HTTP

Hypertext Transfer Protocol (HTTP)

-application layer communications protocol used to access resources (hypertext/hypermedia) on the World Wide Web

-invented by Sir Tim Berners-Lee

-jointly developed by the W3C and the IETF

Version History:

1. HTTP 0.9 (1991)
2. HTTP 1.0 (RFC 1945, May 1966)
3. HTTP 1.1 (RFC 2068 Jan 1997, RFC 2616, June 1999), RFC 7230-7235 (June 2014)
4. HTTP 2 (RFC 7540 May 2015) – patterned after SPDY (made by Google)

Socket – combination of IP Address and port #

HTTP Fundamendals

1. HTTP runs on top of “TCP/IP”, using TCP port 80 by default, or TCP port 443 for HTTPS (HTTP over “SSL/TLS”
2. HTTP is based on a “client-server” architecture
3. Clients a.k.a “user agents (UA)”:

a.1) Web browsers, web crawlers/spiders, other end user tools and apps

https- provides for encrypted communication

encryption – transformation of such data

1. servers

a.1) origin servers

a.2) proxy, servers, gateways, tunnels

1. HTTP uses a “request response” standard protocol
2. The client sends an “HTTP request message” to the server
3. The server processes the request and replies with an “HTTP response”
4. HTTP is a “stateless” communications protocol
5. Servers do not keep information about clients in-between requests
6. HTTP provides support for other functionalities such as:
7. Cache control
8. Content media type “MIME(Multipurpose Internet Mail Extensions)” specification
9. Language and character set specification
10. Content/transfer codings
11. Content negotiation
12. Client-server protocol negotiations
13. Persistent connections – for a few seconds open
14. Request pipelining
15. Authentication/authorization

HTTP Resource Addressing

HTTP Resources are indentified using URIs (RFC 3986), or more specifically, HTTP URLs

* Scheme (http. or https)

URN – Uniform Resource Name

URL – Uniform Resource Locator

* Authority

1. User information or authentication credentials (deprecated)
2. Host

b.1) domain name (resolved to an IP Address using DNS) of the server where the resource resides/ or will be created)

c) port number

* “path” to resource (resolved relative to the “document root” on the server)

1. May refer to a static or dynamic resource

* Query

1. Typically provided as “key=value” pairs, with ampersand (“&”) separators between key/value pairs, may be URL-encoded
2. May be URL-encoded
3. “fragment identifier”

**http://usr:pwd@server.org:81/info/profile.php?id=1234#addr**

**sheme authority path query**

Request Line (CRLF-terminated line consisting of the 3)

* Method
* Request URI
* HTTP Protocol Version

“Message Headers” (general, request, and/or entity headers)

* HTTP requires at least the “Hard” request header to be provided

Empty Circle (CRLF)

Message body/”payload”(optional)

HTTP Response

1. Status Line

* Information (1XX)
* Success (2XX)
* Redirection (3XX)
* Client Error (4XX)
* Server Error (5XX)

HTTP Protocol Version

-Status Code

-Reason Phrase

* Message Headers (general, response, and/or entity headers)
* Empty Line (CRLF)
* Message body (optional)

HTTP Request Methods

Standard Methods

**GET HEAD POST PUT DELETE OPTIONS TRACE**

1. GET – transfer a current selected of the resource identified by the Request URL the retrieve resource
2. HEAD – used to retrieve metadata about the entity required by the request without transferring the entity itself
3. POST – typically used for submitting HTML forms
4. PUT – store the enclosed entity in the message body under the specified request URI
5. DELETE – remove the associated
6. OPTIONS – asking the server “what can I do?” and request information about communication
7. TRACE – request a “loop-back” of the request message (i.e. request the server to “echo” back to the client the received request message), typically used for testing/diagnostics of the request/response chain
8. CONNECT – request the establishment of a “tunnel”

Safe methods – do not modify on the server when you use them

Idempotent methods – used it many times with the same results

Cacheable methods – can be stored in the caches that are currently existing

HTTP Message Headers

1. General Header Fields – can be used by clients and servers
2. Request Header Fields
3. Response Header Fields

* Accept-range
* Age
* ETag
* Location
* Proxy-authenticate
* Retry-after
* Server
* Vary
* www-authenticate

1. Entity Header Fields

* Allow
* Content-encoding
* Content-language
* Content-location
* Content-length
* Content-MD5 (Message Digest Version 5)
* Content-range
* Content-type
* Expires
* Last-modified

Status Code

1. Informational (1XX)

* 100 Continue
* 101 Switching Protocols

1. Success (2XX)

* 200 OK
* 201 Created
* 202 Accepted
* 203 Non-Authoritative Information
* 204 No Content
* 205 Reset Content
* 206 Partial Content

1. Redirection (3XX)

* 300 Multiple Choices
* 301 Moved Permanently
* 302 Found
* 303 See Other
* 304 Not Modified
* 305 Use Proxy
* 306 (unused)
* 307 Temporary Redirect

1. Client Error (4XX)

* 400 Bad Request
* 401 Unauthorized
* 402 Payment Required
* 403 Forbidden
* 404 Not Found
* 405 Method Not Allowed
* 406 Not Acceptable
* 407 Proxy Authentication Required
* 408 Request Time-out
* 409 Conflict
* 410 Gone
* 411 Length Required
* 412 Precondition Failed
* 413 Request Entity Too Large
* 414 Request-URI Too Large
* 415 Unsupported Media Type
* 416 Requested Range Not Satisfiable
* 417 Expectation Failed
* 426 Upgrade Required

1. Server Error (5XX)

* 500 Internal Server Error
* 501 Not Implemented
* 502 Bad Gateway
* 503 Service Unavailable
* 504 Gateway Time-out
* 505 HTTP Version Not Supported