ASSIGNMENT

What is a server?

A server is a computer that provides data to other computers in a network.

Physical server

A physical server is piece of equipment on which data is stored and read. This may be located onsite in your server room, or it could be stored at a data center with a trusted vendor. Physical servers are generally owned, managed, and maintained by the company's staff.

Virtual server

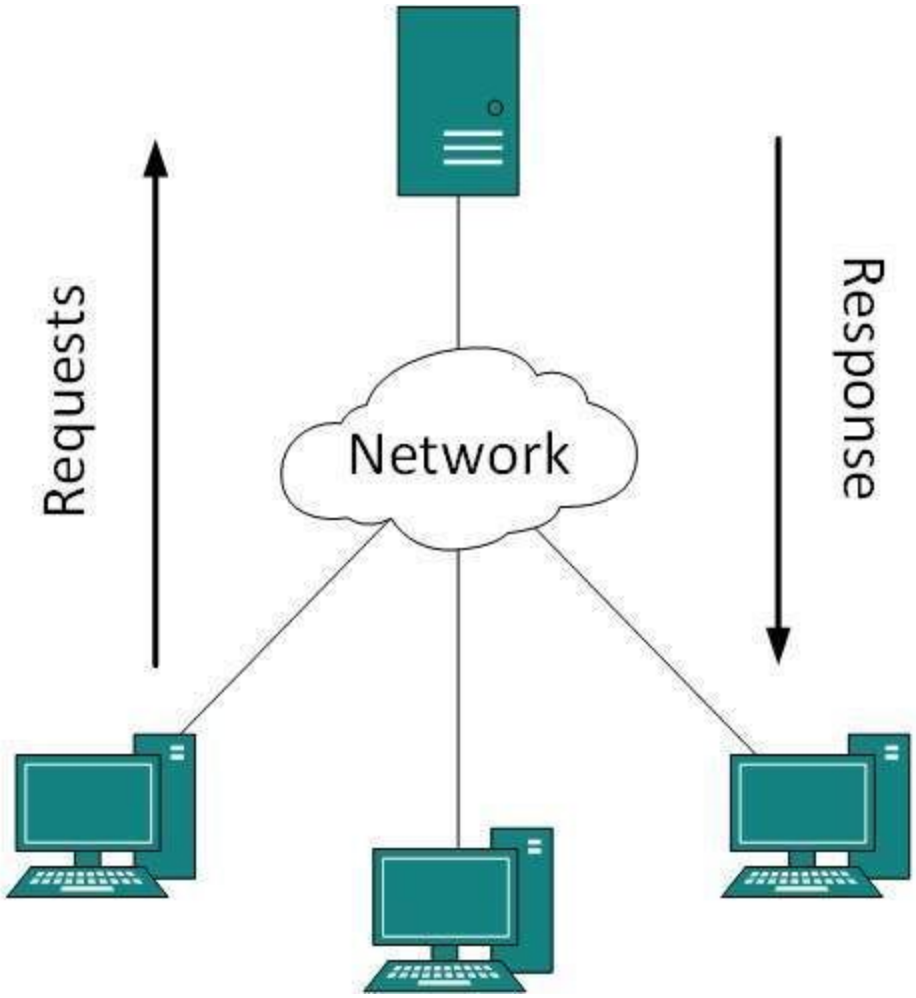
On the Internet, a virtual server is a server at someone else's location that is shared by multiple Web site owners so that each owner can use and administer it as though they had complete control of the server.

What is a client?

A client is a piece of computer hardware or software that accesses a service made available by a server.

Client-server model

The client-server model describes how a server provides resources and services to one or more clients.



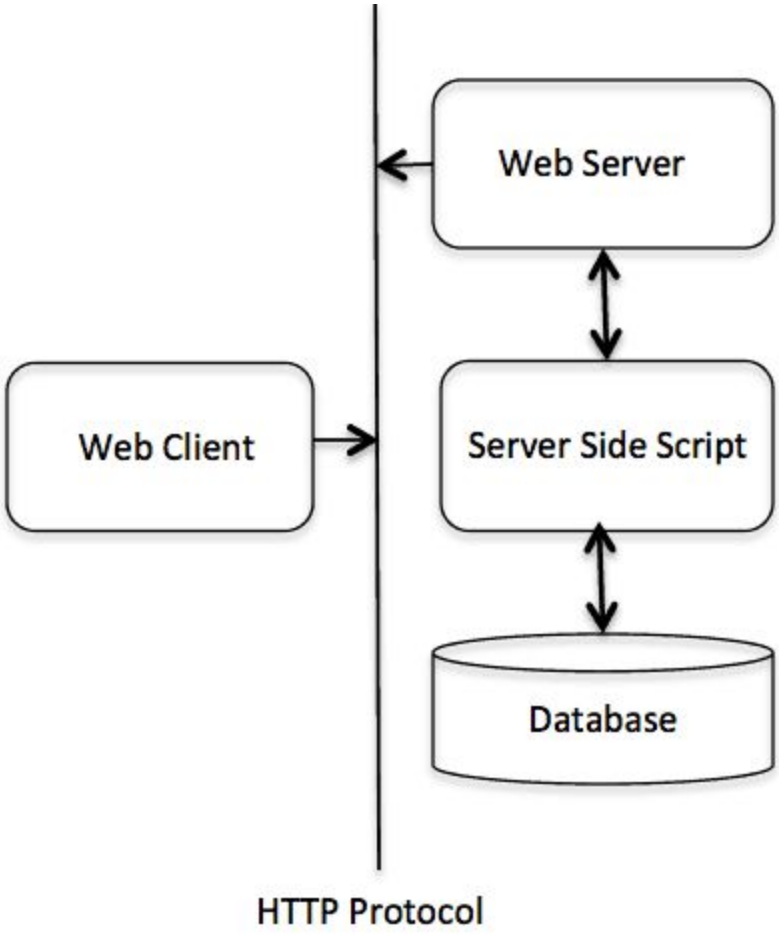
Protocols

A specific set of communication rules is called protocol.

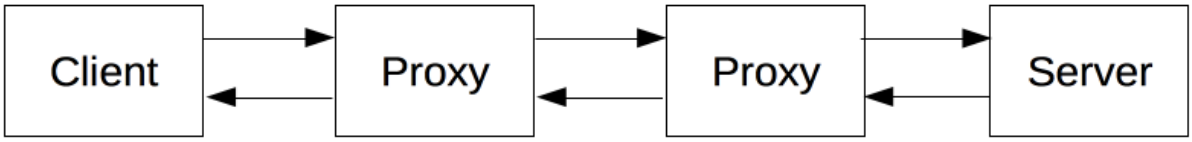
HTTP(Hypertext Transfer Protocol)

HTTP is a set of rules for transferring files(text,graphic images,sound,video, and other multimedia files) on the World Wide Web.

The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.



Each individual request is sent to a server, which will handle it and provide an answer, called the response. Between this request and response there are numerous entities, collectively designated as proxies, which perform different operations and act as gateways or caches, for example.



In reality, there are more computers between a browser and the server handling the request: there are routers, modems, and more. Thanks to the layered design of the Web, these are hidden in the network and transport layers. HTTP is on top at the

application layer. Although important to diagnose network problems, the underlying layers are mostly irrelevant to the description of HTTP.

HTTP methods

1.**GET**

The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only **retrieve data and should have no other effect** on the data.

2**.HEAD**

Same as GET, but transfers the **status line and header section only.**

3.**POST**

A POST request is used to **send data to the server**, for example, customer information, file upload, etc. using HTML forms.

4.**PUT**

Replaces all current representations of the target resource with the uploaded content.

5.**DELETE**

Removes all current representations of the target resource given by a URI.

6**.CONNECT**

Establishes a tunnel to the server identified by a given URI.

7.**OPTIONS**

Describes the communication options for the target resource.

8.**TRACE**

Performs a message loop-back test along the path to the target resource.

REQUEST/RESPONSE MODEL

The interaction between client and server takes place through request/response.Basically a client requests the server for some resource and the server responds to it either by providing the resource or error messages if any. A simple example would be searching for something on a web browser. Here web

browser is the client, as soon as you search something the server responds to the request made by the browser.

HTTP HEADERS

The name-value pairs along with the requests and responses are the headers. HTTP headers are the code that transfers data between a Web server and a browser. HTTP headers are mainly intended for the communication between the server and client in both directions.

There are four types of HTTP message headers,

1. General header-These header fields have general applicability for both request and response messages.
2. Client Request header-These header fields have applicability only for request messages.
3. Server Response header-These header fields have applicability only for response messages
4. Entity-header: These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

TCP(Transmission Control Protocol)

This protocol is used to send the data packets over the network. It is used to create a connection between remote computers by transporting and ensuring the delivery of messages over supporting networks and the Internet.TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination. TCP protocol is classified as stateless, which means each client request is considered new because it is unrelated to previous requests. Being stateless frees up network paths so they can be used continuously.Before transmitting data, TCP creates a connection between the source and destination node and keeps it live until the communication is active. TCP also ensures that the data integrity is intact once it is reassembled at the destination node.TCP is suited for applications that require high reliability, and transmission time is relatively less critical.WEB,telnet,Sending mail,receiving mail uses TCP.

UDP(User Datagram Protocol)

UDP enables process to process communication.UDP is used to send short messages called datagrams but overall, it is an unreliable, connectionless protocol. The protocol assumes that error-checking and correction is not required, thus avoiding processing at the network interface level.
UDP is widely used in video conferencing and real-time computer games. The protocol permits individual packets to be dropped and UDP packets to be received in a different order than that in which they were sent, allowing for better performance.The first eight bytes of a datagram contain header information, while the remaining bytes contain message data.

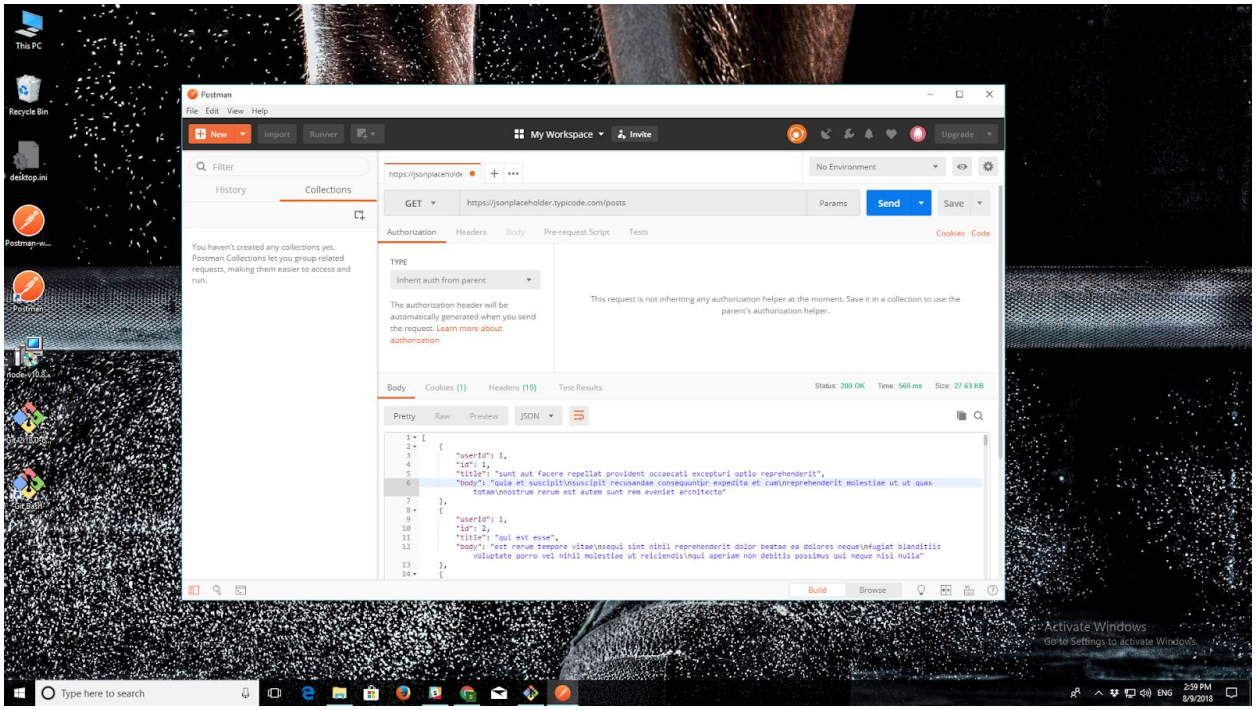
POSTMAN

Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests.
JSONPlaceholder is a free online REST service that you can use whenever you need some fake data. These fake posts are used to test in Postman.

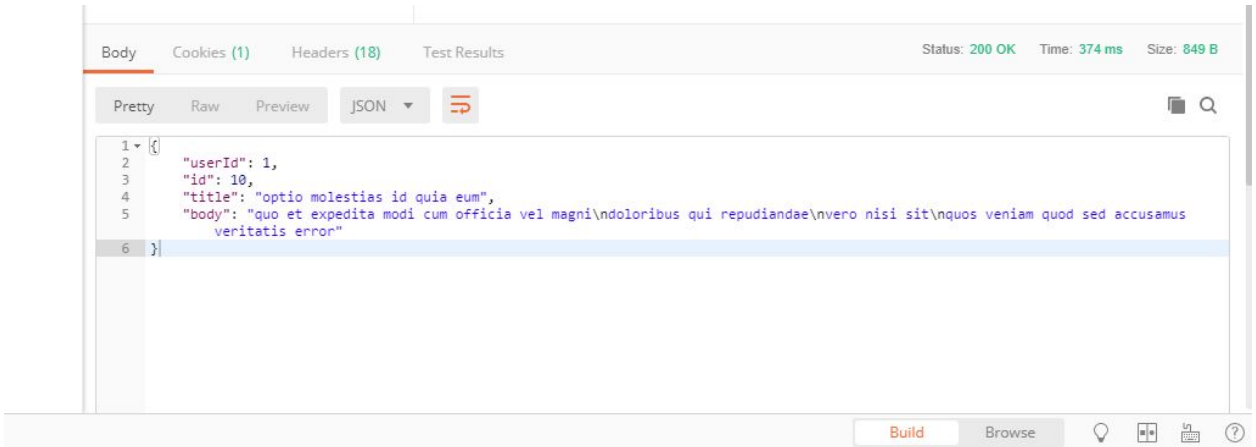
GET

A GET request is used to get the information from the server and does not have any side-effects on the server. Side-effects means there is no updation/deletion/addition of data on the server when you are making a GET request, you just request from the server and the server responds to the request.

A GET request has all its information inside the URL, and since URL is visible all the time, it is advisable not to use GET request while you send some sensitive information such as passwords. For example, when you press search after writing anything in the search box of google.com, you actually go for a GET request because there is no sensitive information and you are just requesting the page with search results , you notice the same search string in URL.

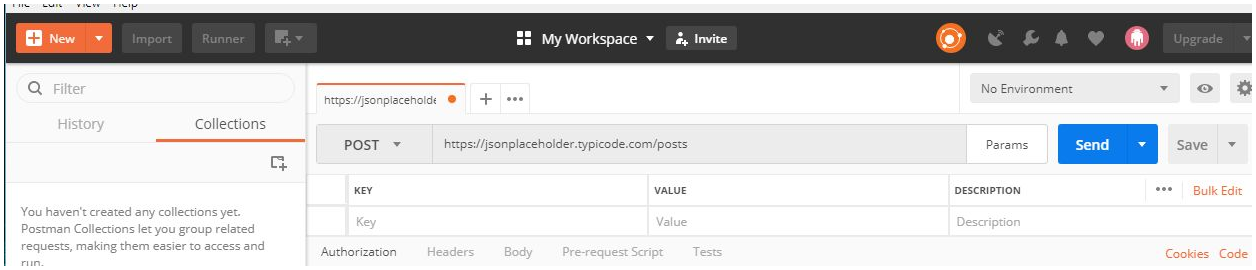


For the above command the output will be,

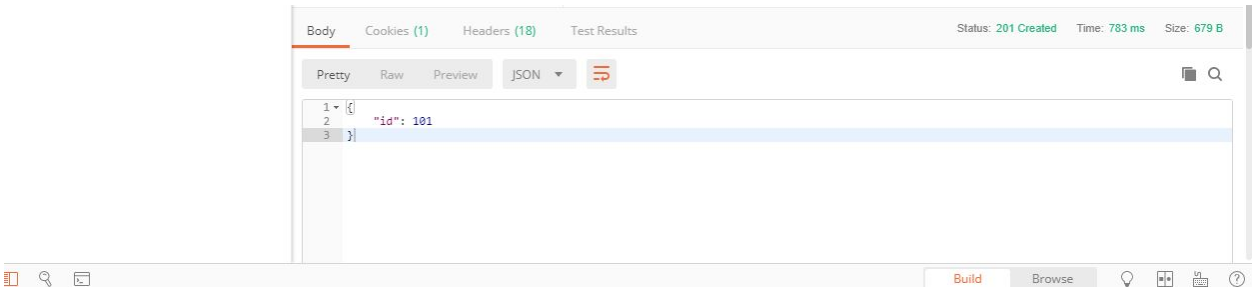


POST

POST request is a method which is used when we need to send some additional information inside the body of the request to the server. When we send a POST request we generally intend to have some modification at the server such as updation, deletion or addition.

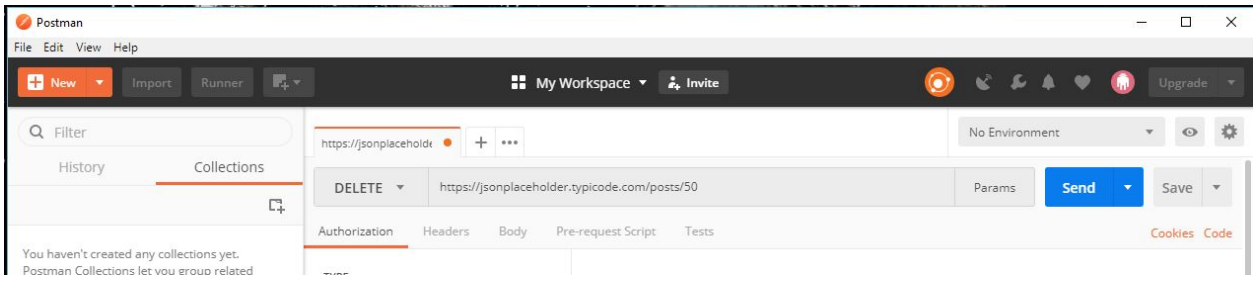


The above request gives following response.

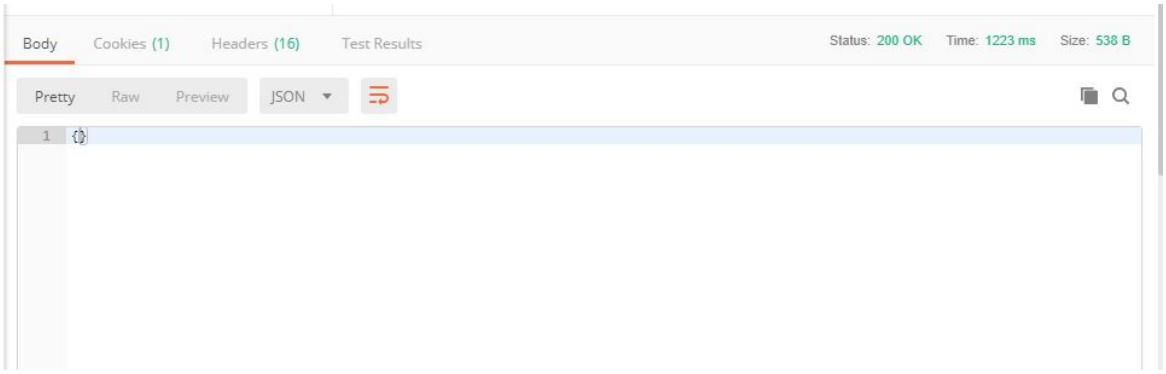


DELETE

DELETE request is a method used when we need to eliminate some information inside the body of the request to the server.



The above request gives following response.



CURL

Curl is used in command lines or scripts to transfer data.

In windows we use GIT BASH to run curl commands.

1.$ curl https://jsonplaceholder.typicode.com/posts
This request gives all the 100 posts in this case.

2.$ curl https://jsonplaceholder.typicode.com/posts/10
This request gives us the post with id=10 as shown below,
{

"userId": 1,

"id": 10,

"title": "optio molestias id quia eum",

"body": "quo et expedita modi cum officia vel magni\ndoloribus qui repudiandae \nvero nisi sit\nquos veniam quod sed accusamus veritatis error"

}

3.$ curl --head https://jsonplaceholder.typicode.com/posts/10
Or
$ curl -I https://jsonplaceholder.typicode.com/posts/10

This request gives us the headers as shown below,

content-type: application/json; charset=utf-8

content-length: 217

set-cookie: __cfduid=df49286035b5322258df2f5e020ea6f271533804286; expires=Fri, 0
9-Aug-19 08:44:46 GMT; path=/; domain=.typicode.com; HttpOnly

x-powered-by: Express

vary: Origin, Accept-Encoding

access-control-allow-credentials: true

cache-control: public, max-age=14400

pragma: no-cache

expires: Thu, 09 Aug 2018 12:44:46 GMT

x-content-type-options: nosniff

etag: W/"d9-iWGkhNuCVU3QUpWXcSkr72qX524"

via: 1.1 vegur

cf-cache-status: HIT

expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"

server: cloudflare

cf-ray: 4478f158bee99463-NRT

4.$ curl -X PUT -d "title=nick" https://jsonplaceholder.typicode.com/posts/10

This request changes the title of the post id=10 as title as shown below,

{

"title": "nick",

"id": 10

}

5.$ curl -X DELETE https://jsonplaceholder.typicode.com/posts/10
This request deletes the post with id=10 as shown below,
{}

DICTIONARIES

Dictionaries are implemented as resizable hash tables. Compared to B-trees, this gives better performance for lookup (the most common operation by far) under most circumstances, and the implementation is simpler.

Dictionaries work by computing a hash code for each key stored in the dictionary using the hash() built-in function. The hash code varies widely depending on the key; for example, "Python" hashes to -539294296 while "python", a string that differs by a single bit, hashes to 1142331976. The hash code is then used to calculate a location in an internal array where the value will be stored. Assuming that you're storing keys that all have different hash values, this means that dictionaries take constant time – O(1), in computer science notation – to retrieve a key. It also means that no sorted order of the keys is maintained, and traversing the array as the .keys() and .items() do will output the dictionary's content in some arbitrary jumbled order.

Every time a key is generated. The key is passed to a hash function. Every hash function has two parts a Hash code and a Compressor.

Hash code is an Integer number (random or nonrandom). In Java every Object has its own hash code. We will use the hash code generated by JVM in our hash function and to compress the hash code we modulo(%) the hash code by size of the hash table. So modulo operator is compressor in our implementation.

The entire process ensures that for any key, we get an integer position within the size of the Hash Table to insert the corresponding value.

So the process is simple, user gives a (key, value) pair set as input and based on the value generated by hash function an index is generated to where the value corresponding to the particular key is stored. So whenever we need to fetch a value corresponding to a key that is just O(1).