UNIT – IV

Overview::

This section provides information about web technologies that relate to the interface between web servers and their clients. This information includes markup languages, programming interfaces and languages, and standards for document identification and display

Objectives::

- Describe about dynamic web content
- Explain about CSS and its usage
- Describe why DHTML is used
- Describe the specialty of XML
- Explain the usage of Web server and Application server

5.1. DYNAMIC WEB CONTENT

Web-based content is increasingly becoming dynamic rather than static. This means Web pages are being generated on demand with content tailored to each user.

A simple example is a personal greeting that pops up when a regular customer returns to a Web site. A more elaborate scheme might provide a customer with a set of recommendations based on past purchases or site interactions.

While dynamic sites provide a far richer experience for users than static sites and have been shown to increase customer retention, generating Web pages on the fly exerts a major toll on server resources. The results are bottlenecks at the server and slower response times for end users.

To address the latencies inherent in dynamic sites, a new class of product is emerging called dynamic content accelerators. These server-side accelerators intelligently feed data to application servers, letting pages be created much faster than they are on no optimized sites.

Basics of content delivery

When a user types in a URL to request a Web page, the page is created by an application server, which executes a script that builds the page. This script contains actions such as calls to database systems to retrieve content. The result is an HTML "blueprint" for the requested page.

The page is then delivered back to the browser in a quick, text-only, no bandwidth-intensive transfer that does not incorporate graphics. Finally, the browser must fetch the graphics, requesting each object from the appropriate server address based on the embedded URLs in the HTML page.

Because browsers are limited to downloading two to four objects at a time and a typical Web page may contain 30 to 40 embedded objects, a good deal of back-and-forth handshaking between the browser and server is required to complete the loading of a page.

During the past few years, object delivery has been successfully optimized through network caching. By storing and serving objects from the network edge, caching slashes the time it takes a browser to load an object. Caching can be deployed as a product (CacheFlow and Inktomi are leading vendors) or as a service (Akamai leads this category).

But caching addresses network latency, not server latency. Caching doesn't effectively address dynamic page generation, which typically accounts for 40% of the time required to deliver a Web page.

The problem is complex, involving a mix of business logic execution, database and/or file system access (from local or remote systems), and content transformation and formatting, such as converting XML to HTML. Each of these tasks requires considerable

server resources, creating load issues. Until the page is created, subsequent steps in the process are stalled.

Emerging alternative

Dynamic content accelerators take advantage of the fact that much of the content in a dynamically generated page is reusable. A dynamic content accelerator caches individual page "components" for faster access. A component is a group of data that is displayed together on a page, such as a set of top news stories or a product's price and attributes.

For each component on a page, the application server makes a request to the dynamic content accelerator. If the data resides in the accelerator's RAM-based cache, it is instantly returned to the application server in ready-to-display HTML format, bypassing the processing and I/O tasks typically required to create the component.

Sites gain three key efficiencies by caching dynamic page content:

? Script routines do not have to be executed.

? Data elements do not have to be retrieved from local or remote database systems.

? Data elements do not have to be converted from formats such as XML or wireless markup language into HTML. Servers and storage subsystems are significantly off-loaded through this approach.

With dynamic content accelerator technology, Web pages are created and delivered to users' browsers in a fraction of the normal time, and the graphics-fetching process can begin. Site scalability is dramatically enhanced as requests for repeatedly accessed content are fulfilled through a high-performance caching engine.

5.2. CASCADING STYLE SHEETS

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, and spacing) to Web documents.

Cascading Style Sheets (**CSS**) is a style sheet language used to describe the presentation of a document written in a markup language. Its most common application is

to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document, including SVG and XUL.

CSS can be used locally by the readers of web pages to define colors, fonts, layout, and other aspects of document presentation. It is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation (written in CSS). This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, and reduce complexity and repetition in the structural content. CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998).

Syntax

CSS has a simple syntax, and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors and a declaration block. A declaration-block consists of a list of semicolon-separated declarations in braces. Each declaration itself consists of a property, a colon (:), a value, then a semi-colon (;).[1]

In CSS, selectors are used to declare which elements a style applies to, a kind of match expression. Selectors may apply to all elements of a specific type, or only those elements which match a certain attribute; elements may be matched depending on how they are placed relative to each other in the markup code, or on how they are nested within the document object model.

In addition to these, a set of pseudo-classes can be used to define further behavior. Probably the best-known of these is :hover, which applies a style only when the user 'points to' the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in a:hover or #elementid:hover. Other pseudo-classes and pseudo-elements are, for example, first-line: visited or: before. A special pseudo-class is: Lang(c), "c".

A pseudo-class selects entire elements, such as :link or :visited, whereas a pseudo-element makes a selection that may consist of partial elements, such as :first-line or :first-letter.

Selectors may be combined in other ways too, especially in CSS 2.1, to achieve greater specificity and flexibility.[2]

Use of CSS

Prior to CSS, nearly all of the presentational attributes of HTML documents were contained within the HTML markup; all font colors, background styles, element alignments, borders and sizes had to be explicitly described, often repeatedly, within the HTML. CSS allows authors to move much of that information to a separate stylesheet resulting in considerably simpler HTML markup.

Headings (h1 elements), sub-headings (h2), sub-sub-headings (h3), etc., are defined structurally using HTML. In print and on the screen, choice of font, size, color and emphasis for these elements is presentational.

Prior to CSS, document authors who wanted to assign such typographic characteristics to, say, all h2 headings had to use the HTML font and other presentational elements for each occurrence of that heading type. The additional presentational markup in the HTML made documents more complex, and generally more difficult to maintain. In CSS, presentation is separated from structure. In print, CSS can define color, font, text alignment, size, borders, spacing, layout and many other typographic characteristics. It can do so independently for on-screen and printed views. CSS also defines non-visual styles such as the speed and emphasis with which text is read out by aural text readers. The W3C now considers the advantages of CSS for defining all aspects of the

presentation of HTML pages to be superior to other methods. It has therefore deprecated the use of all the original presentational HTML markup.

Sources

CSS information can be provided by various sources. CSS style information can be either attached as a separate document or embedded in the HTML document. Multiple style sheets can be imported, and alternative style sheets can be specified so that the user can choose between them. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium.

- Author styles (style information provided by the web page author), in the form of
- external stylesheets, i.e. a separate CSS-file referenced from the document
- embedded style, blocks of CSS information inside the HTML document itself
- inline styles, inside the HTML document, style information on a single element, specified using the "style" attribute.
- User style,
- a local CSS-file specified by the user using options in the web browser, and acting as an override, to be applied to all documents.
- User agent style
- the default style sheet applied by the user agent, e.g. the browser's default presentation of elements.

One of the goals of CSS is also to allow users a greater degree of control over presentation; those who find the red italic headings difficult to read may apply other style sheets to the document. Depending on their browser and the web site, a user may choose from various stylesheets provided by the designers, may remove all added style and view the site using their browser's default styling or may perhaps override just the red italic heading style without altering other attributes.

File highlightheaders.css containing:

h1 { **color**: white; **background**: orange !important; }
h2 { **color**: white; **background**: green !important; }

Such a file is stored locally and is applicable if that has been specified in the browser options. "!important" means that it prevails over the author specifications..

5.3. DHTML

DHTML is NOT a language.

DHTML is a TERM describing the art of making dynamic and interactive web pages.

DHTML combines HTML, JavaScript, DOM, and CSS

JavaScript Alone

If you have studied JavaScript, you already know that the statement:

document.write()

can be used to display dynamic content to a web page.

Example

Using JavaScript to display the current date:

```
<html>
<body>

<script type="text/javascript">
document.write(Date());
</script>

</body>
</html>
```

JavaScript and the HTML DOM

With HTML 4, JavaScript can also be used to change the inner content and attributes of HTML elements dynamically.

To change the content of an HTML element use:

document.getElementById(id).innerHTML=new HTML

To change the attribute of an HTML element use:

document.getElementById(id).attribute=new value

You will learn more about JavaScript and the HTML DOM in the next chapter of this tutorial.

JavaScript and HTML Events

New to HTML 4 is the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element.

To execute code when a user clicks on an element, use the following event attribute:

onclick=JavaScript

You will learn more about JavaScript and HTML Events in a later chapter.

JavaScript and CSS

With HTML 4, JavaScript can also be used to change the style of HTML elements.

To change the style of an HTML element use:

document.getElementById(id).style.property=new style

You will learn more about JavaScript and CSS in a later chapter of this tutorial.

5.4. XML

The **Extensible Markup Language** (**XML**) is a general-purpose specification for creating custom markup languages.[1] It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet,[2] and it is used both to encode documents and to serialize data. In the latter context, it is comparable with other text-based serialization languages such as JSON and YAML.[3]

It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML,[4] RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages.

XML is recommended by the World Wide Web Consortium (W3C). It is a fee-free open standard. The recommendation specifies both the lexical grammar and the requirements for parsing.

Well-formed and valid XML documents

There are two levels of correctness of an XML document:

- **Well-formed**. A well-formed document conforms to all of XML's syntax rules. For example, if a start-tag appears without a corresponding end-tag, it is not well-formed. A document that is not well-formed is not considered to be XML; a conforming parser is not allowed to process it.

- **Valid**. A valid document additionally conforms to some semantic rules. These rules are either user-defined, or included as an XML schema or DTD. For example, if a document contains an undefined element, then it is not valid; a validating parser is not allowed to process it.

Well-formed documents: XML syntax

As long as only well-formedness is required, XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree. The only indispensable syntactical requirement is that the document has exactly one **root element** (alternatively called the **document element**). This means that the text must be enclosed between a root start-tag and a corresponding end-tag. The following is a "well-formed" XML document:

<book>This is a book.... **</book>**

The root element can be preceded by an optional **XML declaration**. This element states what version of XML is in use (normally 1.0); it may also contain information about character encoding and external dependencies.

<?xml version="1.0" encoding="UTF-8"**?>**

The specification requires that processors of XML support the pan-Unicode character encodings UTF-8 and UTF-16 (UTF-32 is not mandatory). The use of more limited encodings, such as those based on ISO/IEC 8859, is acknowledged and is widely used and supported.

Comments can be placed anywhere in the tree, including in the text if the content of the element is text or #PCDATA.

XML comments start with **<!--** and end with **-->**. Two dashes (--) may not appear anywhere in the text of the comment.

<!-- This is a comment. -->

In any meaningful application, additional markup is used to structure the contents of the XML document. The text enclosed by the root tags may contain an arbitrary number of XML elements. The basic syntax for one **element** is:

<name attribute="value">Content**</name>**

The two instances of »name« are referred to as the **start-tag** and **end-tag**, respectively. Here, »content« is some text which may again contain XML elements. So, a generic XML document contains a tree-based data structure. Here is an example of a structured XML document:

```
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="3" unit="cups">Flour</ingredient>
  <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
  <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Bake in the oven at 350(degrees)F for 30 minutes.</step>
  </instructions>
 </recipe>
```

Attribute values must always be quoted, using single or double quotes; and each attribute name must appear only once in any element.

XML requires that elements be properly nested — elements may never overlap, and so must be closed in the opposite order to which they are opened. For example, this fragment of code below cannot be part of a well-formed XML document because the title and author elements are closed in the wrong order:

```
<!-- WRONG! NOT WELL-FORMED XML! -->
<title>Book on Logic<author>Aristotle</title></author>
```

One way of writing the same information in a way which could be incorporated into a well-formed XML document is as follows:

<!-- Correct: well-formed XML fragment. -->
<title>Book on Logic**</title> <author>**Aristotle**</author>**

XML provides special syntax for representing an element with empty content. Instead of writing a start-tag followed immediately by an end-tag, a document may contain an empty-element tag. An empty-element tag resembles a start-tag but contains a slash just before the closing angle bracket. The following three examples are equivalent in XML:

<foo></foo>
<foo />
<foo/>

An empty-element may contain attributes:

<info author="John Smith" genre="science-fiction" date="2009-Jan-01" **/>**

Entity references

An entity in XML is a named body of data, usually text. Entities are often used to represent single characters that cannot easily be entered on the keyboard; they are also used to represent pieces of standard ("boilerplate") text that occur in many documents, especially if there is a need to allow such text to be changed in one place only.

Special characters can be represented either using entity references, or by means of numeric character references. An example of a numeric character reference is "€", which refers to the Euro symbol by means of its Unicode codepoint in hexadecimal.

An entity reference is a placeholder that represents that entity. It consists of the entity's name preceded by an ampersand ("&") and followed by a semicolon (";"). XML has five predeclared entities:

& & ampersand
< < less than
> > greater than

' ' apostrophe

" " quotation mark

Here is an example using a predeclared XML entity to represent the ampersand in the name "AT&T":

<company_name>AT&T**</company_name>**

Additional entities (beyond the predefined ones) can be declared in the document's Document Type Definition (DTD). A basic example of doing so in a minimal internal DTD follows. Declared entities can describe single characters or pieces of text, and can reference each other.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE example [
   <!ENTITY copy "&#xA9;">
   <!ENTITY copyright-notice "Copyright &copy; 2006, XYZ Enterprises">
]>
<example>
   &copyright-notice;
</example>
```

When viewed in a suitable browser, the XML document above appears as:

<example> Copyright © 2006, XYZ Enterprises </example>

Numeric character references

Numeric character references look like entity references, but instead of a name, they contain the "#" character followed by a number. The number (in decimal or "x"-prefixed hexadecimal) represents a Unicode code point. Unlike entity references, they are neither predeclared nor do they need to be declared in the document's DTD. They have typically been used to represent characters that are not easily encodable, such as an Arabic character in a document produced on a European computer. The ampersand in the "AT&T" example could also be escaped like this (decimal 38 and hexadecimal 26 both represent the Unicode code point for the "&" character):

<company_name>AT&T**</company_name>**

<company_name>AT&T**</company_name>**

Similarly, in the previous example, notice that "&#xA9" is used to generate the "©" symbol.

See also numeric character references.

Well-formed documents

In XML, a well-formed document must conform to the following rules, among others:

- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an empty-element (self-closing) tag, such as <IAmEmpty />. This is equal to <IAmEmpty></IAmEmpty>.
- All attribute values are quoted with either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but must not overlap. Each non-root element must be completely contained in another element.
- The document complies with its declared character encoding. The encoding may be declared or implied externally, such as in "Content-Type" headers when a document is transported via HTTP, or internally, using explicit markup at the very beginning of the document. When no such declaration exists, a Unicode encoding is assumed, as defined by a Unicode Byte Order Mark before the document's first character. If the mark does not exist, UTF-8 encoding is assumed.

Element names are case-sensitive. For example, the following is a well-formed matching pair:

<Step> ... </Step>

whereas this is not

<Step> ... </step>

By carefully choosing the names of the XML elements one may convey the meaning of the data in the markup. This increases human readability while retaining the rigor needed for software parsing.

Choosing meaningful names implies the semantics of elements and attributes to a human reader without reference to external documentation. However, this can lead to verbosity, which complicates authoring and increases file size.

Automatic verification

It is relatively simple to verify that a document is well-formed or validated XML, because the rules of well-formedness and validation of XML are designed for portability of tools. The idea is that any tool designed to work with XML files will be able to work with XML files written in any XML language (or XML application). One example of using an independent tool follows:

- load it into an XML-capable browser, such as Firefox or Internet Explorer
    - use a tool like xmlwf (usually bundled with expat)
    - parse the document, for instance in Ruby:

```
irb> require "rexml/document"
irb> include REXML
irb> doc = Document.new(File.new("test.xml")).root
```

Valid documents: XML semantics

By leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customizable schema or DTD, XML provides a syntactic foundation for the creation of purpose-specific, XML-based markup languages. The general syntax of such languages is rigid — documents must adhere to the general rules of XML, ensuring that all XML-aware software can at least read and understand the relative arrangement of information within them. The schema merely supplements the

syntax rules with a set of constraints. Schemas typically restrict element and attribute names and their allowable containment hierarchies, such as only allowing an element named 'birthday' to contain one element named 'month' and one element named 'day', each of which has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the 'month' element's character data may be defined as being a month according to a particular schema language's conventions, perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it were some other type of data.

An XML document that complies with a particular schema/DTD, in addition to being well-formed, is said to be **valid**.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are XML-based, themselves.

Before the advent of generalised data description languages such as SGML and XML, software designers had to define special file formats or small languages to share data between programs. This required writing detailed specifications and special-purpose parsers and writers.

XML's regular structure and strict parsing rules allow software designers to leave parsing to standard tools, and since XML provides a general, data model-oriented framework for the development of application-specific languages, software designers need only concentrate on the development of rules for their data, at relatively high levels of abstraction.

Well-tested tools exist to validate an XML document "against" a schema: the tool automatically verifies whether the document conforms to constraints expressed in the schema. Some of these validation tools are included in XML parsers, and some are packaged separately.

Other usages of schemas exist: XML editors, for instance, can use schemas to support the editing process (by suggesting valid elements and attributes names, etc).

DTD

Main article: Document Type Definition

The oldest schema format for XML is the Document Type Definition (DTD), inherited from SGML. While DTD support is ubiquitous due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressiveness. Certain formal aspects of an XML document cannot be captured in a DTD.
- It uses a custom non-XML syntax, inherited from SGML, to describe the schema.

DTD is still used in many applications because it is considered the easiest to read and write.

XML Schema

Main article: XML Schema (W3C)

A newer XML schema language, described by the W3C as the successor of DTDs, is XML Schema, or more informally referred to by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system, allow for more detailed constraints on an XML document's logical structure, and must be processed in a more robust validation framework. XSDs also use an XML-based format, which makes it possible to use ordinary XML tools to help process them, although XSD implementations require much more than just the ability to read XML.

Criticisms of XSD include the following:

- The specification is very large, which makes it difficult to understand and implement.

- The XML-based syntax leads to verbosity in schema descriptions, which makes XSDs harder to read and write.
- Schema validation can be an expensive addition to XML parsing, especially for high volume systems.
- The modeling capabilities are very limited, with no ability to allow attributes to influence content models.
- The type derivation model is very limited, in particular that derivation by extension is rarely useful.
- Database-related data transfer is supported with arcane ideas such as nillability, but the requirements of industrial publishing are under-supported.
- The key/keyref/uniqueness mechanisms are not type-aware.
- The PSVI concept (Post Schema Validation Infoset) does not have a standard XML representation or Application Programming Interface, thus it works against vendor independence unless revalidation is performed.

[RELAX NG

Main article: RELAX NG

Another popular schema language for XML is RELAX NG. Initially specified by OASIS, RELAX NG is now also an ISO international standard (as part of DSDL). It has two formats: an XML based syntax and a non-XML compact syntax. The compact syntax aims to increase readability and writability but, since there is a well-defined way to translate the compact syntax to the XML syntax and back again by means of James Clark's Trang conversion tool, the advantage of using standard XML tools is not lost. RELAX NG has a simpler definition and validation framework than XML Schema, making it easier to use and implement. It also has the ability to use datatype framework plug-ins; a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

ISO DSDL and other schema languages

The ISO DSDL (Document Schema Description Languages) standard brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes RELAX NG full and compact syntax, Schematron assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based routing of document fragments to different validators. DSDL schema languages do not have the vendor support of XML Schemas yet, and are to some extent a grassroots reaction of industrial publishers to the lack of utility of XML Schemas for publishing.

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide attribute defaults. RELAX NG and Schematron intentionally do not provide these; for example the infoset augmentation facility.

International use

XML supports the direct use of almost any Unicode character in element names, attributes, comments, character data, and processing instructions (other than the ones that have special symbolic meaning in XML itself, such as the open corner bracket, "<"). Therefore, the following is a well-formed XML document, even though it includes both Chinese and Cyrillic characters:

<**?xml** version="1.0" encoding="UTF-8"**?**>
<□□>Данные</□□>

Displaying XML on the web

XML documents do not carry information about how to display the data. Without using CSS or XSL, a generic XML document is rendered as raw XML text by most web browsers. Some display it with 'handles' (e.g. + and - signs in the margin) that allow parts of the structure to be expanded or collapsed with mouse-clicks.

In order to style the rendering in a browser with CSS, the XML document must include a reference to the stylesheet:

<?xml-stylesheet type="text/css" href="myStyleSheet.css"**?>**

Note that this is different from specifying such a stylesheet in HTML, which uses the <link> element.

Extensible Stylesheet Language (XSL) can be used to alter the format of XML data, either into HTML or other formats that are suitable for a browser to display.

To specify client-side XSL Transformation (XSLT), the following processing instruction is required in the XML:

<?xml-stylesheet type="text/xsl" href="myTransform.xslt"**?>**

Client-side XSLT is supported by many web browsers. Alternatively, one may use XSL to convert XML into a displayable format on the server rather than being dependent on the end-user's browser capabilities. The end-user is not aware of what has gone on 'behind the scenes'; all they see is well-formatted, displayable data.

See the XSLT article for an example of server-side XSLT in action.

XML extensions

- **XPath** makes it possible to refer to individual parts of an XML document. This provides random access to XML data for other technologies, including XSLT, XSL-FO, XQuery etc. XPath expressions can refer to all or part of the text, data and values in XML elements, attributes, processing instructions, comments etc. They can also access the names of elements and attributes. XPaths can be used in both valid and well-formed XML, with and without defined namespaces.

- **XInclude** defines the ability for XML files to include all or part of an external file. When processing is complete, the final XML infoset has no XInclude elements, but instead has copied the documents or parts thereof into the final infoset. It uses XPath to refer to a portion of the document for partial inclusions.

- **XQuery** is to XML what SQL and PL/SQL are to relational databases: ways to access, manipulate and return XML.

- **XML Namespaces** enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring.

- **XML Signature** defines the syntax and processing rules for creating digital signatures on XML content.

- **XML Encryption** defines the syntax and processing rules for encrypting XML content.

- **XPointer** is a system for addressing components of XML-based internet media.

XML files may be served with a variety of Media types. RFC 3023 defines the types "application/xml" and "text/xml", which say only that the data is in XML, and nothing about its semantics. The use of "text/xml" has been criticized as a potential source of encoding problems but is now in the process of being deprecated.[5] RFC 3023 also recommends that XML-based languages be given media types beginning in "application/" and ending in "+xml"; for example "application/atom+xml" for Atom. This page discusses further XML and MIME.

Processing XML files

Three traditional techniques for processing XML files are:

- Using a programming language and the SAX API.

- Using a programming language and the DOM API.

- Using a transformation engine and a filter

More recent and emerging techniques for processing XML files are:

- Pull Parsing
- Data binding

Simple API for XML (SAX)

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

DOM

DOM is an interface-oriented Application Programming Interface that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed. DOM is supported in Java by several packages that usually come with the standard libraries. As the DOM specification is regulated by the World Wide Web Consortium, the main interfaces (Node, Document, etc.) are in the package org.w3c.dom.*, as well as some of the events and interfaces for other capabilities like serialization (output). The package com.sun.org.apache.xml.internal.serialize.* provides the serialization (output capacities) by implementing the appropriate interfaces, while the javax.xml.parsers.* package parses data to create DOM XML documents for manipulation.[6]

Transformation engines and filters

A filter in the Extensible Stylesheet Language (XSL) family can transform an XML file for displaying or printing.

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.

- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT stylesheet as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.

- **XQuery** is a W3C language for querying, constructing and transforming XML data.

- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.

Pull parsing

Pull parsing[7] treats the document as a series of items which are read in sequence using the Iterator design pattern. This allows for writing of recursive-descent parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. Examples of pull parsers include StAX in the Java programming language, SimpleXML in PHP and System.Xml.XmlReader in .NET.

A pull parser creates an iterator that sequentially visits the various elements, attributes, and data in an XML document. Code which uses this 'iterator' can test the current item (to tell, for example, whether it is a start or end element, or text), and inspect its attributes (local name, namespace, values of XML attributes, value of text, etc.), and can also move the iterator to the 'next' item. The code can thus extract information from the document as it traverses it. The recursive-descent approach tends to lend itself to keeping data as typed local variables in the code doing the parsing, while SAX, for instance, typically requires a

parser to manually maintain intermediate data within a stack of elements which are parent elements of the element being parsed. Pull-parsing code can be more straightforward to understand and maintain than SAX parsing code.

Data binding

Another form of XML Processing API is data binding, where XML data is made available as a custom, strongly typed programming language data structure, in contrast to the interface-oriented DOM. Example data binding systems include the Java Architecture for XML Binding (JAXB)[8].

[Specific XML applications and editors

The native file format of OpenOffice.org, AbiWord, and Apple's iWork applications is XML. Some parts of Microsoft Office 2007 are also able to edit XML files with a user-supplied schema (but not a DTD), and Microsoft has released a file format compatibility kit for Office 2003 that allows previous versions of Office to save in the new XML based format. There are dozens of other XML editors available.

History

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet.[9][10] By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Jon Bosak developed a charter and recruited collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web. He received support in his efforts from Microsoft.

XML was compiled by a working group of eleven members,[11] supported by an (approximately) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. A record of design decisions and their rationales was

compiled by Michael Sperberg-McQueen on December 4th 1997.[12] James Clark served as Technical Lead of the Working Group, notably contributing the empty-element "<empty/>" syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally Tim Bray and Michael Sperberg-McQueen. Halfway through the project Bray accepted a consulting engagement with Netscape, provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working Group, eventually solved by the appointment of Microsoft's Jean Paoli as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in twenty weeks of intense work between July and November of 1996, when the first Working Draft of an XML specification was published.[13] Further design work continued through 1997, and XML 1.0 became a W3C Recommendation on February 10, 1998.

XML 1.0 achieved the Working Group's goals of Internet usability, general-purpose usability, SGML compatibility, facilitation of easy development of processing software, minimization of optional features, legibility, formality, conciseness, and ease of authoring. Like its antecedent SGML, XML allows for some redundant syntactic constructs and includes repetition of element identifiers. In these respects, terseness was not considered essential in its structure.

Sources

XML is a profile of an ISO standard SGML, and most of XML comes from SGML unchanged. From SGML comes the separation of logical and physical structures (elements and entities), the availability of grammar-based validation (DTDs), the separation of data and metadata (elements and attributes), mixed content, the separation of processing from representation (processing instructions), and the default angle-bracket

syntax. Removed were the SGML Declaration (XML has a fixed delimiter set and adopts Unicode as the document character set).

Other sources of technology for XML were the Text Encoding Initiative (TEI), which defined a profile of SGML for use as a 'transfer syntax'; HTML, in which elements were synchronous with their resource, the separation of document character set from resource encoding, the xml:lang attribute, and the HTTP notion that metadata accompanied the resource rather than being needed at the declaration of a link; and the Extended Reference Concrete Syntax (ERCS), from which XML 1.0's naming rules were taken, and which had introduced hexadecimal numeric character references and the concept of references to make available all Unicode characters.

Ideas that developed during discussion which were novel in XML, were the algorithm for encoding detection and the encoding header, the processing instruction target, the xml:space attribute, and the new close delimiter for empty-element tags.

Versions

There are two current versions of XML. The first, XML 1.0, was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its fourth edition, as published on August 16, 2006. It is widely implemented and still recommended for general use. The second, XML 1.1, was initially published on February 4, 2004, the same day as XML 1.0 Third Edition, and is currently in its second edition, as published on August 16, 2006. It contains features — some contentious — that are intended to make XML easier to use in certain cases[14] - mainly enabling the use of line-ending characters used on EBCDIC platforms, and the use of scripts and characters absent from Unicode 2.0. XML 1.1 is not very widely implemented and is recommended for use only by those who need its unique features. [15]

XML 1.0 and XML 1.1 differ in the requirements of characters used for element and attribute names: XML 1.0 only allows characters which are defined in Unicode 2.0, which includes most world scripts, but excludes those which were added in later Unicode versions. Among the excluded scripts are Mongolian, Cambodian, Amharic, Burmese, and others.

Almost any Unicode character can be used in the character data and attribute values of an XML 1.1 document, even if the character is not defined, aside from having a code point, in the current version of Unicode. The approach in XML 1.1 is that only certain characters are forbidden, and everything else is allowed, whereas in XML 1.0, only certain characters are explicitly allowed, thus XML 1.0 cannot accommodate the addition of characters in future versions of Unicode.

In character data and attribute values, XML 1.1 allows the use of more control characters than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must be expressed as numeric character references. Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace. Whitespace characters are the only control codes that can be written directly.

There are also discussions on an XML 2.0, although it remains to be seen[vague] if such will ever come about. XML-SW (SW for skunk works), written by one of the original developers of XML, contains some proposals for what an XML 2.0 might look like: elimination of DTDs from syntax, integration of namespaces, XML Base and XML Information Set (infoset) into the base standard.

The World Wide Web Consortium also has an XML Binary Characterization Working Group doing preliminary research into use cases and properties for a binary encoding of the XML infoset. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name Fast Infoset[2] for their own binary infoset to avoid confusion (see ITU-T Rec. X.891 | ISO/IEC 24824-1).

Patent claims

In October 2005 the small company Scientigo publicly asserted that two of its patents, U.S. Patent 5,842,213 and U.S. Patent 6,393,426 , apply to the use of XML. The patents cover the "modeling, storage and transfer [of data] in a particular non-hierarchical, non-integrated neutral form", according to their applications, which were filed in 1997 and 1999. Scientigo CEO Doyal Bryant expressed a desire to "monetize" the patents but stated that the company was "not interested in having us against the world." He said that Scientigo was discussing the patents with several large corporations.[16]

XML users and independent experts responded to Scientigo's claims with widespread skepticism and criticism. Some derided the company as a patent troll. Tim Bray described any claims that the patents covered XML as "ridiculous on the face of it".[17]

Because there exists a large amount of prior art relating to XML, including SGML, some legal experts believed it would be difficult for Scientigo to enforce its patents through litigation.[citation needed]

Critique of XML

Commentators have offered various critiques of XML, suggesting circumstances where XML provides both advantages and potential disadvantages.[18]

Advantages of XML

- It is text-based.
- It supports Unicode, allowing almost any information in any written human language to be communicated.
- It can represent common computer science data structures: records, lists and trees.
- Its self-documenting format describes structure and field names as well as specific values.
- The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.
- XML is heavily used as a format for document storage and processing, both online and offline.
- It is based on international standards.
- It can be updated incrementally.

- It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.
- The hierarchical structure is suitable for most (but not all) types of documents.
- It is platform-independent, thus relatively immune to changes in technology.
- Forward and backward compatibility are relatively easy to maintain despite changes in DTD or Schema.
- Its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.

Disadvantages of XML

- XML syntax is redundant or large relative to binary representations of similar data,[19] especially with tabular data.
- The redundancy may affect application efficiency through higher storage, transmission and processing costs.[20][21]
- XML syntax is verbose, especially for human readers, relative to other alternative 'text-based' data transmission formats.[22][23]
- The hierarchical model for representation is limited in comparison to an object oriented graph.[24][25]
- Expressing overlapping (non-hierarchical) node relationships requires extra effort.[26]
- XML namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.[27]
- XML is commonly depicted as "self-documenting" but this depiction ignores critical ambiguities.[28][29]

- The distinction between content and attributes in XML seems unnatural to some and makes designing XML data structures harder

5.5. SERVER SIDE INCLUDES

Server Side Includes (SSI) is a simple server-side scripting language used almost exclusively for the web. As its name implies, its primary use is including the contents of one file into another one dynamically when the latter is served by a web server.

SSI is primarily used to "paste" the contents of one or more files into another. For example, a file (of any type, .html, .txt, etc.) containing a daily quote could be included into multiple SSI-enabled pages throughout a website by placing the following code into the desired pages:

<!--#include virtual="../quote.txt" -->

With one change of the quote.txt file, pages including the snippet will display the latest daily quote. Server Side Includes are useful for including a common piece of code throughout a site, such as a navigation menu.

In order for a web server in a default configuration to recognize an SSI-enabled HTML file and therefore carry out these instructions, the file must end with the .shtml, .stm or .shtm extension. (It is also possible to configure a web server to process files with extension .html.)

SSI is most suitable for simple automatization tasks; more complex server

What are SSI?

SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology.

The decision of when to use SSI, and when to have your page entirely generated by some program, is usually a matter of how much of the page is static, and how much needs to be

recalculated every time the page is served. SSI is a great way to add small pieces of information, such as the current time. But if a majority of your page is being generated at the time that it is served, you need to look for some other solution.

Configuring your server to permit SSI

To permit SSI on your server, you must have mod_include installed and enabled. Additionally, you must have the following directive either in your httpd.conf file, or in a .htaccess file:

```
Options +Includes
```

This tells Apache that you want to permit files to be parsed for SSI directives. Note that most configurations contain multiple Options directives that can override each other. You will probably need to apply the Options to the specific directory where you want SSI enabled in order to assure that it gets evaluated last.

Not just any file is parsed for SSI directives. You have to tell Apache which files should be parsed. There are two ways to do this. You can tell Apache to parse any file with a particular file extension, such as .shtml, with the following directives:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

One disadvantage to this approach is that if you wanted to add SSI directives to an existing page, you would have to change the name of that page, and all links to that page, in order to give it a .shtml extension, so that those directives would be executed.

The other method is to use the XBitHack directive:

```
XBitHack on
```

XBitHack tells Apache to parse files for SSI directives if they have the execute bit set. So, to add SSI directives to an existing page, rather than having to change the file name, you would just need to make the file executable using chmod.

```
chmod +x pagename.html
```

A brief comment about what not to do. You'll occasionally see people recommending that you just tell Apache to parse all .html files for SSI, so that you don't have to mess with .shtml file names. These folks have perhaps not heard about XBitHack. The thing to keep in mind is that, by doing this, you're requiring that Apache read through every single file that it sends out to clients, even if they don't contain any SSI directives. This can slow things down quite a bit, and is not a good idea.

Of course, on Windows, there is no such thing as an execute bit to set, so that limits your options a little.

In its default configuration, Apache does not send the last modified date or content length HTTP headers on SSI pages, because these values are difficult to calculate for dynamic content. This can prevent your document from being cached, and result in slower perceived client performance. There are two ways to solve this:

1. Use the XBitHack Full configuration. This tells Apache to determine the last modified date by looking only at the date of the originally requested file, ignoring the modification date of any included files.
2. Use the directives provided by mod_expires to set an explicit expiration time on your files, thereby letting browsers and proxies know that it is acceptable to cache them.

Basic SSI directives

SSI directives have the following syntax:

```
<!--#element attribute=value attribute=value ... -->
```

It is formatted like an HTML comment, so if you don't have SSI correctly enabled, the browser will ignore it, but it will still be visible in the HTML source. If you have SSI correctly configured, the directive will be replaced with its results.

The element can be one of a number of things, and we'll talk some more about most of these in the next installment of this series. For now, here are some examples of what you can do with SSI

Today's date

```
<!--#echo var="DATE_LOCAL" -->
```

The echo element just spits out the value of a variable. There are a number of standard variables, which include the whole set of environment variables that are available to CGI programs. Also, you can define your own variables with the set element.

If you don't like the format in which the date gets printed, you can use the config element, with a timefmt attribute, to modify that formatting.

```
<!--#config timefmt="%A %B %d, %Y" -->
Today is <!--#echo var="DATE_LOCAL" -->
```

Modification date of the file

```
This document last modified <!--#flastmod file="index.html" -->
```

This element is also subject to timefmt format configurations.

Including the results of a CGI program

This is one of the more common uses of SSI - to output the results of a CGI program, such as everybody's favorite, a ``hit counter."

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

5.6. COMMUNICATION

Communication is the process of transferring information from a sender to a receiver with the use of a medium in which the communicated information is understood by both sender and receiver. It is a process that allows organisms to exchange information by several methods. Communication requires that all parties understand a common language that is exchanged. There are auditory means, such as speaking, singing and sometimes tone of voice, and nonverbal, physical means, such as body language, sign

language, paralanguage, touch, eye contact, or the use of writing. Communication is defined as a process by which we assign and convey meaning in an attempt to create shared understanding. This process requires a vast repertoire of skills in intrapersonal and interpersonal processing, listening, observing, speaking, questioning, analyzing, and evaluating. Use of these processes is developmental and transfers to all areas of life: home, school, community, work, and beyond. It is through communication that collaboration and cooperation occur.[1] Communication is the articulation of sending a message, through different media [2] whether it be verbal or nonverbal, so long as a being transmits a thought provoking idea, gesture, action, etc.

Communication happens at many levels (even for one single action), in many different ways, and for most beings, as well as certain machines. Several, if not all, fields of study dedicate a portion of attention to communication, so when speaking about communication it is very important to be sure about what aspects of communication one is speaking about. Definitions of communication range widely, some recognizing that animals can communicate with each other as well as human beings, and some are more narrow, only including human beings within the parameters of human symbolic interaction.

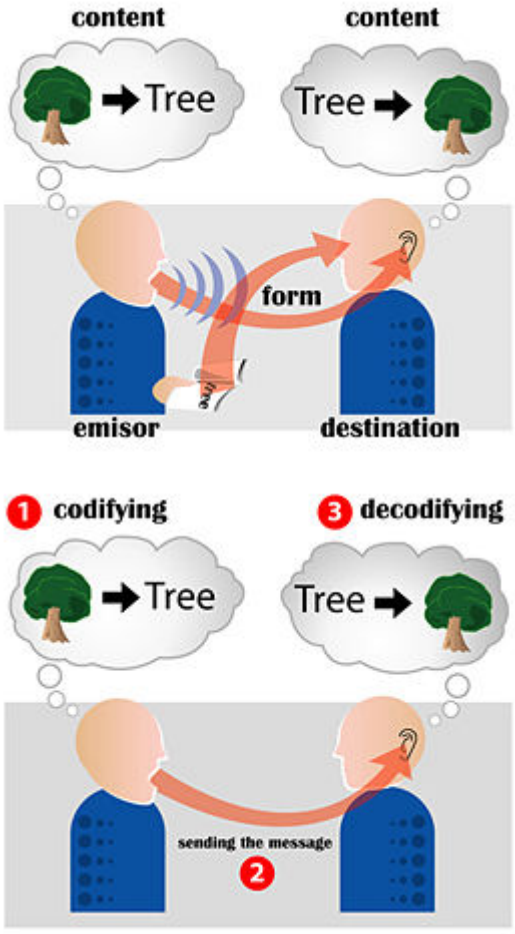
Nonetheless, communication is usually described along a few major dimensions: Content (what type of things are communicated), source, emisor, sender or encoder (by whom), form (in which form), channel (through which medium), destination, receiver, target or decoder (to whom), and the purpose or pragmatic aspect. Between parties, communication includes acts that confer knowledge and experiences, give advice and commands, and ask questions. These acts may take many forms, in one of the various manners of communication. The form depends on the abilities of the group communicating. Together, communication content and form make messages that are sent towards a destination. The target can be oneself, another person or being, another entity (such as a corporation or group of beings).

Communication can be seen as processes of information transmission governed by three levels of semiotic rules:

1. Syntactic (formal properties of signs and symbols),

2. pragmatic (concerned with the relations between signs/expressions and their users) and
3. semantic (study of relationships between signs and symbols and what they represent).

Therefore, communication is social interaction where at least two interacting agents share a common set of signs and a common set of semiotic rules. This commonly held rule in some sense ignores autocommunication, including intrapersonal communication via diaries or self-talk.



In a simple model, information or content (e.g. a message in natural language) is sent in some form (as spoken language) from an emisor/ sender/ encoder to a destination/ receiver/ decoder. In a slightly more complex form a sender and a receiver are linked reciprocally. A particular instance of communication is called a speech act. In the presence of "communication noise" on the transmission channel (air, in this case),

reception and decoding of content may be faulty, and thus the speech act may not achieve the desired effect.

Theories of coregulation describe communication as a creative and dynamic continuous process, rather than a discrete exchange of information.

5.7. ACTIVE AND JAVA SERVER PAGES

JavaServer Pages™ (JSP) and Microsoft Active Server Pages (ASP) technologies have many similarities. Both are designed to create interactive pages as part of a Web-based application. To a degree, both enable developers to separate programming logic from page design through the use of components that are called from the page itself. And both provide an alternative to creating CGI scripts that makes page development and deployment easier and faster.

While JavaServer Pages technology and Microsoft Active Server Pages are similar in many ways, there are also a number of differences that exist. And these differences are just as significant as the similarities, and have far-reaching implications for the developers who use them as well as the organizations that adopt them as part of their overall Web-based architecture.

JSP Technology: An Open Approach

In many ways, the biggest difference between JSP and ASP technologies lies in the approach to the software design itself. JSP technology is designed to be both platform and server independent, created with input from a broader community of tool, server, and database vendors. In contrast, ASP is a Microsoft technology that relies primarily on Microsoft technologies.

Platform and Server Independence

JSP technology adheres to the Write Once, Run Anywhere™ philosophy of the Java™ architecture. Instead of being tied to a single platform or vendor, JSP technology can run on any Web server and is supported by a wide variety of tools from multiple vendors.

Because it uses ActiveX controls for its components, ASP technology is basically restricted to Microsoft Windows-based platforms. Offered primarily as a feature of Microsoft IIS, ASP technology does not work easily on a broader range of Web servers because ActiveX objects are platform specific.

Although ASP technology is available on other platforms through third-party porting products, to access components and interact with other services, the ActiveX objects must be present on the selected platform. If not present, a bridge to a platform supporting them is required.

Open Development Process, Open Source

Sun developed JSP technology using the Java Community Process. Since 1995, Sun has used this open process to develop and revise Java technology and specifications in cooperation with the international Java community. Working with Sun in the JSP effort are authoring tool vendors (such as Macromedia), container companies (such as Apache and Netscape), end users, consultants, and others. Going forward, Sun is licensing the latest versions of JSP and Java™ Servlet (JSP 1.1 and Java Servlet 2.2) source code to Apache to be developed and released under the Apache development process. Apache, Sun, and a number of other companies and individuals will openly develop a robust reference implementation (RI) that is freely available to any company or individual. Additional information can be found at http://jakarta.apache.org/.

The JSP application programming interface (API) has undoubtedly benefited - and will continue to benefit - from the input of this extended community. In contrast, ASP technology is a specifically Microsoft initiative, developed in a proprietary process.

	ASP Technology	**JSP Technology**
Web Server	Microsoft IIS or Personal Web	Any Web server, including

	Server	Apache, Netscape, and IIS
Platforms	Microsoft Windows[1]	Most popular platforms, including the Solaris Operating Environment, Microsoft Windows, Mac OS, Linux, and other UNIX platform implementations

1. Accessing other platforms requires third-party ASP porting products.

For a company selecting the components of a growing, Web-based information architecture, JSP technology provides a flexible, open choice that works with a wide variety of vendors' tools and reflects industry input and collaboration.

The Developer's Perspective

Both ASP and JSP technologies let developers separate content generation from layout by accessing components from the page. ASP supports the COM model, while JSP technology provides components based on JavaBeans™ technology or JSP tags.

As noted previously, the differences outweigh the similarities.

Extensible JSP Tags

The first difference apparent to any page author are the JSP tags themselves. While both ASP and JSP use a combination of tags and scripting to create dynamic Web pages, JSP technology enables developers to extend the JSP tags available. JSP developers can create custom tag libraries, so page authors can access more functionality using XML-like tags and depend less on scripting. With custom tags, developers can shield page authors from the complexities of page creation logic and extend key functions to a broader range of authors.

Reusability Across Platforms

Developers will also notice the focus on reusability. The JSP components (Enterprise JavaBeans™, JavaBeans, or custom JSP tags) are reusable across platforms. An Enterprise JavaBean component accessing legacy databases can serve distributed systems on both UNIX and Microsoft Windows platforms. And the tag extension capability of JSP technology gives developers an easy, XML-like interface for sharing packaged functionality with page designers throughout the enterprise.

This component-based model speeds application development because it enables developers to:

Build quick prototype applications using lightweight subcomponents, then integrate additional functionality as it becomes available

Leverage work done elsewhere in the organization and encapsulate it in a JavaBean or Enterprise JavaBean component

The Java Advantage

JSP technology uses the Java language for scripting, while ASP pages use Microsoft VBScript or JScript. The Java language is a mature, powerful, and scalable programming language that provides many benefits over the Basic-based scripting languages. For example, the Java language provides superior performance to the interpreted VBScript or JScript languages. Because they use Java technology and are compiled into Java servlets, JSP pages provide a gateway to the entire suite of server-side Java libraries for HTTP-aware applications.

The Java language makes the developer's job easier in other ways as well. For example, it helps protect against system crashes, while ASP applications on Windows NT systems are susceptible to crashing. The Java language also helps in the area of memory management by providing protection against memory leaks and hard-to-find pointer bugs that can slow application deployment. Plus, JSP provides the robust exception handling necessary for real-world applications.

Easier Maintenance

Applications using JSP technology are easier to maintain over time than ASP-based applications.

Scripting languages are fine for small applications, but do not scale well to manage large, complex applications. Because the Java language is structured, it is easier to build and maintain large, modular applications with it.

JSP technology's emphasis on components over scripting makes it easier to revise content without affecting logic, or revise logic without changing content.

The Enterprise JavaBeans architecture encapsulates the enterprise logic, such as database access, security, and transaction integrity, and isolates it from the application itself.

Because JSP technology is an open, cross-platform architecture, Web servers, platforms, and other components can be easily upgraded or switched without affecting JSP-based applications. This makes JSP suitable for real-world Web applications, where constant change and growth is the norm.

	ASP Technology	**JSP Technology**
Reusable, Cross-Platform Components	No	JavaBeans, Enterprise JavaBeans, custom JSP tags
Security Against System Crashes	No	Yes
Memory Leak Protection	No	Yes
Scripting Language	VBScript,	Java

	JScript	
Customizable Tags	No	Yes

Scalability in the Enterprise

The Java 2 Platform, Enterprise Edition (J2EE) is the Java architecture for developing multitier enterprise applications. As part of J2EE, JSP pages have access to all J2EE components, including JavaBeans and Enterprise JavaBeans components and Java servlets. JSP pages are actually compiled into servlets, so they have all of the benefits of these flexible, server-side Java applications. The J2EE platform containers manage the complexities of enterprise applications, including transaction management and resource pooling.

JSP pages have access to all of the standard J2EE services, including:

Java Naming and Directory Interface™ API

JDBC™ API (communicating with relational databases)

JavaMail™ (classes supporting Java-based mail and messaging applications)

Java™ Message Service (JMS)

Through J2EE, JSP pages can interact with enterprise systems in many ways. J2EE supports two CORBA-compliant technologies: Java IDL and RMI-IIOP. With Enterprise JavaBeans technology, JSP pages can access databases using high-level, object-relational mappings.

Finally, because JSP technology was developed through the Java Community Process, it has wide support from tool, Web server and application server vendors. This enables

users and partners take a best-of-breed approach, selecting the best tools for their specific applications while protecting their investment in code and in personnel training.

	ASP Technology	JSP Technology
Compatible with Legacy Databases	Yes (COM)	Yes (using JDBC API)
Ability to Integrate with Data Sources	Works with any ODBC-compliant database	Works with any ODBC- and JDBC technology-compliant database
Components	COM components	JavaBeans, Enterprise JavaBeans, or extensible JSP tags
Extensive Tool Support	Yes	Yes

5.8. FIREWALLS

A **firewall** is a device or set of devices configured to permit, deny, encrypt, or proxy all computer traffic between different security domains based upon a set of rules and other criteria.

Function

A firewall is a dedicated appliance, or software running on another computer, which inspects network traffic passing through it, and denies or permits passage based on a set of rules.

A firewall's basic task is to regulate some of the flow of traffic between computer networks of different trust levels. Typical examples are the Internet which is a zone with no trust and an internal network which is a zone of higher trust. A zone with an intermediate trust level, situated between the Internet and a trusted internal network, is often referred to as a "perimeter network" or Demilitarized zone (DMZ).

A firewall's function within a network is similar to firewalls with fire doors in building construction. In the former case, it is used to prevent network intrusion to the private network. In the latter case, it is intended to contain and delay structural fire from spreading to adjacent structures.

Without proper configuration, a firewall can often become worthless. Standard security practices dictate a "default-deny" firewall ruleset, in which the only network connections which are allowed are the ones that have been explicitly allowed. Unfortunately, such a configuration requires detailed understanding of the network applications and endpoints required for the organization's day-to-day operation. Many businesses lack such understanding, and therefore implement a "default-allow" ruleset, in which all traffic is allowed unless it has been specifically blocked. This configuration makes inadvertent network connections and system compromise much more likely.

History

The term "firewall" originally meant a wall to confine a fire or potential fire within a building, c.f. firewall (construction). Later uses refer to similar structures, such as the metal sheet separating the engine compartment of a vehicle or aircraft from the passenger compartment.

Firewall technology emerged in the late 1980s when the Internet was a fairly new technology in terms of its global use and connectivity. The predecessors to firewalls for network security were the routers used in the late 1980s to separate networks from one another [1]. The view of the Internet as a relatively small community of compatible users who valued openness for sharing and collaboration was ended by a number of major internet security breaches, which occurred in the late 1980s.[1]:

- Clifford Stoll's discovery of German spies tampering with his system [1]

- Bill Cheswick's Evening with Berferd" 1992 in which he set up a simple electronic jail to observe an attacker[1]
- In 1988 an employee at the NASA Ames Research Center in California sent a memo by email to his colleagues[citation needed] that read,

> " We are currently under attack from an Internet VIRUS! It has hit Berkeley, UC San Diego, Lawrence Livermore, Stanford, and NASA Ames. "

The Morris Worm spread itself through multiple vulnerabilities in the machines of the time. Although it was not malicious in intent, the Morris Worm was the first large scale attack on Internet security; the online community was neither expecting an attack nor prepared to deal with one.[2]

First generation - packet filters

The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation (**DEC**) developed filter systems known as **packet filter** firewalls. This fairly basic system was the first generation of what would become a highly evolved and technical internet security feature. At AT&T Bell Labs, Bill Cheswick and Steve Bellovin were continuing their research in packet filtering and developed a working model for their own company based upon their original first generation architecture.

Packet filters act by inspecting the "packets" which represent the basic unit of data transfer between computers on the Internet. If a packet matches the packet filter's set of rules, the packet filter will drop (silently discard) the packet, or reject it (discard it, and send "error responses" to the source).

This type of packet filtering pays no attention to whether a packet is part of an existing stream of traffic (it stores no information on connection "state"). Instead, it filters each packet based only on information contained in the packet itself (most commonly using a combination of the packet's source and destination address, its protocol, and, for TCP and UDP traffic, which comprises most internet communication, the port number).

Because TCP and UDP traffic by convention uses well known ports for particular types of traffic, a "stateless" packet filter can distinguish between, and thus control, those types of traffic (such as web browsing, remote printing, email transmission, file transfer), unless the machines on each side of the packet filter are both using the same non-standard ports.

Second generation - "stateful" filters

Main article: stateful firewall

From 1980-1990 three colleagues from AT&T Bell Laboratories, Dave Presetto, Janardan Sharma, and Kshitij Nigam developed the second generation of firewalls, calling them circuit level firewalls.

Second Generation firewalls in addition regard placement of each individual packet within the packet series. This technology is generally referred to as a stateful firewall as it maintains records of all connections passing through the firewall and is able to determine whether a packet is either the start of a new connection, a part of an existing connection, or is an invalid packet. Though there is still a set of static rules in such a firewall, the state of a connection can in itself be one of the criteria which trigger specific rules.

This type of firewall can help prevent attacks which exploit existing connections, or certain Denial-of-service attacks.

Third generation - application layer

Main article: application layer firewall

Publications by Gene Spafford of Purdue University, Bill Cheswick at AT&T Laboratories, and Marcus Ranum described a third generation firewall known as an application layer firewall, also known as a **proxy-based** firewall. Marcus Ranum's work on the technology spearheaded the creation of the first commercial product. The product was released by DEC who named it the DEC SEAL product. DEC’s first major sale was on June 13, 1991 to a chemical company based on the East Coast of the USA.

The key benefit of application layer filtering is that it can "understand" certain applications and protocols (such as File Transfer Protocol, DNS, or web browsing), and it

can detect whether an unwanted protocol is being sneaked through on a non-standard port or whether a protocol is being abused in a known harmful way.

Subsequent developments

In 1992, Bob Braden and Annette DeSchon at the University of Southern California (USC) were refining the concept of a firewall. The product known as "Visas" was the first system to have a visual integration interface with colours and icons, which could be easily implemented to and accessed on a computer operating system such as Microsoft's Windows or Apple's MacOS. In 1994 an Israeli company called Check Point Software Technologies built this into readily available software known as FireWall-1.

The existing deep packet inspection functionality of modern firewalls can be shared by Intrusion-prevention systems (IPS).

Currently, the Middlebox Communication Working Group of the Internet Engineering Task Force (IETF) is working on standardizing protocols for managing firewalls and other middleboxes.

In popular culture

Use of the term "firewall" in relation to computer or network security may have been popularized by its use in the 1983 film WarGames. In the movie, at approximately time index 01:42:00, while attempting to gain access to the WOPR computer as it sought the code required to launch the United States' nuclear arsenal against the Soviet Union, NORAD personnel engaged in the following dialogue:

“ --John, lets feed it a tapeworm.
--Nah, it’s too risky. It might smash the system.
--How’d the kid get in—through the back door?
--We took it out.
--Can we invade the deep logic?
--We keep hitting a damn firewall. ”

Types

There are several classifications of firewalls depending on where the communication is taking place, where the communication is intercepted and the state that is being traced.

Network layer and packet filters

Network layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set. The firewall administrator may define the rules; or default rules may apply. The term "packet filter" originated in the context of BSD operating systems.

Network layer firewalls generally fall into two sub-categories, stateful and stateless. Stateful firewalls maintain context about active sessions, and use that "state information" to speed packet processing. Any existing network connection can be described by several properties, including source and destination IP address, UDP or TCP ports, and the current stage of the connection's lifetime (including session initiation, handshaking, data transfer, or completion connection). If a packet does not match an existing connection, it will be evaluated according to the ruleset for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing.

Stateless firewalls require less memory, and can be faster for simple filters that require less time to filter than to look up a session. They may also be necessary for filtering stateless network protocols that have no concept of a session. However, they cannot make more complex decisions based on what stage communications between hosts have reached.

Modern firewalls can filter traffic based on many packet attributes like source IP address, source port, destination IP address or port, destination service like WWW or FTP. They can filter based on protocols, TTL values, netblock of originator, domain name of the source, and many other attributes.

Commonly used packet filters on various versions of Unix are ipf (various), ipfw (FreeBSD/Mac OS X), pf (OpenBSD, and all other BSDs), iptables/ipchains (Linux).

Application-layer

Main article: Application layer firewall

Application-layer firewalls work on the application level of the TCP/IP stack (i.e., all browser traffic, or all telnet or ftp traffic), and may intercept all packets traveling to or from an application. They block other packets (usually dropping them without acknowledgement to the sender). In principle, application firewalls can prevent all unwanted outside traffic from reaching protected machines.

On inspecting all packets for improper content, firewalls can restrict or prevent outright the spread of networked computer worms and trojans. In practice, however, this becomes so complex and so difficult to attempt (given the variety of applications and the diversity of content each may allow in its packet traffic) that comprehensive firewall design does not generally attempt this approach.

The XML firewall exemplifies a more recent kind of application-layer firewall.

Proxies

Main article: Proxy server

A proxy device (running either on dedicated hardware or as software on a general-purpose machine) may act as a firewall by responding to input packets (connection requests, for example) in the manner of an application, whilst blocking other packets.

Proxies make tampering with an internal system from the external network more difficult and misuse of one internal system would not necessarily cause a security breach exploitable from outside the firewall (as long as the application proxy remains intact and properly configured). Conversely, intruders may hijack a publicly-reachable system and use it as a proxy for their own purposes; the proxy then masquerades as that system to other internal machines. While use of internal address spaces enhances security, crackers may still employ methods such as IP spoofing to attempt to pass packets to a target network.

Network address translation

Main article: Network address translation

Firewalls often have network address translation (NAT) functionality, and the hosts protected behind a firewall commonly have addresses in the "private address range", as defined in RFC 1918. Firewalls often have such functionality to hide the true address of protected hosts. Originally, the NAT function was developed to address the limited number of IPv4 routable addresses that could be used or assigned to companies or individuals as well as reduce both the amount and therefore cost of obtaining enough public addresses for every computer in an organization. Hiding the addresses of protected devices has become an increasingly important defense against network reconnaissance.

5.9. PROXY SERVERS

In computer networks, a **proxy server** is a server (a computer system or an application program) which services the requests of its clients by forwarding requests to other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server provides the resource by connecting to the specified server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it would 'cache' the first request to the remote server, so it could save the information for later, and make everything as fast as possible.

A proxy server that passes all requests and replies unmodified is usually called a gateway or sometimes tunneling proxy.

A proxy server can be placed in the user's local computer or at specific key points between the user and the destination servers or the Internet.

Types and functions

Proxy servers implement one or more of the following functions:

Caching proxy server

A proxy server can service requests without contacting the specified server, by retrieving content saved from a previous request, made by the same client or even other clients. This is called caching. Caching proxies keep local copies of frequently requested resources, allowing large organizations to significantly reduce their upstream bandwidth usage and cost, while significantly increasing performance. There are well-defined rules for caching. Some poorly-implemented caching proxies have had downsides (e.g., an inability to use user authentication). Some problems are described in RFC 3143 (Known HTTP Proxy/Caching Problems).

Web proxy

A proxy that focuses on WWW traffic is called a "web proxy". The most common use of a web proxy is to serve as a web cache. Most proxy programs (e.g. Squid, NetCache) provide a means to deny access to certain URLs in a blacklist, thus providing content filtering. This is usually used in a corporate environment, though with the increasing use of Linux in small businesses and homes, this function is no longer confined to large corporations. Some web proxies reformat web pages for a specific purpose or audience (e.g., cell phones and PDAs).

Content Filtering Web Proxy

Further information: Content-control software

A content filtering web proxy server provides administrative control over the content that may be relayed through the proxy. It is commonly used in commercial and non-commercial organizations (especially schools) to ensure that Internet usage conforms to acceptable use policy.

Common methods used for content filtering include: URL or DNS blacklists, URL regex filtering, MIME filtering, or content keyword filtering. Some products have been known to employ content analysis techniques to look for traits commonly used by certain types of content providers.

A content filtering proxy will often support user authentication, to control web access. It also usually produces logs, either to give detailed information about the URLs accessed by specific users, or to monitor bandwidth usage statistics. It may also communicate to daemon based and/or ICAP based antivirus software to provide security against virus and other malware by scanning incoming content in real time before it enters the network.

anonymizing proxy server

An anonymous proxy server (sometimes called a web proxy) generally attempts to anonymize web surfing. These can easily be overridden by site administrators, and thus rendered useless in some cases. There are different varieties of anonymizers.

Access control: Some proxy servers implement a logon requirement. In large organizations, authorized users must log on to gain access to the web. The organization can thereby track usage to individuals.

Hostile proxy

Proxies can also be installed by online criminals, in order to eavesdrop upon the dataflow between the client machine and the web. All accessed pages, as well as all forms submitted, can be captured and analyzed by the proxy operator. For this reason, passwords to online services (such as webmail and banking) should be changed if an unauthorized proxy is detected.

Intercepting proxy server

An **intercepting proxy** (also known as a "transparent proxy") combines a proxy server with a gateway. Connections made by client browsers through the gateway are redirected through the proxy without client-side configuration (or often knowledge).

Intercepting proxies are commonly used in businesses to prevent avoidance of acceptable use policy, and to ease administrative burden, since no client browser configuration is required.

It is often possible to detect the use of an intercepting proxy server by comparing the external IP address to the address seen by an external web server, or by examining the HTTP headers on the server side.

Transparent and non-transparent proxy server

The term "transparent proxy" is most often used incorrectly to mean "intercepting proxy" (because the client does not need to configure a proxy and cannot directly detect that its requests are being proxied).

However, RFC 2616 (Hypertext Transfer Protocol -- HTTP/1.1) offers different definitions:

> "A 'transparent proxy' is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification".
> "A 'non-transparent proxy' is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering".

Forced proxy

The term "forced proxy" is ambiguous. It means both "intercepting proxy" (because it filters all traffic on the only available gateway to the Internet) and its exact opposite, "non-intercepting proxy" (because the user is forced to configure a proxy in order to access the Internet).

Forced proxy operation is sometimes necessary due to issues with the interception of TCP connections and HTTP. For instance interception of HTTP requests can affect the usability of a proxy cache, and can greatly affect certain authentication mechanisms. This is primarily because the client thinks it is talking to a server, and so request headers required by a proxy are unable to be distinguished from headers that may be required by an upstream server (esp authorization headers). Also the HTTP specification prohibits caching of responses where the request contained an authorization header.

Open proxy server

> Main article: open proxy

Because proxies might be used for abuse, system administrators have developed a number of ways to refuse service to open proxies. many IRC networks automatically test client systems for known types of open proxy. Likewise, an email server may be configured to automatically test e-mail senders for open proxies.

Groups of IRC and electronic mail operators run DNSBLs publishing lists of the IP addresses of known open proxies, such as AHBL, CBL, NJABL, and SORBS.

The ethics of automatically testing clients for open proxies are controversial. Some experts, such as Vernon Schryver, consider such testing to be equivalent to an attacker portscanning the client host. [1] Others consider the client to have solicited the scan by connecting to a server whose terms of service include testing.

Reverse proxy server

Main article: reverse proxy

A **reverse proxy** is a proxy server that is installed in the neighborhood of one or more web servers. All traffic coming from the Internet and with a destination of one of the web servers goes through the proxy server. There are several reasons for installing reverse proxy servers:

- Encryption / SSL acceleration: when secure web sites are created, the SSL encryption is often not done by the web server itself, but by a reverse proxy that is equipped with SSL acceleration hardware. See Secure Sockets Layer.

- Load balancing: the reverse proxy can distribute the load to several web servers, each web server serving its own application area. In such a case, the reverse proxy may need to rewrite the URLs in each web page (translation from externally known URLs to the internal locations).

- Serve/cache static content: A reverse proxy can offload the web servers by caching static content like pictures and other static graphical content.

- Compression: the proxy server can optimize and compress the content to speed up the load time.

- Spoon feeding: reduces resource usage caused by slow clients on the web servers by caching the content the web server sent and slowly "spoon feeds" it to the client. This especially benefits dynamically generated pages.

- Security: the proxy server is an additional layer of defense and can protect against some OS and WebServer specific attacks. However, it does not provide any protection to attacks against the web application or service itself, which is generally considered the larger threat.

- Extranet Publishing: a reverse proxy server facing the Internet can be used to communicate to a firewalled server internal to an organization, providing extranet access to some functions while keeping the servers behind the firewalls. If used in this way, security measures should be considered to protect the rest of your infrastructure in case this server is compromised, as it's web application is exposed to attack from the Internet.

Circumventor

A **circumventor** is a method of defeating blocking policies implemented using proxy servers. Ironically, most circumventors are also proxy servers, of varying degrees of sophistication, which effectively implement "bypass policies".

A circumventor is a web-based page that takes a site that is blocked and "circumvents" it through to an unblocked web site, allowing the user to view blocked pages. A famous example is '**elgooG**', which allowed users in China to use Google after it had been blocked there. elgooG differs from most circumventors in that it circumvents only one block.

Students are able to access blocked sites (games, chatrooms, messenger, offensive material, internet pornography, social networking, etc.) through a circumventor. As fast as the filtering software blocks circumventors, others spring up. However, in some cases the filter may still intercept traffic to the circumventor, thus the person who manages the filter can still see the sites that are being visited.

Circumventors are also used by people who have been blocked from a web site.

Another use of a circumventor is to allow access to country-specific services, so that Internet users from other countries may also make use of them. An example is country-restricted reproduction of media and webcasting.

The use of circumventors is usually safe with the exception that circumventor sites run by an untrusted third party can be run with hidden intentions, such as collecting personal information, and as a result users are typically advised against running personal data such as credit card numbers or passwords through a circumventor.

At schools and offices

Many work places and schools restrict the web sites and online services that are made available in their buildings. Since circumventors are used to bypass censors in computers, social networking and other sites deemed a waste of time or resources have become targets of mass banning.

Proxy Web server creators have become more sophisticated, allowing users to encrypt links and any data going to and from other web servers. This allows users to access websites that would otherwise have been blocked.

A special case of web proxies are "CGI proxies". These are web sites that allow a user to access a site through them. They generally use PHP or CGI to implement the proxy functionality. These types of proxies are frequently used to gain access to web sites blocked by corporate or school proxies. Since they also hide the user's own IP address from the web sites they access through the proxy, they are sometimes also used to gain a degree of anonymity, called "Proxy Avoidance".

Risks of using anonymous proxy servers

In using a proxy server (for example, anonymizing HTTP proxy), all data sent to the service being used (for example, HTTP server in a website) must pass through the proxy server before being sent to the service, mostly in unencrypted form. It is therefore possible, as has been demonstrated, for a malicious proxy server to record everything sent to the proxy: including unencrypted logins and passwords.

By chaining proxies which do not reveal data about the original requester, it is possible to obfuscate activities from the eyes of the user's destination. However, more traces will be left on the intermediate hops, which could be used or offered up to trace the user's activities. If the policies and administrators of these other proxies are unknown, the user may fall victim to a false sense of security just because those details are out of sight and mind.

The bottom line of this is to be wary when using proxy servers, and only use proxy servers of known integrity (e.g., the owner is known and trusted, has a clear privacy policy, etc.), and never use proxy servers of unknown integrity. If there is no choice but to use unknown proxy servers, do not pass any private information (unless it is properly encrypted) through the proxy.

In what is more of an inconvenience than a risk, proxy users may find themselves being blocked from certain Web sites, as numerous forums and Web sites block IP addresses from proxies known to have spammed or trolled the site.

Proxy software

- AlchemyPoint is a user-programmable mashup proxy server that can be used to re-write web pages, emails, instant messenger messages, and other network transmissions on the fly.
- The Apache HTTP Server can be configured to act as a proxy server.
- Blue Coat's (formerly Cacheflow's) purpose-built SGOS proxies 15 protocols including HTTPS/SSL, has an extensive policy engine and runs on a range of appliances from branch-office to enterprise.
- EZproxy is a URL-rewriting web proxy designed primarily for providing remote access to sites that authenticate users by IP address.
- JAP - A local proxy, web anonymizer software connecting to proxy server chains of different organisations

- Microsoft Internet Security and Acceleration Server is a product that runs on Windows 2000/2003 servers and combines the functions of both a proxy server and a firewall.
- Novell BorderManager web proxy server, reverse proxy, Firewall and VPN end point.
- Nginx Web and Reverse proxy server, that can act as POP3 proxy server.
- Privoxy is a free, open source web proxy with privacy and ad-blocking features.
- Proxomitron - User-configurable web proxy used to re-write web pages on the fly. Most noted for blocking ads, but has many other useful features.
- SafeSquid Linux based, complete content filtering HTTP1.1 proxy, allows distribution of 'profiled' internet access.
- SSH Secure Shell can be configured to proxify a connection, by setting up a SOCKS proxy on the client, and tunneling the traffic through the SSH connection.
- sun Java System Web Proxy Server is a caching proxy server running on Solaris, Linux and Windows servers that supports HTTP/S, NSAPI I/O filters, dynamic reconfiguration, SOCKSv5 and reverse proxy.
- Squid is a popular HTTP proxy server in the UNIX/Linux world.
- tinyproxy is a fast light-weight HTTP proxy ideal for embedded use on POSIX operating systems.
- Tor - A proxy-based anonymizing Internet communication system.
- Varnish is designed to be a high-performance caching reverse proxy.
- WinGate is a multi-protocol proxy server and NAT solution that can be used to redirect any kind of traffic on a Microsoft Windows host.

- WWWOFFLE has been around since the mid-1990s, and was developed for storing online data for offline use.

- yProxy is an NNTP proxy server that converts yEnc encoded message attachments to UUEncoding, complete with SSL client support.

- Ziproxy is a non-caching proxy for acceleration purposes. It recompresses pictures and optimizes HTML code.

Summary::

- Web Publishing provides custom web design, web development, hosting, e-commerce, and e-business solutions.
- It can include or can load scripts in languages such as JavaScript which affect the behavior of HTML processors like Web browsers; and Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup.
- The Java 2 Platform, Enterprise Edition (J2EE) is the Java architecture for developing multitier enterprise applications. As part of J2EE, JSP pages have access to all J2EE components, including JavaBeans and Enterprise JavaBeans components and Java servlets. JSP pages are actually compiled into servlets, so they have all of the benefits of these flexible, server-side Java applications. The J2EE platform containers manage the complexities of enterprise applications, including transaction management and resource pooling.
- JSP pages have access to all of the standard J2EE services, including:
- Java Naming and Directory Interface™ API

- JDBC™ API (communicating with relational databases)

- JavaMail™ (classes supporting Java-based mail and messaging applications)

- Java™ Message Service (JMS)

- Through J2EE, JSP pages can interact with enterprise systems in many ways. J2EE supports two CORBA-compliant technologies: Java IDL and RMI-IIOP. With Enterprise JavaBeans technology, JSP pages can access databases using high-level, object-relational mappings.

Key words::

- Dynamic HTML
- User Preferences
- Cookies
- Active-X controls
- XML ,AJAX

Key Term quiz::

1. Query string values can be retrieved from the servlet using one of the following functions
   - getParameter() defined in ServletRequest
   - getServletInfo() defined in ServletContext
   - getInitParameter() defined in ServletConfig
   - none of the above

2 .getParameterNames () is available in the _____object
   - request
   - Response
   - ServletConfig
   - None of the above

3. A servlet uses an instance of _____to send data to the client
   - ServletResponse
   - ServletRequest
   - ServletContext
   - None of the above

4. A HttpServletRequest instance invoking the method, getSession(true) indicates one of the following
   - returns the current HttpSession associated with this request or, if there is no current session returns a new session
   - returns the current HttpSession associated with this request or, if there is no current session returns null
   - always starts a new Session and returns the HttpSession object
   - none of the above

5..A link to another URL form this servlet can be brought about by one of the following method

- encodeRedirectURL()
- encodeURL()
- sendRedirect()
- none of the above

Multiple Choice ::

1. To store a value in the HTTP session the _____method is used
   - HttpSession.storeValue()
   - HttpSession.putValue()
   - HttpSession.writeValue()
   - None of the above

2. The_____method is used to set a cookie to the client
   - Request.setCookie()
   - Request.addCookie()
   - Response.setCookie()
   - Response.addCookie()

3. The class that encapsulates an HTTP cookie is
   - Http Cookie
   - Browser Cookie
   - Cookie
   - ServletCookie

4. Which of the following is true about cookies?
   - cookies are browser dependent
   - cookies are stored both at client and server
   - since cookies are written in the Header they cant be sent after writing content into the packet
   - cookies cannot be set on Netscape Navigator

5 .Which is the correct sequence of method calls for a Get Request for the first time
   - init(),service(),doPost()
   - init(),service(),doGet()
   - init(),doPost()
   - init(),doGet()

6. Is it possible to find out the browser used by a client from a servlet?
   - no
   - yes
   - only if the web server is java web server
   - Only if we have Jrun installed on it.

7. The _____ method is used to inform the browser about the type of data sent to it from a servlet.
   - Response.setDataType()
   - Response.setType()
   - Response.setContentType()
   - ServletConfig.setType()

8. Which of the following methodologies can be used to suppress runtime exception in Servlet?
   - use throws in the service() method
   - use sendRedirect() in the catch block
   - Response.suppressErrors()
   - None of the above

9. Which of the following is true about Http Session?
   - they are browser dependent
   - they are maintained by the browser
   - cookies cannot be set on internet explorer
   - none of the above

10. The return type of Request.getParameterValue () is
   - Enumeration
   - Vector
   - String[]
   - Hash table

Review Questions

In your own words briefly answer the following Questions

1. What are the services provided by the container?
2. Types of transaction?
3. What are transaction attributes?
4. What is JTS?
5. What is connection polling? Is it advantageous?
6. Method and class used for connection pooling?
7. Method in Session Synchronization. Order of Execution
8. What is a transaction, how are they handled in EJBs?
9. What happens when a getPrimaryKey() is called from a Session Bean's remote interface?
10. What are connection pools? How to they affect performance?
11. How to create Transaction on the client side/ on a Stateless bean using code?
12. What happens when getUserTransaction() is called on a stateful bean with container/bean managed transaction?
13. Two beans A, B (Both stateful sessions) what are the possible configurations so that B can be looked up from A or the reverse?
14. What is MVC?
15. Is it mandatory to have at least one servlet in the model-view-controller arch? If so what is the role of that servlet?

Lesson Lab::

Complete the following exercise

1. Create server side page to display results of a student using Servlet
2. Create server side page to display results of a student using Entity Bean (CMP)
3. Develop a web page using ASP Components
4. Create an n-tier Architecture for banking transaction