UNIT- I

1.1 BASIC CONCEPTS IN INTERNET

Internet:

- The internet is a collection of interconnected computer networks, linked by copper wires, fiber optic cables, wireless connections etc.
- Network is an interconnection of systems to share data and information. Internet is a network of network or collection of heterogeneous networks.

Web:

The web is a collection of interconnected documents, linked by hyperlinks and URLs and is accessible using the Internet.

Applications of internet:

- Electronic mail, World Wide Web, FTP, Telnet, Chat & Online transaction

HISTORY OF INTERNET AND WWW

History of the Internet:

- The USSR (Russia) launch of Sputnik spurred the U.S. to create the Advanced Research Project Agency (ARPA) in February 1958 to regain a technological lead.
- ARPA created the information processing technology office (IPTO), to further the research of the semi-Automatic Ground, Which had networked country-wide radar systems together for the first time.
- In late 1960's, one of the authors (HMD) research project MAC was funded by ARPA – of the Department of Defense (DOD).
- ARPA rolled out for networking the main computer systems of about a dozen ARPA – funded universities and research institutions.
- They were to be connected with communications lines operating at a then – stunning 56kbps (i.e., 56.000 bits per second) - this at a time of most people was connecting over telephone lines to computers at a rate of 110 bps.
- ARPA proceeded to implement the ARPANET which eventually evolved into today's internet.
- Researchers to share each other's computers, it rapidly became clear that enabling researchers to communicate quickly and easily (via) became known as electronic mail(e-mail).
- As e-mail facilities communications of all kinds among a billion people worldwide.

Goals for ARPANET:

- The Primary goal of ARPANET is to allow multiple users to send and receive information simultaneously over the same communication paths (e.g., Phone lines).
- The network operated with a technique called packet switching, in which digital data was sent in small bundles called packets.
- The packets contained address, error control and sequencing information. The address information allowed packets to be routed to their destination. The sequencing information helped in reassembling the packets; Packets from different senders were intermixed on the same lines. This packet switching

technique greatly reduced transmission costs, as compared with the cost of the dedicated communication lines.

- The network was designed to operate without centralized control. If a portion of the network failed, the remaining working portions would still route packets from senders to receivers over alternative paths for reliability. The protocol for communicating over the ARPANET became known as TCP – the Transmission Control Protocol.
- TCP ensured that messages were properly routed form sender to receiver and that they arrived intact. Internet involved, organizations worldwide were implementing their own networks for both
    - intero-organization (within the organization)
    - inter- organization (between organization)
- ARPA accomplished with internet protocol (IP), creating a "network of networks", the current architecture of the internet the combined set of protocols is commonly called TCP/IP.

Usage:

- Internet was limited to universities and research institutions; then the military began using the internet.
- The government decided to allow access to the internet for commercial purposes.

History of World Wide Web:

- The World Wide Web (WWW) allows computer users to locate and view multimedia – based documents over the internet.
- In 1989, Tim Berners – Lee of CERN (The European organization for Nuclear research) began to develop a technology for sharing information via a hyperlinked text documents.
- Berners – Lee called his invention the Hyper Text Markup Language(HTML)
- He also wrote communication protocols to form the backbone of his new information system, which he called the World Wide Web
- The Hyper Text Transfer protocol (HTTP) – a communication protocol used to send information over the web.
- Web use exploded with the availability in 1993 of the mosaic browser, which featured a user – friendly graphical interface.
- Mark Andreesen, whose team at NCSA developed mosaic, went on to found Netscape with initiating the explosive internet economy late 1990's.
- In September 1194, Berners-Lee founded the World Wide Web Consortium (W3c), which is the well-known standard making for free software's all over the world.
- The WWW became commercially viable during 1996-98 when a large number of dot-com companies used it for placing their services on the web.
- In past, most computer applications run on computers that were not connected to one another, whereas today's applications can be written to communicate among the world's computers.
- The internet mixes computing & communication technologies. It makes our work easier. It makes information instantly and conveniently accessible worldwide.
- It enables individuals and small business to get worldwide exposure. It is changing the way business is done.

- People can search for the best prices on virtually any product or service special interest communities can stay in touch with one another.
- HTTP – Communication between a web server and a web browser.
- HTTP is used for sending requests from a web client (browser) to a web server, returning web contents (web pages) from the server back to the client.

1.2 VARIOUS CONCEPTS IN HTML

- Hyper Text Markup Language (HTML) is a tag – based language and these tags are added to the pages, there by instructing the web browser about the format in which the page has to be displayed.
- The source code passed, the browser read web pages serially in the text format.
- HTML was display static content like text, with user specific formatting.

Fundamental HTML elements:

- HTML tags are usually used in pairs.
- All tags are enclosed between the angle brackets (< >)
- The start tag is usually between angle brackets, while the end also has a forwarding slash preceding
  the text (< / >).
- The element content is everything between the start and the end tag.
- Most HTML elements can have their attributes.
- HTML elements have empty content.
- Empty elements are closed in the start tag
- HTML tags are case sensitive.

The types of section are

1. Head section
2. Body section

Example:

```
<Html>
    <Head>
        <Title> Example Of Web </Title>
    </Head>
</Html>
```

Elements for the body section:

- The body tags <BODY> …. </BODY> contain the actual page content.
- All page's text, images, forms links and other HTML come in the body block.
- The body tag has 3 commonly used attributes
    1. BGCOLOR
    2. TEXT
    3. LINK
- BGCOLOR specifies the background color for the page.
- TEXT specifies the color of the text on the page.
- LINK specifies the color of the hyperlink on the page.

HTML named colors:

Color values are represented in either hexadecimal numbers or color names. User can specify only 16 by names. It can specify any color by its hexadecimal values.

- White - #FFFFFF

- Yellow - #FFFF00
- Red - #FF00FF
- Green - #008000
- Blue - #0000FF
- Black - #000000

The default background, text and link colors for the web page are white, black and blue.
The various types of tags in HTML are

Example:

```
<HTML>
    <HEAD>
        <TITLE> HEADING OG VARIOUS SIZE </TITLE>
    </HEAD>
    <BODY>
        <H1> IP </H1>
        <H2> Web </H2>
            .
            .
            .
        <H6> Technology </H6>
    </BODY>
</HTML>
```

Text level elements:

-   is used to insert a single blank space.
-
 indicates the line break sometimes user might want to use is to separate text into paragraphs.
- <P> …. </P> - Paragraph tag. These tags add a blank line before and after the block they enclose. Two line break tag can be used consecutively.
- <B>…..</B> - Bold tag
- <I>…..</I> - Italic tag
- <U>…..</U> - Underline tag
- It can effect as indicated in the modification of the font.

Lists of text:

- The most commonly list are of 2 types
    1. Ordered Lists (OL)
    2. Unordered Lists (UL)

- The ordered list tags <OL>……</OL> create ordered list item.
- The unordered list tags <UL>……</UL> create unordered list item.
- For each list item in the list, within either of these set of tag, user can use the listing tag (LI) – List Item, this tag is used singly and does not have an end tag.
- <HR> is used to draw a horizontal line on the web page requires only a single tag.

Definition List (DL) tag:

- Definition list are also commonly used to display information in the form of definitions which have common use to convey information to users.

- A definition has two parts
    1. A term (DT)
    2. A definition (DD)
- Definition list tag <DL>……</DL> - create a definition list
- Within these tags, user can use the definition term tag <DT> - for each list item term.
- Definition data tag <DD> - for each list item's data.
- Both the <DT> and <DD> tag have no associated end tags.

Font tag:

<Font color = "Red" Face = "Times" size = 5>
All data displayed here is red in color and has the times font and the size is 5
</Font>

Embedding Images:

- Images can be embedded into the web page. To add images as a background for the whole web page, the following format is used,

  <body background = "bg.gif">
- To include the image as a normal figure in the web page the <img> tag is used.

Example:

< img src = "url " height = "144" border = "1" width = "200" alt = "An image is here">
</img>

The other related tag is <map> which is used to create hot spots. The syntax is

<map name = " ">
……
</map>

Frames:

- Frame layout is one in which the browser windows is broken into multiple region called frames.
- Each frame contains different HTML documents.
- The <frameset> tag is a container for frames and replaces the body tag. </frameset>
- <frame> tag is used to place the contents into the frame.

Example:

<frameset rows = "value" cols = "value">

- The attribute rows – The window is to be divided in horizontal stripes.
- The attribute cols – The window is to be divided in vertical stripes.

Tables:

- This form tag is the HTMLs best way of arranging information in space and controlling layout
- Table element to format a table
- The various tag in table as

  <table>……</table>
  align = left, center, right

border = make a border around the table & its cells

- The <tr> element (table row) inserts a row in the table.
- The <td> element (table detail) inserts a cell within a row
- The <th> element (table header) to add headings to the rows and columns of the table.

- <caption> tag is used to be added to row and column headings. By default it will be aligned in center of a table.
- <border> is used for draw a border around the tables and the individual cells. Specify the border width in pixels.
- Color can be added to tables by using the bgcolor = and bordercolor = attributes. These attributes are available in the Table, <tr> and <td> elements; so user can apply colors to all the cell in a table, selected rows and individual cells.

Example:

<table border = “1” cellspacing = “10%” cellpadding = “10%”> …. </table>

Spanning rows:

- The two types of attributes are
  - colspan – to span column
  - rowspan – to span rows

HTML Forms:

- Form provides a way to prompt the user for information and to carry out the actions based on the input.
- A form consists of one or more input controls that the user uses to enter text and select choices.
- Once the user provides the input, the form collects the data and sent it to a destination specified in the form element.
- To carry out the requested action, the server must have a script or other service that corresponds to the destination.
- A form can contain inputs like text fields, check boxes, passwords, radio-buttons, submit buttons and reset buttons.
- The <form> tag is used to create an HTML form.

  <form name=” frm1“ action=”index.jsp “ method=”get|post” >
  </form>
- HTML forms are used to pass the data to the server.

Input Types:

- Text Fields, Check Boxes, Password, Radio Button, Submit Button, Reset Button, Image Based Buttons, Scripted Buttons

The syntax is

<input type = “text | passwd” name =”name” value =”default_value” size=”field size”>

Selection List (<select>):

- This form tag is used to set up a list of choices from
- </select>

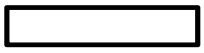
Image Button:

<input type=”image” src=” “ name=”submit” value=”submit”>

Filename:

<input type="file" name="filename" value=" " size=25> Browser

Example:

 Browse

- The form tag can be used in the HTML code.
  <form name=" frm1" action=" hello.jsp " method="get | post" target="response_frame">
  </form>
- The method attribute is used to send the form data to the web server.
- The default attribute value for the method is get which appends the data to the end of the processing script URL.
- If the method has the value post, then the data is sent to the web server as a separate transaction.
- The value post is used when the form data is to be stored in a database or as a processing data in the web server.

1.3 INTERNET PROTOCOLS

- A protocol is a set of rules or an agreement that specifies a common language that computer on a network use for communication with other computers.
- It specifies the conditions under which a particular message should be sent or respond and the particular method of doing it.
- It specifies on how the computer talk with each other.
- The various protocols are
  - Hyper Text Transfer Protocol (HTTP)
  - Simple Mail Transfer Protocol (SMTP)
  - Post Office Protocol Version 3 (POP3)
  - Multipurpose Internet Mail Extension (MIME)
  - File Transfer Protocol (FTP)
  - Internet Protocol (IP)
  - Transport Control Message (TCP)
  - Internet Message Access Protocol (IMAP)

HTTP

- Hyper Text Transfer Protocol (HTTP) is a communication protocol used to send information over the web.
- HTTP protocol is used by the World Wide Web (WWW).
- HTTP defines how messages are formatted and transmitted and what actions web servers and browsers should in response to various commands.
- Example: When you enter a URL in your browser, this actually sends an HTTP command to the web server directing it to fetch and transmit the requested web page.
- It is a standard protocol for communication between the web browsers and the web servers.
- It is a stateless protocol that specifies how a client and a server establish a connection, how the client request data to the server, how the server responds to the client and finally how the connection is closed.
- Sending a request in the form of ACSII string and expects a reply (ASCII).

The basic structures of HTTP communications are:

1. Request Model
2. Response Model

HTTP REQUEST MESSAGE:

- A client sends an HTTP request to server by
  - Clicking a link on the web page
  - Submitting a form
  - Typing a web page address in the browser's address field
- The browser uses the URL (Uniform Resource Locator) to create the request message.

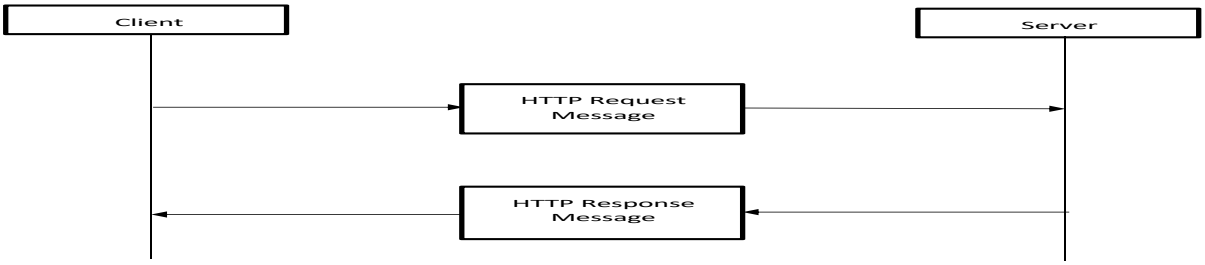


Fig 1.1: HTTP Request Message

The HTTP request message consists of

- Request / start line
- Request header / header field
- Blank line
- Request / message body (optional)

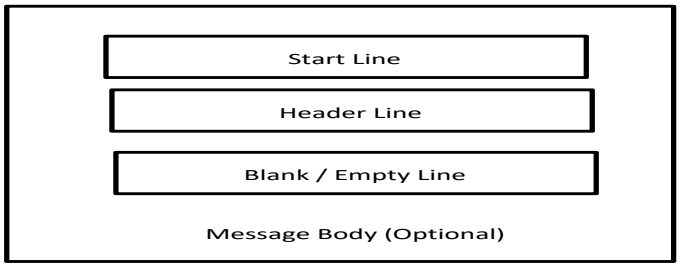


Fig 1.2 HTTP request message

a. Start Line / Request Line

It consists of 3 parts

i. Request Method (Example: GET)
ii. Request URI Portion of Web Address
iii. HTTP Version (Example: HTTP/1.1)
iv.

i. Request Method

The various methods supported by HTTP are:

METHODS	DESCRIPTIONS
GET	Gets a file from the server (or) tells the server the client wants to get some resource.
POST	Sends user information to the server (or) tells the server the client

	wants to get some resources and information will be sent by the client that may modify the request (ex: Submitting the form).
HEAD	Gets information about a file from the server.
PUT	Sends a file to be stored on the server (transfers a file from the client to the server).
DELETE	Deletes a file on the server.
OPTIONS	Requests the available server options.

ii. A Resource Identifier (Request URI)

- Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource on the internet.
- Such identification enables interaction with representations of the resource over a network (WWW) using a specific protocols.
- URI are composed of alpha-numeric names (punctuation characters are permitted) delimited by the character "/"
- URL (Uniform Resource Locator) is used for specifying any kind of information available on the internet.

The four elements of a URL specification are

- Method (Protocol)
- Host (Local hostname or IP address)
- Port (Port number for contacting server)
- Path (Pathname of the resource file)

iii. HTTP Version Identifier

- It specifies the version of the HTTP that the client understands.
- The string starts with the prefix HTTP / and is followed by a version number.
- Example: HTTP/1.1

HTTP RESPONSE MESSAGE:

- A server response is followed by a blank, status line and text of the requested page.

Example:

GET/HTTP/1.0
HTTP/1.0 200 OK
Date: Mon, 12 Dec 2011 11:40:40 GMT
Server: Apache/1.3.3.7 (UNIX)
Last_Modified: Wed, 08 Jan 2003 12:10:10 GMT
E tag: "3f80F-1b6-3e1cb03b"

Server receives a request and uses its URL to decide how to handle it.

- Accept-Range: bytes
- Content-Length: 438
- Connection: close
- Content-Type: text/html

- Etag (Entity Tag) header is used to determine if a cached version of the requested resource is identical to the current version of the resource on the server.
- Content-Type specifies the internet media type of the data conveyed by the HTTP message.
- Content-Length indicates it length in bytes.
- The HTTP/1.1 Web Server publishes its ability to respond to requests for certain byte range of the document – accept ranges bytes.
- Connection: Close is sent in a header. It means that the web server will close the TCP connection immediately after the transfer of this response.

SMTP

- The TCP/IP protocol that supports electronic mail on the internet is called the **Simple Mail Transfer Protocol (SMTP).**
- SMTP is the protocol used by mail servers to exchange email messages.
- It is a system for sending messages to other computer users based on e-mail addresses.

SMTP provides for mail exchanges between users on the same or different computers and supports are

- Sending a single message to one or more recipients.
- Sending messages that include text, audio, video format or graphics file.
- Sending messages to users on networks.

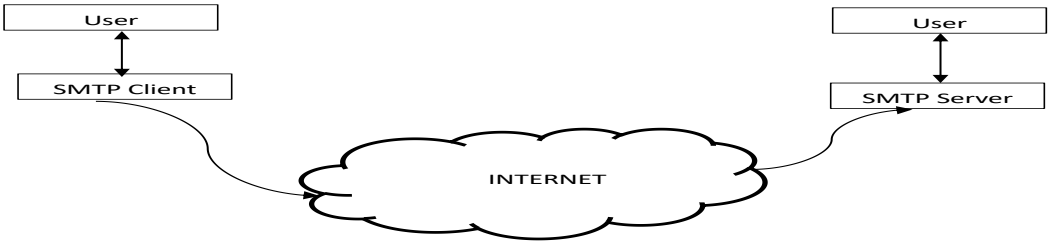


Fig 1.3: SMTP Concept

SMTP client and server has 2 components

a) User Agent (UA)
b) Mail Transfer Agent (MTA)

- The UA prepares the message, create the envelope and puts the message in the envelope.
- The MTA transfers the mail across the internet.

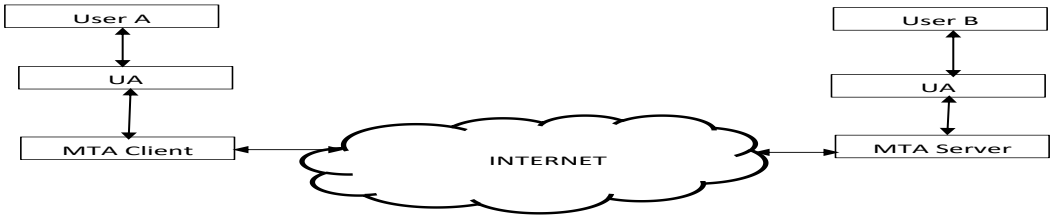


Fig 1.4: MTA transfers the mail across the internet.

- A mail gateway which is a relay MTA that can receive mail prepared by a protocol other than SMTP and transform it to SMTP format before sending it.

- It can also receive mail in SMTP format and change it to another format before sending it.

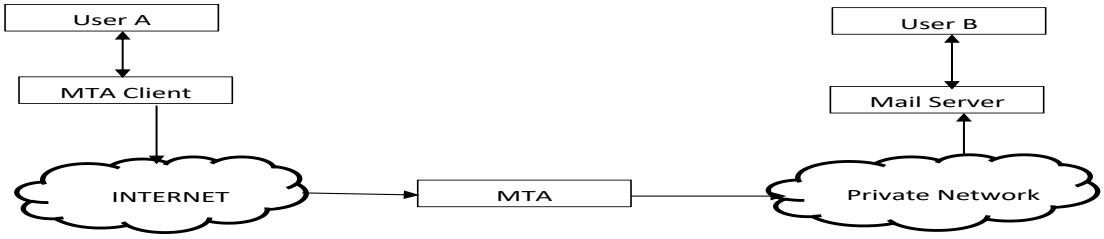


Fig1.5 : Mail gateway

POP3

- Post Office Protocol Version3 (POP3) is used to retrieve e-mail from an internet mailbox.
- POP3 is used to retrieve mail for a single user; typically the POP server has access to a database of email messages created by an SMTP server.
- POP3 is used to download messages from the server.
- POP3 connections require authentication in the form of a secret (i.e.,) shared by the user and the POP server (a password)
- POP3 session passes
  - Authentication
  - Transaction
  - Update
- In the authorization state, the client identifies itself to the server.
- If the authorization is successful, the server opens the client's mailbox and the session enters the transaction state.
- In this state, the client requests the POP3 server provide information or perform an action.
- When it enters the update state, the connection terminates.

POP commands:

- POP commands and replies are formatted as ASCII lines and all replies start with either "+OK" or "-ERR".

  The various POP commands are

Command	Description
USER	Requires a name that identifies the user (Specify username).
PASS	Specify password for the user/server.
STAT	Get mailbox status (no. of messages in the mailbox).
LIST	Get a list of messages and size, one per line, termination line contains a period.
RETR	Retrieves message from mailbox.
DELE	Marks a message for deletion.
LAST	The server returns the highest message no accessed.
RSET	Unmarks all messages marked for deletion.
QUIT	Remove marked messages and close the TCP connection.

- "+OK" is a positive success indicator similar to an ACK message
- The text "-ERR" is a negative success indicator similar to a NAK message

MIME

- MIME is an acronym of Multipurpose Internet Mail extensions.
- It is an international standard that deals with the format of messages exchanged between different e-mail systems.
- It is a specification for formatting non – ASCII messages so that they can be sent over the internet.
- Many e-mail clients now support MIME, which enables them to send and receive graphics, audio and video files via the internet mail system.
- MIME supports messages in character set other than ASCII.
- MIME was defined in 1992 by the internet engineering task force (IETF).
- A new version called s/MIME supports encrypted message.
- The format of Internet mail to allow Non – US – ASCII textual messages, non – textual messages, multipart message bodies and non – US – ASCII information in message headers.

Need for MIME:

- Messages contain only ASCII characters.
- Messages with only 1000 characters.
- Messages should not exceed certain length.

Features of MIME:

MIME allow mail messages to have

- Multiple objects in a single message.
- Message with any no. of lines or unlimited overall length.
- Characters other than ASCII, allowing non – English messages.
- Multi – font messages.
- Binary or application specific files.
- Images, audio, video and multipart messages.

- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet.
- The message at the receiving side is transformed back to the original data.
- NVT→(Network Virtual Terminal)

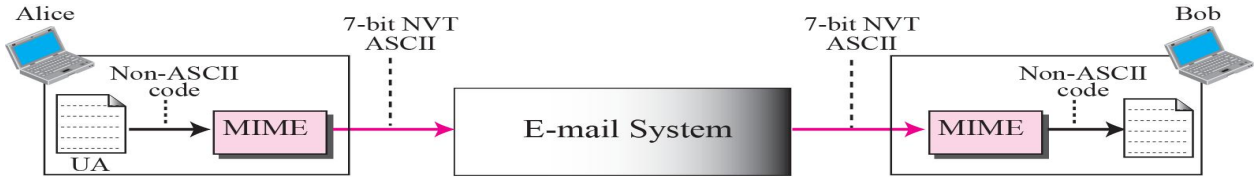


Fig1.6: MIME

MIME Headers:

The 5 header fields to internet e-mail messages.

1. MIME version
2. Content type
3. Content transfer encoding
4. Content id
5. Content description

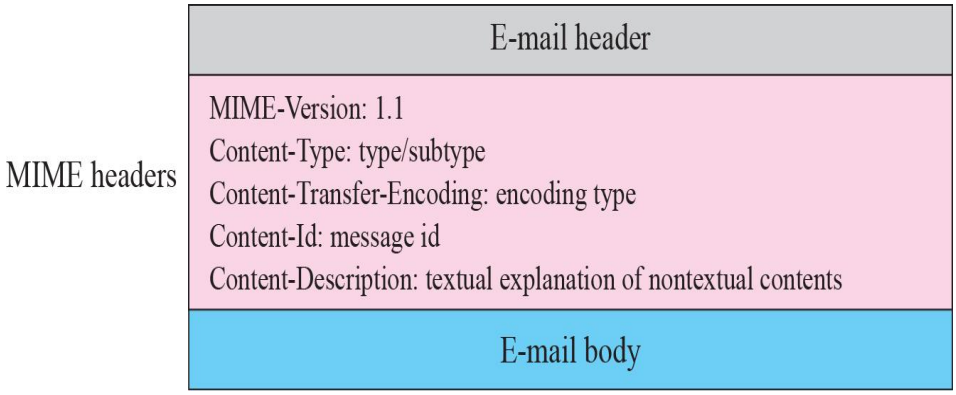


Fig 1.7: MIME Headers

1. **MIME-Version:**
   - This header defines the version of MIME used.
   - The current version is 1.1

     MIME-Version: 1.1

2. **Content type:**
   - This header defines the type of data used in the body of the message.
   - The content type and the content subtype are separated by a slash.
   - Depending on the subtype, the header may contain other parameters.

     Content-Type : < type / sub-type >

IMAP

- IMAP stands for Internet Message Access protocol.
- This protocol is used to access the messages in e-mail or electronic bulletin board that are in mail server
- It is an application layer Internet protocol that allows e-mail client to access e-mail on a remote mail server
- The current version IMAP4 is defined by RFC3501.
- IMAP server on well-known port no.143

The objectives of IMAP are

1. Compatible with internet messaging standards Ex: MIME
2. Allow message access from multiple computers.
3. It supports for online, offline and disconnected access nodes.
4. Support for concurrent access to shared mailboxes.
5. Client software needs no knowledge about the servers file store format.

- The feature of IMAP is the mail messages remain on the server, instead of being downloaded to a computer.
- Checking the mail with a client or web-based environment using the protocol.
- IMAP supports the use of folders for mail organization, but instead of organizing the messages on the local computer, these folders are kept on the server.
- IMAP is quicker access to mail.
- The message headers are initially downloaded so the user can choose to download, open and read only this message.

- Using IMAP and saving messages on the server is that the user will be restricted.

The protocols includes operations for

- Creating mailboxes.
- Deleting mailboxes.
- Renaming mailboxes.
- Checking for new messages.
- Permanently removing messages.
- Setting and clearing flags.
- Selective fetching of message attributes, text and portion of efficiency.

1.4 DNS

Domain Name System (DNS)

- To identify an entity, TCP/IP protocols use the IP address which uniquely identifies the connection of a host to the internet.
- We prefer to use names instead of address.
- We need a system that can map a name to an address and conversely an address to a name. In TCP/IP, this is the domain name system (DNS).
- What internet users use to reference anything by the name of the internet?
- The mechanism by which Internet software translates name to addresses.

The Namespace

- The namespace is the structure of the DNS database.
  - An inverted tree with the root node at the top.
- Each node has a label.
  - The root node has a null node written as " ".

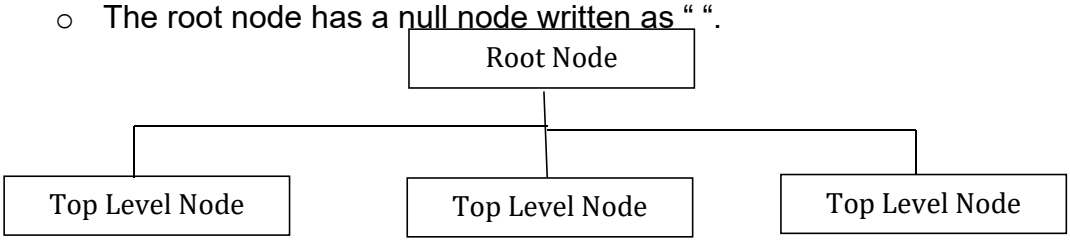


Fig 1.8 : Structure of the DNS Database

DNS in the Internet:

- A domain name is the sequence of labels from a node to the root separated by dots (".""s), read left to right.
  - The namespace has a maximum depth of 127 levels.
  - Domain names are limited to 64 characters in length.
- A node's domain name identifies its position in the namespace.
- DNS is a protocol that can be used in different platforms.

In the internet, the domain namespace (tree) is divided onto 3 different sections.

- Generic domains
- Country domains
- Inverse domains

1. Generic Domains:

- The generic domains define registered host according to their generic behavior.

- Each node in the tree defines a domain, which is an index to the domain namespace database.
- It represents as 3 character labels.
- Ex: com, edu.

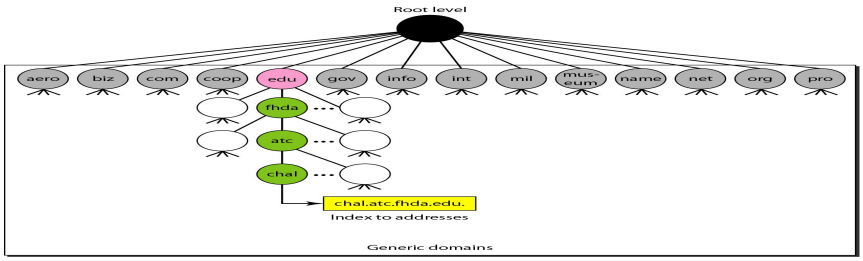


Fig 1.9 : Generic Domain

Sub-Domain

- One domain is a sub domain of another if its apex node is a descendant of the others apex node.
- One domain is a sub domain of another if its domain name is in the other's domain name.
- Ex: sales.google.com is a subdomain of
  - google.com
  - com
- Administrators can create subdomains to group hosts.
- The parent domains retain links to the delegated subdomains.

1.5 WEB BROWSERS AND WEB SERVERS

WEB BROWSERS:

- The internet is an essential medium for communicating and interacting with people worldwide.
- A web browser is used to display web pages. Common web browsers are Netscape navigator and internet explorer.
- The web browser is the interpreter of our web sites.
- Web browsers are software programs that allow users to access the web content.
- Millions of people use web browsers to access the tremendous amount of information available on the web and to store or exchange this content with other users.
- The popular web browsers are
  - Microsoft's Internet Explorer
  - Mozilla's Firefox
  - Apple's Safari
  - Opera Software's Opera
  - Google Chrome
- Web browsers, a software application used to locate, retrieve and also display content on the worldwide web, including web pages, images, videos and other files.
- As a client/server model, the browser is the client run on a computer that contacts the web server and request information.

- The web server sends the information back to the web browser which displays the results on the computer or other internet enabled device that supports a browser.
- Browsers are fully functional software suites that can interpret and display HTML web pages, applications, java scripts, AJAX and other content hosted on web servers.
- Many browsers offer plug-in which extend the capabilities of a browser so it can display multimedia information (including sound and video) or the browser can be used to perform tasks such as video conferencing, to design web pages or add ant phishing filters and other security features to the browser.

Web Servers:

- Web servers are computers that deliver (serves up) web pages.
- Every web server has an IP address and possibly a domain name.
- Ex: if you enter the URL http://www.google.com/index.html in your web browser. This sends a request to the web server whose domain name is google.com. The server then fetches the page named index.html and sends it to your browser.
- Any computer can be turned into a web server by installing server software and connecting the machine to the internet.
- Web server software applications including public domain software from NCSA (National Computer Security Association) and Apache web server.
- The web browser includes 3 categories as
    - Static web document
    - Dynamic web document
    - Active web document

Static Documents:

- Static documents are fixed content documents that are created and stored in a server.
- The client can get only a copy of the document.
- The content in the server can be changed but the user cannot change it.
- When a client accesses the document, a copy of the document is sent. The user then uses a browser to display the content.
- Ex: HTML web page file

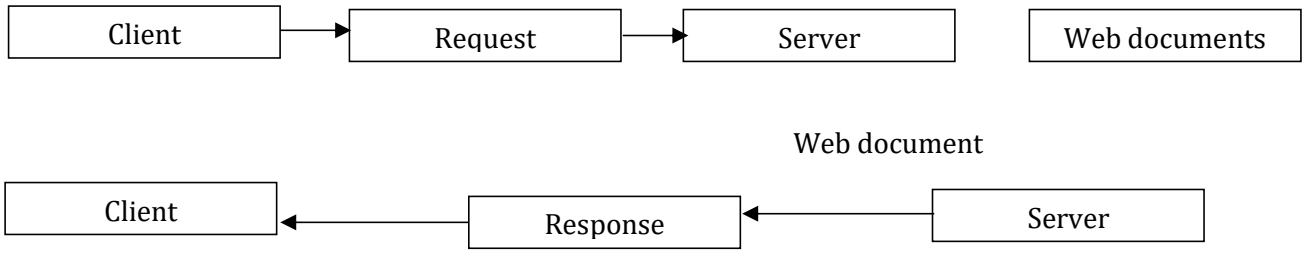
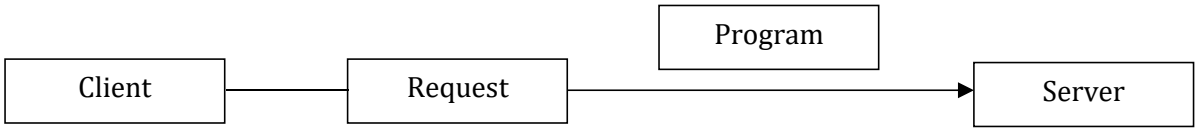


Fig 1.10: Static Document

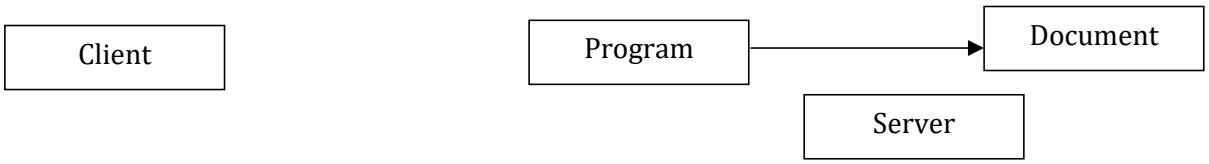
Dynamic Document:

- A dynamic document is created by a web server whenever a browser requests the document.

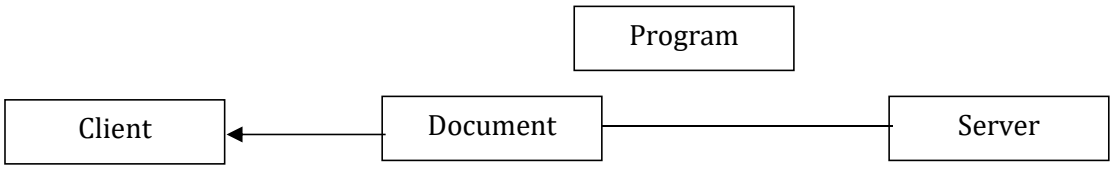
- When a request arrives, the web server runs an application program that creates the dynamic document.
- The server returns the output of the program as a response to the browser that requested the document.
- Because a fresh document is created for each request, the contents of a dynamic document can vary from one request to another.
- Ex: Dynamic document is getting the time and date from the server.
- Time and date are kinds of information that is dynamic and they change from moment to moment.
- Cricket refresh button.



a. Request for running a program



b. Running the program and creating the document



c. Response

Fig 1.10: Dynamic Document

- A server that handles dynamic documents as
  - The server examines the URL to find if it defines a dynamic document.
  - If the URL defines a dynamic document, the server executes the program.
  - The server sends the output of the program to the client (browser).

Common Gateway Interface (CGI):

- It creates and handles dynamic documents.
- CGI is a set of standards that defines how a dynamic document should be written, how input supplied to the program and how the output result is used.
- CGI program is a gateway that can be used to access other resources such as database, graphic, packages.

Active Document:

- We need a program to be a run at the client side.
- Ex: we want to run a program that creates animated graphics on the screen or interacts with the user.

- The program can be run at the client side where the animation or interaction taken place.
- An active document in the server is stored in the form of binary code.
- Ex: Java Applet

1.6 CSS

Introduction to (CSS):

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External style sheets are stored in **CSS files.**
- CSS are rules or styles for organizing the layout of an HTML document including its color, typefaces, margins, links and other formatting elements.
- Style sheets make it easier to create an index because indexing software has only to read the structural elements rather than full content page.
- User includes multiple set of style sheet information.
- It contains rules, composed of selectors and declarations that define how styles are applied.
- The selector (HTML element, class name or ID name) is the link between the HTML document and the style.
    - HTML element tags
    - Attributes (class, ID name)

Rule or Syntax:

A CSS rule set consists of a selector and a declaration block:

Selector { property1: value1; property2: value2; } or **tagname {style_attribute value:}**

```
h1      { color      : blue;
          font-size  : 12px;  }
```

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a property name and a value, separated by a colon.

Example 1:

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly braces:

```
p {
    color:red;
    text-align:center;
  }
```

Example 2:

```
<html>
  <head>
    <style type="text/css" >
      body {background-color:yellow;}
      h1   {font-size:36pt;}
```

```
            h2   {color:blue;}
            p    {margin-left:50px;}
        </style>
</head>
<body>
        <h1>This header is 36 pt</h1>
        <h2>This header is blue</h2>
        <p>This paragraph has a left margin of 50 pixels</p>
</body>
</html>
```

- To assign more than one kind of style information at the same time, separate the styles with semicolon.

1.7 JavaScript

JavaScript is a scripting language mainly used for writing dynamic Web pages. When a script written in JavaScript is embedded in a Web page, it will be executed by the Web browser on the client machine.

JavaScript supports functions as first-class functions - Functions are really objects. Like regular objects, functions can be created during execution, stored in data structure, and passed to other functions as arguments.

- JavaScript is a client – side scripting language for the World Wide Web that is similar to the syntax of the Java programming language.
- JavaScript is designed to provide limited programming functionality.

A simple JavaScript program:

```
<html>
<head>
<Title> Hello World  </Title>
</head>
<body>
<script language="javascript">
document.write("Hello,World wide web");
</script>
</body>
 </html>
```

Client-side JavaScript:

- Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.
- The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.
- JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript:

The merits of using JavaScript are

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript:

The JavaScript as a full-fledged programming language. It lacks the following important features

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.
- JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

1.8 JavaScript Operators

JavaScript

JavaScript is a scripting language mainly used for writing dynamic Web pages. When a script written in JavaScript is embedded in a Web page, it will be executed by the Web browser on the client machine.

JavaScript supports functions as first-class functions - Functions are really objects. Like regular objects, functions can be created during execution, stored in data structure, and passed to other functions as arguments.

- JavaScript is a client – side scripting language for the World Wide Web that is similar to the syntax of the Java programming language.
- JavaScript is designed to provide limited programming functionality.

A simple JavaScript program:

```
<html>
<head>
<Title> Hello World  </Title>
</head>
<body>
<script language="javascript">
document.write("Hello,World wide web");
</script>
</body>
 </html>
```

1.9 JavaScript in Perspective

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow

client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript Syntax

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags.
- JavaScript can be placed in the <body> and the <head> sections of an HTML page.
- The **<script> tag** alerts the browser program to start interpreting all the text between these tags as a script.

A simple syntax of your JavaScript will appear as follows.

```
<script…>
    JavaScript code
</script>
```

The script tag takes two important attributes −

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be Javascript. Although recent versions of HTML have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

The JavaScript segment will look like –

```
<script language="Javascript"  type="text/Javascript">
    JavaScript code
</script>
```

Example Program:

```
<html>
  <body>
    <script language="Javascript" type="text/Javascript">
            document.write("Hello World!")
    </script>
  </body>
</html>
```

Output:

This code will produce the following result –

Hello World!

1.10 JavaScript Variables:

- JavaScript variables are containers for storing data values.
- You can place data into these containers and then refer to the data simply by naming the container.
- Before you use a variable in a JavaScript program, you must declare it.

Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
    var money;
    var name;
```

```
</script>
```

You can also declare multiple variables with the same **var** keyword as follows –

```
<script type="text/javascript">
    var money, name;
</script>
```

JavaScript Identifiers:

- All JavaScript **variables** must be **identified** with **unique names**.
- These unique names are called **identifiers**.
- JavaScript identifiers are case-sensitive.
- Identifiers can be short names (like x and y), or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter.
- Names can also begin with $ and _.
- Names are case sensitive (y and Y are different variables).
- Reserved words (like JavaScript keywords) cannot be used as names.

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
    var name = "Ali";
    var money;
    money = 2000.50;
</script>
```

1.11 JavaScript Data Types:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type.

It can hold any type of values such as numbers, strings etc.

For example:

- var a=40; //holding number
- var b="Rahul"; //holding string

1. JavaScript primitive data types:

There are five types of primitive data types in JavaScript. They are as follows:

(i) JavaScript String:

- A string (or a text string) is a series of characters like "John Doe".

- Strings are written with quotes. You can use single or double quotes:

Example:

```
var carName = "Volvo XC60";          //Using double quotes
var carName = 'Volvo XC60';   // Using single quotes
```

(ii) Javascript Number:

- JavaScript has only one type of numbers.
- Numbers can be written with, or without decimals:

Example:

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

(iii) JavaScript Boolean:

- Booleans can only have two values: true or false.
- Booleans are often used in conditional testing.

Example:

```
var x = true;
var y = false;
```

(iv) Undefined:

In JavaScript, a variable without a value, has the value **undefined**. The typeof is also **undefined**.

Example:

```
var person;                 // Value is undefined, type is undefined
```

Any variable can be emptied, by setting the value to **undefined**. The type will also be **undefined**.

(v) Empty Values

- An empty value has nothing to do with undefined.
- An empty string variable has both a value and a type.

Example:

```
var car = "";               // The value is "", the typeof is string
```

(vi) Null:

- In JavaScript null is "nothing". It is supposed to be something that doesn't exist.
- Unfortunately, in JavaScript, the data type of null is an object.

You can empty an object by setting it to null:

Example:

```
var person = null;          // Value is null, but type is still an object
```

2. JavaScript non-primitive data types:

The non-primitive data types are as follows:

(i) JavaScript Object:

- JavaScript objects are written with curly braces.
- Object properties are written as name:value pairs, separated by commas.

Example:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

(ii) JavaScript Array:

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Example:

```
var cars = ["Saab", "Volvo", "BMW"];
```

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

(iii) JavaScript RegExp:

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax:

*/*pattern*/*modifiers*;*

Example:

```
var patt = /w3schools/i
```

- **/w3schools/i** is a regular expression.
- **w3schools** is a pattern (to be used in a search).
- **i** is a modifier (modifies the search to be case-insensitive).

1.12 JavaScript Statements

Conditional statements are used to perform different actions based on different conditions.

In JavaScript we have the following conditional statements:

i. Use **if** to specify a block of code to be executed, if a specified condition is true.
ii. Use **else** to specify a block of code to be executed, if the same condition is false.
iii. Use **else if** to specify a new condition to test, if the first condition is false.
iv. Use **switch** to specify many alternative blocks of code to be executed.

The if Statement:

Use the **if** statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax:

```
if (condition)
{
   block of code to be executed if the condition is true
}
```

(ii) The else Statement:

Use the **else** statement to specify a block of code to be executed if the condition is false.

Syntax:

```
if (condition)
{
   block of code to be executed if the condition is true
}
else
{
   block of code to be executed if the condition is false
}
```

(iii) The else if Statement:

Use the **else if** statement to specify a new condition if the first condition is false.

Syntax:

```
if (condition1)
{
   block of code to be executed if condition1 is true
}
else if (condition2)
{
   block of code to be executed if the condition1 is false and condition2 is
true
}
else
 {
   block of code to be executed if the condition1 is false and condition2 is
false
}
```

Arithmetic Operators:

Arithmetic operators are used to perform arithmetic on numbers (literals or variables).

Operator	Description
+	Addition

-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Comparison Operators:

JavaScript supports the following comparison operators.

Operator	Description
= =	Equal
!=	Not Equal
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to

Logical (or Relational) Operators:

JavaScript supports the following logical operators.

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

Bitwise Operators:

JavaScript supports the following bitwise operators.

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise Not
<<	Left Shift
>>	Right Shift
>>>	Right shift with Zero

Assignment Operators:

JavaScript supports the following assignment operators.

Operator	Description
=	x=y
+=	x+=y

-=	x-=y
*=	x *= y
/=	x /= y
%=	x %= y

Miscellaneous Operator:

The two Miscellaneous operators are

- conditional operator (? :)
- typeof operator.

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

S.No	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

typeof operator:

- The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.
- The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.
- The list of the return values for the **typeof** Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

1.13 JavaScript Literals

String literals:

A **string literal** is zero or more characters, either enclosed in single quotation (') marks or double **quotation** (") marks. You can also use + operator to join strings. The following are the examples of string literals:

i. string1 = "w3resource.com"
ii. string1 = 'w3resource.com'
iii. string1 = "1000"

iv. string1 = "google" + ".com"

Array literals

In Javascript an **array literal** is a list of expressions, each of which represents an array element, enclosed in a pair of square brackets ' [ ] ' .

When an array is created using an array literal, it is initialized with the specified values as its elements, and its length is set to the number of arguments specified. If no value is supplied it creates an empty array with zero length.

Creating an empty array :

```
var fruits = [ ];
```

Creating an array with four elements

```
var fruits = ["Orange", "Apple", "Banana", "Mango"]
```

Integer literals:

An **integer** must have at least one digit (0-9).

- No comma or blanks are allowed within an integer.
- It does not contain any fractional part.
- It can be either positive or negative, if no sign precedes it is assumed to be positive.

Floating pointing literals:

A floating number has the following parts.

- A decimal integer.
- A decimal point ('.').
- A fraction.
- An exponent.

The exponent part is an "e" or "E" followed by an integer, which can be signed (preceded by "+" or "-").

Boolean literals:

The Boolean type has two literal values:

- true
- false

Object literals:

An **object literal** is zero or more pairs of comma separated list of property names and associated values, enclosed by a pair of curly braces.

In JavaScript an object literal is declared as follows:

1. An object literal without properties:

```
var userObject = {}
```

2. An object literal with a few properties :

```
var student = {
First-name : "Suresy",
Last-name : "Rayy",
Roll-No : 12
};
```

1.14 JavaScript Functions

- A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes.
- Functions allow a programmer to divide a big program into a number of small and manageable functions.
- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

JavaScript Function Syntax:

- A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs.
- The parentheses may include parameter names separated by commas:(***parameter1, parameter2, ...***)

The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3)
 {
   code to be executed
 }
```

- Function **parameters** are the **names** listed in the function definition.
- Function **arguments** are the real **values** received by the function when it is invoked.
- Inside the function, the arguments behave as local variables.

Function Invocation:

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self-invoked)

Function Return:

- When JavaScript reaches a **return statement**, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**. The return value is "returned" back to the "caller":

Example:

Calculate the product of two numbers, and return the result:

```
var x = myFunction(4, 3);        // Function is called, return value will end up in x

function myFunction(a, b) {
   return a * b;                 // Function returns the product of a and b
}
```

The result in x will be:

12

Calling Function:

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
  <head>
       <script type="text/javascript">
       function sayHello()
       {
          document.write ("Hello there!");
       }
     </script>
  </head>
  <body>
     <p>Click the following button to call the function</p>
      <form>
           <input type="button" onclick="sayHello()" value="Say Hello">
     </form>
          <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Output:

Click the following button to call the function

Use different parameters inside the function and then try...

Hello there!

Function Parameters:

There is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Example:

```
<html>
  <head>
    <script type="text/javascript">
      function sayHello(name, age)
      {
         document.write (name + " is " + age + " years old.");
      }
    </script>
   </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body></html>
```

Output:

Click the following button to call the function

Say Hello

Use different parameters inside the function and then try...

Zara is 7 years old.

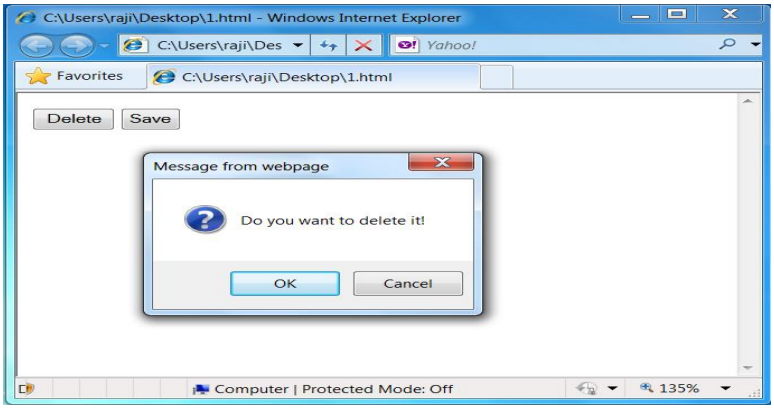
Function arguments:

Through variable you can pass argument to function. The output of the function looks on the arguments given by you.

Example:

```
<html>
<head>
  <script language="javascript">
        function myfunction(text)
        {
           confirm(text)
        }
  </script>
</head>
<body>
        <form>
                <input type="button" onclick="myfunction('Do you want to delete it!')" value="Delete">
                <input type="button" onclick="myfunction('Do you want to save it!')" value="Save">
        </form>
</body>
</html>
```

Output:



1.15 JavaScript objects

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers

- **Encapsulation** − the capability to store related information, whether data or methods, together in an object.
- **Aggregation** − the capability to store one object inside another object.
- **Inheritance** − the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.

- **Polymorphism** − the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties:

The name:values pairs (in JavaScript objects) are called **properties**.
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Object Methods

- Methods are **actions** that can be performed on objects.
- Methods are stored in properties as **function definitions**.

property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

Object Definition:

You define (and create) a JavaScript object with an object literal:

Example:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

Accessing Object Properties:

You can access object properties in two ways:

Syntax:

objectName.propertyName
(or)
objectName["propertyName"]

Accessing Object Methods:

You access an object method with the following syntax:

Syntax:

objectName.methodName()

Example:

name = person.fullName();

The Object() Constructor:

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()**to build the object. The return value of the **Object()** constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

1.16 JavaScript Arrays

- An array is a special variable, which can hold more than one value at a time.
- JavaScript arrays are used to store multiple values in a single variable.

Creating an Array:

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

var *array-name* = [*item1*, *item2*, ...];

Example:

```
var cars = ["Saab", "Volvo", "BMW"];
```

Using the JavaScript Keyword new:

The following example also creates an Array, and assigns values to it:

Example:

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Access the Elements of an Array:

An array element by referring to the **index number**.

This statement accesses the value of the first element in cars:

```
var name = cars[0];
```

Arrays are Objects:

- Arrays are a special type of objects. The **typeof** operator in JavaScript returns "object" for arrays.
- But, JavaScript arrays are best described as arrays.

Arrays use **numbers** to access its "elements". In this example, person[0] returns John:

Array:

```
var person = ["John", "Doe", 46];
```

Objects use **names** to access its "members". In this example, person.firstName returns John:

Object:

```
var person = {firstName:"John", lastName:"Doe", age:46};
```

Array Properties and Methods:

The real strength of JavaScript arrays are the built-in array properties and methods:

Examples:

```
var x = cars.length;          // The length property returns the number of elements in cars
var y = cars.sort();          // The sort() method sort cars in alphabetical order
```

The length Property:

The **length** property of an array returns the length of an array (the number of array elements).

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;                // the length of fruits is
 4
```

Adding Array Elements:

The easiest way to add a new element to an array is using the push method:

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Lemon");         // adds a new element (Lemon) to fruits
```

New element can also be added to an array using the length property:

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Lemon";     // adds a new element (Lemon) to fruits
```

Adding elements with high indexes can create undefined "holes" in an array:

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[10] = "Lemon";         // adds a new element (Lemon) to fruits
```

1.17 JavaScript Built-in Objects

(i) Math Object:

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical methods.

Example:

```
Math.min(0, 150, 30, 20, -8, -200);      // returns -200
```

String HTML Wrapper Methods:

The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color

fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text

(iii) Date object:

- JavaScript's Date object provides methods for date and time manipulations.
- Date objects are created with new Date().
- Date and time processing can be performed based on the computer's local time zone or based on World Time Standard's Coordinated Universal Time (UTC) - formerly called Greenwich Mean Time (GMT).

There are four ways of instantiating a date:

1. var d = new Date();
2. var d = new Date (milliseconds);
3. var d = new Date (dateString);
4. var d = new Date (year, month, day, hours, minutes, seconds, milliseconds);

Date Methods:

- When a Date object is created, a number of **methods** allow you to operate on it.
- Date methods allow you to get and set the year, month, day, hour, minute, second, and millisecond of objects, using either local time or UTC (universal, or GMT) time.

(iv) Boolean objects:

- JavaScript Boolean objetcs can have one of two values: true or false.
- These wrappers define methods and properties useful in manipulating Boolean values.

```
var b = new Boolean ( booleanValue );
```

(v) Number objects:

- JavaScript has only one type of number.
- Numbers can be written with, or without, decimals:

(vi) Array object:

- The Array object is used to store multiple values in a single variable:
- var cars = ["Saab", "Volvo", "BMW"];

(vii) Regular Expression object:

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax:

```
var patt=new RegExp(pattern,modifiers);
            (or)
var patt=/pattern/modifiers;
```

UNIT-2

SERVER SIDE PROGRAMMING

Servlet technology is used to create web application. Servlet technology is robust and scalable because of java language. Before servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language. But there was many disadvantages of this technology.

There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

2.1 SERVLET

Servlet can be described in many ways, depending on the context.

- Servlet is a technology i.e. used to create web application.
- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an interface that must be implemented for creating any servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming request. It can respond to any type of requests.
- Servlet is a web component that is deployed on the server to create dynamic web page.

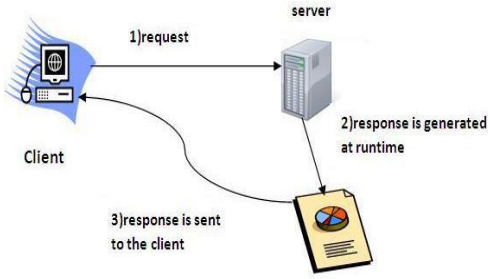


Fig 2.1 Servlet

Web Application

A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter etc. and other components such as HTML. The web components typically execute in Web Server and respond to HTTP request.

CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.

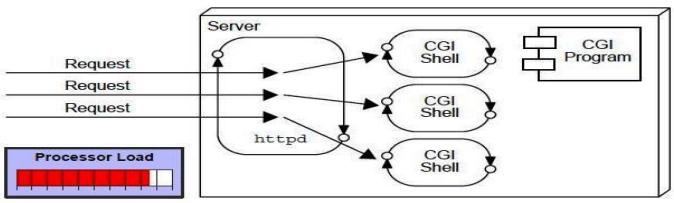


Fig 2.2 Common Gateway Interface

Disadvantages of CGI

There are many problems in CGI technology:

1. If number of client's increases, it takes more time for sending response.
2. For each request, it starts a process and Web server is limited to start processes.
3. It uses platform dependent language e.g. C, C++, Perl.

Advantage of Servlet

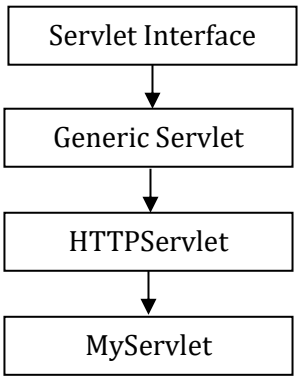
There are many advantages of servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.

The basic benefits of servlet are as follows:

- **Better Performance:** because it creates a thread for each request not process.
- **Portability:** because it uses java language.
- **Robust:** Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
- **Secure:** because it uses java language.

2.2 ARCHITECTURE OF SERVLET

A Servlet is a class, which implements the javax.servlet.Servlet interface. However instead of directly implementing the javax.servlet.Servlet interface we extend a class that has implemented the interface like javax.servlet.GenericServletor javax.servlet.http.HttpServlet.



Servlet Execution

This is the process of a servlet execution takes place when client (browser) makes a request to the webserver.

Fig 2.3 Servlet Execution

Servlet architecture includes:

a) Servlet Interface

To write a servlet we need to implement Servlet interface. Servlet interface can be implemented directly or indirectly by extending **GenericServlet** or **HttpServlet** class.

b) Request handling methods
There are 3 methods defined in Servlet interface: **init(), service() and destroy()**.
The first time a servlet is invoked, the init method is called. It is called only once during the lifetime of a servlet.
The Service method is used for handling the client request. As the client request reaches to the container it creates a thread of the servlet object, and request and response object are also created. These request and response object are then passed as parameter to the service method, which then process the client request. The service method in turn calls the doGet or doPost methods (if the user has extended the class from HttpServlet).

c) Number of instances

Basic Structure of a Servlet

```
Public class firstServlet extends HttpServlet
{
  publicvoidinit()
  {
            /* Put your initialization code in this method,
              * as this method is called only once */
   }
  publicvoid service()
  {
            // Service request for Servlet
   }
  publicvoid destroy()
  {
        // For taking the servlet out of service, this method is called only once
  }
}
```

2.3 Life Cycle of Servlet

The javax.servlet.Servlet interface defines the life cycle methods of servlet such as init(), service() and destroy(). The Web container invokes the init(), service() and destroy() methods of a servlet during its life cycle. The sequence in which the Web container calls the life cycle methods of a servlet is:

1. The Web container loads the servlet class and creates one or more instances of the servlet class.
2. The Web container invokes init() method of the servlet instance during initialization of the servlet. The init() method is invoked only once in the servlet life cycle.
3. The Web container invokes the service() method to allow a servlet to process a client request.
4. The service() method processes the request and returns the response back to the Web container.
5. The servlet then waits to receive and process subsequent requests as explained in steps 3 and 4.
6. The Web container calls the destroy() method before removing the servlet instance from the service. The destroy() method is also invoked only once in a servlet life cycle.

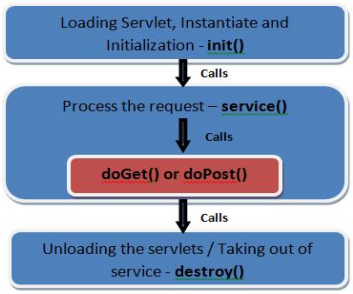


Fig 2.4 Web Container

The init() Method

- The init() method is called during initialization phase of the servlet life cycle. The Web container first maps the requested URL to the corresponding servlet available in the Web container and then instantiates the servlet.
- The Web container then creates an object of the ServletConfig interface, which contains the startup configuration information, such as initialization parameters of the servlet. The Web container then calls the init() method of the servlet and passes the ServletConfig object to it.

- The init() method throws a ServletException if the Web container cannot initialize the servlet resources. The servlet initialization completes before any client requests are accepted. The following code snippet shows the init() method:

```
Public class ServletLifeCycle extends HttpServlet
{
      static int count;
      public void init (ServletConfig config) throws ServletException
      {
            count=0;
      }
}
```

The service() Method

The service() method processes the client requests. Each time the Web container receives a client request, it invokes the service() method. The service() method is invoked only after the initialization of the servlet is complete. When the Web container calls the service() method, it passes an object of the ServletRequest interface and an object of the ServletResponse interface. The ServletRequest object contains information about the service request made by the client. The ServletResponse object contains the information returned by the servlet to the client. The following code snippet shows the service() method:

```
public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException
```

The service() method throws ServletException exception when an exception occurs that interferes with the normal operation of the servlet. The service() method throws IOException when an input or output exception occurs.

The service() method dispatches a client request to one of the request handler methods of the HttpServlet interface, such as the doGet(), doPost(), doHead() or

doPut(). The request handler methods accept the objects of the HttpServletRequest and HttpServletResponse as parameters from the service() method.

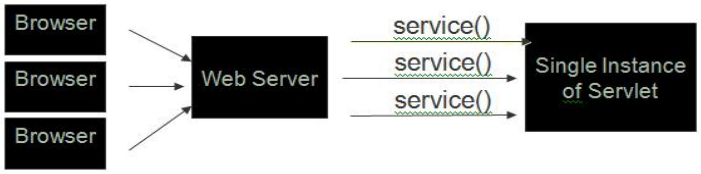


Fig 2.5 Life Cycle of Servlet

The destroy() Method

The destroy() method marks the end of the life cycle of a servlet. The Web container calls the destroy() method before removing a servlet instance from the service.

The Web container calls the destroy() method as

- The time period specified for the servlet has elapsed. The time period of a servlet is the period for which the servlet is kept in the active state by the Web container to service the client request.
- The Web container needs to release servlet instances to conserve memory.
- The Web container is about to shut down.

In the destroy() method we can write the code to release the resources occupied by the servlet. The destroy() method is also used to save any persistent information before the servlet instance is removed from the service.

The following code snippet shows the destroy() method:

```
public void destroy();
```

2.4 GENERIC SERVLET AND HTTP SERVLET

Generic Servlet:

GenericServlet class defines a protocol-independent (HTTP-less) servlet. However while building a website or an online application, you may want to have Http protocol, in that case extend HttpServlet instead of GenericServlet class.

The blue print of GenericServlet class:

```
public abstract class GenericServlet extends java.lang.Object implements Servlet,
ServletConfig,java.io.Serializable
```

An abstract class and it extends Object class & implements Servlet, ServletConfig and Serializable interfaces.

Methods of Generic Servlet class

- **void destroy()**: It is called by servlet container to indicate that the servlet life cycle is finished.
- **java.lang.String getInitParameter (java.lang.String name)**: It returns the value of the specified initialization parameter. If the parameter does not exist then it returns null.
- **java.util.Enumeration getInitParameterNames()**: This method returns all the initialization parameters in form of a String enumeration. By using enumeration methods we can fetch the individual parameter names.
- **ServletConfig getServletConfig()**: It returns this servlet's ServletConfig object.
- **ServletContext getServletContext()**: It returns a reference to the ServletContext in which this servlet is running.

- **java.lang.StringgetServletInfo()**: It returns information about the servlet, such as author, version, and copyright.
- **java.lang.StringgetServletName()**: This method returns the name of this servlet instance.
- **void init()**: This is the first method of servlet life cycle and it is used for initializing a servlet.
- **void init(ServletConfig config)**: Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
- **void log (java.lang.Stringmsg)**: Writes the specified message to a servlet log file, prefixed with servlet's name.
- **void log (java.lang.String message, java.lang.Throwable t)**
- **abstract void service (ServletRequest req, ServletResponse res)**: It is called by the servlet container to allow the servlet to respond to the client's request.

Example of Generic Servlet
index.html

```
<a href="welcome"> Click to call Servlet </a>
```

ExampleGeneric.java

```
import java.io.*;
import javax.servlet.*;
public class ExampleGeneric extends GenericServlet
{
   public void service(ServletRequest req , ServletResponse res) throws IOException,
   ServletException
   {
        res.setContentType("text/html");
        PrintWriterpwriter=res.getWriter();
        pwriter.print("<html>");
        pwriter.print("<body>");
        pwriter.print("<h2> Generic Servlet Example </h2>");
        pwriter.print("</body>");
        pwriter.print("</html>");
   }
}
```

web.xml

```
<web-app>
    <servlet>
        <servlet-name> Beginners book </servlet-name>
        <servlet-class> /ExampleGeneric </servlet-class>
   </servlet>
    <servlet-mapping>
        <servlet-name> Beginners book </servlet-name>
        <url-pattern> welcome </url-pattern>
   </servlet-mapping>
</web-app>
```

HTTP Servlet:

The Http Servlet class implements Serializable interface and extends Generic Servlet class. It provides the definition of HTTP specific methods such as doGet(), doPost(), doTrace() etc.

Methods

The methods of HttpServlet class are as follows:

1. **protected void doGet (HttpServletRequest request, HttpServletResponse response)**: It handles the GET request. It is invoked by the web container.
2. **protected void doPost (HttpServletRequest request, HttpServletResponse response)**: Similar to doGet() method, it also gets invoked by the web container and it handles the POST request.
3. **public void service (ServletRequest request , ServletResponse response)**: There are two types of service method, one is public and other one is protected. The purpose of this public method is to dispatch the request to the protected service method by converting the request and response object into HTTP type.
4. **protected void service (HttpServletRequest request , HttpServletResponse response)**: This method receives the request from the public service() method. It dispatches the request to the doXXX() method depending on the incoming http request type.
5. **protected void doHead (HttpServletRequest request , HttpServletResponse response)**: It handles the HEAD request.
6. **protected void doOptions (HttpServletRequest request , HttpServletResponse response)**: Performs the HTTP OPTIONS operation; the default implementation of this method automatically determines what HTTP Options are supported.
7. **protected void doPut (HttpServletRequest request , HttpServletResponse response)**: Performs the HTTP PUT operation; the default implementation reports an HTTP BAD_REQUEST error.
8. **protected void doTrace (HttpServletRequest request , HttpServletResponse response)**: Performs the HTTP TRACE operation; the default implementation of this method causes a response with a message containing all of the headers sent in the trace request.
9. **protected void doDelete (HttpServletRequest request , HttpServletResponse response)**: Performs the HTTP DELETE operation; the default implementation reports an HTTP BAD_REQUEST error.
10. **protected long getLastModified (HttpServletRequest request)**: Gets the time the requested entity was last modified; the default implementation returns a negative number, indicating that the modification time is unknown and hence should not be used for conditional GET operations or for other cache control operations as this implementation will always return the contents.

2.5 COOKIES

Cookies are short pieces of data sent by web servers to the client browser. The cookies are saved to client's hard disk in the form of small text file. Cookies help the web servers to identify web users, by this way server tracks the user. Cookies pay very important role in the session tracking.

Cookie Class

- In JSP cookie are the objects of the class **javax.servlet.http.Cookie**. This class is used to create a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the

server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

- The getCookies() method of the request object returns an array of Cookie objects. Cookies can be constructed using the following code:

Cookie(java.lang.String name, java.lang.String value)

Cookie objects have the following methods.

Method	Description
getComment()	Returns the comment describing the purpose of this cookie, or null if no such comment has been defined.
getMaxAge()	Returns the maximum specified age of the cookie.
getName()	Returns the name of the cookie.
getPath()	Returns the prefix of all URLs for which this cookie is targeted.
getValue()	Returns the value of the cookie.
setComment(String)	If a web browser presents this cookie to a user, the cookie's purpose will be described using this comment.
setMaxAge(int)	Sets the maximum age of the cookie. The cookie will expire after that many seconds have passed. Negative values indicate the default behavior: the cookie is not stored persistently, and will be deleted when the user web browser exits. A zero value causes the cookie to be deleted
setPath(String)	This cookie should be presented only with requests beginning with this URL.
setValue(String)	Sets the value of the cookie. Values with various special characters (white space, brackets and parentheses, the equals sign, comma, double quote, slashes, question marks, the "at" sign, colon, and semicolon) should be avoided. Empty values may not behave the same way on all browsers.

Example Using Cookies

To write code in JSP file to set and then display the cookie.

Create Form

Here is the code of the form (cookieform.jsp) which prompts the user to enter his/her name.

```
<%@ page language="java" %>
<html>
        <head>
                <title> Cookie Input Form </title>
        </head>
        <body>
                <form method="post" action="setcookie.jsp">
                        <p> <b> Enter Your Name: </b> <input type="text" name="username"> <br>
```

```
                    <input type="submit" value="Submit">
                </form>
        </body>
</html>
```

Above form prompts the user to enter the user name. User input are posted to the setcookie.jsp file, which sets the cookie. Here is the code of **setcookie.jsp file:**

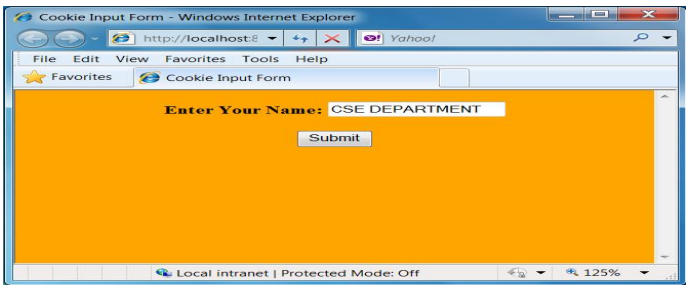
```
<%@ page language="java" import="java.util.*"%>
        <%
                String username=request.getParameter("username");
                if(username==null) username="";
                Date now = new Date();
                String timestamp = now.toString();
                Cookie cookie = new Cookie ("username",username);
                cookie.setMaxAge(365 * 24 * 60 * 60);
                response.addCookie(cookie);
        %>
<html>
        <head>
                <title> Cookie Saved </title>
        </head>
        <body>
                <p> <a href="showcookievalue.jsp"> Next Page to view the cookie value </a> <p>
        </body>
</html>
```

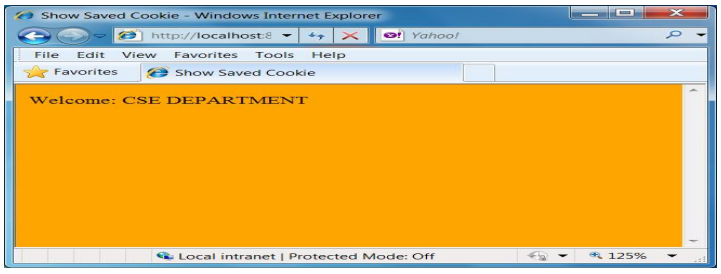
Above code sets the cookie and then displays a link to view cookie page. Here is the code of display cookie page **(showcookievalue.jsp):**

```
<%@ page language="java" %>
        <%
                String cookieName = "username";
                Cookie cookies [] = request.getCookies ();
                Cookie myCookie = null;
                if (cookies != null)
                {
                        for (inti = 0; i<cookies.length; i++)
                        {
                                if (cookies [i].getName().equals (cookieName))
                                {
                                        myCookie = cookies[i];
                                        break;
                                }
                        }
                }
        %>
<html>
        <head>
                <title>Show Saved Cookie</title>
        </head>
```

```
<body>
        <%
                if (myCookie == null)
                {
        %>
                No Cookie found with the name <%=cookieName%>
        <%
                }
                else
                {
        %>
                <p>Welcome: <%=myCookie.getValue()%>.
        <%
                }
        %>
</body> </html>
```

When user navigates to the above the page, cookie value is displayed.





2.6 FILTERS

A filter is an object that is invoked at the preprocessing and post processing of a request. It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc. The servlet filter is pluggable, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet. So maintenance cost will be less.

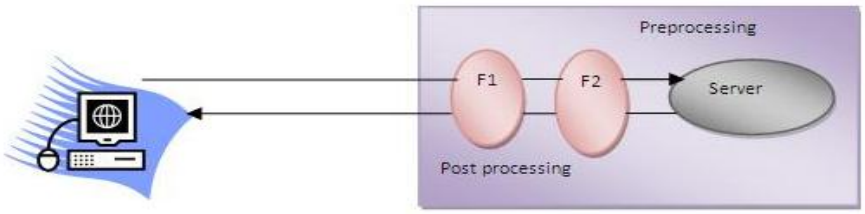


Fig 2.6 Filter

Usage of Filter

- Recording all incoming requests
- Logs the IP addresses of the computers from which the requests originate
- Conversion
- Data compression
- Encryption and decryption
- Input validation etc

Advantage of Filter

- Filter is pluggable.
- One filter don't have dependency onto another resource.
- Less Maintenance

Filter API

The servlet filter have its won API. The javax.servlet package contains the three interfaces of Filter API.

1. Filter
2. FilterChain
3. FilterConfig

1) Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.

Method	Description
public void init (FilterConfig config)	init() method is invoked only once. It is used to initialize the filter.
public void doFilter (HttpServletRequest request, HttpServletResponse response, FilterChain chain)	doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks.
public void destroy()	This is invoked only once when filter is taken out of the service.

2) FilterChain interface

The object of FilterChain is responsible to invoke the next filter or resource in the chain. This object is passed in the doFilter method of Filter interface. The FilterChain interface contains only one method:

public void doFilter(HttpServletRequest request, HttpServletResponse response): it passes the control to the next filter or resource.

How to define Filter

We can define filter same as servlet. Let's see the elements of filter and filter-mapping.

```
<web-app>
    <filter>
        <filter-name> ... </filter-name>
        <filter-class> ... </filter-class>
    </filter>
    <filter-mapping>
        <filter-name> ... </filter-name>
        <url-pattern> ... </url-pattern>
    </filter-mapping>
</web-app>
```

For mapping filter we can use, either url-pattern or servlet-name. The url-pattern elements has an advantage over servlet-name element i.e. it can be applied on servlet, JSP or HTML.

Example of Filter

We are simply displaying information that filter is invoked automatically after the post processing of the request.

index.html

```
<a href="servlet1"> click here </a>
```

MyFilter.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
public class MyFilter implements Filter
{
public void init(FilterConfig arg0) throws ServletException {}
public void doFilter(ServletRequest req, ServletResponse resp,  FilterChain chain) th
rows IOException,
ServletException
{
        PrintWriter out=resp.getWriter();
        out.print("filter is invoked before");
        chain.doFilter(req, resp);              //sends request to next resource
         out.print("filter is invoked after");
 }
 public void destroy() {}
}
```

HelloServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet
```

```
{
public void doGet(HttpServletRequest request, HttpServletResponse response)throw
s ServletException,
IOException
{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print("<br> welcome to servlet <br>");
}
}
```

web.xml

For defining the filter, filter element of web-app must be defined just like servlet.

```
<web-app>
  <servlet>
        <servlet-name> s1 </servlet-name>
        <servlet-class> Hello Servlet </servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name> s1 </servlet-name>
        <url-pattern>/servlet1</url-pattern>
  </servlet-mapping>
  <filter>
        <filter-name> f1 </filter-name>
        <filter-class> MyFilter </filter-class>
  </filter>
  <filter-mapping>
        <filter-name> f1 </filter-name>
        <url-pattern> /servlet1 </url-pattern>
        </filter-mapping>
</web-app>
```

2.7 JAVA SERVER PAGES (JSP)

- JSP abbreviated as Java Server Pages.
- JSP is Sun's solution for developing dynamic web sites.
- JSP provide excellent server side scripting support for creating database driven web applications.
- JSP is a server side technology that enables web programming to generate web pages dynamically in response to client requests.
- JSP uses HTML, XML and Java code, the application are secure, fast and independent of server platforms.
- JSP is nothing but high level abstraction of java servlet technology.
- It allows us to directly embed pure java code in an HTML page.
- JSP pages run under the supervision of an environment called web container.
- The web container compiles the page on the server and generates a servlet, which is loaded in the Java Runtime Environment (JRE).
- JSP enables the developers to directly insert java code into JSP file.
- JSP pages are efficient, they load into the web server's memory on receiving the request at the very first time and the subsequent calls are served within a very short period of time.

- This makes the development of the web applications convenient, simple and fast.
- Maintenance also becomes very easy.
- In today's environment, most website server's dynamic pages are based on user request.
- Database is very convenient way to store the data of users.
- JSP file have the extension .jsp

JSP and ASP

- JSP is portable or running the entire platform. Asp runs only in MS windows platform.
- Similar operation

JSP and JavaScript

JSP

- It dynamically Web Pages can be changed on the server.
- It can access server side resources like database, catalogs, pricing information.
- It can access session tracking and cookies.

JavaScript

- It can generate dynamically on the client side.
- It does not access this type of resources.
- It does not have.

JSP working Model:

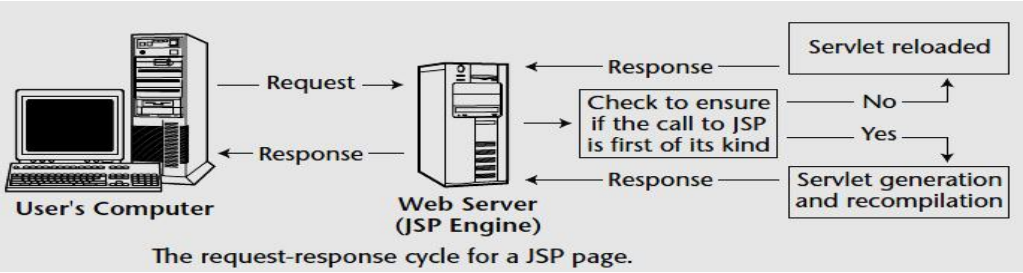


Fig 2.7 JSP Working Model

2.8 JSP Engines

- To Process JSP page, a JSP engine is needed.
- JSP Engine is nothing but a specialized servlet, which runs under the supervision of the servlet engine.
- The JSP Engine is typically connected with a web server or can be integrated inside a web server or an application server.

Many servers are

- Tomcat
- Java Web Server
- WebLogic
- WebSphere

Life Cycle of a Servlet and JSP

- A web server is responsible for initializing, invoking and destroying.
- A web server communicates with a servlet through a simple interfaces as
    1. JspInit()
    2. JspService()
    3. JspDestroy()

How JSP works?

1. Translates the JSP source code into the servlet source code.
2. Complies the servlet source code to generate a class file.
3. Loads the class file and create an instance.
4. Initializes the servlet instance by calling the jspInit() method.
5. Invokes the jspService() method, passing the request and response objects.

2.9 COMPONENTS OF JSP

JSP Syntax:

The syntax of JSP is almost similar to that of XML.

The general rules are

- Tags must have their matching end tags.
- Attributes must appear in the start tags.
- Attribute values in the tag must be quoted.

Components of JSP:

JSP page is built using components such as

- Directives
- Scripting Elements
    - Expressions
    - Declarations
    - Scriptlets
- Standard Actions
- Custom Tags

1. Directives

- A Directives element in a JSP page provides global information amount a particular JSP page.
- The syntax of jsp directive is

  <%@ directive-name attribute="value" %>

The three standard directives available in all JSP environments:

- Page
- include
- Taglib

Page Directives:

- The page directive defines attributes that notify the web container about the general settings of a JSP page.
- It has the following syntax:

  <%@ page attribute="value" attribute="value" %>

Example:

```
<%@ page language="java" %>
<%@ page language="java"  import="java.sql.*" %>
```

Page Directives Examples

```
<%@ page language=”java” %>
<html>
        <head>
                <title> Simple JSP example </title>
        </head>
        <body>
                <h1> Welcome </h1> <hr>
                <h3>Today’s date and time: </h3>
        <%-- This is the JSP content that displays the server time by using the method Date() --%>
                <% =new java.util.Date() %>
        </body>
</html>
```

Output

Include Directives

- Include directives are too used to specify the name of the files to be inserted during the compilation of the JSP page.
- This directive is to include the HTML, JSP or Servlet file into a JSP file.
- This is a static inclusion to a file.

Syntax

<%@ include file="filename" %>

Example

```
<%@ include file="/header.html" %>
<%@ include file="/doc/legal/disclaimer.html" %>
<%@ include file="sortmethod" %>
```

Include Directives Examples

first.jsp

```
<html>
        <head>
                <title> An Include example </title>
        </head>
        <body>
                <h2> This is the content of first.jsp </h2> <hr>
                <% =”Hello, welcome to the world of Java Server Pages!!!” %>
                <%@ include file=”second.jsp” %>
        </body>
```

</html>

Second.jsp

```
<html>
        <head>
                <title> An Include example </title>
        </head>
        <body>
                <h2> This is the content of second. Jsp </h2> <hr>
                <%= “Hope you’re having fun learning JSP” %>
        </body>
</html>
```

Output

Comments

- The JSP comment syntax is
  <%- - This is a hidden JSP comment - - %>
- and the latter looks like this:
  <! - - This is included in the generated HTML - ->

2. Scripting Elements

The 3 different types of Scripting Elements are

1. Expressions
2. Declarations
3. Scriptlets

Expressions

Syntax of JSP Expression is

<% = expression %>

- JSP Expression start with <% = and ends with %>.
- Between these, any number of expressions can be embedded.
- The result of expression is converted to the string to get displayed.

Example

<%=“Hello World!” %>

Expression examples

```
<HTML>
        <HEAD>
```

```
        <TITLE> My first JSP Example </TITLE>
    </HEAD>

    <BODY>
        <H1> My first JSP example </H1> <HR>
        <%-- This is the JSP content --%>
        <%=”Hello World” %>
    </BODY>
</HTML>
```

Declarations:

The syntax of JSP Declaratives is

```
<%!
    // declare all the variables here
%>
```

- JSP declaration begins with <%! and ends with %>.
- We can embed any amount of java code in the JSP declarations.
- Variables and functions in the declarations are class level and can be used anywhere in the JSP page.

2.10 SCRIPLETS:

The Syntax of JSP Scriptlets

```
<%
    //java code
%>
```

Example:

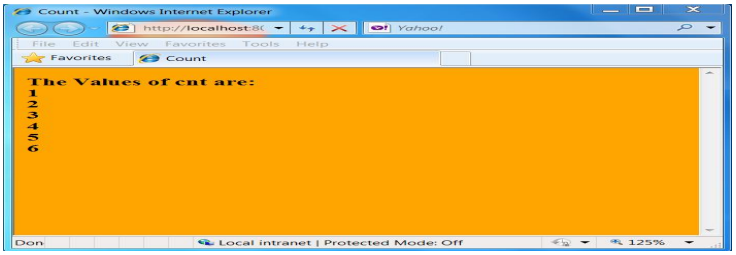
```
<%
    String username=cse;
    Username = request.getParameter(“username”);
%>
```

Program 1:

```
<%@ page language="java"%>
<html>
    <head>
        <title> Count program in JSP </title>
    </head>
    <body>
        <%!
            Int cnt=0;
            private int getcount()
            {
                // increment cnt and return the value
                cnt++;
                return cnt;
            }
        %>
        <p> The Values of cnt are:
```

```
                <%=getcount()%>
                <%=getcount()%>
                <%=getcount()%>
                <%=getcount()%>
        </body>
</html>
```

Output



- We can embed any amount of java code in the JSP Scriptlets.
- JSP Engine places these codes in JspService() methods.
- Variables available to the JSP Scriptlets are

  Request – HttpServletRequest
  Response – HttpServletResponse
  Session – HTTP session object associated with the request
  Out - is an object of output stream and is used to send any output to the client.

2.11 Standard Action:

- Actions are high-level JSP elements that create, modify, or use other objects.
- The syntax for the JSP standard Action is

  <tagname [attr="value" attr="value" ...] >
  ...
  </tag-name>

The various Tag name are

- Include
- Forward
- Param
- Plugin

Include:

- The Inclusion of file is dynamic and the specified file is included in the JSP file at runtime.
- Output of the included file is inserted into the JSP file.

Syntax:

<jsp: include page =“filename”>

Include Example

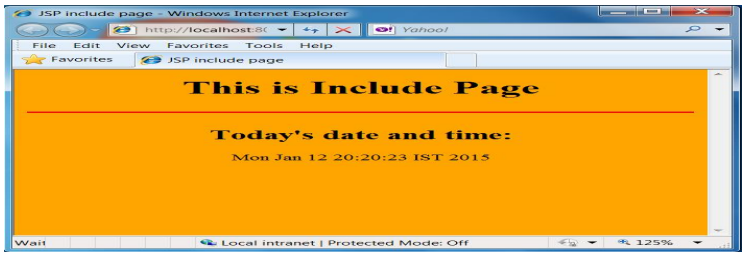
```
<%@page language="java"%>
<html>
```

```
<head>
        <title> JSP include page </title>
</head>
<body>
        <%-- using the <jsp:include> action --%>
        <h1 align="center"> This is Include Page </h1> <hr color="red">
        <jsp:include page="dateutil.jsp" flush="true"/>
</body>
</html>
```

dateutil.jsp

```
<%@ page language="java"%>
<html>
<head>
        <title> A Dynamic Content example </title>
</head>
<body bgcolor="orange">
    <center>
        <h2> Today's date and time: </h2>
        <%= new java.util.Date() %>
     </center>
</body>
</html>
```

Output:



Forward

This will redirect to the different page without notifying browser.

Syntax:

<jsp: forward page =“filename”/>

Param and Plugin:

<jsp:param>	Binds a value to a name and passes the binding to another resource invoked with <jsp:include> or <jsp:forward>. Syntax is <jsp:param name="*name*" value="*value*" />
<jsp:plugin>	Used to generate the appropriate HTML linkage for downloading the Java plugin: <jsp:plugin type="*bean | applet*" code="*objectCode*" codebase="*objectCodebase*" { align="*alignment*" } { archive="*archiveList*" } { height="*height*" } { hspace="*hspace*" } { jreversion="*jreversion*" } { name="*componentName*" } { vspace="*vspace*" } { width="*width*"} { nspluginurl="*url*" } { iepluginurl="*url*" } > { <jsp:params> { <jsp:param name="*name*" value="*value*" /> }+</jsp:params> }}</jsp:plugin>

2.12 CUSTOM TAGS

- The taglib directive makes custom actions available in the current page through the use of a tag library.

The syntax of the directive is

<%@ tagliburi="tagLibraryURI" prefix="tagPrefix" %>

Attribute	Value
tagLibraryURI	The URL of a Tag Library Descriptor.
tagPrefix	A unique prefix used to identify custom tags used later in the page.

Implicit Objects

Variable Name	Value
request	The ServletRequest or HttpServletRequest being serviced.
response	The ServletResponse or HttpServletResponse that will receive the generated HTML output.
pageContext	The PageContext object for this page. This object is a central repository for attribute data for the page, request, session, and application.
session	If the JSP page uses an HttpSession, it is available here under the name session.
application	The servlet context object.
Variable Name	**Value**
out	The character output stream used to generate the output HTML.
config	The ServletConfig object for this servlet context.
page	A reference to the JSP page itself.
exception	An uncaught exception that causes the error page to be invoked. This variable is available only to pages with isErrorPage="true".

2.13 SESSION TRACKING:

Session tracking is a mechanism that servlets use to maintain state about a series of requests from the same user (that is, requests originating from the same browser) across some period of time.

Session-Tracking basics

Every user of a site is associated with a javax.servlet.http.HttpSession object that servlets can use to store or retrieve information about that user
Basic session tracking functionalities,

- Obtain a session (an **HttpSession** object) for a user.
- Store or get data from the HttpSession object.
- Invalidate the session (optional).

Get Session

- A servlet uses its request object's getSession() method to retrieve the current HttpSession object.
- public HttpSessionHttpServletRequest.getSession(boolean create)
- This method returns the current session associated with the user making the request.
- If the user has no current valid session, this method creates one if create is true or returns null if create is false.
- To ensure the session is properly maintained, this method must be called at least once before any output is written to the response.

Put Value

- You can add data to an HttpSession object with the putValue() method:
- public void HttpSession.putValue(String name, Object value)
- This method binds the specified object value under the specified name. Any existing binding with the same name object from a session, use getValue():
- public Object HttpSession.getValue(String name)
- This method returns the object bound under the specified name or null if there is no binding.
- You can also get the names of all of the objects bound to a session with
- getValueNames():
- public String[] HttpSession.getValueNames()
- This method returns an array that contains the names of all objects bound to this session or an empty (zero length) array if there are no bindings.

Remove Value

- Finally, you can remove an object from a session with removeValue():
- public void HttpSession.removeValue(String name)
- This method removes the object bound to the specified name or does nothing if there is no binding
- Each of these methods can throw a java.lang.IllegalStateException if the session being accessed is invalid.

The Session Life Cycle

- Sessions do not last forever.
- A session either expires automatically, after a set time of inactivity (for the Java Web Server the default is 30 minutes), or manually, when it is explicitly invalidated by a servlet.
- When a session expires (or is invalidated), the HttpSession object and the data values it contains are removed from the system.
- Beware that any information saved in a user's session object is lost when the session is invalidated.

- If you need to retain information beyond that time, you should keep it in an external location (such as a database) and store a handle to the external data in the session object (or your own persistent cookie)

URL Rewriting

The Methods involved in managing the session life cycle

1. **Public boolean HttpSession.isNew() -** This method returns whether the session is new or not.
2. **public void HttpSession.invalidate() -**This method causes the session to be immediately invalidated. All objects stored in the session are unbound.
3. **public long HttpSession.getCreationTime() -** This method returns the time at which the session was created.
4. **public long HttpSession.getLastAccessedTime() -** This method returns the time at which the client last sent a request associated with this session.

2.14 DATABASE CONNECTIVITY

The connectivity from MYSQL database with JSP. We take an example of **Books** database. This database contains a table named **books_details**. This table contains three fields- **id**, **book_name**&**author**. We start from very beginning. First we learn how to create tables in MySQl database after that we write a html page for inserting the values in '**books_details**' table in database. After submitting values a table will be showed that contains the book name and author name.

Database

The database in example consists of a single table of three columns or fields. The database name is "books" and it contains information about **books names & authors**.

Table:books_details

ID	Book Name	Author
1.	Java I/O	Tim Ritchey
2.	Java & XML,2 Edition	Brett McLaughlin
3.	Java Swing, 2nd Edition	Dave Wood, Marc Loy

Start MYSQL prompt and type this SQL statement & press Enter-

MYSQL>CREATE DATABASE `books`;

This will create "books" database.

Now we create table a table "books_details" in database "books".

MYSQL>CREATE TABLE `books_details` (`id` INT(11) NOT NULL AUTO_INCREMENT , `book_name` VARCHAR(100) NOT NULL , `author` VARCHAR(100) NOT NULL , PRIMARY KEY (`id`)) TYPE = MYISAM ;

This will create a table "books_details" in database "books"

JSP Code

The following code contains html for user interface & the JSP back end-

```
<%@ page language="java" import="java.sql.*" %>
<%
        String driver = "org.gjt.mm.mysql.Driver";
```

```
        Class.forName(driver).newInstance();
        Connection con=null;
        ResultSetrst=null;
        Statement stmt=null;
        try   {
                String url="jdbc:mysql://localhost/books?user=
user>&password=<password>";
                con=DriverManager.getConnection(url);
                stmt=con.createStatement();
        }
        catch(Exception e) {
                System.out.println(e.getMessage());
        }
        if(request.getParameter("action") != null){
                String bookname=request.getParameter("bookname");
                String author=request.getParameter("author");
                stmt.executeUpdate("insert into books_details(book_name,author)
                      values('"+bookname+"','"+author+"')");
                rst=stmt.executeQuery("select * from books_details");
%>
                <html>
                <body>
                <center>
                        <h2> Books List </h2>
        <table border="1" cellspacing="0" cellpadding="0">
                        <tr>
                                <td> <b> S.No </b> </td>
                                <td> <b> Book Name </b> </td>
                                <td> <b> Author </b>  </td>
                        </tr>
        <%
                        int no=1;
                        while(rst.next()){
        %>

<tr>
                        <td> <%=no%> </td>
                        <td> <%=rst.getString("book_name")%> </td>
                <td> <%=rst.getString("author") </td>
                %>
                </tr>
                <%
                   no++;
        }
        rst.close();
        stmt.close();
        con.close();
            %>
        </table>
        </center>
```

```
        </body>
        </html>
<%}else{%>
        <html>
        <head>
                <title>Book Entry FormDocument</title>
                <script language="javascript">
                   function validate(objForm){
                        if(objForm.bookname.value.length==0){
                        alert("Please enter Book Name!");
                        objForm.bookname.focus();
                        return false;
                        }
                        if(objForm.author.value.length==0){
                        alert("Please enter Author name!");
                        objForm.author.focus();
                        return false;
                        }
                        return true;
                        }
                        </script>
                </head>
                <body>
                <center>
<form action="BookEntryForm.jsp" method="post" name="entry" onSubmit="return
validate(this)">
        <input type="hidden" value="list" name="action">
        <table border="1" cellpadding="0" cellspacing="0">
        <tr>
        <td>
        <table>
        <tr>
        <td colspan="2" align="center">
<h2> Book Entry Form </h2> </td>
        </tr>
        <tr>
        <td colspan="2"> </td>
        </tr>
        <tr>
        <td>Book Name:</td>
<td> <input name="bookname" type="text" size="50"> </td>
        </tr>
        <tr>
        <td> Author: </td> <td> <input name="author" type="text" size="50"> </td>
        </tr>
        <tr>
        <td colspan="2" align="center">
<input type="submit" value="Submit"> </td>
        </tr>
</table>
```

```
        </td>
</tr>
</table>
</form>
        </center>
        </body>
        </html>
<%}%>
```

Now we explain the above codes.

1. Declaring Variables:

Java is a strongly typed language which means, that variables must be explicitly declared before use and must be declared with the correct data types. In the above example code we declare some variables for making connection. These variables are-

Connection con=null;
ResultSetrst=null;
Statement stmt=null;

The objects of type Connection, ResultSet and Statement are associated with the Java sql. "con" is a Connection type object variable that will hold Connection type object. "rst" is a ResultSet type object variable that will hold a result set returned by a database query. "stmt" is a object variable of Statement .Statement Class methods allow to execute any query.

2. Connection to database:

The first task of this programmer is to load database driver. This is achieved using the single line of code:-

```
String driver = "org.gjt.mm.mysql.Driver";
Class.forName(driver).newInstance();
```

The next task is to make a connection. This is done using the single line of code :-

```
String
url="jdbc:mysql://localhost/books?user=<userName>&password=<password>
";
con=DriverManager.getConnection(url);
```

When url is passed into getConnection() method of DriverManager class it returns connection object.

Executing Query or Accessing data from database:

This is done using following code :-

```
stmt=con.createStatement(); //create a Statement object
rst=stmt.executeQuery("select * from books_details");
```

stmt is the Statement type variable name and **rst** is the RecordSet type variable. A query is always executed on a Statement object.

A Statement object is created by calling createStatement() method on connection object **con.**

The two most important methods of this Statement interface are executeQuery() and executeUpdate(). The executeQuery() method executes an SQL statement that returns a single ResultSet object. The executeUpdate() method executes an insert,

update, and delete SQL statement. The method returns the number of records affected by the SQL statement execution.
After creating a Statement, a method executeQuery() or executeUpdate() is called on Statement object **stmt** and a SQL query string is passed in method executeQuery() or executeUpdate().
This will return a ResultSet**rst**related to the query string.

Reading values from a ResultSet:

```
while(rst.next()){
 %>
<tr><td><%=no%></td><td><%=rst.getString("book_name")%></td><td><%=
rst.getString("author")%></td></tr>
  <%
}
```

The ResultSet represents a table-like database result set. A ResultSet object maintains a cursor pointing to its current row of data. Initially, the cursor is positioned before the first row. Therefore, to access the first row in the ResultSet, you use the next() method. This method moves the cursor to the next record and returns true if the next row is valid, and false if there are no more records in the ResultSet object.
Other important methods are getXXX() methods, where XXX is the data type returned by the method at the specified index, including String, long, and int. The indexing used is 1-based. For example, to obtain the second column of type String, you use the following code:

```
resultSet.getString(2);
```

You can also use the getXXX() methods that accept a column name instead of a column index. For instance, the following code retrieves the value of the column LastName of type String.

```
resultSet.getString("book_name");
```

The above example shows how you can use the next() method as well as the getString() method. Here you retrieve the 'book_name' and 'author' columns from a table called 'books_details'. You then iterate through the returned ResultSet and print all the book name and author name in the format " book name | author " to the web page.

UNIT-III

3.1 INTRODUCTION TO XML:

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML Does Not Use Predefined Tags

- The XML language has no predefined tags.
- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
- HTML works with predefined tags like <p>, <h1>, <table>, etc.
- With XML, the author must define both the tags and the document structure.

XML is Extensible

- Most XML applications will work as expected even if new data is added (or removed).
- Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <data>).
- Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed; older version of the application can still work:

```
<note>
      <date> 2015-09-01 </date>
      <hour> 08:30 </hour>
      <to> Tove </to>
      <from> Jani </from>
      <body> Don't forget me this weekend! </body>
</note>
```

Note
To: Tove
From: Jani
Date: 2015-09-01 08:30
Head: (none)
Don't forget me this weekend!

3.2 XML DOCUMENT

XML Document Types:

1. An XML document with correct syntax is called "Well Formed".
2. A "Valid" XML document must also conform to a document type definition.

(i) Well Formed XML Documents:

- XML with correct syntax is Well Formed XML.

The syntax rules

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <note>
                <to> Tove </to>
                <from> Jani </from>
                <heading> Reminder </heading>
                <body> Don't forget me this weekend! </body>
        </note>
```

(ii) Valid XML Documents

- A "valid" XML document is not the same as a "well formed" XML document.
- A "valid" XML document must be well formed. In addition it must conform to a document type definition.
- Rules that define the legal elements and attributes for XML documents are called Document Type Definitions (DTD) or XML Schemas.

There are two different document type definitions that can be used with XML:

1. DTD - The original Document Type Definition
2. XML Schema - An XML-based alternative to DTD

(ii) XML DTD

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
        <to> Tove </to>
        <from> Jani </from>
        <heading> Reminder </heading>
        <body> Don't forget me this weekend! </body>
```

```
</note>
```

The DOCTYPE declaration, in the example above, is a reference to an external DTD file.

3.3 DTD

- A DTD is a Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document.

Why Use a DTD?

- With a DTD, independent groups of people can agree on a standard DTD for interchanging data.
- An application can use a DTD to verify that XML data is valid.

(i) An Internal DTD Declaration

If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

XML document with an internal DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
   <to> Tove </to>
   <from> Jani </from>
   <heading> Reminder </heading>
   <body> Don't forget me this weekend </body>
</note>
```

(ii) An External DTD Declaration

If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

XML document with a reference to an external DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
        <note>
                <to> Tove </to>
                <from> Jani </from>
                <heading> Reminder </heading>
                <body> Don't forget me this weekend! </body>
```

```
</note>
```

And here is the file "note.dtd", which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

3.4 XML SCHEMA

- An XML Schema describes the structure of an XML document.
- The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="note">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="to" type="xs:string"/>
                        <xs:element name="from" type="xs:string"/>
                        <xs:element name="heading" type="xs:string"/>
                        <xs:element name="body" type="xs:string"/>
                    </xs:sequence>
            </xs:complexType>
    </xs:element>
</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XML Schemas Support Data Types

XML Schemas is the support for data types.

- It is easier to describe allowable document content
- It is easier to validate the correctness of data
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- It is easier to convert data between different data types

XML Schemas use XML Syntax

XML Schemas is that they are written in XML.

- You don't have to learn a new language

- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schema with the XML DOM
- You can transform your Schema with XSLT

XML Schemas are extensible, because they are written in XML.
With an extensible Schema definition you can:

- Reuse your Schema in other Schemas
- Create your own data types derived from the standard types
- Reference multiple schemas in the same document

XML Schemas Secure Data Communication

- When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content.
- With XML Schemas, the sender can describe the data in a way that the receiver will understand.
- A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.

However, an XML element with a data type like this:

```
<date type="date">2004-03-11</date>
```

ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".

3.5 XML NAMESPACE

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

```
<table>
    <tr>
        <td> Apples </td>
        <td> Bananas </td>
    </tr>
</table>
```

This XML carries information about a table (a piece of furniture):

```
<table>
    <name> African Coffee Table </name>
    <width> 80 </width>
    <length> 120 </length>
</table>
```

Solving the Name Conflict Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
      <h:td> Apples </h:td>
      <h:td> Bananas </h:td>
  </h:tr>
</h:table>
<f:table>
      <f:name> African Coffee Table </f:name>
      <f:width> 80 </f:width>
      <f:length> 120 </f:length>
</f:table>
```

In the example above, there will be no conflict because the two <table> elements have different names.

XML Namespaces - The xmlns Attribute

- When using prefixes in XML, a **namespace** for the prefix must be defined.
- The namespace can be defined by an **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax **xmlns:prefix="URI".**

```
<root>
    <h:table xmlns:h="http://www.w3.org/TR/html4/">
       <h:tr>
                <h:td> Apples </h:td>
                <h:td> Bananas </h:td>
       </h:tr>
    </h:table>
    <f:table xmlns:f="http://www.w3schools.com/furniture">
                 <f:name> African Coffee Table </f:name>
                <f:width> 80 </f:width>
                 <f:length> 120 </f:length>
    </f:table>
</root>
```

In the example above:

- The xmlns attribute in the first <table> element gives the h: prefix a qualified namespace.
- The xmlns attribute in the second <table> element gives the f: prefix a qualified namespace.
- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

Namespaces can also be declared the XML root element:

```
<root  xmlns:h="http://www.w3.org/TR/html4/"
   xmlns:f="http://www.w3schools.com/furniture">
<h:table>
```

```
        <h:tr>
            <h:td> Apples </h:td>
             <h:td> Bananas </h:td>
        </h:tr>
    </h:table>
    <f:table>
        <f:name> African Coffee Table </f:name>
        <f:width> 80 </f:width>
        <f:length> 120 </f:length>
    </f:table>
    </root>
```

- The namespace URI is not used by the parser to look up information.
- The purpose of using an URI is to give the namespace a unique name.
- However, companies often use the namespace as a pointer to a web page containing namespace information.

Default Namespaces:

Defining a default namespace for an element saves us from using prefixes in all the child elements.

It has the following syntax:

xmlns="*namespaceURI*"

This XML carries HTML table information:

```
<table xmlns="http://www.w3.org/TR/html4/">
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
</table>
```

XFILES:

3.6 XLINKS

XLink and its attributes

- XLink is used to create hyperlinks within XML documents.
- Any element in an XML document can behave as a link.
- With XLink, the links can be defined outside the linked files.

The anchor element, <a>, within HTML indicates a link to another resource on an HTML page. This could be a location within the same document or a document located elsewhere. In HTML terms, the anchor element creates a hyperlink to another location.

The hyperlink can either appear as straight text, a clickable image, or a combination of both. Although HTML anchor elements contain a lot of functionality, they are still limiting—they require the use of the anchor element (<a>) itself, and they basically sit there waiting for someone to click them before navigating to the specified location.

The **XML Linking Language**, XLink, addresses and overcomes these limitations by allowing a link to another document to be specified on any element within an XML document. The links to other documents can be much more complex than the simple links supported by the HTML specification.

Example for XLink

```
<?xml version="1.0" encoding="UTF-8"?>
<homepages xmlns:xlink="http://www.w3.org/1999/xlink">
  <homepage xlink:type="simple" xlink:href="http://www.w3schools.com"> Welcome
</homepage>
  <homepage xlink:type="simple" xlink:href="http://www.w3.org"> to achariya
</homepage>
</homepages>
```

Output

Welcome to achariya

XLink Attributes

The XML Linking Language creates a link to another resource through the use of attributes specified on elements, not through the actual elements themselves. The XML Linking Language specification supports the attributes listed in the below table

Attribute	Description
xlink:type	This attribute must be specified and indicates what type of XLink is represented or defined.
xlink:href	This attribute contains the information necessary to locate the desired resource.
xlink:role	This attribute describes the function of the link between the current resource and another.
xlink:arcrole	This attributes describes the function of the link between the current resource and another.
xlink:title	This attribute describes the meaning of the link between the resources.
xlink:show	This attribute indicates how the resource linked to should be displayed
xlink:actuate	This attribute specifies when to load the linked resource.
xlink:label	This attribute is used to identify a name for a target resource.
xlink:from	This attribute identifies the starting resource.
xlink:to	This attribute identifies the ending resource.

TYPES OF XLINK

The XML Linking Language offers two major types of links:

1. Simple links
2. Extended links

Within XLink, a simple link is a convenient, by which to associate two resources. These resource - one local and one remote - are connected by an arc, always making

a simple link an outbound link. An extended link associates any number of resources together. Furthermore, those resources may be both local and remote.

3.7 XPATH

The **XML Path Language (XPath)** is a standard for creating expressions that can be used to find specific pieces of information within an XML document.

XPath expressions are used by both XSLT (for which XPath provides the core functionality) and XPointer to locate a set of nodes.

To understand how XPath works, it helps to imagine an XML document as a tree of nodes consisting of both elements and attributes. An XPath expression can then be considered a sort of roadmap that indicates the branches of the tree to follow and what limbs hold the information desired.

XPath expressions have the ability to locate nodes based on the nodes' type, name, or value or by the relationship of the nodes to other nodes within the XML document. In addition to being able to find nodes based on these criteria, an XPath expression can also return any of the following:

- ✓ A node set
- ✓ A Boolean value
- ✓ A string value
- ✓ A numeric value

XML documents are a hierarchical tree of nodes. There is a similarity between URLs and XPath expressions.

- URLs represent a navigation path of a hierarchical file system.
- XPath expressions represent a navigation path for a hierarchical tree of nodes.

The priority for evaluating XPath expressions is as follows:

1. Grouping
2. Filters
3. Path operations

XPath Syntax:

The XML Path Language provides a declarative notation, termed a *pattern*, used to select the desired set of nodes from XML documents. Each pattern describes a set of matching nodes to select from a hierarchical XML document. Each pattern describes a "navigation" path to the desired set of nodes similar to the Uniform Resource Identifier (URI) syntax. However, instead of navigating a file system, the XML Path Language navigates a hierarchical tree of nodes within an XML document.

Each "query" of an XML document occurs from a particular starting node that defines the context for the query. The context for the query has a very large impact on the results. For instance, the pattern that locates a node from the root of an XML document will most likely be a very different pattern when looking for the same node from somewhere else in the hierarchy.

One possible result from performing an XPath query is a node set, or a collection of nodes matching specified search criteria. To receive these results, a "location path" is needed to locate the result nodes. These location paths select the resulting node set relative to the current context. A location path is, itself, made up of one or more location steps. Each step is further comprised of three pieces:

- An axis
- A node test
- A Predicate

Therefore, the basic syntax for an XPath expression would be something like this:

axis::node test[predicate]

Using this basic syntax and the XML document in operators table, we could locate all the <c> nodes by using the following XPath expression:

/a/b/child::*

Alternatively, we could issue the following abbreviated version of the preceding expression:

/a/b/c

All XPath expressions are dependent on the current context. The context is the current location within the tree of nodes. Therefore, if we're currently on the second <b> element within the XML document, select all the <c> elements contained within that <b> element by using the following XPath expression:

./c

This is what's known as a "relative" XPath expression.

3.8 XPOINTER

- XPointer allows links to point to specific parts of an XML document.
- XPointer uses XPath expressions to navigate in the XML document.
- XPointer describes a location within an external document.
- XPointer can target a point within that XML document or a range within the target XML document.

The **XML Pointer Language (XPointer)** describes a location within an external document, an XPointer can target a point within that XML document or a range within the target XML document. XPointer builds on the XPath specification, the location steps within an XPointer are comprised of the same elements that make up XPath location steps.

XPointer provides two more important node tests:

- point()
- range()

The XPointer specification added the concept of a location within an XML document. Within XPointer, a location can be an XPath node, a point, or a range.

A **point** can represent the location immediately before or after a specified character or the location just before or just after a specified node.

A **range** consists of a start point and an endpoint and contains all XML information between those two points. In fact, the XPointer specification extends the node types to include points and ranges. XPointer expressions also allow predicates to be specified as part of a location step in much the same fashion XPath expressions allow for them. As with XPath expressions, XPointer expressions have specific functions to deal with each specific predicate type.

However, the XPointer specification also adds an additional function named unique().This new function indicates whether an XPointer expression selects a single location rather than multiple locations or no locations at all.

For an XPath expression, the result from a location step is known as a ***node set***; for an XPointer expression, the result is known as a ***location set***. To reduce the confusion, the XPointer specification uses a different term for the results of an expression:

Because an XPointer expression can yield a result consisting of points or ranges, the idea of the *node set* had to be extended to include these types. Therefore, to prevent confusion, the results of an XPointer expression are referred to *location sets*.

Four of the functions that return location sets, id(), root(), here(), and origin(),

Some XPointer Functions That Return Location Sets

Function	description
id()	Selects all nodes with the specified ID
root()	Selects the root element as the only location in a location set
here()	Selects the current element location in a location set
origin()	Selects the current element location for a node using an out-of-line link

3.9 XML WITH XSL

Creating the XSL Style Sheet

The next step is to create the XSL style sheet. XSL style sheets are XML documents; as a result, they must be well formed. An XSL style sheet has the following general structure:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="URI" version="1.0">
    <!--XSL-T CONVERSION RULES-->
</xsl:stylesheet>
```

The <xsl:stylesheet> element defines how the XSLT processor should process the current XSL document. The xmlns attribute is the namespace definition. The XSL Transformation engine reads the xmlns attribute and determines whether it supports the given namespace. The xmlns attribute specifies the XSL prefix. All XSL elements and types in the document use the prefix.

The xmlns attribute value contains a Uniform Resource Identifier (URI), which serves as a generic method for identifying entities on the World Wide Web.

The XSL style sheet contains HTML text and XSL elements. The HTML text forms the basis of the desired output page. The XSL elements are template rules

for the XSLT processor. A template is associated with a given element in the XML document. In our example, a template is defined to match on the <book> element using the following code:

```
<xsl:template match="/book">
    <!--static text and xsl rules -->
</xsl:template>
```

XSLT defines the <xsl:value-of> element for retrieving data from a XML document. The <xsl:value-of> element contains a select attribute. This attribute value is the name of the actual XML element you want to retrieve. For example, the following code will retrieve the title of the book:

```
<xsl:value-of select="title" />
```

To create the file book_view.xsl. This style sheet will create an HTML page that contains information about the book, which is stored in the file book.xml. The style sheet contains the basic format of an HTML page and uses XSL elements to retrieve the data. Currently, the XSL elements are merely placeholders in the document. Once an XSL processor accesses the XSL style sheet, the processor executes the XSL elements and replaces them with the appropriate data from the XML document. Below code contains the complete code for book_view.xsl.

Example

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/book">
  <html>
    <body>
      <b> Title: </b> <xsl:value-of select="title" />
      <p/>
      <b> By: </b> <xsl:value-of select="author" />
      <p/>
      <b> Cost: </b> <xsl:value-of select="price" />
      <p/>
      <b> Category: </b> <xsl:value-of select="category" />
      <p/>
      <b> Description </b>
      <p/>
      <i> <xsl:value-of select="summary" /> </i>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

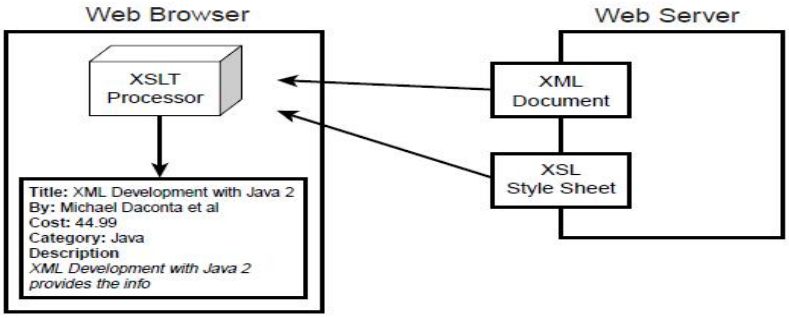
The XSLT Processor

Two techniques are available for performing the XSLT processing:

1. Client-side processing
2. Server-side processing

Client-side XSLT processing commonly occurs in a Web browser. The Web browser includes an XSLT processor and retrieves the XML document and XSL style sheet.

The client-side technique offloads the XSLT processing to the client machine. This minimizes the workload on the Web server. However, the disadvantage is that the Web browser must provide XSLT support. Netscape Communicator 6 and Microsoft Internet Explorer 6 support the XSLT 1.0 specification.



Processing XSLT in a Web browser

Server-side XSLT processing occurs on the Web server or application server. A server-side process such as an Active Server Page (ASP), Java Server Page (JSP), or Java servlet will retrieve the XML document and XSL style sheet and pass them to an XSLT processor. The output of the XSLT processor is sent to the client Web browser for presentation. The output is generally a markup language, such as HTML, that is understood by the client browser. The application interaction is shown below figure.

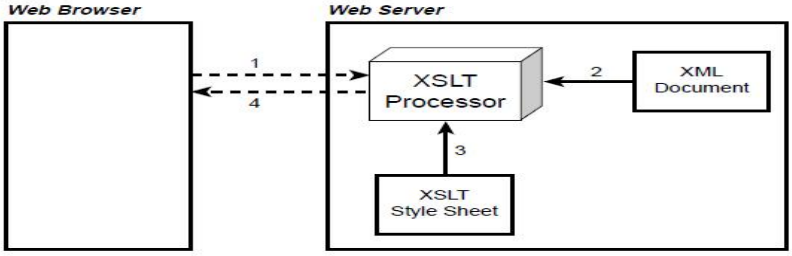


Fig 3.1: Processing XSLT on the server side

An advantage of the server-side technique is browser independence. The output document is simply an HTML file. This technique supports the older browser versions and makes the application more robust and versatile.

The server-side technique, the application can support a diverse collection of clients. The application can detect the user-agent, such as a WAP-enabled mobile phone, and send back a document containing a Wireless Markup Language (WML) tag. The WAP-enabled phone can render the content using the built-in WML mini-browser.

3.10 XSL FORMATTING OBJECTS

- The XSL technology is also composed of **XSL Formatting Objects (XSL-FO).**

- XSL-FO was designed to assist with the printing and displaying of XML data. The main emphasis is on the document layout and structure. This includes the dimensions of the output document, including page headers, footers, and margins.
- XSL-FO also allows the developer to define the formatting rules for the content, such as font, style, color, and positioning.XSL-FO is a sophisticated version of Cascading Style Sheets (CSS). XSL-FO borrows a lot of the terminology and elements from CSS.
- XSL-FO documents are well-formed XML documents. An XSL-FO formatting engine processes XSL-FO documents.
  The two techniques for creating XSL-FO documents.
  - The first is to simply develop the XSL-FO file with the included data.
  - The second technique is to dynamically create the XSL-FO file using an XSLT translation.

XSL-FO Formatting Engines

Many of the XSL-FO formatting engines implement a subset of the XSL-FO specification. Also, the browser support for XSL-FO is nonexistent. Engines are available that allow you to experiment with the basic features of XSL-FO. To use the Apache XSL-FOP to generate PDF documents from XML.

XSL-FO Engine	Web Site
Apache XSL-FOP	xml.apache.org
XEP	www.renderx.com
iText	www.lowagie.com/iText/
Unicorn	www.unicorn-enterprises.com

To create a simple XSL-FO document. Once the document is created, we will use the Apache XSL-FOP formatter to convert the document to a PDF file. The application interaction is illustrated below figure.

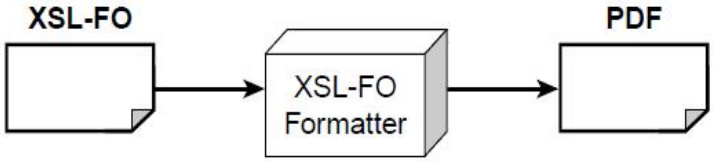


Fig 3.2: Apache XSL-FOP generating PDF documents

Basic Document Structure

An XML-FO document follows the syntax rules of XML; as a result, it is well formed. XSL-FO elements use the following namespace:

http://www.w3.org/1999/XSL/Format

The following code snippet shows the basic document setup for XSL-FO:

```
<?xml version=”1.0” encoding=”utf-8”?>
  <fo:root xmlns:fo=”http://www.w3.org/1999/XSL/Format”>
    <!-- layout master set -->
     <!-- page masters: size and layout -->
```

```
    <!-- page sequences and content -->
</fo:root>
```

The **element <fo:root>** is the root element for the XSL-FO document. An XSL-FO document can contain the following components:

1. Page master
2. Page master set
3. Page sequences

3.11 PARSING XML USING DOM

- The **Document Object Model (DOM)** provides a way of representing an XML document in memory so that it can be manipulated by your software.
- DOM is a standard application programming interface (API) that makes it easy for programmers to access elements and delete, add, or edit content and attributes.
- DOM was proposed by the World Wide Web Consortium (W3C) in August of 1997 in the User Interface Domain. The Activity was eventually moved to the Architecture Domain in November of 2000.
- DOM interfaces are defined independent of any particular programming language.
- DOM code in just about any programming language, such as Java, ECMAScript (a standardized version of JavaScript/JScript), or C++.

The XML DOM is:

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
  - ✓ The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.
  - ✓ In other words: **The XML DOM is a standard for how to get, change, add, or delete XML elements.**

DOM Core

- DOM is a tree structure that represents elements, attributes, and content.

Consider a simple XML document

```
<purchase-order>
        <customer> James Bond </customer>
        <merchant> Spies R Us </merchant>
        <items>
                <item> Night vision camera </item>
                <item> Vibrating massager </item>
        </items>
</purchase-order>
```

Tree structure representing the XML document as

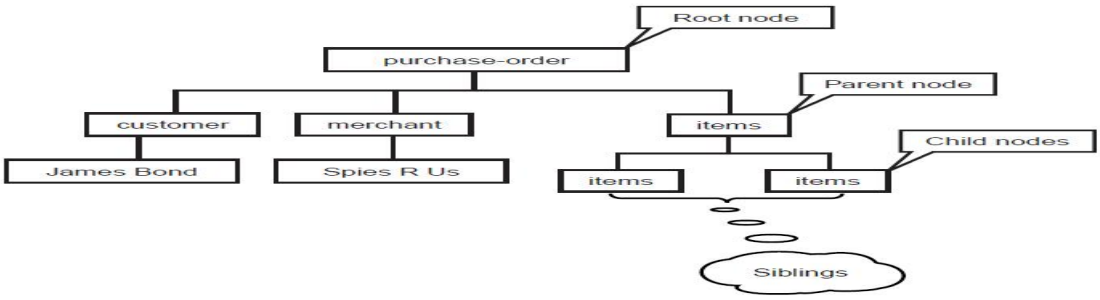


Fig 3.3 Tree Structure of XML Document

The DOM says:

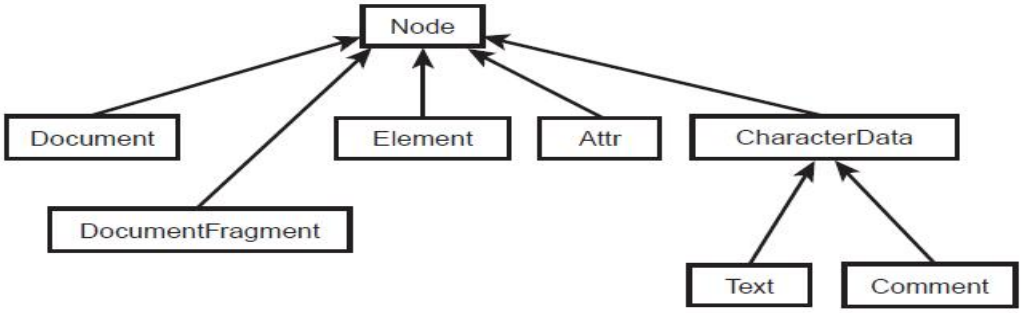
- The entire document is a document node
- Every XML element is an element node
- The text in the XML elements are text nodes
- Every attribute is an attribute node
- Comments are comment nodes

DOM Interfaces

DOM interfaces are defined in IDL so that they are language neutral.

Interface	Description
Node	The primary interface for the DOM. It can be an element, attribute, text, and so on, and contains methods for traversing a DOM tree.
NodeList	An ordered collection of Nodes.
NamedNodeMap	An unordered collection of Nodes that can be accessed by name and used with attributes.
Document	An Node representing an entire document. It contains the root Node.
DocumentFragment	A Node representing a piece of a document. It's useful for extracting or inserting a fragment into a document.
Element	A Node representing an XML element.
Attr	A Node representing an XML attribute.
CharacterData	A Node representing character data.
Text	A CharacterData node representing text.
Comment	A CharacterData node representing a comment.
DOMException	An exception raised upon failure of an operation.
DOMImplementation	Methods for creating documents and determining whether an implementation has certain features.

The relationships among the interfaces



3.12 SAX

- The **Simple API for XML (SAX)** can only be used for parsing existing documents.
- SAX (Simple API for XML) is an event-driven online algorithm for parsing XML documents, with an API interface developed by the XML-DEV mailing list.
- SAX parsers operate on each piece of the XML document sequentially.
- SAX is an API that can be used to parse XML documents. A *parser* is a program that reads data a character at a time and returns manageable pieces of data.
- For example, a parser for the English language might break up a document into paragraphs, words, and punctuation.
- SAX provides a framework for defining event listeners, or ***handlers*.**
- These handlers are written by developers interested in parsing documents with a known structure. The handlers are registered with the SAX framework in order to receive events.
- Events can include start of document, start of element, end of element, and so on. The handlers contain a number of methods that will be called in response to these events. Once the handlers are defined and registered, an input source can be specified and parsing can begin.

SAX Basics

To illustrate how SAX works, consider a simple document:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<fiction>
        <book author=”Herman Melville”> Moby Dick </book>
</fiction>
```

If you want to parse this document using SAX, you would build a *content handler* by creating a Java class that implements the Content Handler interface in the org.xml.saxmpackage.

Once you have a content handler, you simply register it with a SAX XMLReader, set up the input source, and start the parser. Next, the methods in your content handler will be called when the parser encounters elements, text, and other data. Specifically, the events generated by the preceding example will look something like this:

```
start document
start element: fiction
start element: book (including attributes)
characters: Moby Dick
end element: book
```

end element: fiction
end document

The events reported follow the content of the document in a linear sequence. There are a number of other events that might be generated in response to processing instructions, errors, and comments.

XML Processing with SAX:

A parser that implements SAX (i.e., *a SAX Parser*) functions as a stream parser, with an event-driven API. The user defines a number of callback methods that will be called when events occur during parsing.

The SAX events include

- XML Text nodes
- XML Element Starts and Ends
- XML Processing Instructions
- XML Comments

SAX Packages

- The SAX 2.0 API is comprised of two standard packages and one extension package.
- The standard packages are
  - org.xml.sax
  - org.xml.helpers

org.xml.sax package

The org.xml.sax package contains the basic classes, interfaces, and exceptions needed for parsing documents.

Name	Description
Interfaces	
Attributes	Interface for a list of XML attributes.
ContentHandler	Receives notification of the logical content of a document.
DocumentHandler	ContentHandler interface, which includes namespace support.
DTDHandler	Receives notification of basic DTD-related events.
EntityResolver	Basic interface for resolving entities.
ErrorHandler	Basic interface for SAX error handlers.
Locator	Interface for associating a SAX event with a document location.
XMLFilter	Interface for an XML filter.
XMLReader	Interface for reading an XML document using callbacks.
Classes	
HandlerBase	DocumentHandler interface.
InputSource	A single input source for an XML entity.
Exceptions	
SAXException	Encapsulates a general SAX error or warning.

SAXNotRecognizedException	Exception class for an unrecognized identifier.
SAXNotSupportedException	Exception class for an unsupported operation.
SAXParseException	Encapsulates an XML parse error or warning.

(ii) org.xml.sax.helpers package

The org.xml.sax.helpers package contains additional classes that can simplify some of your coding and make it more portable

Class	Description
AttributesImpl	Default implementation of the Attributes interface.
DefaultHandler	Default base class for SAX2 event handlers.
LocatorImpl	Provides an optional convenience implementation of Locator.
NamespaceSupport	Encapsulate namespace logic for use by SAX drivers.
ParserAdapter	Adapts a SAX1 Parser as a SAX2 XMLReader.
XMLFilterImpl	Base class for deriving an XML filter.
XMLReaderAdapter	Adapts a SAX2 XMLReader as a SAX1 Parser.
XMLReaderFactory	Factory for creating an XML reader.

3.13 XML INTEGRATING WITH DATABASE

XML and database integration is important because XML provides a standard technique to describe data.

XML Database Solutions

A large number of XML database solutions are available, and they generally come in two flavors:

1. database mapping
2. native XML support

(i) XML Database Mapping

The XML database solution provides a mapping between the XML document and the database fields. The system dynamically converts SQL result sets to XML documents. Depending on the sophistication of the product, it may provide a graphical tool to map the database fields to the desired XML elements. Other tools support a configuration file that defines the mapping. These tools continue to store the information in relational database management system (RDBMS) format. The simply provide an XML conversion process that is normally implemented as a server-side Web application.

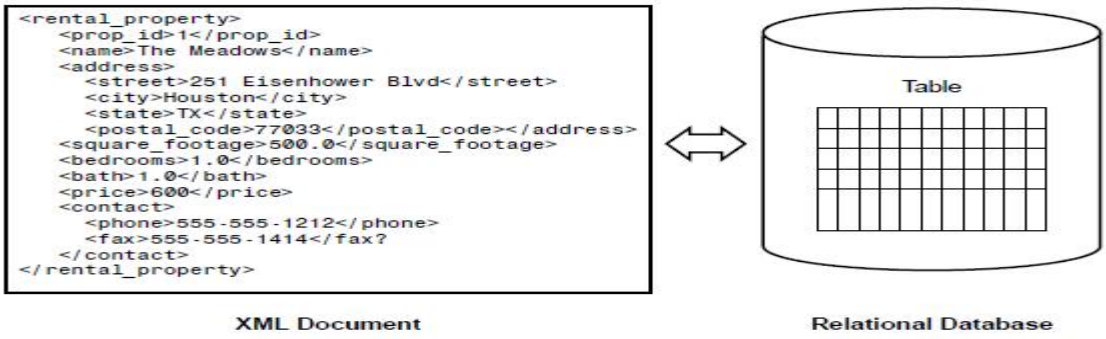


Fig 3.4: Mapping XML documents to database fields

(ii) Native XML Support:

The XML database solution actually stores the XML data in the document in its native format. Each product uses its own proprietary serialization technique to store the data. However, when the data is retrieved, it represents an XML document.

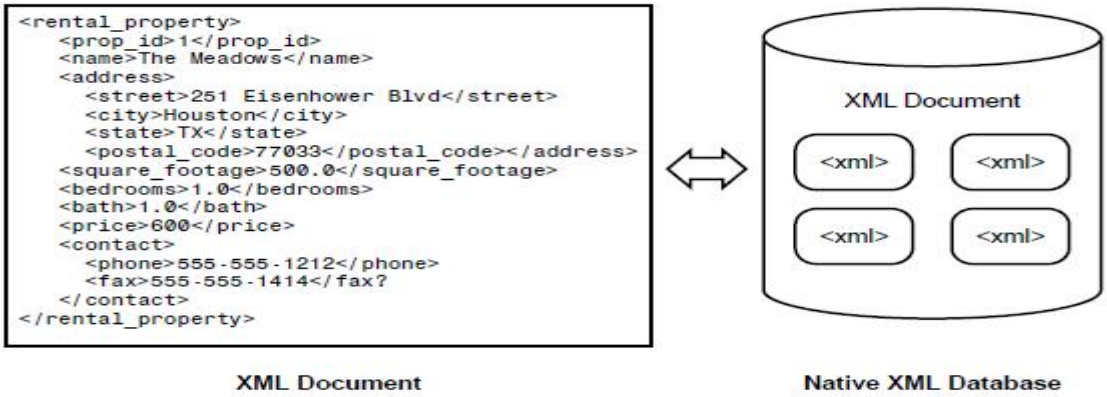


Fig 3.5: Native XML databases

Modeling Databases in XML:

Learn how to model a database in XML using Java. When we model a database, we provide an external representation of the database contents. For our sample program, we'll utilize a database that contains information on rental properties. We'll model the rental property database as an XML document.

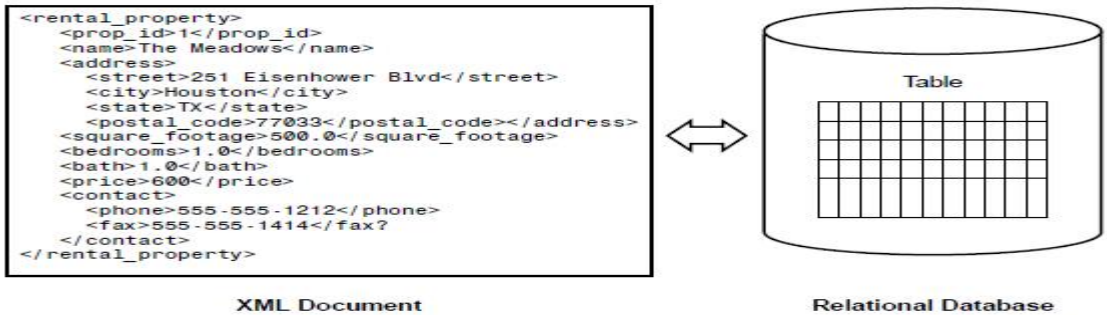


Fig 3.6: Modeling Databases in XML

One possible solution is to use Java servlets and JDBC. Java servlets are server-side components that reside in a Web server or application server. Java servlets are commonly used to handle requests from Web browsers using the HTTP protocol.

A key advantage to using servlets is the thin-client interface. The servlets handle the request on the server side and respond by generating an HTML page dynamically. This lowers the requirement on the client browser. The browser only has to provide support of HTML. As a result, there is zero client-side administration.

Java applets require the browser to support the correct version of the Java Virtual Machine (JVM). This has been a thorny issue with the Java community since the early days of applet development. If the browser doesn't support Java, the applet will not execute. Of course, there are a number of workarounds, such as the Java Plug-In and Java Web Start.

We can develop a servlet that uses JDBC. The servlet will make the appropriate query to the database and use Java database Connectivity (JDBC) API result set metadata to create the elements.

3.14 FORMATTING XML FOR THE WEB

(i) XML Presentation Using CSS:

```
<!-- This style sheet will be referenced as notestyle.css -->
Note
{
    display: block
}
From, To
{
   display:block;
   font-family:verdana;
   font-size:15px;
   margin-bottom:5px
}
Subject
{
    display:block;
    font-family:verdana;
    font-size:13px;
    font-weight:bold;
    margin-bottom:10px
}
Body
{
    display:block;
    font-family:verdana;
    font-size:12px
}
```

In this listing are five style selectors: Note, From, To, Subject, and Body. Each selector listed represents the name of an XML element. The styles associated with each selector will be applied to XML elements that have matching names A CSS style sheet may be attached to an XML document through the use of the special XML processing instruction <?xml-stylesheet?>. There are two attributes to the xml-stylesheet processing instruction: type and href. The type attribute sets the MIME type for the CSS style sheet. Its value should always be text/css. The href attribute gives the URL for the location of the CSS style sheet.

The href attribute of the xml-stylesheet processing instruction assumes that the CSS style sheet is located in the same directory. The href attribute value could also be a relative URL or an absolute URL.

Applying CSS to an XML Document

```
<?xml version=”1.0”?>
<!--
    This is referencing the style sheet from calling it notestyle.css here
-->
<?xml-stylesheet type=”text/css” href=”notestyle.css”?>
<Note>
    <From> From: Bob </From>
    <To> To: Jenny </To>
    <Subject> Subject: Hello Friend! </Subject>
    <Body> Just thought I would drop you a line. </Body>
</Note>
```

Output

From: Bob
To: Jenny
Subject: Hello Friend!
Just thought I would drop you a line.

(ii) XHTML

- XHTML stands for E**X**tensible **H**yper **T**ext **M**arkup **L**anguage
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Variants of XHTML

XHTML, a document must be validated against one of three DTDs

1. Strict
2. Transitional
3. Frameset

Strict DTD

A strictly conforming XHTML document that references the Strict DTD will have the following Document Type Declaration:

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Strict//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>

Strict DTD means that the XHTML document will have the following characteristics:

- There will be a strict separation of presentation from structure. Style sheets are used for formatting, and the XHTML markup is very clean and uncluttered. There are no optional vendor-specific HTML extensions.
- The Document Type Definition must be present and placed before the <html> element in the document.
- The root element of the document will be <html>.
- The <html> element will have the xmlns attribute in order to designate the XHTML namespace.
- The document is valid according to the rules defined in the Strict DTD.

UNIT-IV

MULTIMEDIA AND WEB APPLICATION

4.1 MULTIMEDIA IN WEB DESIGN

A few years back, the typical desktop computer's power, although considered substantial at the time, made it impossible to think of integrating high-quality audio and video into applications. Today's computers typically include CD-ROMs, sound cards and other hardware and special software which have make computer multimedia a reality.

Economical desktop machines are so powerful that they can store and play DVD-quality sound and video and we expect to see a huge advance in the kinds of programmable multimedia capabilities available through programming languages.

The multimedia revolution occurred first on the desktop computer, with the widespread availability of CD-ROMs. This platform is rapidly evolving towards DVD technology, but our focus in this chapter is on the explosion of sound and video technology that appears on the World Wide Web. In general, we expect the desktop to lead with the technology, because the Web is so dependent on bandwidth, and, for the foreseeable future, Internet bandwidths for the masses are likely to lag considerably behind those available on the desktop. One thing that Deitel has learned—having been in this industry for nearly four decades now—is to plan for the impossible. In the computer and communications a field, the impossible has repeatedly become reality so many times that it is almost routine at this point.

We discuss how to add sound, video and animated characters to Web based applications. Your first reaction may be a sense of caution because you realize that these are complex technologies and most readers have had little if any education in these areas. This is one of the beauties of today's programming languages. They give the programmer easy access to complex technologies and hide most of the complexity.

Multimedia files can be quite large. Some multimedia technologies require that the complete multimedia file be downloaded to the client before the audio or video begins playing. With streaming technologies, audio and video can begin playing while the files are downloading, to reduce delays. Streaming technologies are becoming increasingly popular.

Creating audio and video to incorporate into Web pages often requires complex and powerful software such as Adobe™ After Effects or Macromedia™ Director. Rather than discuss how to create media clips, this chapter focuses on using existing audio and video clips to enhance Web pages. The chapter also includes an extensive set of Internet and World Wide Web resources. Some of these Web sites display examples of interesting multimedia enhancements; others provide instructional information for developers planning to enhance their own sites with multimedia.

4.2 AUDIO AND VIDEO SPEECH SYNTHESIS AND RECOGNITION

Audio and Video:

- Audio and video can be used in Web pages in a variety of ways. Audio and video files can be embedded in a Web page or placed on a Web server such that they

can be downloaded “on-demand.” A variety of audio and video file formats are available for different uses.

- Common video file formats include MPEG (Moving Pictures Experts Group), Quick- Time, RealPlayer, AVI (Video for Windows) and MJPEG (Motion JPEG). Audio formats include MP3 (MPEG Layer 3), MIDI (Musical Instrument Digital Interface), WAV (Windows Waveform) and AIFF (Audio Interchange File Format—Macintosh only).

Encoding and compression determine a file’s format. An **encoding algorithm** or CODEC compresses media files by taking the raw audio or video and transforming it into a format that Web pages can read. Different encoding levels and formats produce file sizes that are ideal for different applications.
Some CODECs are available to the public in the form of **encoding applications**. Most encoding applications compress audio and video files. Some serve as **format converters**, converting one file format into another.

Adding Background Sounds with the bgsound Element:

- Some Web sites provide background audio to create a particular “atmosphere” on the site. Various ways exist to add sound to a Web page, the simplest is the bgsound element. The src property specifies the URL of the audio clip to play. Internet Explorer supports a wide variety of audio formats.
- The loop property specifies the number of times the audio clip will play. The value -1 (the default) specifies that the audio clip should loop until users browse a different Web page or click the browser’s Stop button. A positive integer indicates the exact number of times the audio clip should loop. Negative values (except -1) and zero values for this property cause the audio clip to play once.
- The balance property specifies the balance between the left and right speakers. The value for this property is between -10000 (sound only from the left speaker) and 10000 (sound only from the right speaker). The default value, 0, indicates that the sound should be balanced between the two speakers.
- The XHTML document of Fig demonstrates the bgsound element and scripting the element’s properties. This example’s audio clip came from the Microsoft Developer Network’s downloads site, msdn.microsoft.com/downloads/default.asp

Adding Video with the img Element’s dynsrc Property:

Users can tremendously enhance the multimedia presentations by incorporating a variety of video formats into their Web pages. The img element incorporates both images and videos in a Web page. The src property, shown previously, indicates that the source is an image. The dynsrc (i.e., dynamic source) property indicates that the source is a video clip. The dynsrc property may have other properties such as loop, which is similar to the bgsound loop property. The XHTML document of demonstrates the img element and its dynsrc property.

```
<?xml version = "1.0"?>
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Fig: Dynamicimg.html -->
<!-- Demonstrating the img element's dynsrc property -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
        <title>An Embedded Video Using the dynsrc Property</title>
        <bgsound src = "http://msdn.microsoft.com/downloads/sounds/carib.MID"
        loop = "-1"></bgsound>
</head>
<body>
        <h1>An Embedded Video Using the img element's dynsrc Property</h1>
        <h2>Car and Carribean Music</h2>
   <table>
        <tr>
                <td><img dynsrc = "car_hi.wmv"start = "mouseover" width =
        "180"height = "135"
                loop = "-"alt = "Car driving in circles" />
                </td>
                <td>This page will play the audio clip and video in a loop.<br />
                The video will not begin playing until you move the mouse over the
        video.<br />
                Press  <strong>Stop</strong> to stop playing the sound and the
        video.
                </td>
        </tr>
        </table>
</body>
</html>
```

Output:

- The img element in lines 21–24 uses the dynsrc property to load and display the video car_hi.wmv. Property start specifies when the video should start playing.

- There are two possible start events—fileopen indicates that the video should play as soon as it loads into the browser, and mouseover indicates that the video should play when users first position the mouse over the video.

Adding Audio or Video with the embed Element:

- Previously, we used elements bgsound and img to embed audio and video in a Web page. In both cases, users of the page have little control over the media clip. In this section, we introduce the *embed* element, which embeds a media clip (audio or video) into a Web page. The embed element displays a graphical user interface that gives users direct control over the media clip. When the browser encounters a media clip in an embed element, the browser plays the clip with the player registered to handle that media type on the client computer. For example, if the media clip is a wave file (i.e., a Windows Wave file), Internet Explorer typically uses the Windows Media Player ActiveX control to play the clip.
- The Windows Media Player has a GUI that enables users to play, pause and stop the media clip. Users can also control the volume of audio and move forward and backward through the clip using the GUI.
- The embed element is supported by both Microsoft Internet Explorer and Netscape navigator; however it is not part of the XHTML 1.0 recommendation. Documents written in XHTML using the embed element should render properly in either browser, however, errors may occur when trying to validate the document using the World Wide Web Consortium's XHTML 1.0 validator.
- The XHTML document of fig modifies the wave filter example by using an embed element to add audio to the Web page.
- Line 58 uses the embed element to specify that the audio file humming.wav should be embedded in the Web page. The loop property indicates that the media clip should loop indefinitely. The width and height properties define the size of the controls for the sound clip. By default, the GUI for the media player is displayed. To prevent the GUI from appearing in the Web page, add the hidden property to the <embed> element. To script the element, specify a scripting name by adding the id property to the <embed> element. The embed element can specify video clips as well as audio clips.

Demonstrates an embedded video:

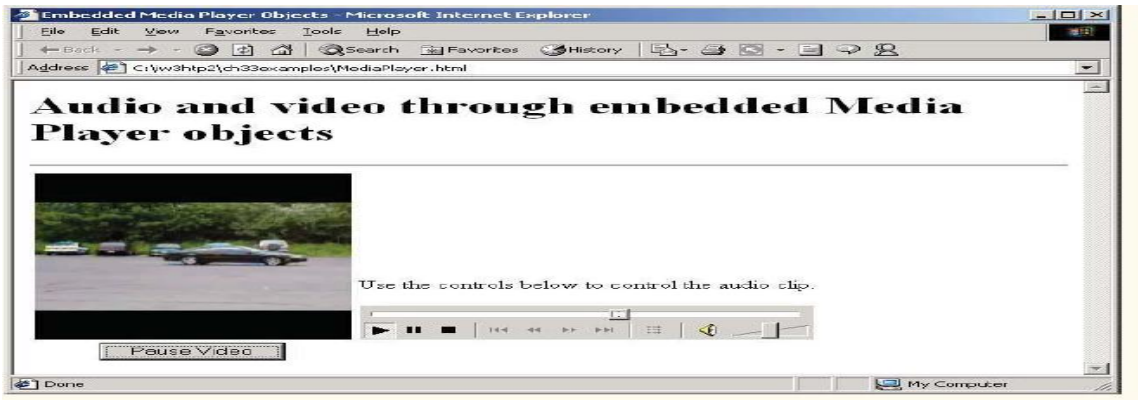
The embed element that loads and plays the video is located in line 18.

```
<?xml version = "1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- Fig: EmbeddedAudio.html -->
 <!-- Background Audio via the embed Element -->
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
        <title>Background Audio via the embed Element</title>
       <style type = "text/css">
              span { width: 600 }
```

```
                .big { color: blue; font-family: sans-serif;15 font-size: 50pt;16 font-
            weight: bold }
            </style>
             <script type = "text/javascript">
                <!--
                var TimerID;
                var updown = true;
                var str = 1;
                function start()
                {
                        TimerID = window.setInterval( "wave()", 100 );
                 }

                 function wave()
                {
                         if ( str > 20 || str < 1 )
                                updown = !updown;
                         if ( updown )
                                str++;
                         else
                                str--;
                        wft.filters( "wave" ).phase = str * 30;
                        wft.filters( "wave" ).strength = str;
                }
                 // -->
            </script>
        </head>
<body onload = "start()">
        <h1>Background Audio via the embed Element</h1>
        <p>Click the text to stop the script.</p>
        <p class = "big" align = "center">
        <span onclick = "window.clearInterval( TimerID )"
                id = "wft" style = "filter:wave( add = 0, freq = 3, light = 0, phase = 0,
        strength = 5)">
                 WAVE FILTER EFFECT</p>
        </span>
        <p>These controls can be used to control the audio.</p>
        <embed src = "humming.wav" loop = "true"></embed>
</body>
</html>
```

Embedding audio with the embed element:



4.3 ELECTRONIC COMMERCE

Any web site that offers products and/or services for sale is a commercial web site. There are thousands of commercial web sites on the Internet. Some of them have been successful, and some weren't so lucky. What elements make up a good commercial web site? Of course, web pages should look attractive to a customer. However, even the most attractive web pages will not make a person come back to a web site where it takes too long to find the right product or where order forms don't work. In this lecture we will discuss what functionality is needed for a successful commercial web site and what technology implements various web site elements.

Examples of commercial web sites

Let's browse the following two web sites, which are fairly typical web sites for online shopping.

- Online bakery. A web site for a small business.
- Trains.com Online store of model trains and related products. An example of a larger commercial web site.

Components of a well-functioning commercial web site

A well-organized web site should be easy for a customer to use and easy for the owner to maintain. It takes a lot to build a well-organized web site. Below is a list of the most important elements that a good web site should have:

- **A well-organized collection of products and/or services.** Smaller web sites can just list all their products on one or several web pages. Larger sites provide indices of products and search engines so that customers can find what they need. There should be a way for customers to get all necessary information about products, compare several products, get an advice on related products that they might want to get, etc.
    - To maintain a web site, the owner should be able to change product information easily. A web site where a price change of a single product requires changing 3 web pages is just bound to have inconsistencies. The owner should be able to add or remove products based on every day's availability, change prices, add product cross-references, etc. without making the web site inconsistent even for a minute.
- **A convenient way for a customer to select products.** Usually implemented as a shopping cart or a shopping basket. The customer should be able to select and delete products while browsing the web site.
- **Convenient order forms.** The form should be flexible enough to allow specifying a different address for the product delivery, a gift message, etc. It should have as few required fields as possible. A returning customer should be provided default information so that not to type it every time. For both the customer and the owner it is essential that the form catches simple typos (s.a. 4 digits in a zip code).
- **Convenient ways of payment.** There should be options of paying by a credit card, by a check (not everyone has credit cards!), and by a credit card over the phone if the customer is not comfortable sending his/her credit card number. The options may include some electronic payment systems.
  For the owner, there should be a quick way to verify the credit card information or in some other way to check that the payment is valid.
- **Secure communication system** not only to protect transmission of a credit card number, but also to guarantee privacy of the customer (including details of the purchase). A web site might have a user registration system with a password, in which case all transactions by the user should be private. It is also important to prevent unauthorized access to the web site (by a hacker or accidental).
- **Some way of storing information about customers.** This is convenient for customers so that they don't need to reenter their information every time they access the site. It also allows to "customizing" the web site for someone's interests. This can be done via customer registration or by means of "cookies", The owner can benefit a lot from storing information about a customer: he/she can customize ads based on the customer's profile or send an e-mail advertising a new product (but keep in mind that many people don't like this!) However, the greatest benefit is the owner's ability to monitor customer's behavior: which pages have the customer visited and which purchases (if any) he/she have made afterwards.
- **A way of keeping information about orders.** This allows customers to track their orders, and for owners to get all kinds of financial and statistical information. It is also important to keep order information in case of later disputes.
- The last, but not the least, **customer support and feedback**. There should be online documentation for all products ever sold on the web site, various FAQs,

and, ideally, a way of customers to post their opinion about the product. Easy access to this information may make a difference between a frequently visited web site and a lonely online looser.
This aspect of a web site cannot be completely mechanized: a human being has to answer e-mail, judge the relevance of customer's comments and organize comments by topics, and so on. However, there is a lot that can be computer-aided in this process, for instance sorting incoming messages by their title and/or return address to forward them to an appropriate customer support person.

We are only considering business-to-customer interactions here (so-called **B2C**), leaving out all the features related to business-to-business (**B2B**) interactions. While such interactions are basically similar to B2C, they can be optimized if a company has stable business partners which have more powerful computers than an average customer does. Even a customer oriented web site has a B2B part when dealing with vendors of products. We are not considering B2B connections in designing a web site.

Some technologies used to implement all this

- Convenient storage and retrieval of information about products and customers require a **database**.
- **Some terminology:** A database is a collection of data organized in such a way that it can be easily accessed, managed, and updated. A query is a request to a database written in a form that's supported by the database. Every brand of databases has its own language of queries (a query language).
- A well-organized database allows you to store each piece of information only once, so if you need to change it, you change it in one place. Databases allow you to store information about orders, needed both for you and for customers. For instance, customers can track their orders by requesting information from the database, which will automatically reply whether the order has been shipped. You can use customer information to fill in forms for returning customers with their recorded information.
- You can store statistics about customers (how many times they have visited the web sites, which pages, what did they buy, and so on). Customer's feedback about products can be organized and displayed later when someone else is interested in the same product. It's hard to imagine an interesting web site that does not have a database behind it.
  There are various kinds of databases, from a simple one, which is a collection of "flat" files storing data, to very sophisticated commercial products, s.a. ORACLE. The most common kind is a relational database.
- A relational database is a database organized as a collection of tables.
- A convenient interface to the database from the web site. A customer does not need to know anything even about the existence of the database, not to mention details of its organization. He/she should be able to get all necessary information by typing in keywords and filling in electronic forms.
- Various programming languages (for instance, Java, JavaScript) provide libraries to implement a remote connection to a database server.

- A server is a computer that provides a remote access to some service, for instance a web page server "serves" web pages, i.e. sends HTML files, graphics files, etc. in response to http requests, a database server provides responses to database queries, and so on. One machine can provide several different services at the same time.
- A client is a computer that makes a request for a service.
- We will use a Java **JDBC (Java Database Connectivity)** package to facilitate a connection to a relational database. It does not depend on a particular kind of the database, as long as the database supports **SQL (Structured Query Language).**
- SQL is a standard for a query language of a relational database**.**
- An interface has to provide way for customers to fill in forms, press buttons, etc. We will use javax.swing package to build an interactive graphical interface.
- We need to be able to process user's forms, for instance order forms, and produce various web pages, depending on the request. Ways of processing electronic forms include various server-side scripts, s.a. CGI, PERL, asp (Active Server Pages, an extension of JavaScript) and others. We will use Java Servlets for this purpose.
- A web site responds to requests for web pages. It doesn't "know" where the requests have come from, so it can't tell if two requests for web pages have been made by the same user. To keep track of a user during a session (and sometimes between sessions), a web server uses **cookies.**
- A cookie is a small text file placed by a web server on the client machine. The file gets sent back every time the client requests a web page from the server. A cookie has an expiration time, which may be just for the session or longer.
- Java Servlet package also implements cookies.
- **Encryption and security.** A common way secure communications are implemented is via SSL (Secure Sockets Layer), which allows various forms of encryption, depending on the maximum level of encryption provided by the server and the client and on geographic location of the machines (a different level of encryption is allowed in the US for connections to domestic and foreign computers). We will study and compare several encryption algorithms. On the practical side, we will use Java packages which implement encryption and secure communications.
- **Electronic payment systems** are based on customer's accounts with one of trusted vendors. A customer obtains certificates "signed" by the vendor which are analogous to checks ("personal money") and cash ("anonymous money"). The implementation should be such that mere copying of any piece of information transmitted in the transaction does not allow the thief to use the "money". We will study protocols for verifying validity of "checks" and "cash" in electronic transactions. We will also study ways of implementing "coins" on the Internet, i.e. certificates for small amounts of money.

4.4 E-BUSINESS model

Learning Objectives

- Identify the key components of e-commerce business models.
- Describe the major B2C business models.
- Describe the major B2B business models.
- Recognize business models in other emerging areas of e-commerce.
- Understand key business concepts and strategies applicable to e-commerce.

Components of e-Business Models

Fig 4.1 E-Business Model

Price Competition

- Price for books and CDs sold on the Internet less than conventional channel.
  - Average 9-16%
- Price increments
  - Price change on the Internet is smaller than conventional channel.
- Price dispersion
  - Substantial differences in price across retailers on the Internet.
  - Heterogeneity in consumer awareness
  - Heterogeneity in retailer branding and trust

E-Commerce Business Models

- Business model
  - a set of planned activities designed to result in a profit in a marketplace.
- E-commerce business model
  - a business model that aims to use and leverage the unique qualities of the Internet and the World Wide Web.

Value Position

- Defines how a company's product or service fulfills the needs of customers.
- Questions
  - Why will customers choose to do business with your firm instead of another company?
  - What will your firm provide that other firms do not and cannot?

Revenue Model

- Describes how the firm will earn revenue, produce profits, and produce a superior return on invested capital.
- E-commerce revenue models include:
  - advertising model
  - subscription model
  - transaction fee model
  - sales model
  - affiliate model
- Advertising revenue model
  - a company provides a forum for advertisements and receives fees from advertisers (Yahoo).
- Subscription revenue model
  - a company offers it users content or services and charges a subscription fee for access to some or all of it offerings (Consumer Reports or Wall Street Journal).
- Transaction fee revenue model
  - a company receives a fee for enabling or executing a transaction (eBay or E-Trade).
- Sales revenue model
  - a company derives revenue by selling goods, information, or services (Amazon or DoubleClick).
- Affiliate revenue model
- a company steers business to an affiliate and receives a referral fee or percentage of the revenue from any resulting sales (MyPoints).

Market Opportunity

- Market opportunity
  - refers to the company's intended market space and the overall potential financial opportunities available to the firm in that market space.
  - defined by the revenue potential in each of the market niches where you hope to compete.
- Market space
  - the area of actual or potential commercial value in which a company intends to operate.

Competitive Environment

- Refers to the other companies operating in the same marketplace selling similar products.
- Influenced by:
  - how many competitors are active
  - how large are their operations
  - the market share of each competitor
  - how profitable these firms are
  - how they price their products

Market Strategy

- The plan you put together that details exactly how you intend to enter a new market and attract new customers.
- Best business concepts will fail if not properly marketed to potential customers.

Organizational Development

- Describes how the company will organize the work that needs to be accomplished.
- Work is typically divided into functional departments.
- Move from generalists to specialists as the company grows.

Management Team

- Employees of the company responsible for making the business model work
- Strong management team gives instant credibility to outside investors
- A strong management team may not be able to salvage a weak business model
- Should be able to change the model and redefine the business as it becomes necessary

Portal

- offers powerful search tools plus an integrated package of content and services.
- typically utilizes a combines subscription/advertising revenues/transaction fee model may be general or specialize (portal).

E-tailer

- online version of traditional retailer
  It includes
  - virtual merchants (online retail store only)
  - clicks and mortar e-tailers (online distribution channel for a company that also has physical stores)
  - catalog merchants (online version of direct mail catalog)
  - online malls (online version of mall)
  - Manufacturers selling directly over the Web

Content Provider

- information and entertainment companies that provide digital content over the Web.
- typically utilizes an advertising, subscription, or affiliate referral fee revenue model.

Transaction Broker

- processes online sales transactions typically utilizes a transactions feel revenue model.

Market Creator

- uses Internet technology to create markets that bring buyers and sellers together.
- typically utilizes a transaction fee revenue model.
- E.g. Auction

- English auction
- Dutch auction
- Sealed-bid auction
- Double auction

B2B Hub

- also known as marketplace/exchange electronic marketplace where suppliers and commercial purchasers can conduct transactions may be a general (horizontal marketplace) or specialized (vertical marketplace)

E-distributor

- supplies products directly to individual businesses

B2B Service Provider

- sells business services to other firms

Matchmaker

- links businesses together
- charges transaction or usage fees

Infomediary

- gather information and sells it to businesses

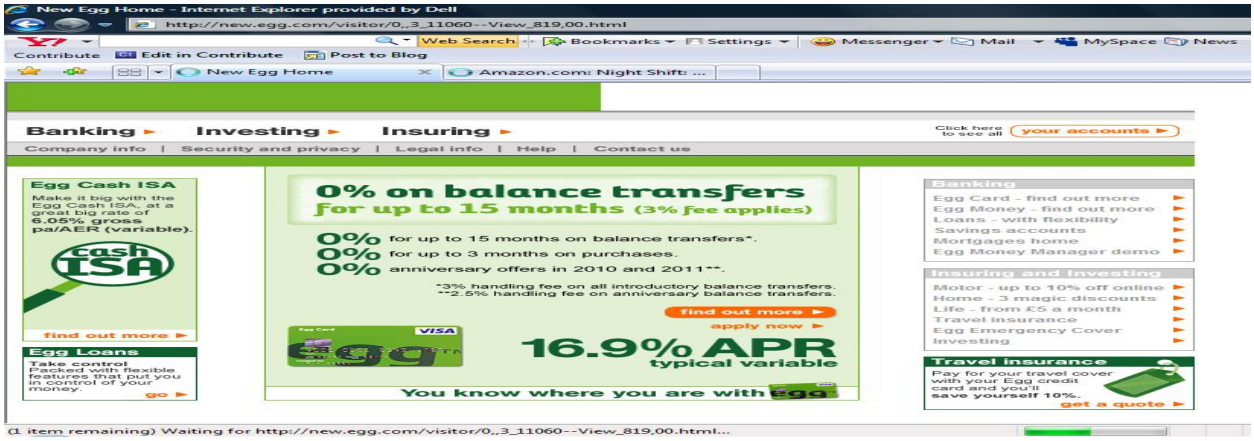
4.5 E-MARKETING

- These 7 Cs e-marketing are fundamental to understanding the intricacies of Internet marketing and to transforming a venture from being a mere web presence to a highly successful e-venture. They are also instrumental in shaping the overall business strategy and the economic model that an organization needs to adopt.
- The value propositions of products and services offered in the physical world are essentially limited “point solutions” that meet only part of a consumer’s need or want. In the online world, even a simple banner advertisement can be both an advertisement and a direct marketing service. The banner raises the passive consumer’s awareness of a product. Yet it also encourages the consumer to pursue action by clicking on it.
- **E-marketing** must be defined to include the management of the consumer’s online experience of the product, from first encounter through purchase to delivery and beyond. Digital marketers should care about the consumer’s online experiences for the simple reason that all of them -- good, bad, or indifferent -- influence consumer perceptions of a product or a brand. The web offers companies’ ownership and control of all interactions with customers and thus creates both the ability and the need to improve their overall experience.
- There are two reasons for building the concept of e-marketing around consumer experiences. First, this approach forces marketers to adopt the consumer’s point of view. Second, it forces managers to pay attention to all aspects of their digital brand’s interactions with the consumer, from the design of the product or service

to the marketing message, the sales and fulfillment processes, and the after-sales customer service effort.

The 7 Cs of E-Marketing

- The Internet allows for the entire sales cycle to be conducted on one medium, nearly instantaneously. From making the consumer aware of the product to providing additional information to transacting the final purchase, the Internet can accomplish it all.
- The Internet is like one big point-of-sales display, with easy access to products and the ability for impulse shopping. Impulse shoppers have found a true friend in the Internet. Within seconds from being made aware of a product, consumers can purchase it online. Further, with the targeting techniques available to advertisers, consumers who turn down a product because of the price can be identified and served a special offer more likely to result in a purchase. In the right hands, with the right tools, the Internet really is an advertiser's dream come true.



As opposed to the 4 Ps of brick-and-mortar marketing, the changing outlook in the area of e-marketing can be explained on the basis of 7 Cs of e-marketing.

- **Contract:** The e-marketer's first goal is to communicate a core promise for a truly distinctive value proposition appealing to the target customers.

- **Content:** refers to whatever appears on the website itself and on hot linked websites. If chosen appropriately, it can increase both the rates at which browsers are converted into buyers and their transactions.

- **Construction:** The promises made by e-marketers are not unique to the Internet, but the medium's interactive capabilities make it easier for them to deliver on their promises quickly, reliably, and rewardingly.
    - In practice, this means that promises must be translated into specific interactive functions and Web design features collectively giving

consumers a seamless experience. Such design features as one-click ordering and automated shopping help deliver the promise of convenience.

- **Community:** Through site-to-user and user-to-user forms of interactivity (such as chat rooms), e-marketers can develop a core of dedicated customers who become avid marketers of the site too.

- **Concentration:** Targeting through online behavioral profiling. Advertisers have known for some time that behavioral targeting (a.k.a., profiling) is vastly superior to simple demographic targeting. Knowledge of a consumer's past purchases interests, likes/dislikes, and behavior in general allows an advertiser to target an advertisement much more effectively. Department stores have long kept track of consumers' past purchases. They are thus able to project what other types of products a consumer might be interested in and then send an appropriate coupon or sale offer.
    - Credit card companies are the ultimate gatherers of behavioral targeting information. They maintain vast databases of cardholders' past transactions, and they sell lists of this data to advertisers. The same type of behavioral model is forming on the Internet. Publishers and advertisement networks monitor the items that a consumer has expressed interest in or purchased on a site (or network of sites) in the past and target advertisements based on this information.

- **Convergence:** We will soon enter the next round of the E-marketing battle as broadband reaches the masses. The Internet will become more ubiquitous and wireless; televisions will become more interactive; video/data/voice appliances will converge; brand advertising and direct marketing practices will integrate; domestic brands, commerce and marketing will become even more global; and big marketing spenders will spend more money online. Many companies that are well positioned today will need to continue to evolve to take advantage of the opportunities.
    - The success of Internet advertising companies will largely be driven by how they maneuver among the coming developments. Rich media, brought on by broadband, will allow advertisers much greater creativity by bringing in new types of advertising to the Internet, as well as enhancing some of the more traditional forms. Broadband technology will allow the convergence of television and the Internet.
    - Dubbed "interactive TV," in its simplest form, will consist of a television with some interactive capabilities. Basically, a user will see a television screen that is three-quarters traditional television, but with a frame that has Internet capabilities. This frame will allow users to access up-to-the-minute sports scores or news on the Web, for example. More importantly for E- marketers, it would allow viewers to immediately leap to the website of an advertiser whose ad was being shown. The user could find out more information or order the product right there.

- **Commerce:** The last emerging fundamental of e-marketing is commerce, whether it includes offering goods and services directly, or marketing those of another company for a fee, thus helping to cover the fixed costs of site operations and to offset customer acquisition costs.

To be successful on the Internet, e-marketers will have to do more than reproduce their off-line business models on line because these business models work only at considerable scale. Interestingly, It is possible for online marketers to be profitable even at lower sales volume if they exploit efficiencies in e-marketing and synergies with the off-line business, with examples as follows.

Exploiting more than one channel to close the transaction

- Although early winners on the web might belong to an exclusive club of Internet start-up companies, established players in the off-line industry can catch them and even overtake them by offering a choice of channels.

Leveraging low customer acquisition costs:

- Traditional brick-and mortar companies can bring their existing customers online at a much lesser cost than Internet start-up companies who must lay out a hefty amount per head to acquire customers.

Exploiting alternative revenue streams:

- An online presence offers an E-marketer a wider variety of sales opportunities. For web-based retailers, acting as an agent on behalf of the customer can become a revenue source in the future.

Purchasing scale at low volumes:

- E-marketers can cut down on their purchasing cost and shorten their procurement cycle by replacing EDI tools with Internet based ones that facilitate product comparison, streamline logistics, and help B2B vendors aggregate their retailer's back office purchases.

Reducing customer churn:

- Given the high cost of replacing established customers, losing them is expensive. A web presence supplies the personalized attention that could keep customers loyal.

Maximizing the pricing potential:

- It has been reported by consumer researchers that buyers shop online more for convenience than for cost. In view of this relative indifference to price, e-marketers can capture some margin premium, at least in the early days of their sites.

Challenges in e-marketing:

- Every online fulfillment operation, large or small, faces four main challenges: controlling customer data, integrating on- and off-line orders, delivering the goods cost effectively, and handling returns.

Controlling customer data:

- As outsourcing arrangements proliferate and delivery services become more expert in using information technology, e-marketers risk losing their lock on consumer data. In an economy where knowledge is revenue as well as power, e-marketers must consider how to strike a balance between the efficiencies offered by the out-sourcing of fulfillment and the confidentiality that keeping data in-house preserves.

Integrating on and off-line orders:

- When the volume of orders is high, companies must decide how much integration they need. In a totally integrated system, Internet orders would be automatically transmitted through a processing center and transferred to the supplier's manifest. An integrated system with full ERP (enterprise resource-planning) capabilities, for example, can ensure that surges in demand don't retard key fulfillment operations such as data entry, inventory, and packing.

Delivering the goods cost-effectively:

- At present, every single transaction challenges e-marketers to deliver the goods quickly, cheaply, and conveniently. But this is largely a technical and logistical problem, and it will be possible (though perhaps expensive) to solve it by developing new sorting and scanning equipment and by deploying larger delivery vehicles. Making contact with the recipient is a trickier problem but one that must be resolved if the full potential of "eimpulse" orders is to be realized, for an impulse purchase loses its power to gratify if the product or service takes too long to appear. But since each missed delivery adds as much as a full day to the fulfillment process, spanning that "final mile" to the home can take longer than traveling the rest of the fulfillment loop.

Handling returns:

- E- Marketers, with their emphasis on convenience and customization, must match the high standard of service exhibited by some physical marketers regarding returns. At present, they do not. To begin with, few ecommerce retailers (or mail order companies, for that matter) design their packaging for easy returns. Customers often have to find new packing materials; calls to arrange credits and refunds, and physically take packages to delivery services. Each step represents an inconvenience that, however minor, can combine with others to create negative feelings about the vendor. Even if a convenient solution for returns were developed, e-marketers might discover that impulse sales carry hidden costs. The implication is that fulfillment costs must be driven down to preserve profitability.

Choice of Marketing Strategy:
An online company's choice of marketing strategy will depend on four main variables: the nature of the customer's interaction with the product and seller; the current capabilities of the business; the capabilities that are (or will become) "commodity" operations, in which competitive advantage cannot be sustained; and the trade-off between time and control. This is essentially a value chain concept (as propounded by Porter) whereby e-marketers look at each component of the value chain and the support activities to determine where and in what form can they add value to the customer. This translates into their competitive advantage. And in this entire process, it is the 7 Cs of e-marketing that act as fundamental guiding principles.

Rethinking the business model:

- As e-marketers align the "contract" and the "construction," they must also align the economic model that will sustain their businesses. For most of them, the very process of taking the brand online will force a fundamental reconsideration of the business.

Channel supporter:

- E-marketers can use the Internet more to support their existing channels than to generate additional sales. Beyond cross-channel promotions, many brick-and-mortar companies can use the web to increase their customers' understanding of their products and services. Others can harness the web's interactivity to improve their product development and product mixes by inviting customer responses on their web sites.

Advisory and information service providers:

- An expert (such as an investment adviser or a personal shopper) can offer consumers unbiased advice for a fee. A business can also collect, process, and sell information through the Internet.

Retail model:

- Vendors or products can be aggregated to facilitate transactions for buyers. Many companies can also achieve success as online auctioneers. Sellers of goods and services can provide the content; community may come from matching sellers with buyers and setting bidder against bidder; and commissions on sales and advertising revenue can generate the commerce.

Vertical model:

- The business model that may take the greatest advantage of the Internet is the vertical model, which specializes, in a particular category or a product. It might provide specialized information and advice as well as access to a community with common interests.

4.6 ONLINE PAYMENTS AND SECURITY

The information technology, the Internet high speed development, electronic commerce has caused the current distribution realm significant transformation

gradually. In the electronic commerce practice, the online electronic payment is the electronic commerce essential link, also is the foundation condition which electronic commerce can smoothly develop. Not the corresponding real-time electron payment means coordinate, electronic commerce only can be does not have the practical significance "the hypothesized commerce", but is unable to realize on the genuine net the transaction.

The on-line electronic payment is the electronic commerce development core, is completes on the net the transaction essential step, also is at present restricts the domestic network application development a bottleneck.

Online Electronic Payment

- Online electronic payments are not tantamount to electronic payments. In the emergence of e-commerce, credit cards have long been represented by electronic means of payment, credit cards in shopping malls.
- Many hotels and other places and items could swipe of the card, POS terminals Regulations, ATM cash forms of payment. And online electronic payments, online payments also known as electronic currency, broadly speaking, refer to a transaction in the online exchange of funds; It is a network-based electronic financial, a business card transactions for all types of electronic tools and media, the electronic computer and communications technologies as a means Electronic data (binary data) stored in the bank's computer system and through the computer network system in the form of the flow of electronic information transfer and payment.
- Electronic Payment System is the basis for online payments, and online payments system development is a higher form of electronic payment. It makes electronic payment may, at any time, through the Internet directly to the transfer, settlement and form e-business environment.

Common Online Electronic Payment System

In online shopping online electronic payment function is the key issue to ensure the consumers are fast and convenient, we have to ensure the safety and secrecy of the parties to a transaction, which requires a complete electronic trading systems. Currently, several online electronic payment systems used for:

Internet Bank Card Payment System

Including online credit card, smart card (IC card) payment systems are established in accordance with the standards set shopping and payment system. Internet users in specific ways: sending banks coast and password encryption sent to the bank for payment. And the payment process for customers, merchants and verify the legitimacy of a request for payment. At present, domestic banks had set up such a bank cards for online payments.

Based on the bank card payment the following four models:

1. No security model

Its features: users complete control of the bank card business information, the transmission of messages without bank card security.

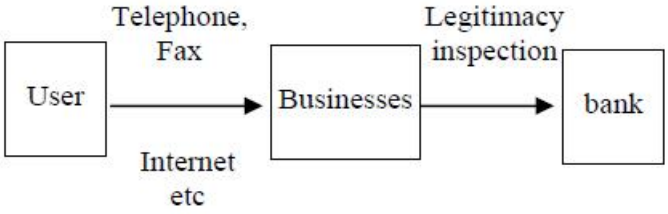


Fig 4.1: No security model

2. through third-party brokers paid model

Its characteristic is as follows: Bank card information is not open to the transmission network, is paid by users. Both businessmen trusted third party (agents) to complete.

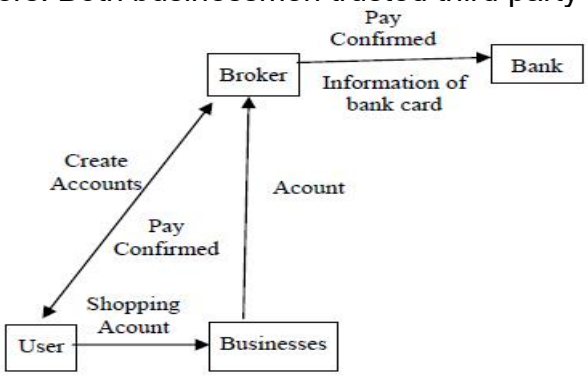


Fig 4.2 Third-Party Brokers Paid Model

3. Simple encrypted payment system model

Its characteristic is as follows: the use of encryption technology to bank cards and other critical information encrypted digital signature to confirm the authenticity of the message. Business servers and the need for software support services.

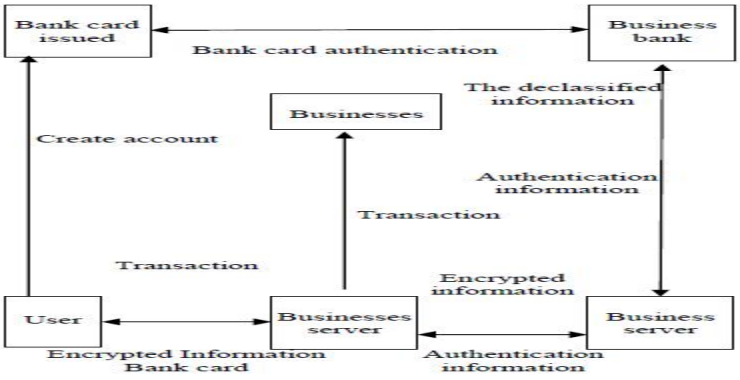


Fig 4.3 Encrypted Payment System Model

4. SET (security electronic transaction) model

"Secure Electronic Transactions," and referred to the SET. In an open Internet is a realization of the international agreements and standards for secure electronic

transactions. Their characteristics are as follows: SET transactions participants to provide certification to ensure data security, integrity and no repudiation of transactions, in particular to ensure that no information leaked to the cardholder's account for the businesses. Guarantee the safety of the SET.

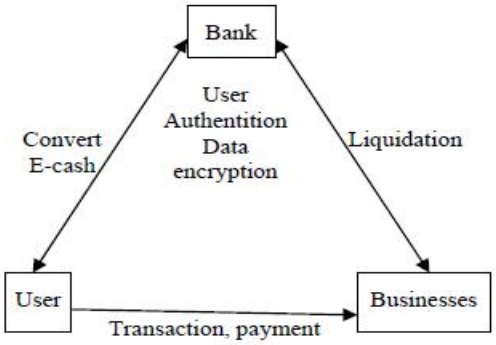


Fig 4.4 SET Model

E-Cash (Electronic-cash) Internet Payment System:

- E-cash is a form of data, the currency in circulation, there is electronic cash currency; it can be converted to cash a series of encrypted numerical sequence number, and then use these sequences to show the value of all sizes.
- Its characteristic is as follows: an agreement between the banks and businesses and authorization, identity verification by e-cash to complete it, electronic cash can be kept, admission, and transfer to smaller transactions.

E-cash and e-payment systems also have the advantage of cash, mainly as follows:

- Anonymity;
- Not shadowing;
- Savings on transaction costs;
- Savings on transmission costs;
- Poor risk;
- Pay flexibility;
- Prevent forgery and repeatability.

E-purse Internet Payment System

Users use e-purse shopping, the first in a personal bank account and users into a certain amount; then the corresponding electronic wallet service system free software to download and install an electronic purse; then download the corresponding website to apply online and access the cardholder "electronic safety certificate". Users shopping, the only direct hits "electronic wallet" icon and following the importation of their coast. Corresponding information such as passwords by e-purse will pay to complete the follow-up work. Modern foreign companies such as a smart card as an e-purse to the online payment system. E-purse is sporadic small payment transactions always used in conjunction with bank cards to help users complete the entire shopping process.

Electronic check (E_check) Internet Payment System

Electronic check transfer payments from paper checks to the merits of using digital transmission o transfer money from one account to another account. These electronic check payments in businesses and banks linked to the online password transmission. Most common use encryption keys handwritten signature or personal identification numbers instead of signatures. Thus ensuring the safety of this form of payment.

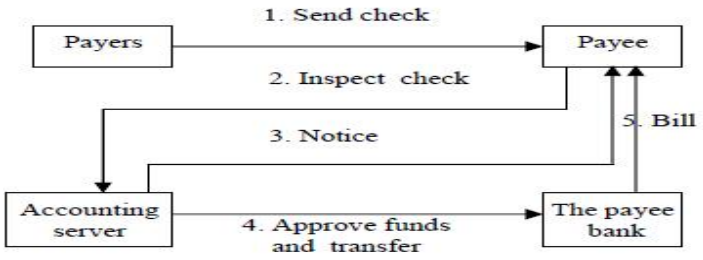


Fig 4.5 E_check Internet Payment System

STRATEGY OF E-COMMERCE SECURITY:

As e-commerce security problems caused by many factors, to solve the security problem from different aspects, offers a variety of countermeasures.

1. Security Strategy

- To ensure the safety communications must be the necessary measures to guard against them.
- Communications links, we can use a firewall, proxy server, Virtual Private Network (VPN) technology; in the identification and authentication, encryption and authentication techniques.

2. Legal Protection

- As e-commerce activities are a commodity transaction and security issues should be protected by law must ensure that the legal status of electronic contracts and digital signatures, electronic contracting parties to the contract approved Electronic Contract denied or modified to ensure that electronic contracts can be implemented.

3. Social Moral Norms

- As e-commerce transactions are not direct, face-to face features Transactions are often seen in the traditional process of e-commerce fraud is bound to have security implications. Thus, the healthy development of ecommerce depends on the establishment and perfection of social ethics.

4. Perfect Management Strategy

- As e-commerce transaction system is a highly integrated man-machine system, in addition to network security, and management is also very important, but the factors that play a decisive role. Thus, the whole system of power distribution management and supervision, management training and assessment, ethical and professional standards must draw up complete training regulations, management jobs in order to enhance the spirit of love.

4.7 WORKING OF SEARCH ENGINES

Three basic tasks are performed

1. Searching the internet, selecting documents based on important words
2. Keep an index of the words they find, and where they find them.
3. Allow users to look for the words or combinations of words found in that index.

Gathering pages (web crawling)

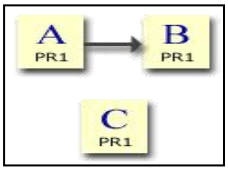
- Programs called Spiders; robots etc. collect pages of the Web for indexing.
- Start processing with a certain number of starting point URLs and follow other links.
- Visited pages or portions of pages are saved for analysis.

Indexing pages

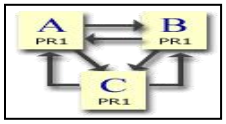
- Indexing – determining what a page is about.
- Looks for the components of a page like
  - <title>
  - <meta> tags
  - Link texts
  - Texts in headings and body
  - Comment text
- Tries to distill the meaning of the page.
- Position or frequency of the text will be considered.
- Stop words, the more common and little useful ones, are ignored.
- Pages with similar key words are ranked and are stored in the database.

Page Rank

- A numeric value that represents how important a page is, on the web.
- When one page links to another, it is effectively casting a vote for the other page.
- The more votes cast on a page, the more important the page is.
- Importance of the page that casts the vote also determines the importance of the vote.
- A page's importance is calculated from the votes cast for it.
- Page A has voted for Page B but not to Page C.
- Hence Page Rank for each page is
  Page A = 0.15
  Page B = 1
  Page C = 0.15



- Consider the linking of all pages to all pages.
- This produces page rank as
  Page A = 1
  Page B = 1
  Page C = 1 resulting in maximum Page Rank.



Providing a Search Mechanism

- Deals with search interface and result page

- Search page – an interface, the user makes their query form.
- Contains primary query text box, varies to advanced searches.
- Basic and advanced searches differ in interfaces.

4.8 DESIGNING A SEARCH INTERFACE:

- Search form must fit the type of data being searched.
- HTML and Scripting languages are used.
- Primary element – Search query field.
- Second aspect – Button to execute the search. (Form buttons or Custom buttons)

Result Page Design

- Must contain as much information as possible relevant to user's query.
- Elements of a result page
  - Original query
  - Scope of the search and the results found
  - Page or document titles
  - Page summaries
  - URL of returned page
  - Size and Type of result
  - Relevancy of results
  - Keyword matches

  - Navigation

Negative results page

- Tries to help the user identify what went wrong and says why a query went wrong.
- Possibilities – nothing may match the search terms; search facility may be used incorrectly.
- Functions:
  - Clear failure message
  - Search again mechanism
  - Help information

UNIT-V

WEB SERVICES

- Web Services can convert your application into a Web-application, which can publish its function or message to the rest of the world.
- The basic Web Services platform is XML + HTTP.

5.1 INTRODUCTION TO WEB SERVICES

- Web Services can convert your applications into Web-applications.
- Web Services are published, found, and used through the Web**.**

What are Web Services?

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- XML is the basis for Web services

How does it Work?

- The basic Web services platform is XML + HTTP.
- XML provides a language which can be used between different platforms and programming languages and still express complex messages and functions.
- The HTTP protocol is the most used Internet protocol.

Web services platform elements:

1. SOAP (Simple Object Access Protocol)
2. UDDI (Universal Description, Discovery and Integration)
3. WSDL (Web Services Description Language)

Why Web Services?

Interoperability has Highest Priority

- When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed.
- Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.

Web Services take Web-applications to the Next Level

- By using Web services, your application can publish its function or message to the rest of the world.
- Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).
- With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.

Web Services have Two Types of Uses

1. **Reusable application-components**
   - Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.
2. **Connect existing software**
   - Web services can help to solve the interoperability problem by giving different applications a way to link their data.
   - With Web services you can exchange data between different applications and different platforms.

Web Services Platform Elements

Web Services have three basic platform elements:

1. SOAP
2. WSDL and
3. UDDI

5.2 UDDI

Universal Description, Discovery and Integration (UDDI) are a directory service where businesses can register and search for Web services.

What is UDDI

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.

- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- UDDI is built into the Microsoft .NET platform

What is UDDI Based On?

- UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols.
- UDDI uses WSDL to describe interfaces to web services
- Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

UDDI Benefits

- Any industry or businesses of all sizes can benefit from UDDI.
- Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.

Problems the UDDI specification can help to solve

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers

- Expanding offerings and extending market reach
- Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
- Describing services and business processes programmatically in a single, open, and secure environment

How can UDDI be Used

- If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory.
- Travel agencies could then search the UDDI directory to find the airline's reservation interface.
- When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

Who is Supporting UDDI?

- UDDI is a cross-industry effort driven by all major platform and software providers like Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun, as well as a large community of marketplace operators, and e-business leaders.
- Over 220 companies are members of the UDDI community.

UDDI has two parts:

1. A registry of all a web service's metadata including a pointer to the WSDL description of a service.
2. A set of WSDL port type definitions for manipulating and searching that registry.

UDDI Elements

A business or company can register three types of information into a UDDI registry. This information is contained into three elements of UDDI.
These three elements are:

(1) White pages:

This category contains:

- Basic information about the Company and its business.
- Basic contact information including business name, address, contact phone number etc.
- A unique identifiers for the company tax IDs. This information allows others to discover your web service based upon your business identification.

(2) Yellow pages:

This category contains:

- This has more details about the company, and includes descriptions of the kind of electronic capabilities the company can offer to anyone who wants to do business with it.
- It uses commonly accepted industrial categorization schemes, industry codes, product codes, business identification codes and the like to make it easier for companies to search through the listings and find exactly what they want.

(3) Green pages:
This category contains technical information about a web service. This is what allows someone to bind to a Web service after it's been found. This includes:

- The various interfaces
- The URL locations
- Discovery information and similar data required to find and run the Web service.

UDDI Technical Architecture:
The UDDI technical architecture consists of three parts:

1. **UDDI data model:**
   - An XML Schema for describing businesses and web services. The data model is described in detail in the "UDDI Data Model" section.

2. **UDDI API Specification:**
   - A Specification of API for searching and publishing UDDI data.

3. **UDDI cloud services:**
   - This is operator sites that provide implementations of the UDDI specification and synchronize all data on a scheduled basis.

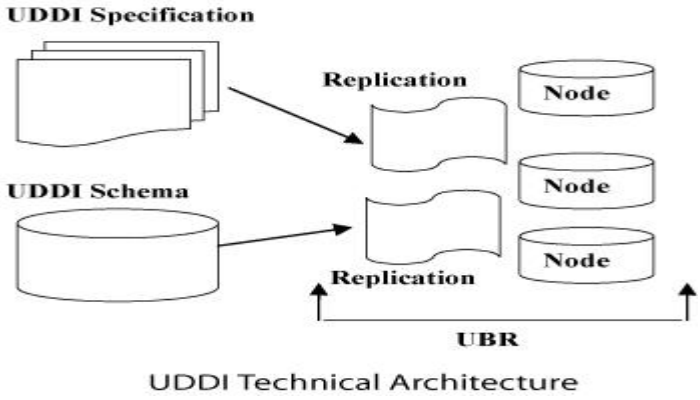


Fig 5.1: UDDI Architecture

- The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes that has their data synchronized through replication.
- The current cloud services provide a logically centralized, but physically distributed, directory. This means that data submitted to one root node will automatically be replicated across all the other root nodes. Currently, data replication occurs every 24 hours.
- UDDI cloud services are currently provided by Microsoft and IBM.

It is also possible to set up private UDDI registries. For example, a large company may set up its own private UDDI registry for registering all internal web services. As these registries are not automatically synchronized with the root UDDI nodes, they are not considered part of the UDDI cloud.

UDDI Data Model:
UDDI includes an XML Schema that describes four five data structures:

1. businessEntity
2. businessService
3. bindingTemplate
4. tModel
5. publisherAssertion

Business Entity data structure:

- The business entity structure represents the provider of web services. Within the UDDI registry, this structure contains information about the company itself, including contact information, industry categories, business identifiers, and a list of services provided.

Business Service Data Structure:

- The business service structure represents an individual web service provided by the business entity. Its description includes information on how to bind to the web service and what type of web service it is.

Binding Template Data Structure:

- Binding templates are the technical descriptions of the web services represented by the business service structure. A single business service may have multiple binding templates. The binding template represents the actual implementation of the web service.

5.3 SOAP

SOAP INTRODUCTION:

- SOAP is a simple XML-based protocol to let applications exchange information over HTTP.
- SOAP is a protocol for accessing a Web Service.

What is SOAP?

- SOAP stands for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is for communication between applications
- SOAP is a format for sending messages
- SOAP communicates via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to get around firewalls
- SOAP is a W3C recommendation

Why SOAP?

- It is important for application development to allow Internet communication between programs.
- Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

- A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP SYNTAX:

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

1. An **Envelope element** that identifies the XML document as a SOAP message.
2. A **Header element** that contains header information.
3. A **Body element** that contains call and response information.
4. **A Fault element** containing errors and status information.

All the elements above are declared in the default namespace for the SOAP envelope:

http://www.w3.org/2001/12/soap-envelope

and the default namespace for SOAP encoding and data types is:

http://www.w3.org/2001/12/soap-encoding

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message MUST use the SOAP Encoding namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instruction

Skeleton SOAP Message

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Header>
      ...
    </soap:Header>
    <soap:Body>
        ...
        <soap:Fault>
          ...
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

1. SOAP Envelope Element:

- The required SOAP Envelope element is the root element of a SOAP message.
- This element defines the XML document as a SOAP message.

Example

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

The xmlns:soap Namespace

- Notice the xmlns:soap namespace in the example above. It should always have the value of: "http://www.w3.org/2001/12/soap-envelope".
- The namespace defines the Envelope as a SOAP Envelope.
- If a different namespace is used, the application generates an error and discards the message.

The encoding Style Attribute

- The encoding Style attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and applies to the element's contents and all child elements.
- SOAP message has no default encoding.

Syntax

soap:encodingStyle="URI"

Example

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

2. SOAP Header:

The SOAP Header Element

- The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.
- If the Header element is present, it must be the first child element of the Envelope element.

Note: All immediate child elements of the Header element must be namespace-qualified.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
    <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
```

```
        soap:mustUnderstand="1">234
      </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

- The example above contains a header with a "Trans" element, a "must understand" attribute with a value of 1, and a value of 234.
- SOAP defines three attributes in the default namespace ("http://www.w3.org/2001/12/soap-envelope"). These attributes are: must Understand, actor, and encoding Style.
- The attributes defined in the SOAP Header defines how a recipient should process the SOAP message.

The must Understand Attribute

- The SOAP must understand attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process.
- If you add must Understand="1" to a child element of the Header element it indicates that the receiver processing the Header must recognize the element. If the receiver does not recognize the element it will fail when processing the Header.

Syntax

soap:mustUnderstand="0|1"

The actor Attribute

- A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path. However, not all parts of a SOAP message may be intended for the ultimate endpoint, instead, it may be intended for one or more of the endpoints on the message path.
- The SOAP actor attribute is used to address the Header element to a specific endpoint.

Syntax

soap:actor="URI"

The encoding Style Attribute

- The encoding Style attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and it will apply to that element's contents and all child elements.
- A SOAP message has no default encoding.

Syntax

soap:encodingStyle="URI"

3. SOAP Body Element

- The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.
- Immediate child elements of the SOAP Body element may be namespace-qualified.

Example

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
</soap:Envelope>
```

The example above requests the price of apples. Note that the m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.
A SOAP response could look something like this:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

4. SOAP Fault Element

- The SOAP Fault element holds errors and status information for a SOAP message.
- The optional SOAP Fault element is used to indicate error messages.
- If a Fault element is present, it must appear as a child element of the Body element. A Fault element can only appear once in a SOAP message.

SOAP Example

In the example below, a GetStockPrice request is sent to a server. The request has a StockName parameter, and a Price parameter that will be returned in the response. The namespace for the function is defined in "http://www.example.org/stock".

SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
```

```
            </soap:Body>
        </soap:Envelope>
```

SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
    <?xml version="1.0"?>
        <soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
        soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
            <soap:Body xmlns:m="http://www.example.org/stock">
                <m:GetStockPriceResponse>
                <m:Price>34.5</m:Price>
                </m:GetStockPriceResponse>
            </soap:Body>
        </soap:Envelope>
```

5.4 WSDL

WSDL (Web Services Description Language) is an XML-based language for describing Web services and how to access them.

Introduction to WSDL

- WSDL is an XML-based language for describing Web services and how to access them.

What is WSDL?

- WSDL stands for Web Services Description Language
- WSDL is written in XML
- WSDL is an XML document
- WSDL is used to describe Web services
- WSDL is also used to locate Web services
- WSDL is a W3C recommendation

WSDL Describes Web Services

- WSDL stands for Web Services Description Language.
- WSDL is a document written in XML. The document describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes.

WSDL Documents

- A WSDL document is just a simple XML document.
- It contains set of definitions to describe a web service.

The WSDL Document Structure:

A WSDL document describes a web service using these major elements:

Element	Description

<types>	A container for data type definitions used by the web service
<message>	A typed definition of the data being communicated
<portType>	A set of operations supported by one or more endpoints
<binding>	A protocol and data format specification for a particular port type

The main structure of a WSDL document looks like this:

```
<definitions>
    <types>
        data type definitions.....
    </types>
    <message>
        definition of the data being communicated....
    </message>
    <portType>
        set of operations.....
    </portType>
    <binding>
        protocol and data format specification....
    </binding>
</definitions>
```

A WSDL document can also contain other elements, like extension elements, and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

WSDL Ports:

- The **<portType>** element is the most important WSDL element.
- It describes a web service, the operations that can be performed, and the messages that are involved.
- The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language.

WSDL Messages

- The **<message>** element defines the data elements of an operation.
- Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

WSDL Types

- The **<types>** element defines the data types that are used by the web service.
- For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Bindings

- The **<binding>** element defines the data format and protocol for each port type.

WSDL Example

This is a simplified fraction of a WSDL document:

```
<message name="getTermRequest">
    <part name="term" type="xs:string"/>
```

```
</message>
<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
        <operation name="getTerm">
                <input message="getTermRequest"/>
               <output message="getTermResponse"/>
        </operation>
</portType>
```

- In this example the **<portType>** element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.
- The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".
- The **<message>** elements define the **parts** of each message and the associated data types.
- Compared to traditional programming, glossaryTerms is a function library, "getTerm" is a function with "getTermRequest" as the input parameter, and getTermResponse as the return parameter.

WSDL PortType

The <portType> element is the most important WSDL element.

WSDL - The <portType> Element

- The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.
- <portType> defines the connection point to a web service. It can be compared to a function library (or a module, or a class) in a traditional programming language. Each operation can be compared to a function in a traditional programming language.

Operation Types

The request-response type is the most common operation type, but WSDL defines four types:

Type	Definition
One-way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

One-Way Operation

A one-way operation example:

```
<message name="newTermValues">
        <part name="term" type="xs:string"/>
```

```
        <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
        <operation name="setTerm">
        <input name="newTerm" message="newTermValues"/>
        </operation>
</portType >
```

- In the example above, the portType "glossaryTerms" defines a one-way operation called "setTerm".
- The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.

Request-Response Operation

A request-response operation example:

```
<message name="getTermRequest">
        <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
    <operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
    </operation>
</portType>
```

- In the example above, the portType "glossaryTerms" defines a request-response operation called "getTerm".
- The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".

5.5 Web Service Architecture (WSA)

Web services are

- Applications that enable remote procedure calls over a network or the Internet often using XML and HTTP.
- Web services architecture is an interoperability architecture-it identifies those global elements of the global web services network that are required in order to ensure interoperability between web services.

Web Service Model:

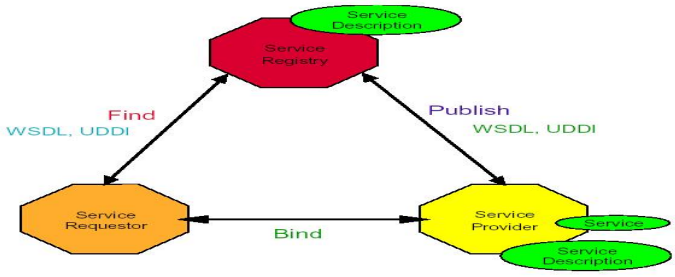


Fig 5.2 Web Service

Roles in Web Service architecture

1. Service provider
   - Owner of the service.
   - Platform that hosts access to the service.
2. Service requestor
   - Business that requires certain functions to be satisfied.
   - Application looking for and invoking an interaction with a service.
3. Service registry
   - Searchable registry of service descriptions where service providers publish their service descriptions.

Operations in Web Service Architecture

- Publish
  Service descriptions need to be published in order for service requestor to find them.
- Find
  Service requestor retrieves a service description directly or queries the service registry for the service required.
- Bind
  Service requestor invokes or initiates an interaction with the service at runtime.

Web services specification and technologies

- Security - for Web services confidentiality, integrity, authentication, and authorization.
- Process flow - for arranging the flow of execution across Web services.
- Transactions - for coordinating the results of multiple Web services.
- Messaging - for configuring message paths and routing messages reliably across multiple network hops.

Web Services involve three major roles

1. Service Provider
2. Service Registry
3. Service Consumer

Three major operations surround web services

- Publishing – making a service available
- Finding – locating web services
- Binding – using web services

Security

- Security is one of the most important and most complex issues in the Internet and Web services.
- Basic security issues include
  - data confidentiality and integrity—to ensure your credit card number,
  - Authentication/authorization, which deals with the rights of individuals or groups to access a certain resource, such as a given Web service interface.

SAML

- Security Assertions Markup Language (SAML) provides a standard way to profile information in XML documents and to define user identification and authorization information.

XKMS

- The XML Key Management Specification (XKMS) defines a protocol for distributing and registering public keys used in encrypting and decrypting messages transmitted using SOAP.
- XML-based security standards
  - authentication and authorization (SAML)
  - public key management (XKMS).
  - WS-License and WS-Security
  - fundamental to all Internet security,
  - the firewalls that protect private networks

5.6 AJAX –improving web page performance using AJAX

AJAX = Asynchronous JavaScript and XML.

- AJAX is not a new programming language, but a new way to use existing standards.
- AJAX is the art of exchanging data with a server, and updating parts of a web page – without reloading the whole page.

AJAX Introduction

- AJAX is about updating parts of a web page, without reloading the whole page.

What is AJAX?

- **AJAX = Asynchronous JavaScript and XML.**
- AJAX is a technique for creating fast and dynamic web pages.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.
- Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

How AJAX Works

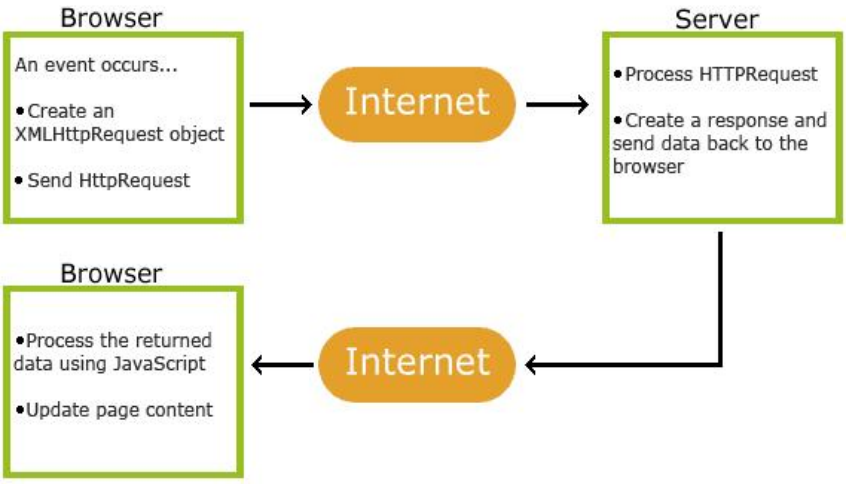


Fig 5.3 Working of AJAX

AJAX is based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)
- AJAX applications are browser and platform-independent!

Google Suggest

- AJAX was made popular in 2005 by Google, with Google Suggest.
- Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

AJAX Example

To understand how AJAX works, we will create a small AJAX application:

```
<html>
    <head>
        <script>
        function loadXMLDoc()
        {
            var xmlhttp;
            if (window.XMLHttpRequest)
            {    // code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
            }
            else
            {    // code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.onreadystatechange=function()
            {
                if (xmlhttp.readyState==4 && xmlhttp.status==200)
                {
```

```
        document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
                        }
                }
                xmlhttp.open("GET","ajax_info.txt",true);
                xmlhttp.send();
        }
        </script>
</head>
<body>
        <div id="myDiv"><h2>Let AJAX change this text</h2></div>
        <button type="button" onclick="loadXMLDoc()">Change Content</button>
</body>
</html>
```

OUTPUT:

Let AJAX change this text

AJAX XMLHttpRequest:

AJAX - Create an XMLHttpRequest Object

- The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object (IE5 and IE6 use an ActiveXObject).
- The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (IE7+, Firefox, Chrome, Safari, and Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

variable=new XMLHttpRequest();

Old versions of Internet Explorer (IE5 and IE6) uses an ActiveX Object:

variable=new ActiveXObject("Microsoft.XMLHTTP");

To handle all modern browsers, including IE5 and IE6, check if the browser supports the XMLHttpRequest object. If it does, create an XMLHttpRequest object, if not, create an ActiveXObject:

Example

```
var xmlhttp;
   if (window.XMLHttpRequest)
     {    // code for IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp=new XMLHttpRequest();
      }
   else
```

```
{   // code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
 }
```

AJAX - Send a Request To a Server

- The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

Method	Description
open(method,url,async)	Specifies the type of request, the URL, and if the request should be handled asynchronously or not. method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)
send(string)	Sends the request off to the server. string: Only used for POST requests

GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

- A cached file is not an option (update a file or database on the server)
- Sending a large amount of data to the server (POST has no size limitations)
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET

AJAX - Server Response

Server Response

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

The responseText Property

- If the response from the server is not XML, use the responseText property.
- The responseText property returns the response as a string, and you can use it accordingly:

Example

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

The responseXML Property

- If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property:

Example

Request the file cd_catalog.xml and parse the response:

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
 {
 txt=txt + x[i].childNodes[0].nodeValue + "<br>";
 }
document.getElementById("myDiv").innerHTML=txt;
```

5.7 PROGRAMMING IN AJAX

```
<html>
	<head>
		<script type="text/javascript">
			function loadXMLDoc()
			{
				var xmlhttp;

				if (window.XMLHttpRequest)
				{	// code for IE7+, Firefox, Chrome, Opera, Safari
					xmlhttp=new XMLHttpRequest();
				}
				else
				{	// code for IE6, IE5
					xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
				}
				xmlhttp.onreadystatechange=function()
				{
					if (xmlhttp.readyState==4 && xmlhttp.status==200)
					{

	document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
					}
				}
					xmlhttp.open("GET","ajax_info.txt",true);
					xmlhttp.send();
			}
		</script>
	</head>
	<body>
		<div id="myDiv"><h2>Let AJAX change this text</h2></div>
		<button type="button" onclick="loadXMLDoc()">Change
Content</button> </body></html>
```

Output:

AJAX is not a new programming language.

AJAX is a technique for creating fast and dynamic web pages.

Load an XML file with AJAX

```
<html>
	<head>
		<script type="text/javascript">
			function loadXMLDoc(url)
			{
				var xmlhttp;
			if (window.XMLHttpRequest)
			  {	// code for IE7+, Firefox, Chrome, Opera, Safari
				 xmlhttp=new XMLHttpRequest();
			  }
			else
			  {	// code for IE6, IE5
				  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
			  }
			xmlhttp.onreadystatechange=function()
			{
			  if (xmlhttp.readyState==4 && xmlhttp.status==200)
			{

document.getElementById('A1').innerHTML=xmlhttp.status;

document.getElementById('A2').innerHTML=xmlhttp.statusText;

	document.getElementById('A3').innerHTML=xmlhttp.responseText;
			 }
			}
			xmlhttp.open("GET",url,true);
			xmlhttp.send();
			}
		</script>
	</head>
<body>
	<h2>Retrieve data from XML file</h2>
	<p><b>Status:</b><span id="A1"></span></p>
	<p><b>Status text:</b><span id="A2"></span></p>
	<p><b>Response:</b><span id="A3"></span></p>
	<button onclick="loadXMLDoc('note.xml')">Get XML data</button>
</body>
</html>
```

Output:

Retrieve data from XML file
Status: 200
Status text: OK
Response: Tove Jani Reminder Don't forget me this weekend! s