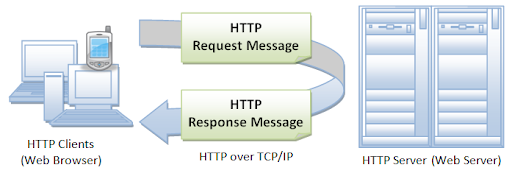
**HTTP**

**Hypertext Transfer Protocol (HTTP)** is an [application-layer](https://en.wikipedia.org/wiki/Application_Layer) protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical [client-server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model), with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a [stateless protocol](https://en.wikipedia.org/wiki/Stateless_protocol), meaning that the server does not keep any data (state) between two requests.

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files -- such as text, images, sound, video and other multimedia files -- over the web. As soon as a user opens their web browser, they are indirectly using HTTP. HTTP is an application protocol that runs on top of the [TCP/IP](https://techtarget.com/searchnetworking/definition/TCP-IP) suite of protocols, which forms the foundation of the internet.



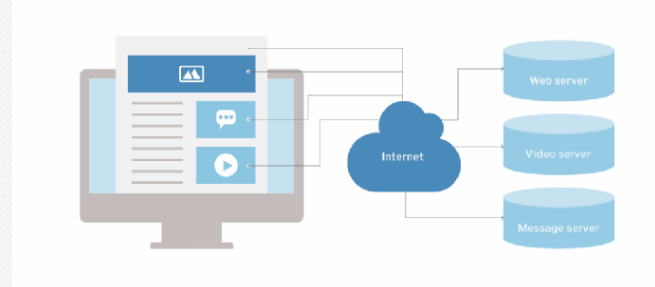
**Purpose of HTTP**

HTTP was invented alongside HTML to create the first interactive, text-based web browser: the original World Wide Web. Today, the protocol remains one of the primary means of using the Internet.

#### How does HTTP work?

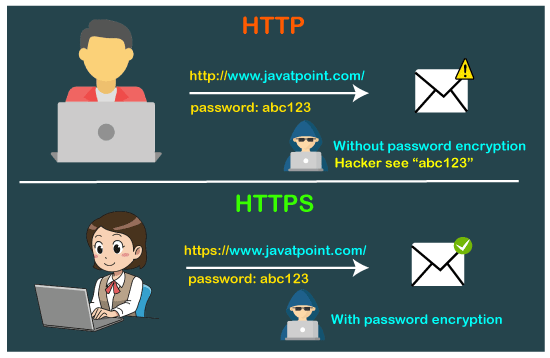
Through the HTTP protocol, resources are exchanged between client devices and servers over the internet. Client devices send requests to servers for the resources needed to load a web page; the servers send responses back to the client to fulfil the requests. Requests and responses share sub-documents -- such as data on images, text, text layouts, etc. -- which are pieced together by a client web browser to display the full web page file.

In addition to the web page files it can serve, a [web server](https://whatis.techtarget.com/definition/Web-server) contains an HTTP [daemon](https://whatis.techtarget.com/definition/daemon), a program that waits for HTTP requests and handles them when they arrive. A web browser is an HTTP client that sends requests to servers. When the browser user enters file requests by either "opening" a web file by typing in a URL or clicking on a hypertext [link](https://whatis.techtarget.com/definition/link), the browser builds an HTTP request and sends it to the Internet Protocol address ([IP address](https://searchwindevelopment.techtarget.com/definition/IP-address)) indicated by the URL. The HTTP daemon in the destination server receives the request and sends back the requested file or files associated with the request. HTTP clients generally use Transmission Control Protocol (TCP) connections to communicate with servers.



HTTP utilizes [specific request methods](https://tools.ietf.org/html/rfc2616#section-5.1.1) in order to perform various tasks. All HTTP servers use the GET and HEAD methods, but not all support the rest of these request methods:

* GET requests a specific resource in its entirety
* HEAD requests a specific resource without the body content
* POST adds content, messages, or data to a new page under an existing web resource
* PUT directly modifies an existing web resource or creates a new URI if need be
* DELETE gets rid of a specified resource
* TRACE shows users any changes or additions made to a web resource
* OPTIONS shows users which HTTP methods are available for a specific URL
* CONNECT converts the request connection to a transparent TCP/IP tunnel
* PATCH partially modifies a web resource



**HTTP-get**

**Source Code:**

**HttpServer.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class HttpServer

{

private static final String USER\_AGENT = "Google Chrome";

static String sendGET(String GET\_URL) throws IOException

{

URL obj = new URL(GET\_URL);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", USER\_AGENT);

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

if (responseCode == HttpURLConnection.HTTP\_OK)

{

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null)

{

response.append(inputLine);

}

in.close();

System.out.println(response.toString());

return(response.toString());

}

else

{

System.out.println("GET request not worked");

return (null);

}}

public static void main(String a[]) throws Exception

{

ServerSocket ss=new ServerSocket(6789);

try{

while(true){

Socket consoc= ss.accept();

BufferedReader ifc =new BufferedReader(new InputStreamReader(consoc.getInputStream()));

DataOutputStream otc =new DataOutputStream(consoc.getOutputStream());

String cs=ifc.readLine()+'\n';

System.out.println("RECEIVED : "+cs);

String GET\_URL = cs;

otc.writeBytes(sendGET(GET\_URL)+'\n');

System.out.println("GET DONE");

}}

catch(Exception e){

e.printStackTrace();

}}}

**HttpClient.java**

import java.io.\*;

import java.net.\*;

class HttpClient

{

public static void main(String a[]) throws Exception {

try {

BufferedReader ifu =new BufferedReader(new InputStreamReader(System.in));

Socket clientSocket=new Socket("localhost",6789);

DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());

BufferedReader ifs = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

System.out.println("\nGET url : ");

String sentence =ifu.readLine();

ots.writeBytes(sentence+'\n');

String ms=ifs.readLine();

ms=ifs.readLine();

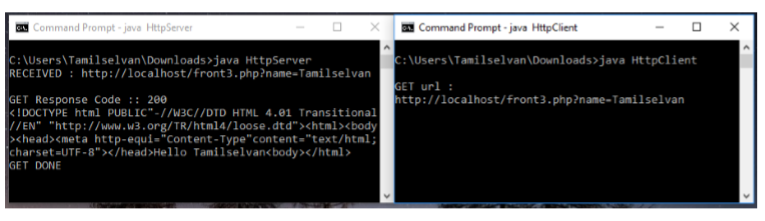
clientSocket.close();

}

catch(Exception e) {

e.printStackTrace();

}}}



**HTTP-post**

**Source Code:**

**HttpServer.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class HttpServer {

private static final String USER\_AGENT = "Google Chrome";

static String sendPOST(String POST\_URL) throws IOException{

URL obj = new URL(POST\_URL);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

con.setRequestMethod("POST");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setDoOutput(true);

OutputStream os = con.getOutputStream();

int responseCode = con.getResponseCode();

System.out.println("POST Response Code :: " + responseCode);

if (responseCode == HttpURLConnection.HTTP\_OK) {

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

System.out.println(response.toString());

return(response.toString());

}

else

{

System.out.println("POST request not worked");

return(null);

}}

public static void main(String a[]) throws Exception {

ServerSocket ss=new ServerSocket(6789);

Try {

while(true) {

Socket consoc= ss.accept();

BufferedReader ifc =new BufferedReader(new InputStreamReader(consoc.getInputStream()));

DataOutputStream otc =new DataOutputStream(consoc.getOutputStream());

String ps=ifc.readLine()+'\n';

System.out.println("RECEIVED : "+ps);

String POST\_URL = ps;

otc.writeBytes(sendPOST(POST\_URL)+'\n');

System.out.println("POST DONE");

}}

catch(Exception e){

e.printStackTrace();

}}

}

**HttpClient.java**

import java.io.\*;

import java.net.\*;

class HttpClient {

public static void main(String a[]) throws Exception {

Try {

BufferedReader ifu =new BufferedReader(new InputStreamReader(System.in));

Socket clientSocket=new Socket("localhost",6789);

DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());

BufferedReader ifs = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

System.out.println("\nPOST url : ");

String sentence =ifu.readLine();

ots.writeBytes(sentence+'\n');

String ms=ifs.readLine();

clientSocket.close();

}

catch(Exception e) {

e.printStackTrace();

}}}

