
<%
Dim dHour
dHour = Hour(Now())
```

```
If dHour < 12 Then

Response.Write "Good Morning!"

Else

Response.Write "Hello!"

End If

%>
```

Using ASP Directives

ASP provides directives that are not part of the scripting language:

- the output directive and
- the processing directive.

The ASP *output directive*<%= expression %> displays the value of an expression. This output directive is equivalent to using Response.Write.

The ASP *processing directive*<%@ keyword %> gives ASP the information it needs to process an .asp file. For example:

```
<%@ Language= "VBScript" %>
```

The processing directive must appear on the first line of an .asp file. To add more than one directive to a page, the directive must be within the same delimiter. You must use a space between the @ sign and the keyword. The important keywords for the processing directive are:

- The @LANGUAGE keyword sets the scripting language for the .asp file.
- The @ENABLESESSIONSTATE keyword specifies whether an .asp file uses session state variable.
- The @CODEPAGE keyword sets the code page (the character encoding) for the .asp file.
- The @LCID keyword sets the locale identifier for the file.

If the primary scripting language is either VBScript or JScript, ASP removes white space from commands. For all other scripting languages, ASP preserves white space. White space includes spaces, tabs, returns and line feeds.

There are around 100 pre-defined functions in VBScript and these can be used in ASP. The string functions are simple and useful.

Example

```
<%@ Language= "VBScript" %>

<html>

<head>

<title>ASP Example </title>

</head>

<body>

<font face="MS Gothic">

<H3>Changing Address</H3>
```

```
<%
'Create some variables.
dim strString
dim strSearchFor
dim strReplaceWith
'Set the variables, otherwise they could be fetched from the server database.
strString = "IGNOU<BR>SOCIS<BR>Maidan Garhi, New Delhi<BR>"
'This is set here but could be input by the client
strSearchFor = "SOCIS"
'This is set here but could be input by the client
strReplaceWith = "MPDD"
'Verify that the String to search exists... (otherwise this could fetch from the server database)
If Instr(strString, strSearchFor) Then
Response.Write "Found<BR>"
Else
Response.Write "Not found"
End If
Response.Write "<BR>Original Address:- <BR>" & strString & "<BR>"
Response.Write "<BR>Changed Address:- <BR>"
Response.Write Replace(strString, strSearchFor, strReplaceWith) & "<BR>"
%>
</font>
</body>
</html>
```

Output

```
Changing Address
Found
Original Address
IGNOU
SOCIS
Maidan Garhi, New Delhi
Changed Address:
IGNOU
MPDD
Maidan Garhi, New Delhi
```

Notes: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of this Unit.

1) What is the primary scripting language for ASP?

.....

.....

.....

.....

3.5 ASP.NET

ASP.NET has evolved from ASP. However, there is a sea change in ASP.NET in that it uses the .NET framework. Some features remain, as the ease of Visual Basic is maintained. A high level of integration is available in ASP.NET.

3.5.1 Overview

Configuring the web server for ASP.NET. However, with Microsoft WebMatrix user interface, one can get started with a website, use ASP.NET (Razor syntax), add a database and analyze the performance of the web application. This is a very easy way to write ASP.NET code without bothering to configure the web server. The web server IIS Express is integrated with WebMatrix.

The WebMatrix can be downloaded from Microsoft website for free. The proceeding instructions (given below) are contained in the Microsoft website. After installation, run it and choose **Site from Template – Empty site**, giving the name of the proposed site, say "WebMatrixDemo". Select the **Files** workspace and click on **New**. Choose **CSHTML** and name the new page as **Default.cshtml**. The extension .cshtml indicates that it is an ASP.NET page. Add "Hello World Page" to the Title and "Hello World" to the Body in the HTML markup in the Default.cshtml page and run the page to check the working. Now, it is possible to write any ASP.NET code on it and run it. Simple!

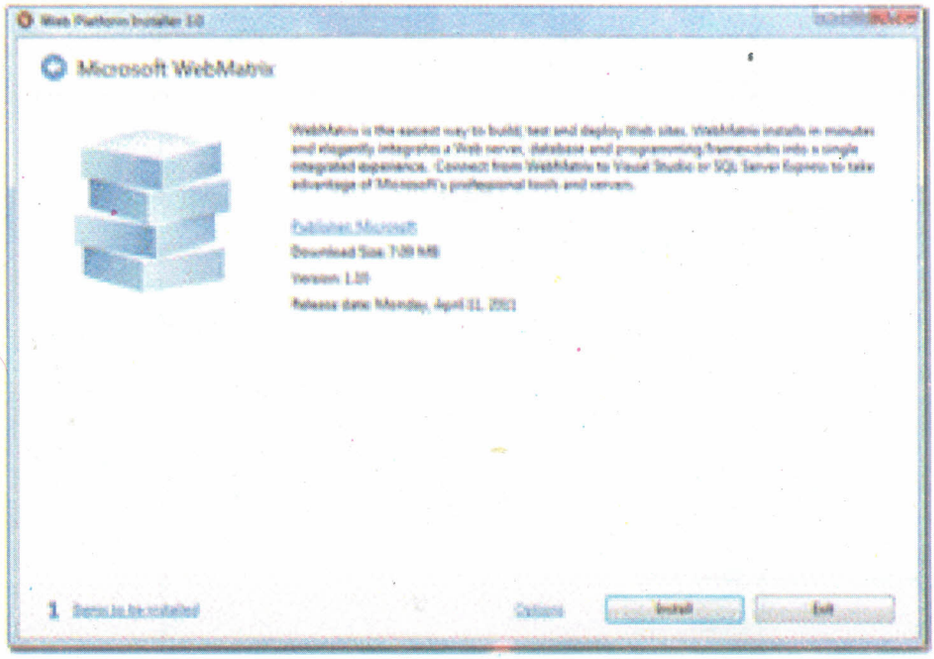


Fig. 5

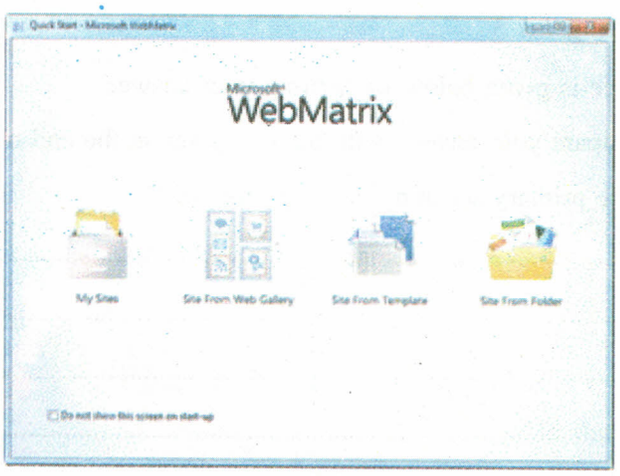


Fig. 6

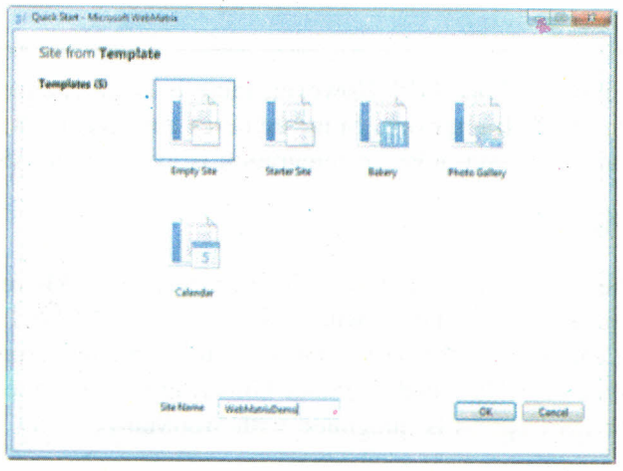


Fig. 7

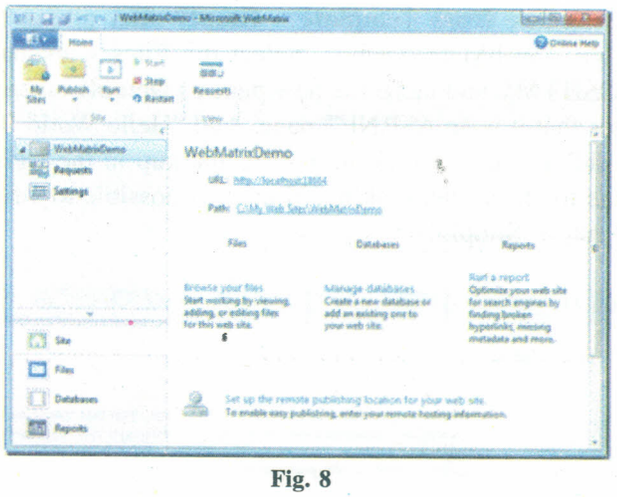


Fig. 8

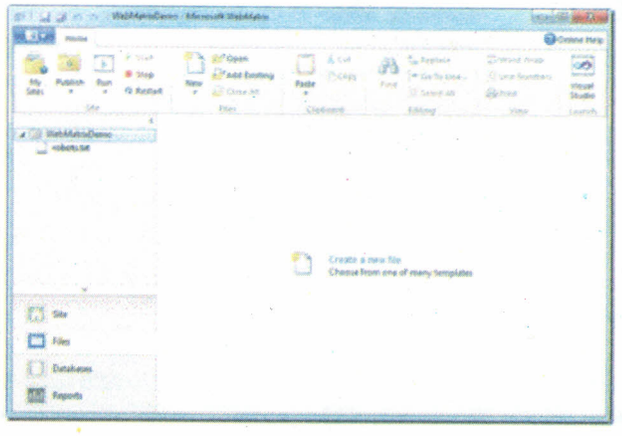


Fig. 9

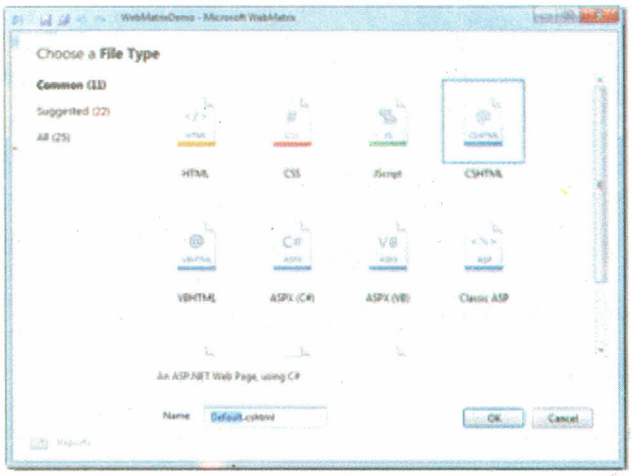


Fig. 10

Fig. 11

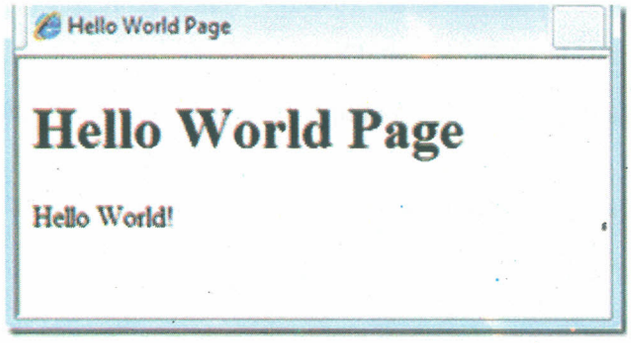


Fig. 12

3.5.2 ASP.NET Using the Razor Syntax

Razor syntax is based on Microsoft's ASP.NET, which in turn is based on the Microsoft .NET Framework. The.NET Framework is a comprehensive programming framework from Microsoft for developing virtually any type of computer application. ASP.NET is the part of the .NET Framework that's specifically designed for creating web applications and has the file-name extension .aspx.

The Razor syntax is based on the C# programming language and also supports the Visual Basic language.

Writing Razor code

The following example makes an overview of Razor syntax:

```
<!-- Single statement blocks  -->
@{ var total = 7; }
@{ var myMessage = "Hello World"; }
<!-- Inline expressions -->
<p>The value of your account is: @total </p>
<p>The value of myMessage is: @myMessage</p>
<!-- Multi-statement block -->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + "Today is:" + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```

Output

```
The value of your account is: 7
The value of myMessage is: Hello World
The greeting is: Welcome to our site! Today is: Monday
```

We can see how variables are used to store values as in:

```
var total = 7;
```

We see how literal string values are enclosed in double quote marks:

```
var greeting = "Welcome to our site!";
```

The difference among single and multiple statement blocks and inline expressions is also seen.

The following simple program accepts input of two whole numbers and displays the sum:

```
@{
var totalNum = 0;
var totalString = "";
if(IsPost) {
// Retrieve the numbers that the user entered.
var number1 = Request["text1"];
var number2 = Request["text2"];
// Convert the entered strings into integers numbers and add.
totalNum = number1.AsInt() + number2.AsInt();
totalString = "Total is" + totalNum;
    }
```

```
}
<!DOCTYPE html>
<html lang="en">
<head>
<title>Add Numbers</title>
<meta charset="utf-8" />
<style type="text/css">
body {background-color: beige; font-family: Verdana, Arial; margin: 50px;}
form {padding: 10px; border-style: solid; width: 250px;}
</style>
</head>
<body>
<p>Enter two whole numbers and then click the button <strong>Add</strong>.</
p>
<form action="" method="post">
<p><label for="text1">First Number:</label>
<input type="text" name="text1" />
</p>
<p><label for="text2">Second Number:</label>
<input type="text" name="text2" />
</p>
<p><input type="submit" value="Add" /></p>
</form>
<p>@totalString</p>
</body>
</html>
```

Here are some points to be noted:

- The @ character starts the first block of code and it precedes the totalString variable (the output) near the bottom of the page.
- The block at the top of the page is enclosed in braces and all statements end with a semicolon.
- The literal string value assigned to the totalString variable is in double quotation marks.
- The code is case-sensitive.
- The <form> tag includes a method="post" attribute. This specifies that when the user clicks on the **Add** button, the page will be sent to the server using the HTTP POST method. When the page is submitted, the if(IsPost) test returns true.

Save the page and run it in a browser. (Make sure the page is selected in the **Files** workspace before you run it.) Enter two whole numbers and then click the **Add** button.

Converting and Testing Data Types

Although ASP.NET can usually determine a data type automatically, sometimes it cannot do so. As a rule, user input comes to us as strings. *Even if we have prompted users to enter a number and even if a digit is entered, the data is in string format.* To convert the whole number values in string to integers, we called the AsInt method in the above example. The following table lists some common conversion and test methods for variables.

Method	Description	Example

AsInt(), IsInt()	Converts a string that represents a whole number (like "593") to an integer.	var myIntNumber = 0;
		var myStringNum = "539";
		if(myStringNum.IsInt()=true){
		myIntNumber =
		myStringNum.AsInt();
		}

AsBool(), IsBool()	Converts a string like "true" or "false" to a Boolean type.	var myStringBool = "True"; var myVar = myStringBool.AsBool();
AsFloat(), IsFloat()	Converts a string that has a decimal value like "1.3" or "7.439" to a floating-point number.	var myStringFloat = "41.432895"; var myFloatNum = myStringFloat.AsFloat();
AsDecimal(), IsDecimal()	Converts a string that has a decimal value like "1.3" or "7.439" to a decimal number. (In ASP.NET, a decimal number is more precise than a floating-point number.)	var myStringDec = "10317.425"; var myDecNum = myStringDec.AsDecimal();
AsDateTime(), IsDateTime()	Converts a string that represents a date and time value to the ASP.NET DateTime type.	var myDateString = "12/27/2010"; var newDate = myDateString.AsDateTime();

ToString()	Converts any other data type to a string.	int num1 = 17;
		int num2 = 76;
		// myString is set to 1776
		string myString =
		num1.ToString() +
		num2.ToString();

Operator

Operator	Description	Examples
+ - * /	Math operators used in numerical expressions.	@(5 + 13) @{ var netWorth = 150000; } @{ var newTotal = netWorth * 2;} @(newTotal/2)

=	Assignment. Assigns the value on the right side of a statement to the object on the left side.	var age = 17;

==	Equality. Returns true if the values are equal. (Notice the distinction between the = operator and the =operator.)	var myNum = 15;
		if (myNum = 15) {
		// Do something.
		}

!=	Inequality. Returns true if the values are not equal.	var theNum = 13;
		if (theNum != 15) {
		// Do something.
		}

<
>
<=
>=
	Less-than,
	greater-than,
	less-than-or-equal, and
	greater-than-or-equal.
		if (2 < 3) {
		// Do something.
		}
		var currentCount = 12;
		if(currentCount >= 12) {
		// Do something.
		}

+	Concatenation, which is used to join strings. ASP.NET knows the difference between this operator and the addition operator based on the data type of the expression.	// The displayed result is "abcdef". @("abc" + "def")
+= -=	The increment and decrement operators, which add and subtract 1 (respectively) from a variable.	int theCount = 0; theCount += 1; // Adds 1 to count
.	Dot. Used to distinguish objects and their properties and methods.	var myUrl = Request.Url; var count = Request["Count"].AsInt();
()	Parentheses. Used to group expressions and to pass parameters to methods.	@(3 + 7) @Request.MapPath (Request.FilePath);
[]	Brackets. Used for accessing values in arrays or collections.	var income = Request["AnnualIncome"];

!	Not. Reverses a true value to false and vice versa. Typically used as a shorthand way to test for false (that is, for not true).	bool taskCompleted = false;
		// Processing.
		if(!taskCompleted) {
		// Continue processing
		}

&&
||
	Logical AND and OR, which are used to link conditions together.	bool myTaskCompleted = false;
		int totalCount = 0;
		// Processing.
		if(!myTaskCompleted &&
		totalCount < 12) {
		// Continue processing.
		}

WebMatrix is a very integrated way of writing ASP.NET and includes a very simplified way to access SQL databases by simply selecting the **Databases** workspace (as against the Files workspace) and using the built-in **table data editor** and **WebGrid** helper. However, a more conventional way to access SQL databases from ASP.NET is given below.

To access SQL databases from ASP.NET

1) Create a database connection using the **SqlConnection class**.

2) Select a set of records from the database using the **SqlDataAdapter** class.

3) Fill a new **DataSet** using the SqlDataAdapter class.

4) If you are selecting data from a database for non-interactive display only, it is recommended that you use a read-only, forward-only **SqlDataReader** (or OleDbDataReader for non-SQL databases) for best performance. When using a SqlDataReader, select the records using a **SqlCommand** query and create a SqlDataReader that is returned from the SqlCommand object's **ExecuteReader** method.

   In some cases, such as when you want to sort or filter a set of data, you might also want to create a new DataView based on a DataSet for the desired table.

5) Bind a server control, such as a **DataGrid**, to the **DataSet**, **SqlDataReader** or **DataView**.

Reusing code through developing user controls

Ease of Code Re-use is the most powerful feature of ASP.NET. This can be done through user controls. The syntax of a user control is very similar to that of a Web Forms page. The primary differences are that the user controls use a **@ Control** directive in place of a **@ Page** directive and that the user controls do not include the <html>, <body> and <form> elements around the content.

To create a user control

1) Create a new file and give name it with the extension .ascx.

2) Create a @ Control directive at the top of the page and specify the programming language you want to use for the control (if any).

   <%@ Control Language="VB" %>

(**Note:** In Visual Studio .NET, all pages and user controls in the application must be in the same programming language.)

3) Create user interface elements (controls) that you want the user control to display.

4) Create properties in the control to share information between the user control and the hosting page.

The following example shows a complete user control that displays a text box where users can enter a name and a label where the name is displayed. The user control also exposes a Name property so that the name can be set in the hosting page. **(MSDN)**

```
<%@ Control Language="VB" %>
<script runat="server">
Public Property Name As String
Get
Return labelOutput.Text
End Get
Set
textName.Text = Server.HtmlEncode(value)
labelOutput.Text = Server.HtmlEncode(value)
End Set
End Property
Public Sub buttonDisplayName_Click(sender As Object, e As EventArgs)
labelOutput.Text = textName.Text
End Sub
</script>
<table>
<tbody>
<tr>
<td>
<b>Enter your name:</b></td>
</tr>
<tr>
<td>
<asp:TextBox id="textName"
runat="server">
</asp:TextBox>
</td>
</tr>
<tr>
<td>
<asp:button id="buttonDisplayName"
onclick="buttonDisplayName_Click
```

```
runat="server" text="Submit">
</asp:button>
</td>
</tr>
<tr>
<td><b>Hello,
<asp:Label id="labelOutput"
runat="server">
</asp:Label>.</b>
</td>
</tr>
</tbody>
</table>
[C#]
<%@ Control Language="C#" %>
<script runat="server">
public String Name {
get
{
return labelOutput.Text;
}
set
{
textName.Text = Server.HtmlEncode(value);
labelOutput.Text = Server.HtmlEncode(value);
}
}
void buttonDisplayName_Click(object sender, EventArgs e) {
labelOutput.Text = textName.Text;
}
</script>
<table>
<tbody>
<tr>
<td>
```

```
<b>Enter your name:</b></td>
</tr>
<tr>
<td>
<asp:TextBox id="textName"
runat="server">
</asp:TextBox>
</td>
</tr>
<tr>
<td>
<asp:button id="buttonDisplayName"
onclick="buttonDisplayName_Click"
runat="server" text="Submit">
</asp:button>
</td>
</tr>
<tr>
<td><b>Hello,
<asp:Label id="labelOutput"
runat="server">
</asp:Label>.</b>
</td>
</tr>
</tbody>
</table>
```

To include a user control in a Web Forms page

1) In the containing Web Forms page, declare an @ Register directive that includes:
   - **A tagprefix attribute:** which associates a prefix with the user control. This prefix will be included in opening tag of the user control element.
   - **A tagname attribute:** which associates a name with the user control. This name will be included in the opening tag of the user control element.
   - **A Src attribute:** which defines the virtual path to the user control file that you are including in the Web Forms page.

   Note: The tilde (~) character represents the root directory of the application.

For example, the following code registers a user control defined in the file Login1.ascx with tag prefix Acme and tag name Login. The file is located in Controls directory.

<%@ Register TagPrefix="Acme" TagName="Login" Src="~\controls\login1.ascx" %>

2) Declare the user control element between the opening and closing tags of the HtmlForm server control (<form runat=server></form>). For example, to declare the control imported in the previous step, use the following syntax:-

<html><body>

<form runat="server">

<Acme:Login id="MyLogin" runat="server"/>

</form>

</body></html>

Note: Regardless of how many ASP.NET server controls (user controls and any others) you include on your Web Forms page, you should include only one HtmlForm server control on a Web Forms page. So, include all server controls between the opening and closing tags of this control. (MSDN)

Check Your Progress 4

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) Why is AsInt method used even though numeric value is input in form?

.....

.....

.....

.....

.....

2) Differentiate between SqlConnection and SqlDataAdapter.

.....

.....

.....

.....

.....

3.6 LET US SUM UP

The generic web server architecture contains several sub-systems with the basic purpose of responding to the client request. This is required because the client can not be exposed to all the resources in the web server. Accordingly server side scripts are so designed so as to hide the internal working of the server and also provide scripting embedded within the HTML markup. JSP and ASP are popular server side scripting technologies and ASP.NET provides the functionality of .NET Framework. Testing of ASP.NET has become quicker with WebMatrix.

3.7 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) The generic sub-systems of a web server are:
   a) The Reception sub-system
   b) The Request Analyzer sub-system
   c) The Access Control sub-system
   d) The Resource Handler sub-system
   e) The Transaction Log sub-system
   f) The Utility sub-system and
   g) The Operating System Abstraction Layer.
2) server.xml.

Check Your Progress 2

1) Scripting elements, directives and actions.
2) Expressions, Scriptlets and Declarations.
3) Code re-use.

Check Your Progress 3

1) VBScript or JScript.

Check Your Progress 4

1) Data input is taken as string even though numeric value is entered in the form.
2) The former provides the database connection while the later provides a set of records.

3.8 SUGGESTED READINGS

- Hassan and Holt, *A reference architecture for web servers*, University of Waterloo, www.bauhaus-stuttgart.de/dagstuhl/HassanHolt.pdf.
- http://java.sun.com/developer/technicalArticles/Programming/jsp/.
- http://msdn.microsoft.com/en-us/library/26db8ysc(v=VS.71).aspx.
- http://msdn.microsoft.com/en-us/library/aa286483.aspx.
- http://msdn.microsoft.com/en-us/library/sbz9etab(v=VS.71).aspx.
- http://www.apache.org/httpd.html.
- http://www.microsoft.com/web/post/create-an-aspnet-website-from-scratch.

UNIT 4 ATTACKS ON WEB APPLICATION

Structure

4.0 INTRODUCTION

Attacks on web applications refer to threat at the application-level as against operating system-level and services-level. Therefore, when we use a hardware firewall or an Intrusion Detection System for network security, it does not mean that we are protecting against attacks on web applications.

So, what is web application security about? It is about securing:

a) the code in the web application

b) the backend systems

c) the web and application servers

d) the users.

4.1 OBJECTIVES

After studying this unit, you should be able to:

- know the various types of threats to web applications;
- gather a view about the threats which are more prevalent;
- understand an important type of hacking in more detail; and
- know the precautions and remedies to these threats.

4.2 TYPES OF WEB APPLICATION ATTACKS

One way of categorizing the various web application attacks is:

- Unvalidated input
- Broken access control
- Broken account/session management
- Cross-site scripting flaws
- Buffer overflows
- Injection flaws
- Error handling problems
- Insecure storage/use of cryptography
- Denial of service
- Insecure configuration management

However, the issue is closely followed and researchers have identified the threats and ascribed scores to them. One such is the 2011 CWE/SANS Top 25 list of the most widespread and critical errors that can lead to serious vulnerabilities in software. This is detailed ahead.

4.2.1 The Top 25 Threats to Web Applications

Rank	Score	Name
[1]	93.8	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	Missing Authentication for Critical Function
[6]	76.8	Missing Authorization
[7]	75.0	Use of Hard-coded Credentials
[8]	75.0	Missing Encryption of Sensitive Data
[9]	74.0	Unrestricted Upload of File with Dangerous Type
[10]	73.8	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	Execution with Unnecessary Privileges
[12]	70.1	Cross-Site Request Forgery (CSRF)
[13]	69.3	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	Download of Code Without Integrity Check

[15]	67.8	Incorrect Authorization
[16]	66.0	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	Incorrect Permission Assignment for Critical Resource
[18]	64.6	Use of Potentially Dangerous Function
[19]	64.1	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	Incorrect Calculation of Buffer Size
[21]	61.5	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	Uncontrolled Format String
[24]	60.3	Integer Overflow or Wraparound
[25]	59.9	Use of a One-Way Hash without a Salt

1) **Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**

Summary

Weakness Prevalence	High	Consequences	Data loss, Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

If you use SQL queries in security controls such as authentication, attackers could alter the logic of those queries to bypass security. They could modify the queries to steal, corrupt or otherwise change your underlying data. They will even steal data one byte at a time if they have to and they have the patience and know-how to do so. In 2011, SQL injection was responsible for the compromises of many high-profile organizations, including Sony Pictures, PBS, MySQL.com, security company HBGary Federal and many others.

2) **Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')**

Summary

Weakness Prevalence	Medium	Consequences	Code execution
Remediation Cost	Medium	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

Your software is often the bridge between an outsider on the network and the internals of your operating system. When you allow untrusted inputs to be fed into the OS command string that you generate for executing a program, then attackers can execute their own commands instead of yours.

3) **Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**

Summary

Weakness Prevalence	High	Consequences	Code execution, Denial of service, Data loss
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

Copying an untrusted input without checking the size of that input is the simplest error to make. That is why this type of buffer overflow is often referred to as "classic".

4) **Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')**

Summary

Weakness Prevalence	High	Consequences	Code execution, Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

Cross-site scripting (XSS) can happen because we combine the stateless nature of HTTP, the mixture of data and script in HTML, lots of data passing between web sites, diverse encoding schemes and feature-rich web browsers. Attackers can inject Javascript or other browser-executable content into a web page that your application generates. Your web page is then accessed by other users, whose browsers execute that malicious script as if it came from you. The attacker can use a variety of techniques to get the input directly into your server.

5) **Missing Authentication for Critical Function**

Summary

Weakness Prevalence	Common	Consequences	Security bypass
Remediation Cost	Low to High	Ease of Detection	Moderate
Attack Frequency	Sometimes	Attacker Awareness	High

Discussion

To break into a house one would choose to crawl through pipes, scaling elevator shafts or noiselessly break glass windows. Nobody would prefer walking across security guards asking for identification. Attackers seek to find a critical function in which some authentication mechanism is missing or weak.

6) **Missing Authorization**

Summary

Weakness Prevalence	High	Consequences	Security bypass
Remediation Cost	Low to Medium	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

If we have invited friends to a party but one of the friends peeks into our almirah or safe without permission then it is a case of missing authorization. In May 2011, Citigroup revealed that it had been compromised by hackers who were able to steal details of hundreds of thousands of bank accounts by changing the account information that was present in fields in the URL; authorization would check that the user had the rights to access the account being specified. Earlier, a similar missing-authorization attack was used to steal private information of iPad owners from an AT&T site.

7) **Use of Hard-coded Credentials**

Summary

Weakness Prevalence	Medium	Consequences	Security bypass
Remediation Cost	Medium to High	Ease of Detection	Moderate
Attack Frequency	Rarely	Attacker Awareness	High

Discussion

Hard-coding a secret password or cryptograpic key into your program is very risky. Being hard-coded, it is usually a huge pain for sysadmins to fix. The high-profile Stuxnet worm, which caused operational problems in an Iranian nuclear site, used hard-coded credentials in order to spread. Another way that hard-coded credentials arise is through unencrypted or obfuscated storage in a configuration file, registry key or other location that is only intended to be accessible to an administrator.

8) **Missing Encryption of Sensitive Data**

Summary

Weakness Prevalence	High	Consequences	Data loss
Remediation Cost	Medium	Ease of Detection	Easy
Attack Frequency	Sometimes	Attacker Awareness	High

Discussion

Encryption can prevent data theft in transmission. Otherwise software sniffers can read the data packets and there are tools to reach the databases. If your software sends private data or authentication credentials, that can compromise security. Many of the credit card data thefts are due to unencrypted storage. In June 2011, the LulzSec group grabbed headlines by grabbing and publishing unencrypted data.

9) **Unrestricted pload of File with Dangerous Type**

Summary

Weakness Prevalence	Commonz	Consequences	Code execution
Remediation Cost	Medium	Ease of Detection	Moderate
Attack Frequency	Sometimes	Attacker Awareness	Medium

Discussion

Uploading of innocent-looking files such as images in the server can be a risk. These files may even have innocent extensions. A picture may be associated with malicious code which may be uploaded in the server.

10) **Reliance on Untrusted Inputs in a Security Decision**

Summary

Weakness Prevalence	High	Consequences	Security bypass
Remediation Cost	Medium	Ease of Detection	Moderate
Attack Frequency High	Often		Attacker Awareness

Discussion

A fake address proof may get a fake driving license which may grant more fake credentials. In granting access to restricted resources it is important that the information sought for authentication and authorization should be worth trusting. Credentials submitted for requesting a login ID or for requesting an authorization may be fake.

11) **Execution with Unnecessary Privileges**

Summary

Weakness Prevalence	Medium	Consequences	Code execution
Remediation Cost	Medium	Ease of Detection	Moderate
Attack Frequency	Sometimes	Attacker Awareness	High

Discussion

Often we leave the house keys to some house maid or some known person. The set of keys may contain all the keys including that of the safe. The developer can grant unnecessary privileges for ease of programming. But this makes other programs vulnerable.

12) **Cross-Site Request Forgery (CSRF)**

Summary

Weakness Prevalence	High	Consequences	Data loss, Code execution
Remediation Cost	High	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	Medium

Discussion

In Cross-site request forgery the attacker tricks a user into activating a request that goes to your site. The user may be unaware of the request being sent. The attacker masquerades as a legitimate user and can gain access that the user has. This is especially handy when the user has administrator privileges. When combined with Cross-site scripting (XSS), the result can be extensive. If you've heard about XSS worms that stampede through very large web sites in a matter of minutes (like Facebook), there's usually CSRF feeding them.

13) **Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')**

Summary

Weakness Prevalence	Widespread	Consequences	Code execution, Data loss, Denial of service

Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

While data is often exchanged using files, sometimes you don't intend to expose every file on your system while doing so. When you use an outsider's input while constructing a filename, the resulting path could point outside of the intended directory. An attacker could combine multiple ".." or similar sequences to cause the operating system to navigate out of the restricted directory and into the rest of the system.

14) **Download of Code Without Integrity Check**

Summary

Weakness Prevalence	Medium	Consequences	Code execution
Remediation Cost	Medium to High	Ease of Detection	Moderate
Attack Frequency	Rarely	Attacker Awareness	Low

Discussion

Maybe you only access a download site that you trust, but attackers can perform all sorts of tricks to modify that code before it reaches you. They can hack the download site, impersonate it with DNS spoofing or cache poisoning, convince the system to redirect to a different site or even modify the code in transit as it crosses the network. This scenario even applies to cases in which your own product downloads and installs its own updates. Integrity check can prevent the damage.

15) **Incorrect Authorization**

Summary

Weakness Prevalence	High	Consequences	Security bypass
Remediation Cost	Low to Medium	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

While the lack of authorization is more dangerous, incorrect authorization can be just as problematic. For example, once a person has logged in to a web application, the developer may store the permissions in a cookie. By modifying the cookie, the attacker can access other resources. Alternately, the developer might perform authorization by delivering code that gets executed in the web client, but an attacker could use a customized client that removes the check entirely.

16) **Inclusion of Functionality from Untrusted Control Sphere**

Summary

Weakness Prevalence	High	Consequences	Security bypass
Remediation Cost	Low to Medium	Ease of Detection	Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

It is common in these days of Web 2.0 to build a large program from many small programs. One of these small programs could be from the attacker's server.

17) **Incorrect Permission Assignment for Critical Resource**

Summary

Weakness Prevalence	Medium	Consequences	Data loss, Code execution
Remediation Cost	Low to High	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

If there are critical programs, data stores or configuration files with incorrect permissions like reading or writing, it may be difficult for sysadmin to notice it.

18) **Use of Potentially Dangerous Function**

Summary

Weakness Prevalence	High	Consequences	Data loss, Code execution
Remediation Cost	Medium	Ease of Detection	Easy
Attack Frequency	Rarely	Attacker Awareness	High

Discussion

The programmer's toolbox is full of power tools, including library or API functions that make assumptions about how they will be used, with no guarantees of safety if they are abused. If potentially-dangerous functions are not used properly, then things can get dangerous.

19) **Use of a Broken or Risky Cryptographic Algorithm**

Summary

Weakness Prevalence	High	Consequences	Data loss, Security bypass
Remediation Cost	Medium to High	Ease of Detection	Moderate
Attack Frequency	Rarely	Attacker Awareness	Medium

Discussion

Cryptography can protect from reading sensitive data across a communication channel. You may be tempted to develop your own encryption scheme but this is most likely an invite to attackers to break it. It is better to use proven and trusted algorithms.

20) **Incorrect Calculation of Buffer Size**

Summary

Weakness Prevalence	High	Consequences	Code execution, Denial of service, Data loss
Remediation Cost	Low	Ease of Detection	Easy to Moderate
Attack Frequency	Often	Attacker Awareness	High

Discussion

In languages such as C, in which memory management is the programmer's responsibility, there are many opportunities for error. If not calculated, then the buffer may be too small to write the data into. This may be exploited by an attacker.

21) **Improper Restriction of Excessive Authentication Attempts**

Summary

Weakness Prevalence	Consequences
Remediation Cost	Ease of Detection
Attack Frequency	Attacker Awareness

Discussion

Attackers may try to break into your account by writing programs that repeatedly try to guess different passwords. Some kind of protection is required against brute force techniques.

22) **URL Redirection to Untrusted Site ('Open Redirect')**

Summary

Weakness Prevalence	High	Consequences	Code execution, Data loss, Denial of service
Remediation Cost	Medium	Ease of Detection	Easy
Attack Frequency	Sometimes	Attacker Awareness	Medium

Discussion

The World Wide Web is about sharing and following links between web sites, but there are threats. Firstly, the victim could be automatically redirected to a malicious site that tries to attack the victim through the web browser. Alternately, a phishing attack could be conducted, which tricks victims into visiting malicious sites that are posing as legitimate sites. They could then collect vital information of the user when the user fills in a form, say.

23) **Uncontrolled Format String**

Summary

Weakness Prevalence	Consequences
Remediation Cost	Ease of Detection
Attack Frequency	Attacker Awareness

Discussion

Format strings are often used to send or receive well-formed data. By manipulating a format string, the attacker can control the input or output – sometimes, even, to execute harmful code.

24) **Integer Overflow or Wraparound**

Summary

Weakness Prevalence	Common	Consequences	Denial of service, Code execution, Data loss
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Sometimes	Attacker Awareness	High

Discussion

Integer Overflow, infinite loops can arise out of faulty code. These can be manipulated and can cause crashes.

25) **Use of a One-Way Hash without a Salt**

Summary

Weakness Prevalence	Medium	Consequences	Security bypass
Remediation Cost	Medium to High	Ease of Detection	Moderate
Attack Frequency	Rarely	Attacker Awareness	High

Discussion

Salt is the one-way hash applied to passwords so as to avoid storing passwords in plain text. This randomizes the output and keeps the password database safe.

Check Your Progress 1

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What is CSRF?

.....

.....

.....

.....

2) What is the difference between SQL Injection and OS Command Injection?

.....

.....

.....

.....

3) What is Buffer Overflow?

.....

.....

.....

.....

4) What is Path Traversal threat?

.....

.....

.....

.....

4.3 CASE STUDY

As stated earlier, SQL Injection can manipulation the SQL through exploratory input. This type of attack is the most prevalent among all attacks.

4.3.1 SQL Injection

The various observed types/variants of SQL Injection are given below:-

Observed Examples

Description
Chain: SQL injection in library intended for database authentication allows SQL injection and authentication bypass.
SQL injection through an ID that was supposed to be numeric.
SQL injection through an ID that was supposed to be numeric.
SQL injection via user name.
SQL injection via user name or password fields.
SQL injection in security product, using a crafted group name.
SQL injection in authentication library.

Now let us see some actual code and some varieties of the attack.

...

string userName = ctx.getAuthenticatedUserName();

string query = "SELECT * FROM items WHERE owner = '" + userName + "' AND itemname = '" + ItemName.Text + "'";

sda = new SqlDataAdapter(query, conn);

DataTable dt = new DataTable();

sda.Fill(dt);

...

The query that this code intends to execute follows:

SELECT * FROM items WHERE owner = <userName> AND itemname = <itemName>;

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string:

name' OR 'a'='a

for itemName, then the query becomes the following:

SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name' OR 'a'='a';

The addition of the:

OR 'a'='a'

condition causes the WHERE clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

SELECT * FROM items;

This simplification of the query allows the attacker to bypass the requirement that the query only return items owned by the authenticated user; the query now returns all entries stored in the items table, regardless of their specified owner.

Example

This example examines the effects of a different malicious value passed to the query constructed and executed in the previous example.

If an attacker with the user name wiley enters the string:

name'; DELETE FROM items; --

for itemName, then the query becomes the following two queries:

SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name';

DELETE FROM items;

--'

Many database servers, including Microsoft(R) SQL Server 2000, allow multiple SQL statements separated by semicolons to be executed at once. While this attack string results in an error on Oracle and other database servers that do not allow the batch-execution of statements separated by semicolons, on databases that do allow batch execution, this type of attack allows the attacker to execute arbitrary commands against the database.

Notice the trailing pair of hyphens (--), which specifies to most database servers that the remainder of the statement is to be treated as a comment and not executed. In this case the comment character serves to remove the trailing single-quote left over from the modified query. On a database where comments are not allowed to be used in this way, the general attack could still be made effective using a trick similar to the one shown in the previous example.

If an attacker enters the string

name'; DELETE FROM items; SELECT * FROM items WHERE 'a'= 'a

Then the following three valid statements will be created:

SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name';

DELETE FROM items;

SELECT * FROM items WHERE 'a'='a';

One traditional approach to preventing SQL injection attacks is to handle them as an input validation problem and either accept only characters from a whitelist of safe values or identify and escape a blacklist of potentially malicious values. Whitelisting can be a very effective means of enforcing strict input validation rules, but parameterized SQL statements require less maintenance and can offer more guarantees with respect to security. As is almost always the case, blacklisting is riddled with loopholes that make it ineffective at preventing SQL injection attacks. For example, attackers can:

- Target fields that are not quoted
- Find ways to bypass the need for certain escaped meta-characters
- Use stored procedures to hide the injected meta-characters.

Manually escaping characters in input to SQL queries can help, but it will not make your application secure from SQL injection attacks.

Another solution commonly proposed for dealing with SQL injection attacks is to use stored procedures. Although stored procedures prevent some types of SQL injection attacks, they do not protect against many others. For example, the following PL/SQL procedure is vulnerable to the same SQL injection attack shown in the first example.

procedure get_item (itm_cv IN OUT ItmCurTyp, usr in varchar2, itm in varchar2)

is open itm_cv for

' SELECT * FROM items WHERE ' || 'owner = '|| usr || ' AND itemname = ' || itm || ';

end get_item;

Stored procedures typically help prevent SQL injection attacks by limiting the types of statements that can be passed to their parameters. However, there are many ways around the limitations and many interesting statements that can still be passed to stored procedures. Again, stored procedures can prevent some exploits, but they will not make your application secure against SQL injection attacks.

MS SQL has a built in function that enables shell command execution. An SQL injection in such a context could be disastrous. For example, a query of the form:

SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY ='$user_input' ORDER BY PRICE

Where $user_input is taken from an untrusted source.

If the user provides the string:

'; exec master..xp_cmdshell' 'dir' --

The query will take the following form:

SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY=''; exec master..xp_cmdshell 'dir' --' ORDER BY PRICE

Now, this query can be broken down into:

1) a first SQL query: SELECT ITEM, PRICE FROM PRODUCT WHERE ITEM_CATEGORY='';

2) a second SQL query, which executes the dir command in the shell: exec master..xp_cmdshell 'dir'

3) an MS SQL comment: --' ORDER BY PRICE

As can be seen, the malicious input changes the semantics of the query into a query, a shell command execution and a comment.

Check Your Progress 2

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) What is the benefit of the OR clause in SQL Injection?

.....

.....

.....

.....

2) What is the use of the trailing pair of hyphens (--) in SQL Injection?

.....

.....

.....

.....

4.4 DETECTION

There can be two ways of managing the risks or threats described above. One way is to detect it when an attack takes place. This is similar to diagnosis of illness so that appropriate medicines can be taken. The other way is to prevent it. This is similar to vaccination and healthy living. Here we understand the former, that is, detecting an attack. Detection can be of two types:- Static and Dynamic.

4.4.1 Static Detection

Static detection is about detecting an attack which has happened after observing the logs. It is similar to a forensic examination of the transaction logs. This can be done by analyzing the logs of web server, application server and operating system.

One limitation of this detection method is that all request and response transactions are not fully logged. Another is that HTTP headers and POST data are not fully recorded.

However, this method is easier to use and implement.

4.4.2 Dynamic Detection

Dynamic detection involves detecting attacks when they are occurring. This can be done by matching against signatures of known attacks or matching against known traffic patterns. The former is signature based and easier to implement. The latter is anomaly based and finds abnormal transaction spurts.

Check Your Progress 3

Notes: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of this Unit.

1) Differentiate between Static and Dynamic Detection.

.....

.....

.....

.....

4.5 PREVENTION AND SOLUTION

Most of the prevention measures have emerged after observation of the attacks. Therefore, there can be no complete prevention or complete solution from the attacks to web applications, as newer attack methods may emerge. However, quite a significant body of precautionary measures have emerged. Various checklists are available for security checks.

Microsoft provides a very comprehensive summary which is presented below for reference. The idea in presenting the details below is not to understand each remedial method but to gather a general view of the gamut of possibilities in preventive measures against attacks on web applications. Of these, prevention of SQL Injection, prevention of Cross-Site Scripting and security of Application Server are important.

4.5.1 Architecture and Design Solutions

How to identify and evaluate threats?

Use threat modeling to systematically identify threats rather than applying security in a haphazard manner. Next, rate the threats based on the risk of an attack or occurrence of a security compromise and the potential damage that could result. This allows you to tackle threats in the appropriate order.

How to create secure designs?

Use tried and tested design principles. Focus on the critical areas where the correct approach is essential and where mistakes are often made. These as application vulnerability categories. They include input validation, authentication, authorization, configuration management, sensitive data protection, session management, cryptography, parameter manipulation, exception management and auditing and logging considerations. Pay serious attention to deployment issues including topologies, network infrastructure, security policies and procedures.

How to perform architecture and design review?

Review your application's design in relation to the target deployment environment and associated security policies. Consider the restrictions imposed by the underlying infrastructure layer security, including perimeter networks, firewalls, remote application servers and so on. Use application vulnerability categories to help partition your application and analyze the approach taken for each area.

4.5.2 Development Solutions

How to write secure managed code?

Use strong names to digitally sign your assemblies and to make them tamperproof.

At the same time you need to be aware of strong name issues when you use strong name assemblies with ASP.NET. Reduce your assembly attack profile by adhering to solid object oriented design principles and then use code

access security to further restrict which code can call your code. Use structured exception handling to prevent sensitive information from propagating beyond your current trust boundary and to develop more robust code. Avoid canonicalization issues, particularly with input file names and URLs.

How to handle exceptions securely?

Do not reveal internal system or application details, such as stack traces, SQL statement fragments and so on. Ensure that this type of information is not allowed to propagate to the end user or beyond your current trust boundary. Fail securely in the event of an exception and make sure your application denies

access and is not left in an insecure state. Do not log sensitive or private data such as passwords, which could be compromised. When you log or report exceptions, if user input is included in exception messages, validate it or sanitize it. For example, if you return an HTML error message, you should encode the output to avoid script injection.

How to perform security reviews of managed code?

Use analysis tools such as FxCop to analyze binary assemblies and to ensure that they conform to the .NET Framework design guidelines. Fix any security vulnerabilities identified by your analysis tools. Use a text search facility to scan your source code base for hard-coded secrets such as passwords. Then, review specific elements of your application including Web pages and controls, data access code, Web services, serviced components and so on. Pay particular attention to SQL injection and cross-site scripting vulnerabilities. Also review the use of sensitive code access security techniques such as link demands and asserts.

How to secure a developer workstation?

You can apply a methodology when securing your workstation. Secure your accounts, protocols, ports, services, shares, files and directories and registry. Most importantly, keep your workstation current with the latest patches and updates. If you run Internet Information Services (IIS) on Microsoft Windows® XP or Windows 2000, then run IISLockdown. IISLockdown applies secures IIS configurations and installs the URLScan Internet Security Application Programming Interface (ISAPI) filter, which detects and rejects potentially malicious HTTP requests. You may need to modify the default URLScan configuration, for example, so you can debug Web applications during development and testing.

How to use code access security with ASP.NET?

With.NET Framework version 1.1, you can set ASP.NET trust levels either in Machine.config or Web.config. These trust levels use code access security to restrict the resources that ASP.NET applications can access, such as the file system, registry, network, databases and so on. In addition, they provide application isolation.

How to write least privileged code?

You can restrict what code can do regardless of the account used to run the code. You can use code access security to constrain the resources and operations that your code is allowed to access, either by configuring policy or how you write your code. If your code does not need to access a resource or perform a sensitive operation such as calling unmanaged code, you can use declarative security attributes to ensure that your code cannot be granted this permission by an administrator.

How to constrain file I/O?

You can use code access security to constrain an assembly's ability to access areas of the file system and perform file I/O. For example, you can constrain a Web application so that it can only perform file I/O beneath its virtual directory hierarchy. You can also constrain file I/O to specific directories. You can do this programmatically or by configuring code access security policy.

How to prevent SQL injection?

Use parameterized stored procedures for data access. The use of parameters ensures that input values are checked for type and length. Parameters are also treated as safe literal values and not executable code within the database. If you cannot use stored procedures, use SQL statements with parameters. Do not build SQL statements by concatenating input values with SQL commands. Also, ensure that your application uses a least privileged database login to constrain its capabilities in the database.

How to prevent cross-site scripting?

Validate input for type, length, format and range and encode output. Encode output if it includes input, including Web input. For example, encode form fields, query string parameters, cookies and so on and encode input read from a database (especially a shared database) where you cannot assume the data is safe. For free format input fields that you need to return to the client as HTML, encode the output and then selectively remove the encoding on permitted elements such as the <b> or <i> tags for formatting.

How to manage secrets?

Look for alternate approaches to avoid storing secrets in the first place. If you must store them, do not store them in clear text in source code or in configuration files. Encrypt secrets with the Data Protection Application Programming Interface (DPAPI) to avoid key management issues.

How to call unmanaged code securely?

Pay particular attention to the parameters passed to and from unmanaged APIs and guard against potential buffer overflows. Validate the lengths of input and output string parameters, check array bounds and be particularly careful with file path lengths. Use custom permission demands to protect access to unmanaged resources before asserting the unmanaged code permission. Use caution if you use SuppressUnmanagedCodeSecurityAttribute to improve performance.

How to perform secure input validation?

Constrain, reject and sanitize your input because it is much easier to validate data for known valid types, patterns and ranges than it is to validate data by looking for known bad characters. Validate data for type, length, format and range. For string input, use regular expressions. To perform type checks, use the .NET. Framework type system. On occasion, you may need to sanitize input. An example is encoding data to make it safe.

How to secure Forms authentication?

Partition your Web site to separate publicly accessible pages available to anonymous users and restricted pages which require authenticated access. Use Secure Sockets Layer (SSL) to protect the forms authentication credentials and the forms authentication cookie. Limit session lifetime and ensure that the authentication cookie is passed over HTTPS only. Encrypt the authentication cookie, do not persist it on the client computer and do not use it for personalization purposes; use a separate cookie for personalization.

4.5.3 Administration Solutions

How to implement patch management?

Use the Microsoft Baseline Security Analyzer (MBSA) to detect the patches and updates that may be missing from your current installation. Run this on a regular basis and keep your servers current with the latest patches and updates. Back up servers prior to applying patches and test patches on test servers prior to installing them on a production server. Also, use the security notification services provided by Microsoft and subscribe to receive security bulletins via e-mail.

How to make the settings in Machine.config and Web.config more secure?

Do not store passwords or sensitive data in plaintext. For example, use the Aspnet_setreg.exe utility to encrypt the values for <processModel>, <identity> and <sessionState>. Do not reveal exception details to the client. For example do not use mode="Off" for <customErrors> in ASP.NET because it causes detailed error pages that contain system-level information to be returned to the client. Restrict who has access to configuration files and settings. Lock configuration settings if necessary, using the <location> tag and the allowOverride element.

How to secure a Web server running the .NET Framework?

Apply a methodology to systematically configure the security of your Web server. Secure your accounts, protocols, ports, services, shares, files and directories and registry. You can use IISLockdown to help automate some of the security configuration. Use a hardened Machine.config configuration to apply stringent security to all .NET Framework applications installed on the server. Most importantly, keep your server current with the latest patches and updates.

How to secure a database server?

Apply a common methodology to evaluate accounts, protocols, ports, services, shares, files and directories and the registry. Also evaluate SQL Server™ security settings such as the authentication mode and auditing configuration. Evaluate your authorization approach and use of SQL Server logins, users and roles. Make sure you have the latest service pack and regular monitor for operating system and SQL Server patches and updates.

How to secure an application server?

Evaluate accounts, protocols, ports, services, shares, files and directories and the registry. Use Internet Protocol Security (IPSec) or SSL to secure the communication channel between the Web server and the application server and between the application server and the database server. Review the security of your Enterprise Services applications, Web services and remoting applications. Restrict the range of ports with which clients can connect to the application server and consider using IPSec restrictions to limit the range of clients.

How to host multiple ASP.NET applications securely?

Use separate identities to allow you to configure access control lists (ACLs) on secure resources to control which applications have access to them. On the Microsoft Windows Server 2003 operating system, use separate process identities with IIS 6 application pools. On Windows 2000 Server, use multiple anonymous Internet user accounts and enable impersonation. With the .NET Framework version 1.1 on both platforms, you can use partial trust levels and use code access security to provide further application isolation. For example, you can use these methods to prevent applications from accessing each other's virtual directories and critical system resources.

How to secure Web services?

In cross-platform scenarios and where you do not control both endpoints, use the Web Services Enhancements 1.0 for Microsoft .NET (WSE) to implement message level security solutions that conform to the emerging WS-Security standard. Pass authentication tokens in Simple Object Access Protocol (SOAP) headers. Use XML encryption to ensure that sensitive data remains private. Use digital signatures for message integrity. Within the enterprise where you control both endpoints, you can use the authentication, authorization and secure communication features provided by the operating system and IIS.

How to secure Enterprise Services?

Configure server applications to run using least privileged accounts. Enable COM+ role-based security and enforce component-level access checks. At the minimum, use call-level authentication to prevent anonymous access. To secure the traffic passed to remote serviced components, use IPSec encrypted channels or use remote procedure call (RPC) encryption. Restrict the range of ports that Distributed COM (DCOM) dynamically allocates or use static endpoint mapping to limit the port range to specific ports. Regularly monitor for Quick Fix Engineer (QFE) updates to the COM+ runtime.

How to secure Microsoft .NET Remoting?

Disable remoting on Internet-facing Web servers by mapping .rem and .soap extensions to the ASP.NET HttpForbiddenHandler HTTP module in Machine.config. Host in ASP.NET and use the HttpChannel type name to benefit from ASP.NET and IIS authentication and authorization services. If you need to

use the TcpChannel type name, host your remote components in a Windows service and use IPSec to restrict which clients can connect to your server. Use this approach only in a trusted server situation, where the remoting client (for example a Web application) authenticates and authorizes the original callers.

How to secure session state?

You need to protect session state while in transit across the network and while in the state store. If you use a remote state store, secure the communication channel to the state store using SSL or IPSec. Also encrypt the connection string in Machine.config. If you use a SQL Server state store, use Windows authentication when you connect to the state store and limit the application login in the database. If you use the ASP.NET state service, use a least privileged account to run the service and consider changing the default port that the service listens to. If you do not need the state service, disable it.

How to manage application configuration securely?

Remote administration should be limited or avoided. Strong authentication should be required for administrative interfaces. Restrict access to configuration stores through ACLs and permissions. Make sure you have the granularity of authorization required to support separation of duties.

How to secure against denial of service attacks?

Make sure the TCP/IP stack configuration on your server is hardened to protect against attacks such as SYN floods. Configure ASP.NET to limit the size of accepted POST requests and to place limits on request execution times.

How to constrain file I/O?

You can configure code access security policy to ensure that individual assemblies or entire Web applications are limited in their ability to access the file system. For

example, by configuring a Web application to run at the Medium trust level, you prevent the application from being able to access files outside of its virtual directory hierarchy. Also, by granting a restricted file I/O permission to a particular assembly you can control precisely which files it is able to access and how it should be able to access them.

How to perform remote administration?

Terminal Services provides a proprietary protocol (RDP.) This supports authentication and can provide encryption. If you need a file transfer facility, you can install the File Copy utility from the Windows 2000 Server resource kit. The use of IIS Web administration is not recommended and this option is removed if you run IISLockdown. You should consider providing an encrypted channel of communication and using IPSec to limit the computers that can be used to remotely administer your server. You should also limit the number of administration accounts.

Check Your Progress 4

Notes: a) Space is given below for writing your answers.

b) Compare your answers with the one given at the end of this Unit.

1) Mention some method how SQL Injection can be prevented?

.....

.....

.....

.....

.....

2) Mention some method how cross-site scripting can be prevented?

.....

.....

.....

.....

.....

3) Mention some method how application server can be secured?

.....

.....

.....

.....

.....

4.6 LET US SUM UP

Nothing is absolutely secure in the internet. The various threats to web applications are analyzed and classified into groups. SQL Injection is one such important threat. The attacks can be detected statically or dynamically. Many preventive measures and solutions have emerged.

4.7 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Refer to Sub-Section 4.2.1.

2) SQL Injection attacks the user input validation vulnerability. OS Command Injection attacks the Operating System exposure vulnerability.

3) Refer to Sub-Section 4.2.1.

4) Refer to Sub-Section 4.2.1.

Check Your Progress 2

1) a) always evaluates to true. b) user permission can be bypassed.

2) The trailing single-quote is ignored as the paired hyphen is for comment.

Check Your Progress 3

1) Refer to Section 4.4.

Check Your Progress 4

1) Refer to Sub-Section 4.5.2.

2) Refer to Sub-Section 4.5.2.

3) Refer to Sub-Section 4.5.3.

4.8 SUGGESTED READINGS

- http://msdn.microsoft.com/en-us/library/ff649874.aspx.
- http://projects.webappsec.org/w/page/13246978/Threat%20Classification.
- http://www.oracle.com/technetwork/database/features/plsql/overview/how-to-write-injection-proof-plsql-1-129572.pdf.
- http://www.sans.org/reading_room/whitepapers/logging/detecting-attacks-web-applications-log-files_2074.
- http://cwe.mitre.org/top25/index.html.

Student Satisfaction Survey

ignou THE PEOPLE'S UNIVERSITY

Student Satisfaction Survey of IGNOU Students

Enrollment No.	
Mobile No.	
Name	
Programme of Study	
Year of Enrolment	
Age Group	☐ Below 30 ☐ 31-40 ☐ 41-50 ☐ 51 and above
Gender	☐ Male ☐ Female
Regional Centre	
States	
Study Center Code	

Please indicate how much you are satisfied or dissatisfied with the following statements

Sl. No.	Questions	Very Satisfied	Satisfied	Average	Dissati-sfied	Very Dissati-sfied
1.	Concepts are clearly explained in the printed learning material	☐	☐	☐	☐	☐
2.	The learning materials were received in time	☐	☐	☐	☐	☐
3.	Supplementary study materials (like video/audio) available	☐	☐	☐	☐	☐
4.	Academic counselors explain the concepts clearly	☐	☐	☐	☐	☐
5.	The counseling sessions were interactive	☐	☐	☐	☐	☐
6.	Changes in the counseling schedule were communicated to you on time	☐	☐	☐	☐	☐
7.	Examination procedures were clearly given to you	☐	☐	☐	☐	☐
8.	Personnel in the study centers are helpful	☐	☐	☐	☐	☐
9.	Academic counseling sessions are well organized	☐	☐	☐	☐	☐
10.	Studying the programme/course provide the knowledge of the subject	☐	☐	☐	☐	☐
11.	Assignments are returned in time	☐	☐	☐	☐	☐
12.	Feedbacks on the assignments helped in clarifying the concepts	☐	☐	☐	☐	☐
13.	Project proposals are clearly marked and discussed	☐	☐	☐	☐	☐
14.	Results and grade card of the examination were provided on time	☐	☐	☐	☐	☐
15.	Overall, I am satisfied with the programme	☐	☐	☐	☐	☐
16.	Guidance from the programme coordinator and teachers from the school	☐	☐	☐	☐	☐

After filling this questionnaire send it to:
Programme Coordinator, School of Vocational Education and Training,
Room no. 19, Block no. 1, IGNOU, Maidangarhi, New Delhi- 110068

MPDD-IGNOU/P.O. 1T/September 2011

ISBN-978-81-266-5617-2