HTTP

\* Application layer communications protocol used to access resources (hypertext/hypermedia) on the World Wide Web

\* Invented by Tim Berners-Lee

\* Jointly developed by the W3C and the IETF

\* Version history

• HTTP 0.9 (1991) - GET

• HTTP 1.0 (RFC 1945, May 1996)

HTTP 1.1 (RFC 2068 January 1997), (RFC 2616 June 1999), (RFC 7230-7235 June 2014)

• HTTP 2 (RFC 7540 May 2015)

• SPDY – Protocol designed by Google

HTTP Fundamentals

\* HTTP runs on top of TCP/IP, using TCP port 80 by default, or TCP port 443 for HTTPS (HTTP over SSL/TLS)

\* HTTP is based on a client-server, architecture

• clients, aka user agents (UA):

\* Web browsers, web crawlers/spiders, other end user tools and applications

o NOTE: port number – unique number assigned by operating system (0-64K)

o IP address + port number = socket

• servers:

> Origin servers

> Proxy servers, gateways, tunnels

\* HTTP uses a request-response standard protocol

• the client sends an HTTP request message to the server

• the server processes the request and replies with an HTTP response message

\* HTTP is a stateless communications protocol

• servers do not keep information about clients in-between requests

\* HTTP provides support for other functionalities, such as:

• cache control

• content media type (MIME-Multipurpose Internet Mail Extensions) specification

• e.g. text/html (.xls), img/jpg (.jpg)

• language and character set specification

• content/transfer coding

• content negotiation

• client-server protocol negotiations

\* Locality of Reference Protocol

• persistent connections

• request pipelining

• authentication/authorization

• and others

HTTP Resource Addressing

\* HTTP Resources are identified using URIs (RFC 3986) or more specifically, HTTP URLs

URN – naming without regard to where it is (Ex. ISBN)

URL – how to access (mechanism of finding)

• scheme (http or https) - protocol

• authority

-user info or authentication credentials (deprecated)

-host (host name – where the resources)

-port number (default- 80)

o user information or authentication credentials (deprecated)

o host

\* domain name (resolved to an IP address using DNS) of the server where the resource resides (or will be created)

• path to resource (resolved relative to the document root on the server)

o may refer to a static or dynamic resource (static – html itself; dynamic – scripts)

• query (starts with ? ) Ex. Products.php?

o typically provided as key=value pairs, with ampersand (&) separators between key/value pairs

o May be URL encoded

• Fragment Identifier

• # - identify id

• + - space

• & - separation keys

• e.g

products.php?id=123&color=red&size=xL

HTTP Request Message

• Request Line (CRLF terminated line connecting three spaces separated values)

o Method

o Request URI

o HTTP Protocol Versions

• Message Headers (general, request, and/or entity leaders)

o HTTP 1.1 requires at least the Host request header to be provided (1 header required)(Ex. Fieldname: value)

1. General headers fields – can be used by clients and server.

2. Request header fields – used by clients.

3. Response header fields – server

4. Entity header fields

• Empty Line (CRLF)

• Message Body AKA payload (optional)(message body – entity)

HTTP Response Message

• Status Line (CRLF-terminated line consisting of three space-separated values)

o HTTP Protocol Version

o Status Code

- Information (1xx)

- Success (2xx)

- Redirection (3xx)

- Client error (4xx)

- Server error (5xx)

o Reason Phrase (Ex. Success 2xx – 200 OK)

• Message Headers (general, response, and/or entity headers)

• Empty Line (CRLF)

• Message Body (optional) – have body more than request message.

HTTP Request Methods (HTTP verbs)

• Standard Methods

GET

o Transfer a current selected representation of the resource identified by the Request URI; the retrieved resource is returned in the message of the response as an entity

o Most commonly used HTTP method

o Must be supported by all compliant general-purpose servers

HEAD

o Same as GET, except that the entity is not included in the response (ie. Returns only the status line and headers returned by a GET request, without the message body)

o Used to retrieve metadata about the entity implied by the request without transferring the entity itself (eg. To test for link validity or resource modification)

o Like GET, must be supported by all general-purpose servers

POST

o Perform resource-specific processing of the entities enclosed in the message body by the resource identified by the Request URI

o Typically used in submitting HTML form data

PUT

o Store the enclosed entity in the message body under the specified Request URI (ie the resource identified by the Request URI is either created or replaced, using the enclosed entity)

Extension Method

\* Eg : Web DAV (RFC 4918)

\* PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK

OPTIONS

o Request information about the communication option available for the resource

DELETE

o Remove the resource associated with request URI

TRACE

o Request a loop-back of the request message (ie request the server to echo back to the client the received request message)

o Typically used for teaching/diagnostics of the request/response shown

CONNECT

o Request the establishment of a tunnel to the destination origin server, and if successful, thereafter restrict its behaviour to blind-forwarding of packet, in both directions, until the tunnel is closed

o Commonly used to create an end-to-end virtual connection

• Safe Methods, Idempotent Methods, Cacheable Methods

HTTP Message Headers

• General Header Fields

o Cache-Control

o Connection

o Date

o Pragma

o Trailer

o Transfer-Encoding

o Upgrade

o Via

o Warning

• Request Header Fields

o Accept

o Accept-Charset

o Accept-Encoding

o Accept-Language

o Authorization

o Expect

o From

o Host

o If-Match \*

o If-Modified-Since \*

o If-None-Match \*

o If-Range \*

o If-Unmodified-Since

o Max-Forwards

o Proxy-Authorization

o Proxy-Authorization

o Range

o Referrer

o TE

o User-Agent

\* Conditional Headers

• Response Header Fields

o Accept-Ranges

o Age

o ETag (Entity Tag)

o Location

o Proxy-Authenticate

o Retry-After

o Server

o Vary

o WWW-Authenticate

• Entity Header Fields

o Allow

o Content-Encoding

o Content-Language

o Content-Length

o Content-Location

o Content-MD5 (Message Digest)

o Content-Range

o Content-Type

o Expires

o Last-Modified

/ GROUP MATES, put all of this STATUS CODES in a CSS TABLE PLEASE \*/

HTTP Status Codes

• Informational (1xx)

100 Continue

101 Switching Protocols

• Success (2xx)

200 Ok

201 Created

202 Accepted

203 Non-Authoritative Information

204 No Content

205 Reset Content

206 Partial Content

• Redirection (3xx)

300 Multiple Choices

\*301 Moved Permanently

\*302 Found

\*303 See Other

304 Not Modified

305 Use Proxy

306 (unused)

\*307 Temporary Redirect

\* Different version of Redirection

• 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 Request Range Not Satisfiable

417 Expectation Failed

426 Upgrade Required (newly added. 2014 version)

• 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