**FUNDAMENTALS:**

**Web- Introduction to the World Wide Web:**

- The "Web," short for "World Wide Web" (which gives us the acronym www), is the name for one of the ways that the internet lets people browse documents connected by hypertext links.

- The concept of the Web was perfected at CERN (Conseil Européen pour la Recherche Nucléaire, or the European Organization for Nuclear Research) in 1991 by a group of researchers which included Tim-Berners Lee, the creator of the hyperlink, who is today considered the father of the Web.

- The principle of the Web is based on using hyperlinks to navigate between documents (called web pages) with a program called a browser. A web page is a simple text file written in a markup language (called HTML) that encodes the layout of the document, graphical elements, and links to other documents, all with the help of tags.

- Besides the links which connect formatted documents to one another, the web uses the HTTP protocol to link documents hosted on distant computers (called web servers, as opposed to the client represented by the browser).

- On the internet, documents are identified with a unique address, called a URL, which can be used to locate any resource on the internet, no matter which server may be hosting it.

**What is a website?**

- A website (also called an internet site or a home page in the case of a personal site) is a group of HTML files that are stored on a hosting computer which is permanently connected to the internet (a web server).

- A website is normally built around a central page, called a welcome page, which offers links to a group of other pages hosted on the same server, and sometimes "external" links, which lead to pages hosted by another server.

- **Web server** is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

- Web site is collection of web pages while web server is a software that respond to the request for web resources

- Web server respond to the client request in either of the following two ways:
  - Sending the file to the client associated with the requested URL.
  - Generating response by invoking a script and communicating with database
• When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.

• If the requested web page is not found, web server will the send an **HTTP response: Error 404 Not found.**

• If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

• Web Server Architecture follows the following two approaches:

  • **Concurrent Approach**

  • **Single-Process-Event-Driven Approach.**

  **Concurrent Approach**

  • Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

  • Multi-process

  • Multi-threaded

  Hybrid method.

  • Multi-processing
• In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

• It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

• Multi-threaded

• Unlike Multi-process, it creates multiple single-threaded process.

• Hybrid

• It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

Web Application:

• A website is a collection of static files(webpages) such as HTML pages, images, graphics etc. A Web application is a web site with dynamic functionality on the server. Google, Facebook, Twitter are examples of web applications.

HTTP (Hypertext Transfer Protocol)

• HTTP is a protocol that clients and servers use on the web to communicate.

• It is similar to other internet protocols such as SMTP(Simple Mail Transfer Protocol) and FTP(File Transfer Protocol) but there is one fundamental difference.

• HTTP is a stateless protocol i.e HTTP supports only one request per connection. This means that with HTTP the clients connect to the server to send one request and then disconnects. This mechanism allows more users to connect to a given server over a period of time.

• The client sends an HTTP request and the server answers with an HTML page to the client, using HTTP.
Request for communication options that are available on the request/response chain.

GET - Request to retrieve information from server using a given URI.

HEAD - Identical to GET except that it does not return a message-body, only the headers and status line.

POST - Request for server to accept the entity enclosed in the body of HTTP method.

DELETE - Request for the Server to delete the resource.

CONNECT - Reserved for use with a proxy that can switch to being a tunnel.

PUT - This is same as POST, but POST is used to create, PUT can be used to create as well as update. It replaces all current representations of the target resource with the uploaded content.

**General Difference between PUT and POST methods:**

- Following are some basic differences between the PUT and the POST methods:
  - **POST** to a URL creates a child resource at a server defined URL while **PUT** to a URL creates/replaces the resource in its entirety at the client defined URL.
  - POST creates a child resource, so POST to /books will create a resources that will live under the /books resource. Eg. /books/1. Sending the same post request twice will create two resources.
  - PUT is for creating or replacing a resource at a URL known by the client.
  - PUT must be used for CREATE when the client already knows the url before the resource is created.
  - PUT replaces the resource at the known url if it already exists, so sending the same request twice has no effect. In other words, calls to PUT are **idempotent**.

**Anatomy of an HTTP GET request**

- Get request contains path to server and the parameters added to it.
Anatomy of an HTTP POST request:

- Post requests are used to make more complex requests on the server. For instance, if a user has filled a form with multiple fields and the application wants to save all the form data to the database. Then the form data will be sent to the server in POST request body, which is also known as Message body.

Message format in web technology:

- There are a number of web services that use markup languages:
  - JSON-RPC.
• JSON-WSP.
• Web template.
• Web Services Description Language (WSDL) from the W3C.
• XML Interface for Network Services (XINS) provides a POX-style Web service specification format.
• Web Services Conversation Language (WSCL)
• Web Services Flow Language (WSFL) (superseded by BPEL)
• WS-MetadataExchange
• Representational state transfer (REST) versus remote procedure call (RPC)
• XML-RPC - XML - Remote Procedure Call

Web service change management:

• Work related to the dealing with the visualization and capturing changes in a Web service. Visualization and computation of changes can be done in the form of intermediate artifacts (Subset WSDL). The insight on computation of change impact is helpful in testing, top down development and reduces regression testing. Automated Web Service Change Management (AWSCM) is a tool that identifies subset operations in a WSDL file to construct a subset WSDL.

Non-persistent and persistent connections:

Non-persistent

• HTTP/1.0
• server parses request, responds, and closes TCP connection
• 2 RTTs to fetch each object
• Each object transfer suffers from slow start

Persistent:

• default for HTTP/1.1
• on same TCP connection: server, parses request, responds, parses new request,..
• Client sends requests for all referenced objects as soon as it receives base HTML.
• Fewer RTTs and less slow start.
• But most 1.0 browsers use parallel TCP connections.

HTTP 1.0:

• Under HTTP 1.0, there is no official specification for how keepalive operates. It was, in essence, added to an existing protocol. If the client supports keep-alive, it adds an additional header to the request:

  • Connection: keep-alive  Then, when the server receives this request and generates a response, it also adds a header to the response:

  • Connection: keep-alive  following this, the connection is not dropped, but is instead kept open. When the client sends another request, it uses the same connection

  • This will continue until either the client or the server decides that the conversation is over, and one of them drops the connection.

HTTP 1.1:

• In HTTP 1.1, all connections are considered persistent unless declared otherwise. The HTTP persistent connections do not use separate keepalive messages, they just allow multiple requests to use a single connection. However, the default connection timeout of Apache httpd 1.3 and 2.0 is as little as 15 seconds and just 5 seconds for Apache httpd 2.2 and above.

• The advantage of a short timeout is the ability to deliver multiple components of a web page quickly while not consuming resources to run multiple server processes or threads for too long

  Advantages:

  • Lower CPU and memory usage (because fewer connections are open simultaneously).

  • Enables HTTP pipelining of requests and responses.

  • Reduced network congestion (fewer TCP connections).

  • Reduced latency in subsequent requests (no handshaking).

  • Errors can be reported without the penalty of closing the TCP connection.

  • These advantages are even more important for secure HTTPS connections, because establishing a secure connection needs much more CPU time and network round-trips.

According to RFC 7230, section 6.4, "a client ought to limit the number of simultaneous open connections that it maintains to a given server". The previous version of the HTTP/1.1 specification stated specific maximum values but in the words of RFC 7230 "this was found to be
impractical for many applications... instead... be conservative when opening multiple connections". These guidelines are intended to improve HTTP response times and avoid congestion. If HTTP pipelining is correctly implemented, there is no performance benefit to be gained from additional connections, while additional connections may cause issues with congestion.

Disadvantages:

- If the client does not close the connection when all of the data it needs has been received, the resources needed to keep the connection open on the server will be unavailable for other clients. How much this affects the server's availability and how long the resources are unavailable depend on the server's architecture and configuration.

Web cache:

- A web cache (or HTTP cache) is an information technology for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. A web cache system stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met. A web cache system can refer either to an appliance, or to a computer program.
Web cache:

- A web cache (or HTTP cache) is an information technology for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. A web cache system stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.[1] A web cache system can refer either to an appliance, or to a computer program.

Cache control:

- HTTP defines three basic mechanisms for controlling caches: freshness, validation, and invalidation.

- Freshness allows a response to be used without re-checking it on the origin server, and can be controlled by both the server and the client. For example, the Expires response header gives a date when the document becomes stale, and the Cache-Control: max-age directive tells the cache how many seconds the response is fresh for. Validation can be used to check whether a cached response is still good after it becomes stale.

- For example, if the response has a Last-Modified header, a cache can make a conditional request using the If-Modified-Since header to see if it has changed. The ETag (entity tag) mechanism also allows for both strong and weak validation. Invalidation is usually a side effect of another request that passes through the cache. For example, if a URL associated with a cached response subsequently gets a POST, PUT or DELETE request, the cached response will be invalidated. Many CDNs and manufacturers of network equipment have replaced this standard HTTP cache control with dynamic caching.

Creating a web service:


- Select File->New->Website. Select ASP.NET Web service as the type of the project and enter InventoryWS as the name of the project. Visual Studio provides Service.cs and Service.asmx files along with others. Delete these two files.

- We add a new web service to the project using Website -> Add new item -> Web Service and enter name as InventoryService.

- Visual Studio creates InventoryService.aspx in root directory and InventoryService.cs in App_Code directory.

- Creating Data Access Layer
• Data Access Layer is set of classes used to access database. We centralize the entire data access to DAL.

Creating Proxy in Java:

• Now, let us create a proxy in Java to access InventoryService Web Service. Start NetBeans IDE 6.5.
• Select File->New Project. Select Java in category and Java Application in projects. Click on Next.
• Enter InventoryClient as the name of the application. Change name of the class in Create Main Class option to Client
• Click on Finish.
• Select InventoryClient project in Projects window. Right click to invoke context menu. Select New->Other. Select Web service in categories and Web Service Client in File Types.
• In the next window, select WSDL URL radio button (as shown below) and enter the URL at which InventoryService is running. Click on Finish. NetBeans creates required classes to access web service.

Java – Networking:

• The term network programming refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.
• The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.
• The java.net package provides support for the two common network protocols –
  • TCP – TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
  • UDP – UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.
• In this chapter gives a good understanding on the following two subjects –
  • Socket Programming – This is the most widely used concept in Networking and it has been explained in very detail.
  • URL Processing – This would be covered separately. Click here to learn about URL Processing in Java language.
Socket Programming:

- Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

- When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket.

- The java.net.Socket class represents a socket, and the java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.

Looking Up Internet Addresses:

- DNS, IP Addresses, and All That

- Devices connected to the Internet are called nodes. Nodes that are computers are called hosts. Each node or host is identified by at least one unique 32-bit number called an Internet address, an IP address, or a host address, depending on who you talk to. This takes up exactly four bytes of memory. An IP address is normally written as four unsigned bytes, each ranging from 0 to 255, with the most significant byte first. Bytes are separated by periods for the convenience of human eyes. For example, the address for hermes.oit.unc.edu is 152.2.21.1. This is called the dotted quad format.

Looking Up Internet Addresses:

- DNS, IP Addresses, and All That

- Devices connected to the Internet are called nodes. Nodes that are computers are called hosts. Each node or host is identified by at least one unique 32-bit number called an Internet address, an IP address, or a host address, depending on who you talk to. This takes up exactly four bytes of memory. An IP address is normally written as four unsigned bytes, each ranging from 0 to 255, with the most significant byte first. Bytes are separated by periods for the convenience of human eyes. For example, the address for hermes.oit.unc.edu is 152.2.21.1. This is called the dotted quad format.

- IP addresses are great for computers, but they are a problem for humans, who have a hard time remembering long numbers. In the 1950s, it was discovered that most people could remember about seven digits per number; some can remember as many as nine, while others remember as few as five. This is why phone numbers are broken into three- and four-digit pieces with three-digit area codes. Obviously an IP address, which can have as many as 12 decimal digits, is beyond the capacity of most humans to remember. I can remember about two IP addresses, and then only if I use both daily and the second is a simple permutation of the first.
Email client:

- In Internet, an email client, email reader or more formally mail user agent (MUA) is a computer program in the category of groupware environments used to access and manage a user’s email.

- Client is meant to be a role. For example, a web application which provides message management, composition, and reception functions may internally act as an email client; as a whole, it is commonly referred to as webmail. Likewise, email client may be referred to a piece of computer hardware or software whose primary or most visible role is to work as an email client.

Remote messages

- POP3 has an option to leave messages on the server. By contrast, both IMAP and webmail keep messages on the server as their method of operating, albeit users can make local copies as they like. Keeping messages on the server has advantages and disadvantages.

Advantages

- Messages can be accessed from various computers or mobile devices at different locations, using different clients.

- Some kind of backup is usually provided by the server.

Disadvantages

- With limited bandwidth, access to long messages can be lengthy, unless the email client caches a local copy.

- There may be privacy concerns, since messages that stay on the server at all times have more chances to be casually accessed by IT personnel, unless end-to-end encryption is used.

Protocols:

- While popular protocols for retrieving mail include POP3 and IMAP4, sending mail is usually done using the SMTP protocol.

- Another important standard supported by most email clients is MIME, which is used to send binary file email attachments. Attachments are files that are not part of the email proper, but are sent with the email.

- Most email clients use a User-Agent header field to identify the software used to send the message. According to RFC 2076, this is a common but non-standard header field.

- RFC 6409, Message Submission for Mail, details the role of the Mail submission agent.
• RFC 5068, *Email Submission Operations: Access and Accountability Requirements*, provides a survey of the concepts of MTA, MSA, MDA, and MUA. It mentions that "Access Providers MUST NOT block users from accessing the external Internet using the SUBMISSION port 587" and that "MUAs SHOULD use the SUBMISSION port for message submission."

**Remote Method Invocation:**

• Java’s Remote Method Invocation (commonly referred to as RMI) is used for client and server models. RMI is the object oriented equivalent to RPC (Remote procedure call).

• The Java Remote Method Invocation (RMI) system allows an object running in one Java Virtual Machine (VM) to invoke methods of an object running in another Java VM. RMI provides for remote communication between programs written in the Java programming language.

• RMI is only defined for use with the Java platform. If you need to call methods between different language environments, use CORBA. With CORBA a Java client can call a C++ server and/or a C++ client can call a Java server. With RMI that can not be done.

• STUB and SKELETON

• The remote method invocation goes through a STUB on the client side and a so called SKELETON on the server side.

• `CLIENT --> STUB --> ... Network ... --> SKELETON --> REMOTE OBJECT Prior to Java 1.2 the skeleton had to be explicitly generated with the `rmic` tool. Since 1.2 a dynamic skeleton is used, which employs the features of Java Reflection to do its work.

• `rmiregistry[edit]`

• Remote objects can be listed in the RMI Registry. Clients can get a reference to the remote object by querying the Registry. After that, the client can call methods on the remote objects. (Remote object references can also be acquired by calling other remote methods. The Registry is really a 'bootstrap' that solves the problem of where to get the initial remote reference from.)

• The RMI Registry can either be started within the server JVM, via the `LocateRegistry.createRegistry()` API, or a separate process called `rmiregistry` that has to be started before remote objects can be added to it, e.g. by the command line in Unix.

```java
import java.rmi.registry.LocateRegistry; 2 import java.rmi.registry.Registry; 3 4  public class HelloClient{ 5 6 private HelloClient() {} 7 8 public static void main(String[] args) { 9 String host = (args.length < 1) ? null : args[0]; 10 try { 11 Registry registry = LocateRegistry.getRegistry(host); 12 Hello stub = (Hello) registry.lookup("Hello"); 13 String response = stub.sayHello(); 14 System.out.println("response: " + response); 15 } catch (Exception e) { 16 System.err.println("Client exception: " + e.toString()); 17 e.printStackTrace(); 18 } 19 }
```