Http Header

- Http is request/response based protocol.
- Headers are nothing but metadata that means data about data.

Why http header is needed?

❖ Client tell the server Hey, look I am Firefox, I am install on the windows machine, i support these fonts, when you are responding to me tell me or talk to me in English.
❖ It is very important to communicate between client and server.

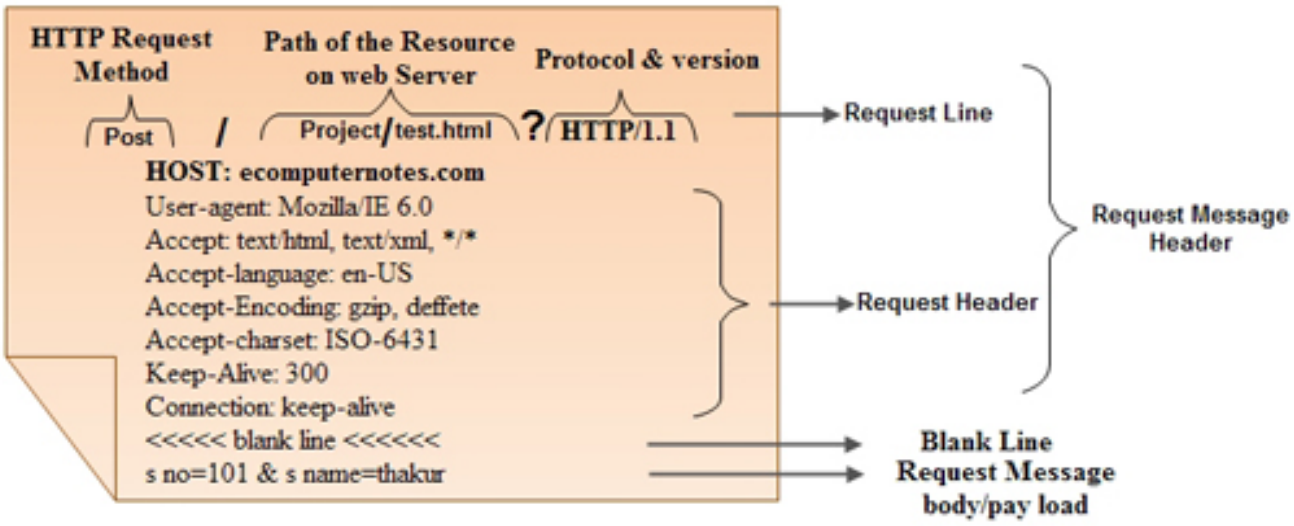
How Http header works?

1. **HTTP Request:**

   <HTTP-method> <what resource is requested for> <HTTP-version>

   ex: GET /servlet/SomeName HTTP/1.1 (then new line char called carriage)
   Host:.....
   Header2:.....
   ..
   HeaderN:
   (Blank Line) line feed char

   then the body starts if it is applicable.



2. HTTP Response:

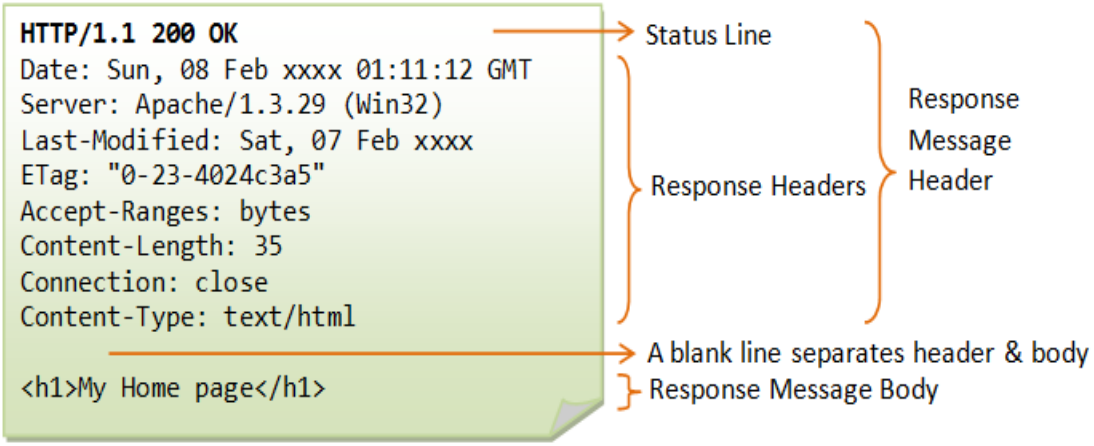
Server checks for the incoming request and the corresponding response is sent back.

<HTTP-Version> <HTTP status-code> <Reason-Phrase>

<Header section>
ex: HTTP/1.1 200 OK (okay, I am also communicating on HTTP/1.1 protocol and the response code is 200 that means your request is successful)
Content-Type: Text/HTML (The content type i am sending in text/html format)
Header2:...
..
HeaderN:
(blank line)
</Header section>

<Body section>

</Body section>



Tools to look into the HTTP headers?

- **Browser plugin**
  1. Live HTTP headers
  2. HTTP FOX

- **Browser developer tool**
- **Proxy tools**
  1. Fiddler
  2. Charles
  3. Burp
  4. Wireshark

Request Headers

Are basically name value pair which is segregated by ":" in between**.**

- **Accept:** it tells what content that acceptable. when browser sends header, and says okay I accept plain/text or I accept image/jpg file.
- **Accept-charset:** that means what are the different charset that are acceptable. ex: Accept-charset: UTF-8
- **Accept-encoding:** it tells the list of encoding that are supported by the browser. Client says that I support gzip, D-flate encoding. so that the client's responsibility to send it to the server. ex: Accept-encoding: gzip
- **Accept-language:** what all human languages in which server should respond with. ex: Accept-language: en-us
- **Authorization:** credentials when the server asks for HTTP based authentication. ex: Authorization: Basic<hash> that hash is just a base 64 encoding format of username/password
- **Cache-Control:** (HTTP/1.1) It's a request header as well as response header. Since the request, response goes through the intermediate proxies and CDN servers. Both way should tell by pass the cache or serve from the cache.
- **Connection:** It is also a request header and response header. way of telling the partner whether to continue with this TCP connection or to close it. Client may say connection: Keep-Alive, that means whenever the transaction is complete, don't terminate the connection. use the same TCP connection for further communication. Since HTTP/1.1 forward by default the connection is persistent. so if you are not sending connection: Keep-Alive then also the connection will be alive for the fixed amount of time that is defined by connection timeout.
- **Cookie:** these are just the flat files which stores the session information. Types: persistent, non-persistent. So if the client is connecting to the server next time it shows the cookies sent by server in previous communication and says don't ask for the username/password again. Server validate the cookie value and sends the content back if it's applicable.
- **Content-length:** which tells server whatever is the content sent what is the length of it. It's bytes.
- **Content-Type:** It tells mime-type or body of this header. Even it's it PUT request, Post request you have to tell server what is the mime-type I am understanding of the object.
- **Date:** It is the date and time that the message was originated.
- **Host:** The measure difference between HTTP/1.0 and HTTP/1.1 is introduction of host header. It's a name of the domain or the server or the virtual hosting to which the client is communicated to. If there are multiple sites hosted on the same server this is the way of telling the server from client even though you have single IP address and you host multiple website I need to see the content of may be www. abc.com. Server receive this header match this request to the configured Document Root.

- **If-modified-Since:** Okay I have this object with me, serve me if this object was last modified since this time. Server gets this request and it checks the last modified value of the object, if it matches it serve 304 response code not modified. Server says, dude you have the object even I have the same obj. No need to download this object again. In a scenario, the object is different 200 OK response is served and the object along with that.
- **If-unmodified-Since:** The If-Modified-Since request-header field is used with a method to make it conditional: if the requested variant has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body. for ex: If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

{The both above will only work when the server has the last modified value of the object}

- **If-match:** 304 response code if it matches. If not, 200 OK response codes with the value of the object
- **If-none-match:** The If-None-Match request-header field is used with a method to make it conditional. A client that has one or more entities previously obtained from the resource can verify that none of those entities is current by including a list of their associated entity tags in the If-None-Match header field. The purpose of this feature is to allow efficient updates of cached information with a minimum amount of transaction overhead. It is also used to prevent a method (e.g. PUT) from inadvertently modifying an existing resource when the client believes that the resource does not exist. As a special case, the value "*" matches any current entity of the resource. For ex: If-None-Match = "If-None-Match" ":" (“*” | 1#entity-tag)
- **Pragma:** (HTTP/1.0 cache-control) This is typically being used by proxy to debugging purposes. ex: pragma: no-cache it tells proxy don't cache its content.
- **Range:** Http/1.1 onwards server started supporting byte range. Client tell i don't want the entire object to serve me the byte from 1 Mb to 2 Mb.
- **User-Agent:** tell the browser client is using and version of the browser and most important is the which OS is being used by client.
- **Via:** I am coming through different proxies. and proxy typically add this header in between. Server gets this header and sees the value in it. it might content server name in it.
- **X-Forwarded For:** this is typically added by proxies in between. RFC says whenever there is a request passes through proxy server, proxy has to add X-Forwarded For header and add the client's IP address. So even if you see that the request is coming from this proxy but along with that you will also see that this request is forwarded for this client IP.

Response Headers

- **Access-Control-Allow-Origin (A-C-A-O):** this header tells which website can participate in cross origin resource sharing. It's a browser security machanism that by default you are not allowed to load content from different website. What all content of which website could be accepted. ex: Access-Control-Allow-Origin:*

- **Accept-Ranges:** what kind of range i accept, do i accept bytes, so it will tell Accept-Ranges: bytes
- **Age:** It tells that the age of the object in proxy servers in <seconds>. How long the object was cached in the intermediate proxies. If it's 0 that means it's not cache.
- **Allow:** it tells what are the different http methods are allowed ex: Allow: GET, HEAD other than that if you are using any header then server will respond 405 means the method that you are trying is not allowed.
- **Cache-Control:** Server or client is allowed to cache the object, ex: Cache-Control: max-age=3600 means this object is save to cache for 3600 sec.
- **Connection:** do you want to keep this connection open or close. ex: Connection: close
- **Content-Disposition:** It tell the browser don't load this content but download it instead. It an opportunity todays to download the file in a known file type in the browser.
- **Content-Encoding:** The Content-Encoding entity-header field is used as a modifier to the media-type. When present, its value indicates what additional content coding have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a document to be compressed without losing the identity of its underlying media type. for ex: Content-Encoding: gzip
- **Content-Language:** The Content-Language entity-header field describes the natural language(s) of the intended audience for the enclosed entity. Note that this might not be equivalent to all the languages used within the entity-body.
- **Content-Length:** what is the length of the response body. in request, you will see content-length during POST, because it's a post body that has some length. what is the size of the response could be GET or POST doesn't matter.
- **Content-Type:** It tells mime-type or body of this header. Even it's it PUT request, Post request you have to tell server what is the mime-type I am understanding of the object.

  {MIME: multi-purpose mail extension, media type}

- **Date:**
- **Etag:** The ETag response-header field provides the current value of the entity tag for the requested variant. The entity tag MAY be used for comparison with other entities from the same resource. For ex: ETag: "xyzzy"
- **Expires:** it tells the proxy servers or browser when this response is going to expire. after that don't use it because it is expired.
- **Last-Modified:** work similar to Etag, it tells what was the last modified time of the object on the server side. using this value when the request comes as if-modified or if-unmodified since. tells client whatever the request good to go.
- **Location:** in the case of redirection, location value contains where the redirect will happen, suppose if a client access www.abc.com this request reaches to the server and server says this is not machine where the resource present and redirects this request to the actual resource and add the location header with the value of URL of the actual resource server.
- **retry-After:** It the entity is not available on my side please don't try until n no of sec. ex: Retry-After:100, retry after 100 sec.

- **Server:** Similar to user-agent header. It is the identity of a server. What server is used for delivering the content. it contains value like apache, IIA, nginx.
- **Set-Cookie:** tells the client okay I got this session or information, you should store in the local cache/local drive in a plain text format. it can also contain the value when it's going to expire. for what part, it's valid for and what domain it's valid for. It might be restricted only a subdomain or it might be available for entire necked domain.
- **Transfer-Encoding:** the form of encoding used to safely transfer the data to the user. It may be the chunk response, might not be the complete response to the end user. ex: Transfer-Encoding: Chunk
- **Vary:** It tells the downstream proxy how to match future request header to display in cache we can use rather than requesting server in a fresh manner.
- **Via:** 1.0 apache
- **WWW-Authenticate:** indicate authentication scheme that should be used to access the requested entity. ex: WWW-Authenticate: Basic
- **Strict-Transport-Security**:

Reference

https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
https://www.youtube.com/watch?v=vAuZwirKjWs
https://www.youtube.com/watch?v=kSfxRk0XCXs&list=PLmCsXDGbJHdjxhhl8eWDNCJcGxazln4Zj