📥

Day11: Http request, Http response, WebServer, Webapplication v/s WebServices Architecture

Web Servers:

Web pages are a collection of data, including images, text files, hyperlinks, database files etc., all located on some computer (also known as server space) on the Internet. A web server is dedicated software that runs on the server-side. When any user requests their web browser to run any web page, the webserver places all the data materials together into an organized web page and forwards them back to the web browser with the help of the Internet. Therefore, we can conclude that: -

> A web server is a dedicated computer responsible for running websites sitting out on those computers somewhere on the Internet. They are specialized programs that circulate web pages as summoned by the user. The primary objective of any web server is to collect, process and provide web pages to the users.

One of the more popular web servers in the Java ecosystem is Apache Tomcat .

HTTP (Hyper Text Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems. It is the data communication protocol used to establish communication between client and server.

HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80. It provides the standardized way for computers to communicate with each other.
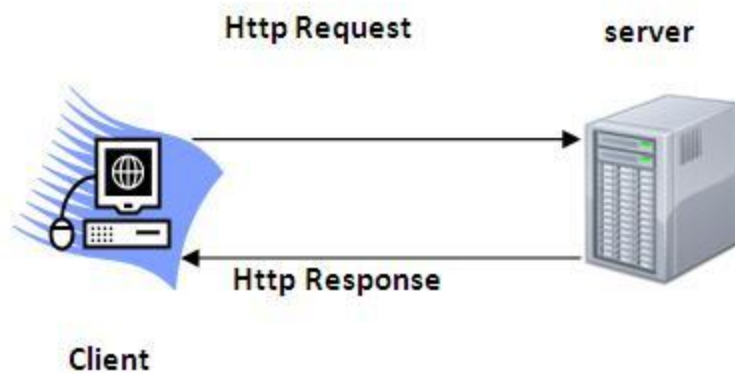
The Basic Characteristics of HTTP (Hyper Text Transfer Protocol):

- It is the protocol that allows web servers and browsers to exchange data over the web.

- It is a request response protocol.

- It uses the reliable TCP connections by default on TCP port 80.

- It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.

The Basic Features of HTTP (Hyper Text Transfer Protocol):

There are three fundamental features that make the HTTP a simple and powerful protocol used for communication:

- **HTTP is media independent:** It specifies that any type of media content can be sent by HTTP as long as both the server and the client can handle the data content.

- **HTTP is connectionless:** It is a connectionless approach in which HTTP client i.e., a browser initiates the HTTP request and after the request is sent the client disconnects from server and waits for the response.

- **HTTP is stateless:** The client and server are aware of each other during a current request only. Afterwards, both of them forget each other. Due to the stateless nature of protocol, neither the client nor the server can retain the information about different request across the web pages.

HTTP Requests:

he request sent by the computer to a web server, contains all sorts of potentially interesting information; it is known as HTTP requests.

The HTTP client sends the request to the server in the form of request message which includes following information:

- The Request-line

- The analysis of source IP address, proxy and port

- The analysis of destination IP address, protocol, port and host

- The Requested URI (Uniform Resource Identifier)

- The Request method and Content

- The User-Agent header

- The Connection control header

- The Cache control header

The HTTP request method indicates the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**. This method is case-sensitive and should be used in uppercase.

The HTTP request methods are:

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can response

Anatomy of Get Request:

| Path to the source on Web Server | Parameters to the server | Protocol Version Browser supports |

The HTTP Method

GET /RegisterDao.jsp?user=ravi&pass=java HTTP/1.1

The Request Headers
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept-text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive

Anatomy of Post Request

The HTTP Method — Path to the source on Web Server — Protocol Version Browser supports

Post /RegisterDao.jsp HTTP/1.1
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive:300
Connection:keep-alive
User=ravi&pass=java

The Request Headers

Message body

Http Response:

HTTP Response sent by a server to the client. The response is used to provide the client with the resource it requested. It is also used to inform the client that the action requested has been carried out. It can also inform the client that an error occurred in processing its request.

An HTTP response contains the following things:

1. Status Line

2. Response Header Fields or a series of HTTP headers

3. Message Body

In the request message, each HTTP header is followed by a carriage returns line feed (CRLF). After the last of the HTTP headers, an additional CRLF is used and then begins the message body.

Status Line

In the response message, the status line is the first line. The status line contains three items:

he status line is the first line in the response message. It consists of three items:

1. The HTTP version number, showing the HTTP specification to which the server has tried to make the message comply.
2. A **status code**, which is a three-digit number indicating the result of the request.
3. A **reason phrase**, also known as status text, which is human-readable text that summarizes the meaning of the status code.

An example of a response line is:

```
HTTP/1.1 200 OK
```

In this example:

- the HTTP version is `HTTP/1.1`
- the status code is `200`
- the reason phrase is `OK`

Status codes and reason phrases explains more about these elements of the status line.

HTTP headers

The HTTP headers for a server's response contain information that a client can use to find out more about the response, and about the server that sent it. This information can assist the client with displaying the response to a user, with storing (or caching) the response for future use, and with making further requests to the server now or in the future. For example, the following series of headers tell the client when the response was sent, that it was sent by CICS®, and that it is a JPEG image:

```
Date: Thu, 09 Dec 2004 12:07:48 GMT
Server: IBM_CICS_Transaction_Server/3.1.0(zOS)
Content-type: image/jpg
```

In the case of an unsuccessful request, headers can be used to tell the client what it must do to complete its request successfully.

An empty line (that is, a CRLF alone) is placed in the response message after the series of HTTP headers, to divide the headers from the message body.

Message body

The message body of a response may be referred to for convenience as a response body.

Message bodies are used for most responses. The exceptions are where a server is responding to a client request that used the HEAD method (which asks for the headers but not the body of the response), and where a server is using certain status codes.

For a response to a successful request, the message body contains either the resource requested by the client, or some information about the status of the action requested by the client. For a response to an unsuccessful request, the message body might provide further information about the reasons for the error, or about some action the client needs to take to complete the request successfully.

What is Web Service:

A **Web Service** is can be defined by following ways:

- It is a client-server application or application component for communication.
- The method of communication between two devices over the network.
- It is a software system for the interoperable machine to machine communication.
- It is a collection of standards or protocols for exchanging information between two devices or application.
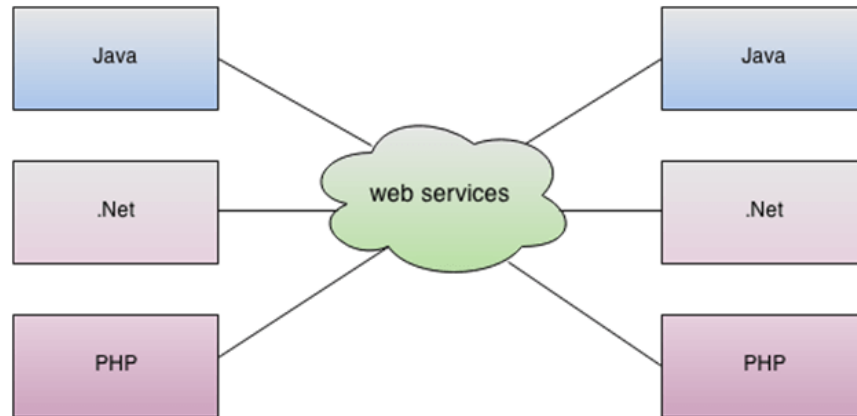
As you can see in the figure, Java, .net, and PHP applications can communicate with other applications through web service over the network. For example, the Java application can interact with Java, .Net, and PHP applications. So web service is a language independent way of communication.

Types of Web Services

There are mainly two types of web services.

1. SOAP web services.

2. RESTful web services.
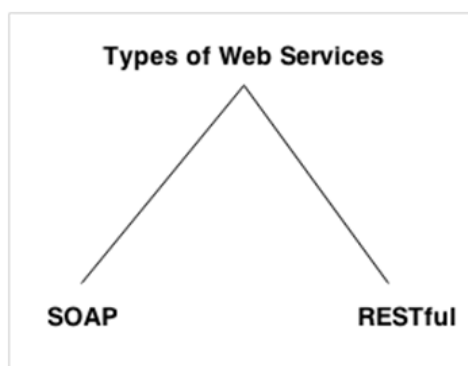
SOAP Web Services

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.

SOAP is a W3C recommendation for communication between two applications.

SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, you will be able to interact with other programming language applications.

Disadvantages of Soap Web Services

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

RESTful Web Services

REST stands for REpresentational State Transfer.

REST is an architectural style not a protocol.

Advantages of RESTful Web Services

Fast: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

Language and Platform independent: RESTful web services can be written in any programming language and executed in any platform.

Permits different data format: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.