**HTTP Protocol-Request**

Http messages: How data is exchanged between a server and a client

Request: Sent by the client to trigger an action on the server

Response: The answer from the server

HTTP/1.1 – Messages openly sent across the connection

HTTP/2 – Human readable message is now divided up into HTTP frames, provides optimization and performance improvements.

Binary framing mechanism has been designed to not require any alteration of the APIs or config files applied

***HTTP REQUEST***

Structure:

1.Start line: Status of successful or failure

Contain HTTP method (GET, PUT, POST) or noun like (HEAD, OPTIONS)

Get: URL or absolute path of the protocol, port and domain

Examples:

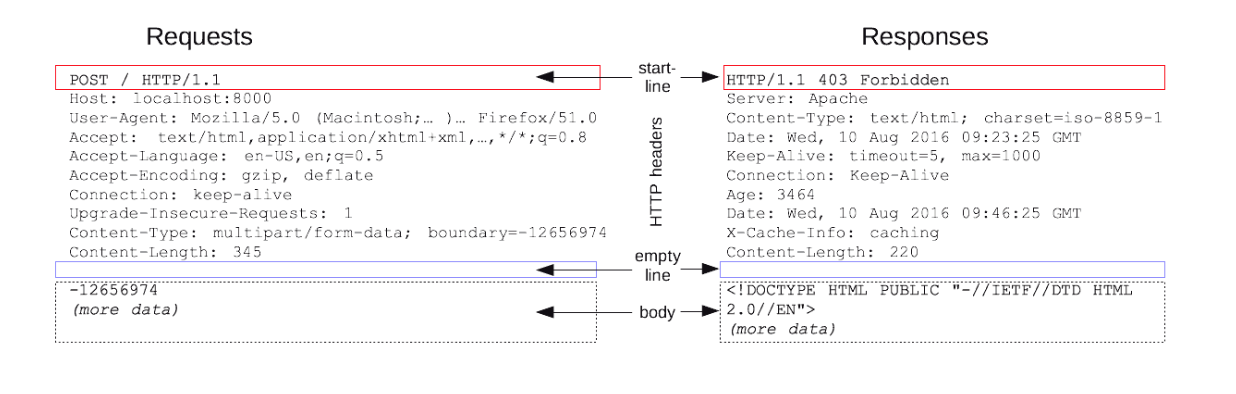
POST/HTTP 1.1

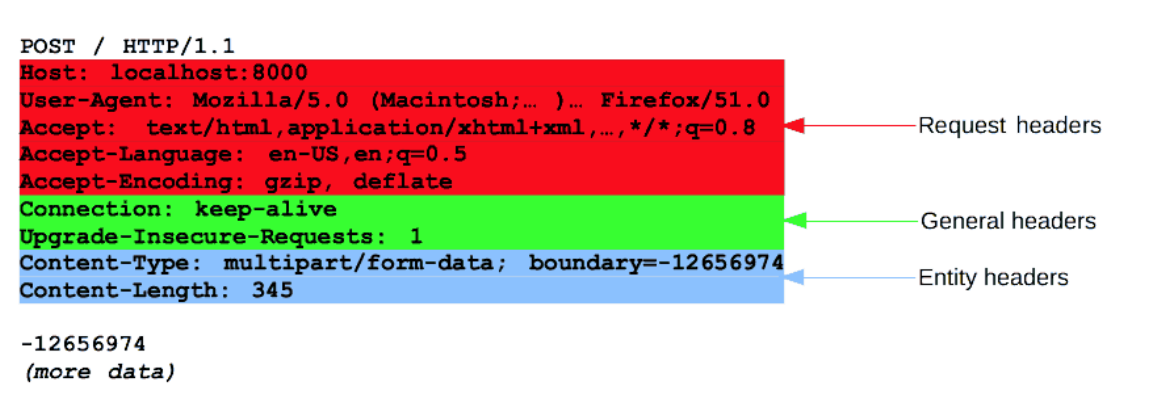
GET /background.png HTTP/1.0

HEAD /test.html?query=alibaba HTTP/1.1

OPTIONS /anypage.html HTTP/1.0

When we add ‘\*’ represents the server as a whole



2. HTTP headers describing the body included in the messag 

3. Optional body containing data associated with the request

Divided into two categories:

Single resource bodies, one single file: *Content-Type* and *Content-Length*

Multiple-resources bodies: Multipart body, associated with HTML forms

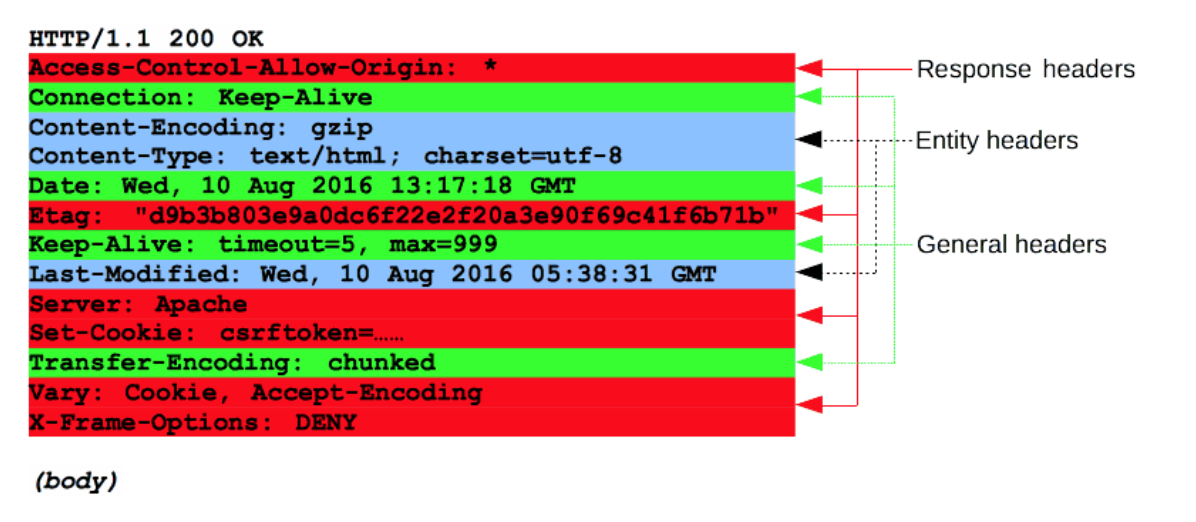
***HTTP response***

Status line:

-Protocol version

-Status code: 200, 404, 302

-Status text: Textual description of the status code to help understand HTTP message



Body: Divided into three categories:

**HTTP/2 frames**

HTTP/1.x disadvantages:

Headers are uncompressed

No multiplexing can be done (several connections need opening on the same server)

Divides HTTP/1.x into frames, embedded in a stream

Data and header frames are separated, this allows header compression

**Multiplexing:** Several streams can be combined together

Request is serialized to bites and sent to the host, the response is an HTML or whatever we asked for

It is important that we learn how to use the part of the TIMING on console, we can know the performance of our server.