2025.09.19

HTTP

network

HTTP

CONTENTS

01

HTTP란?

HTTP란?

HyperText Transfer Protocol

- 웹에서 서버/클라이언트 모델을 따라 데이터를 주고 받기 위한 통신 규약
- 네트워크 통신을 작동하게 하는 기본 기술

HTTP의 특징

클라이언트 서버 구조

클라이언트가 서버에 요청을 보내고,
서버는 요청에 대해 응답을 보내는 방식으로 동작

텍스트 기반 프로토콜

요청/응답 메시지 구조가 사람이 읽을 수 있는
형식으로 디버깅이 비교적 쉬움

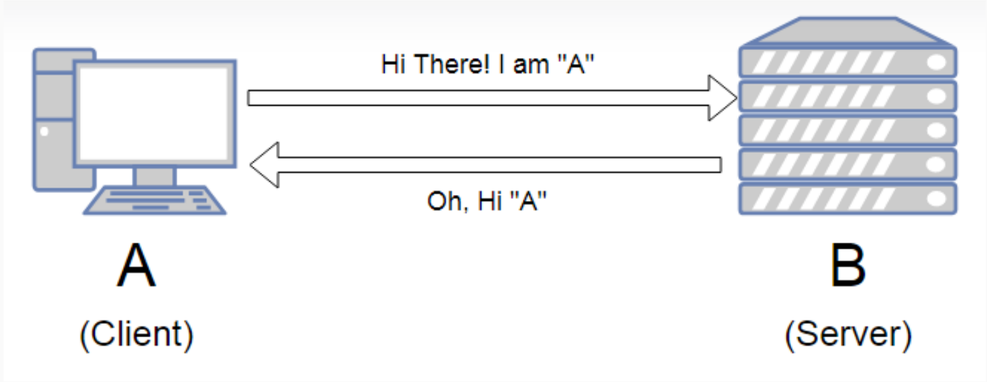
비연결성

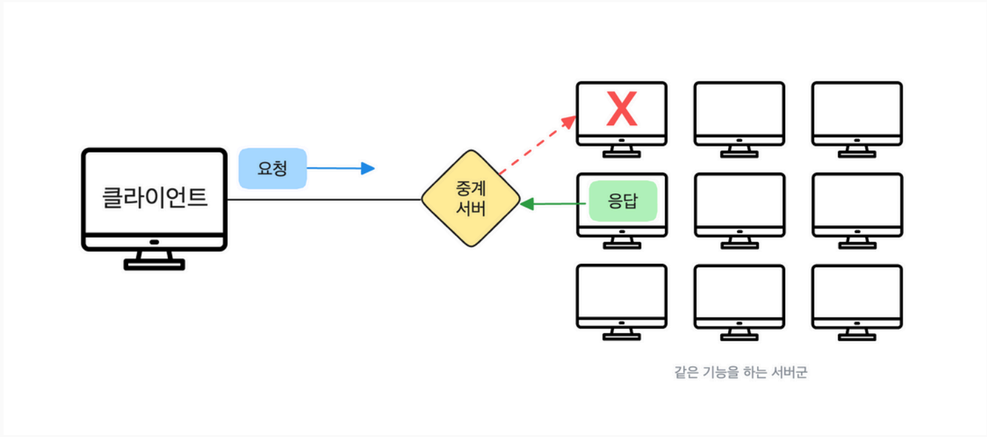
클라이언트와 서버가 연결을 맺고, 클라이언트 요청에
서버가 응답을 마치면 연결을 끊어버리는 성질

무상태성

서버가 이전의 요청에 대한 상태를 보존하지 않는 성질

HTTP의 특징



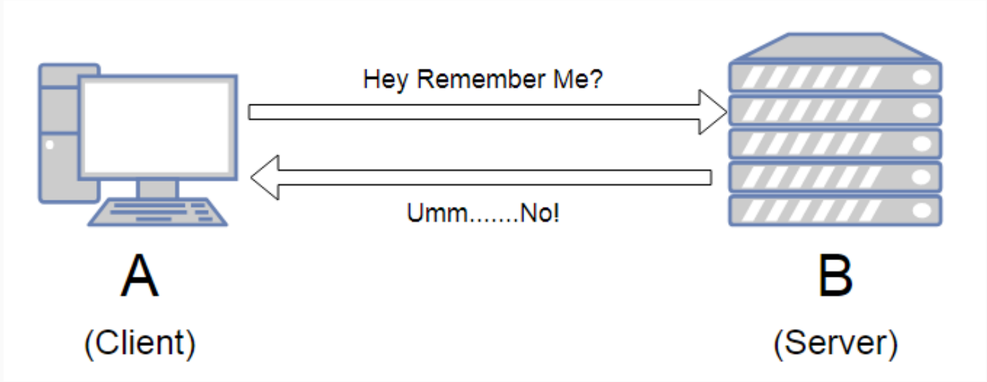


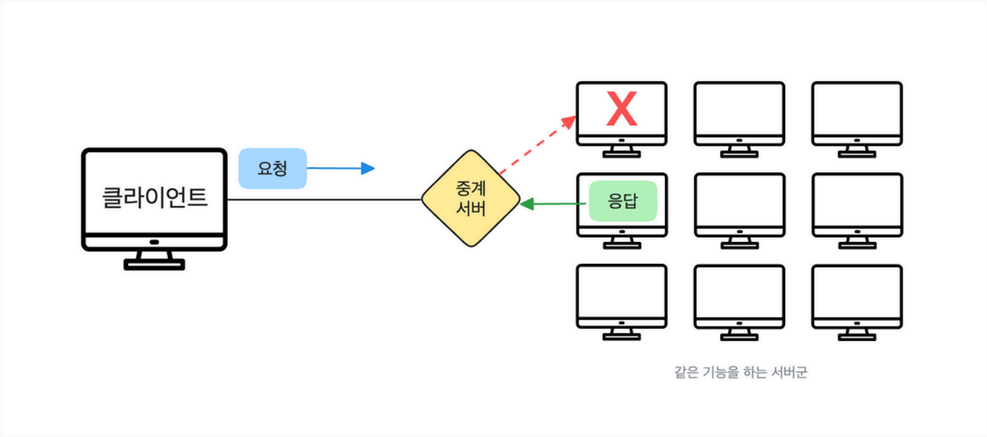
상태 유지 (Stateful)

- 클라이언트가 서버에 요청을 보내면 서버는 클라이언트 상태 정보를 저장함

- 장점
  - 더 빠르고 효율적인 통신 가능
- 단점
  - 서버가 모든 클라이언트 상태를 관리해야 함
  - 특정 서버에 상태가 종속됨

HTTP의 특징



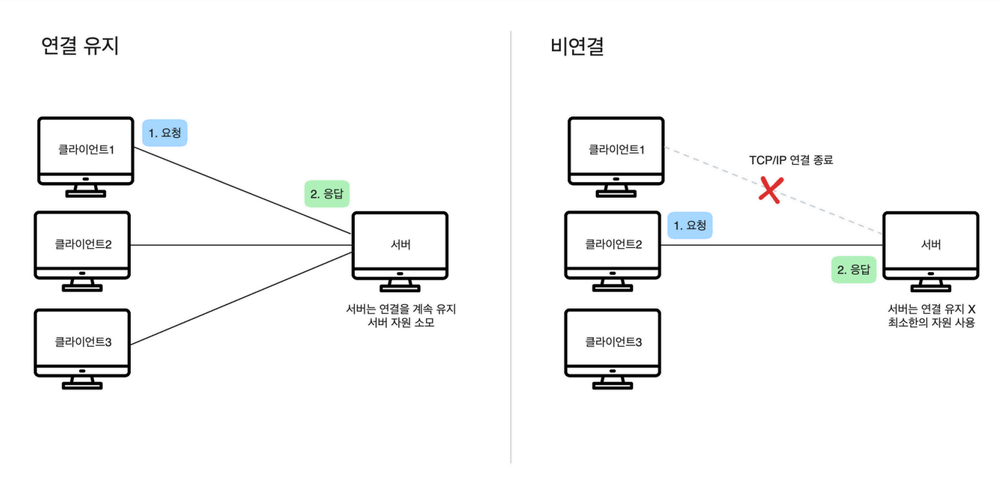


무상태 (Stateless)

- 서버는 클라이언트의 요청에 응답만 할 뿐, 그 상호작용을 기록하지 않음

- 장점
  - 서버의 상태를 관리할 필요가 없음
  - 다른 서버가 요청을 이어받아 처리할 수 있음
- 단점
  - 매 요청마다 필요한 모든 정보를 담아야 함

HTTP의 특징

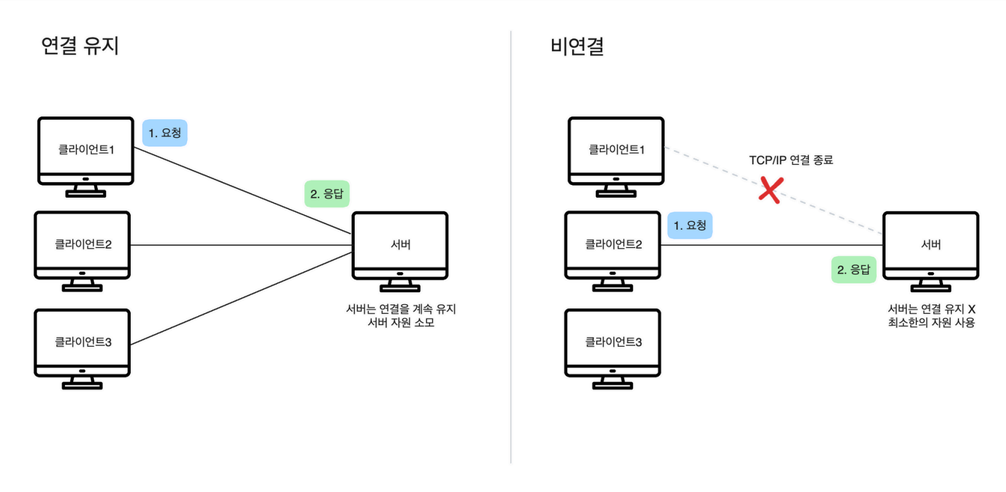


연결 유지(Connection-Oriented)

- 데이터를 전송하기 전에 통신 경로를 먼저 설정
- 통신이 끝나면 연결을 해제하는 방식

- 장점
  - 데이터가 순서대로 전달되는 것을 보장함
  - 중간에 데이터 유실 시 재전송 요청
- 단점
  - 연결 설정/해제 과정에 시간과 자원 소요

HTTP의 특징

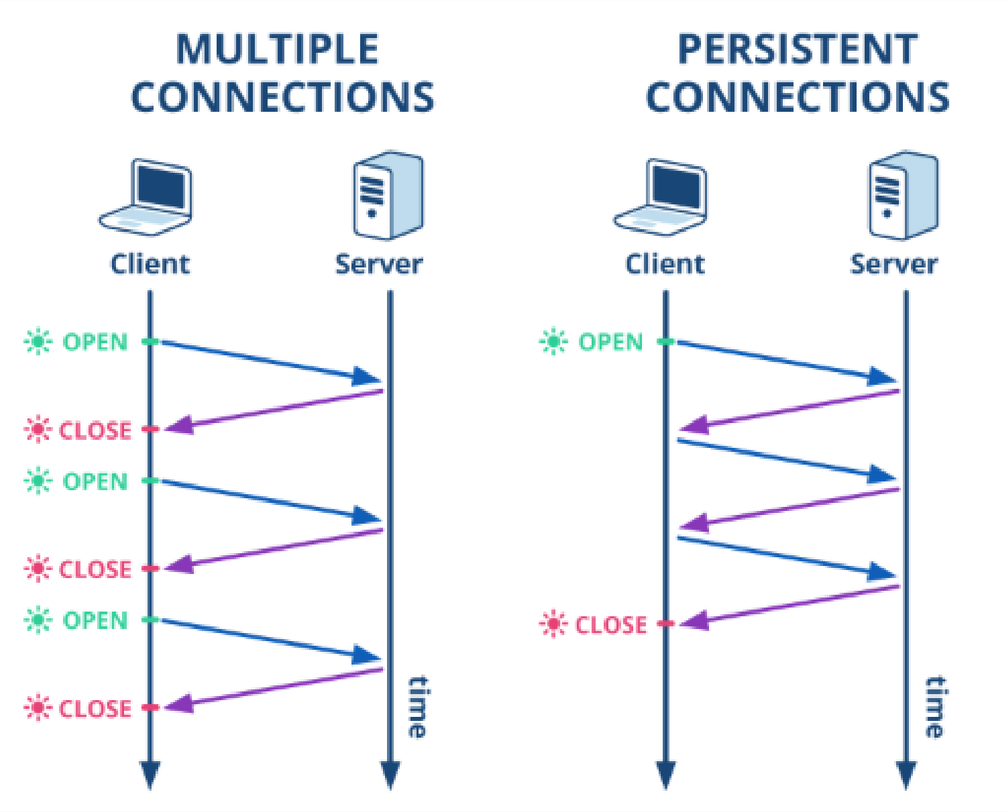


비연결 (Connectionless)

- 연결 없이 데이터 보내고 싶을 때 바로 전송

- 장점
  - 연결 설정 없어 오버헤드가 적음
  - 데이터를 빠르게 보낼 수 있음
- 단점
  - 데이터가 손실되거나 순서가 뒤바뀔 수 있음

HTTP의 특징



지속 연결 (Persistent Connection)

- 한 번 맺은 TCP 연결을 끊지 않고, 여러 요청과 응답을 같은 연결에서 처리 하는 방식

- 장점
  - 오버헤드 감소 및 네트워크 자원 절약
  - 웹 페이지의 로딩 속도 향상
- 단점
  - 요청을 안 보냈는데 열려있다면? 비효율적

02

HTTP의 장단점

HTTP의 장점

단순하고 확장 가능함

- 사람이 읽을 수 있는 메시지 형식을 사용하여 이해하기 쉬움
- 새로운 헤더나 메서드를 추가하여 기능을 확장 용이

서버 부하 감소

- 비연결성으로 동시에 많은 클라이언트 요청 처리 가능
- 서버 자원의 효율적인 관리

유연성

- HTML 문서, 이미지, 영상, JSON 등 다양한 형태의 데이터 전송 가능

HTTP의 단점

보안에 취약함

- 통신 내용이 암호화되지 않아 위험함
- 보안을 위해 HTTPS 권장

상태 유지의 어려움

- 무상태성으로 이전 통신 내용을 기억하지 못함
- 매 요청마다 인증 정보를 같이 보내야 함

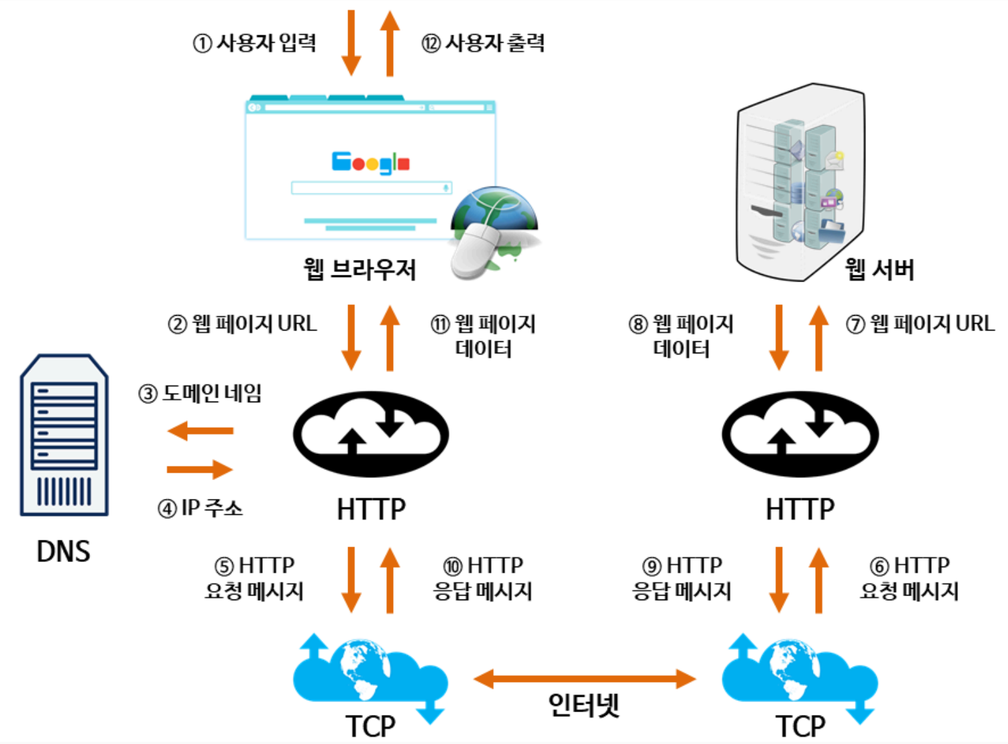
연결 오버헤드

- 비연결성으로 매 요청마다 새로운 연결 설정/해제 과정에서 시간과 자원 소모
- 대규모나 반복 요청 시 비효율성이 커짐

03

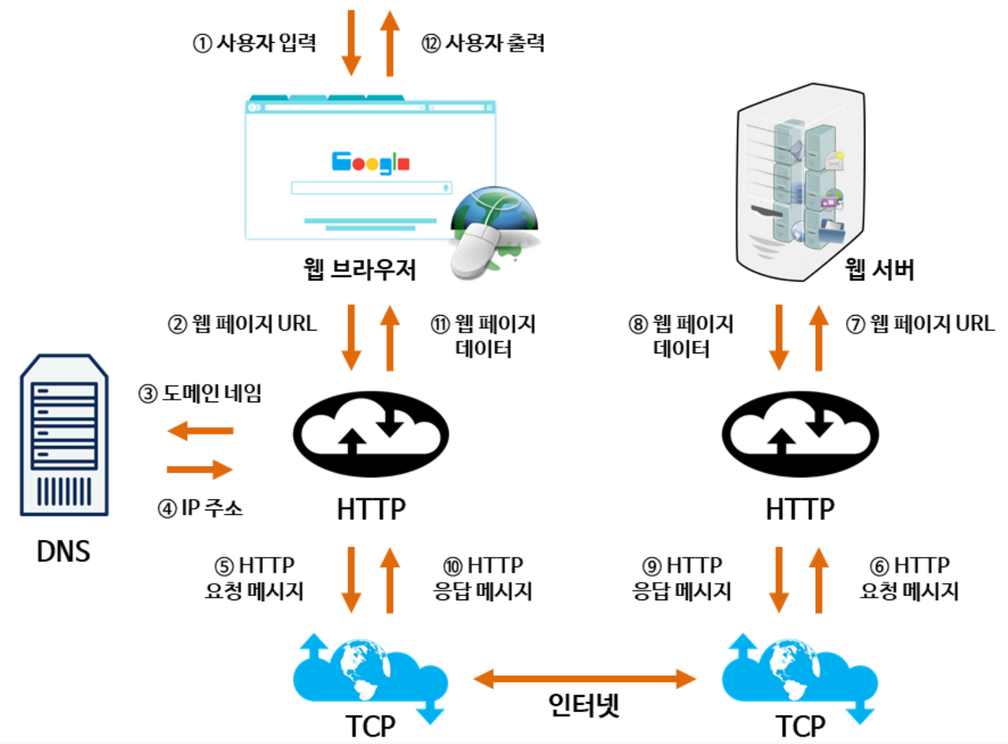
HTTP의 작동 방식

HTTP의 작동 방식



1. 사용자는 웹 브라우저를 통해 찾고 싶은 웹 페이지의 URL 주소 입력
2. 브라우저는 사용자가 입력한 URL을 받아 통신 준비
3. 사용자가 입력한 URL 주소 중에서 도메인 네임 부분을 DNS 서버에서 검색
4. DNS 서버에서 해당 도메인 네임에 해당하는 IP 주소를 찾아 사용자가 입력한 URL 정보와 함께 전달
5. 웹 페이지 URL 정보와 전달 받은 IP 주소는 HTTP 프로토콜을 사용하여 HTTP 요청 메시지를 생성
6. HTTP 요청 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 해당 IP 주소의 컴퓨터로 전송
7. HTTP 요청 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 URL 정보로 변경

HTTP의 작동 방식



8. 웹 서버는 도착한 웹 페이지 URL 정보에 해당하는 데이터를 검색
9. 검색된 웹 페이지 데이터는 또 다시 HTTP 프로토콜을 사용하여 HTTP 응답 메시지를 생성
10. 생성된 HTTP 응답 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 원래 컴퓨터로 전송
11. 도착한 HTTP 응답 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 데이터로 변환
12. 변환된 웹 페이지 데이터는 웹 브라우저에 의해 출력되어 사용자가 볼 수 있게 됨

04

HTTP 메시지

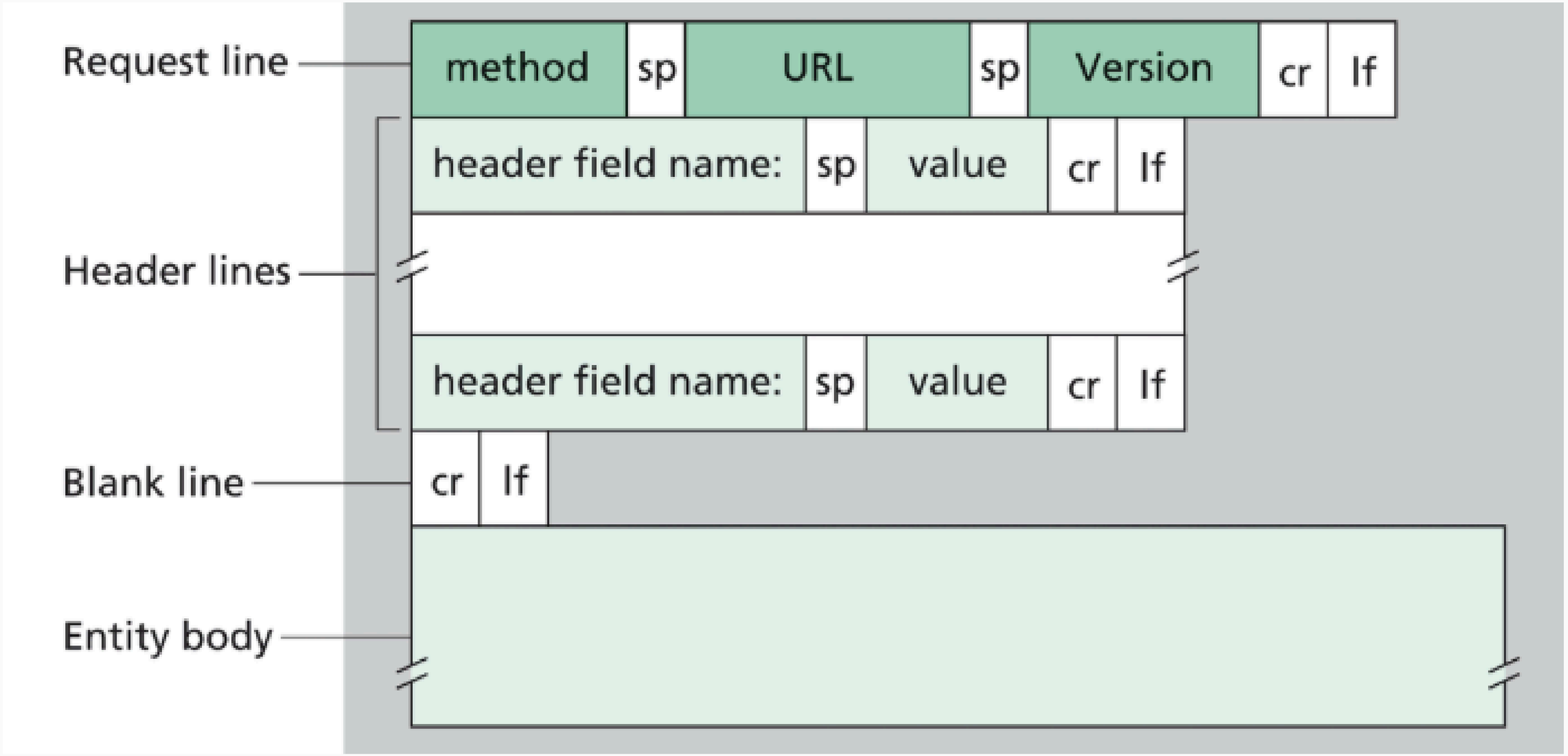
HTTP 메시지

HTTP Message

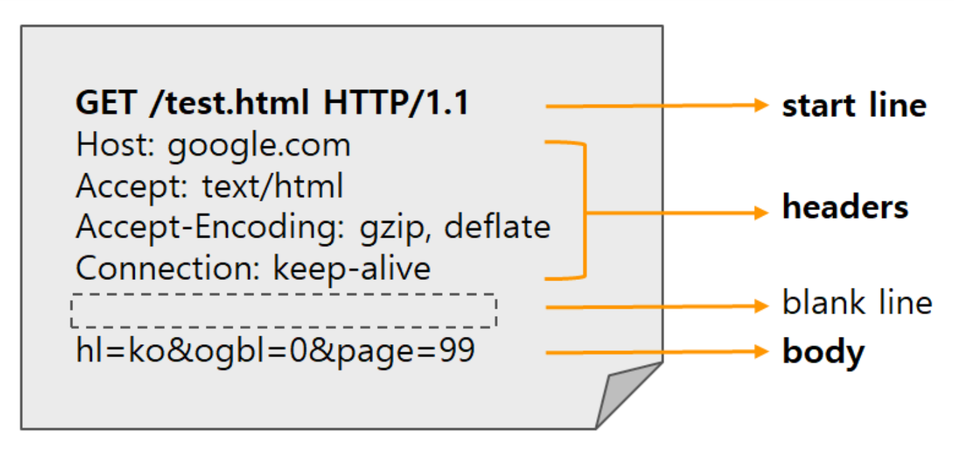
- 서버와 클라이언트 간에 데이터가 교환되는 방식

- 요청 메시지
  - 클라이언트가 서버로 전달해서 서버의 액션이 일어나게끔 하는 메시지
- 응답 메시지
  - 요청에 대한 서버의 답변

HTTP 요청 메시지


Request line
Header lines
Blank line
Entity body
method
sp
URL
sp
Version
cr
lf
header field name:
sp
value
cr
lf
header field name:
sp
value
cr
lf
cr
lf

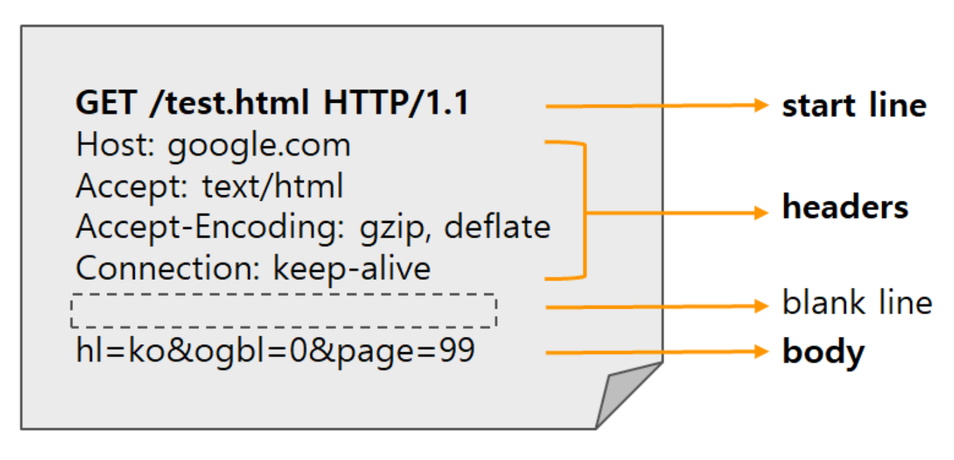
HTTP 요청 메시지



start line

- HTTP method
  - 서버가 수행해야 할 동작을 나타냄
- request target
  - URL, 프로토콜, 포트, 도메인의 절대 경로로 나타낼 수 있으며, 요청 컨텍스트에 의해 특정됨
- HTTP version
  - 응답 메시지에서 사용 할 HTTP 버전을 알려줌

HTTP 요청 메시지

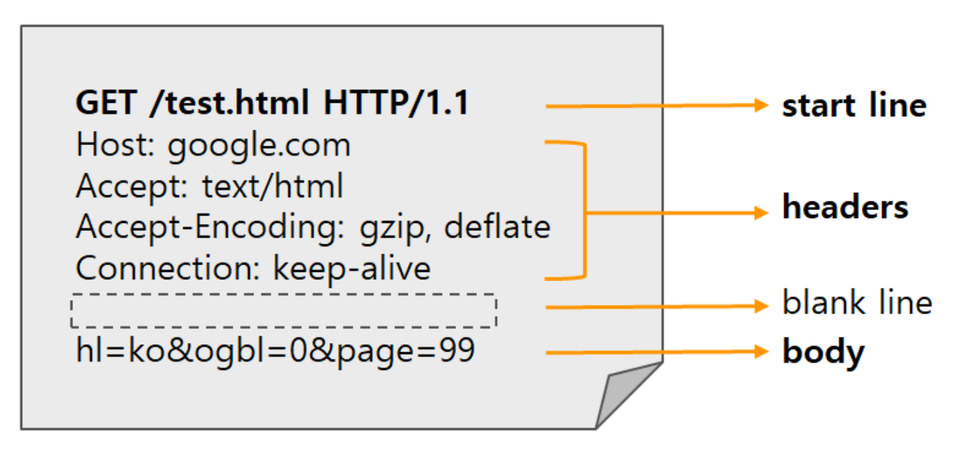


request target

- origin 형식
- 가장 일반적인 형식으로 절대 경로와 '?'와 같이 사용

- 예제
  - POST / HTTP/1.1
  - GET /background.png HTTP/1.0
  - HEAD /test.html?query=alibaba HTTP/1.1
  - OPTIONS /anypage.html HTTP/1.0

HTTP 요청 메시지

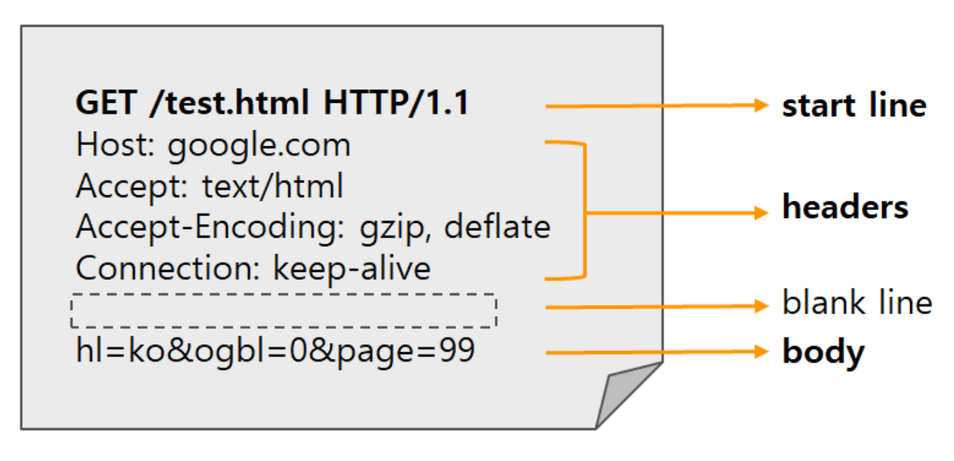


request target

- absolute 형식
- 완전한 URL, 대부분 GET과 같이 사용

- 예제
  - GET http://developer.mozilla.org/ko/docs/Web/HTTP/Messages HTTP/1.1

HTTP 요청 메시지

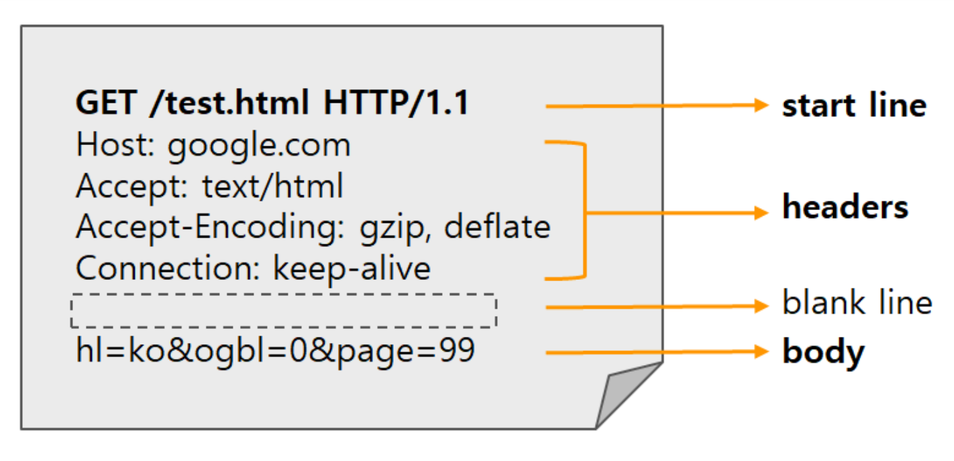


request target

- authority 형식
- 도메인 이름 및 옵션 포트로 이루어진 URL 인증 컴포넌트, CONNECT와 함께 사용 가능

- 예제
  - CONNECT developer.mozilla.org:80 HTTP/1.1

HTTP 요청 메시지

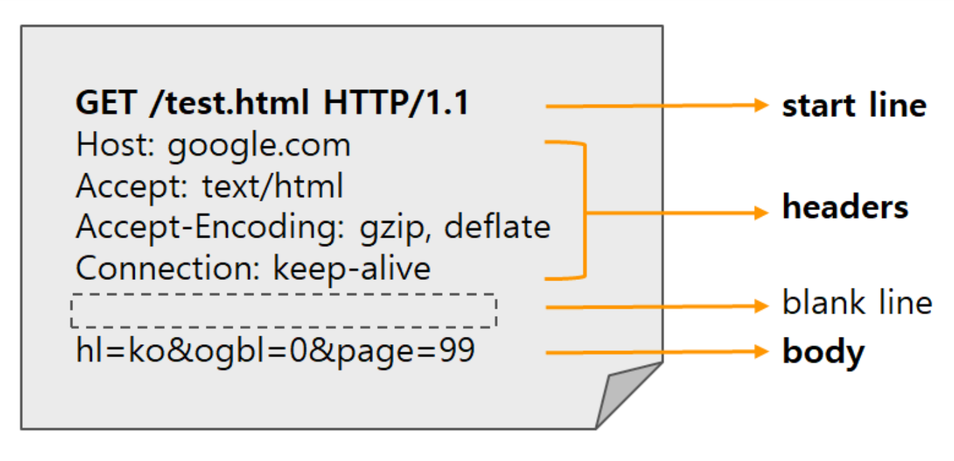


request target

- asterisk 형식
- ‘*’로 서버 전체를 나타냄, OPTION과 함께 사용

- 예제
  - OPTIONS * HTTP/1.1

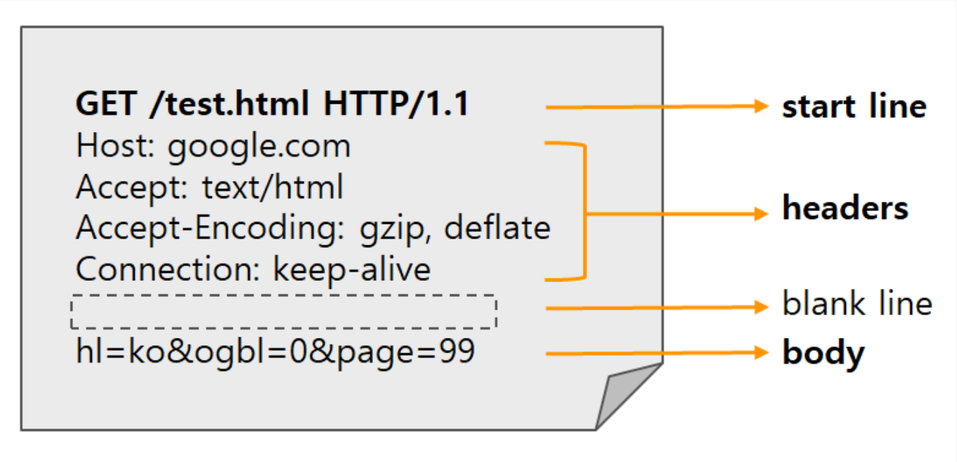
HTTP 요청 메시지



headers

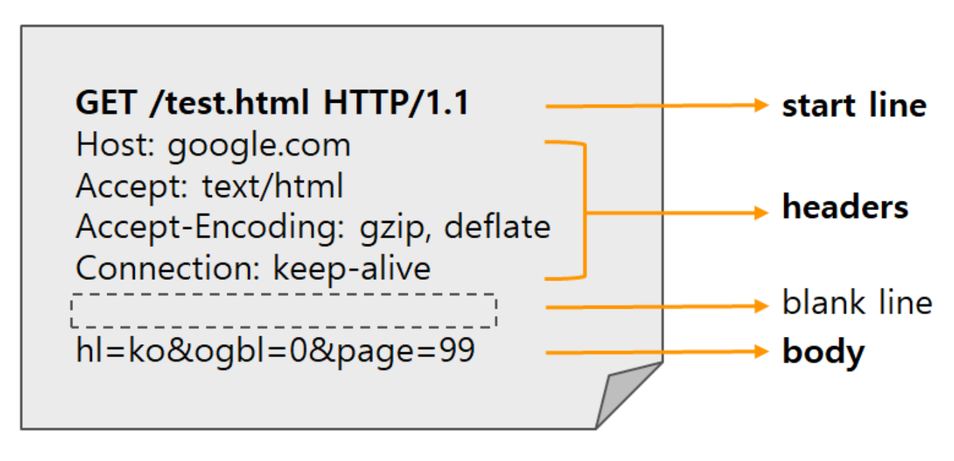
- general header
  - HTTP 헤더를 가리키는 데 사용되는 구식 용어
- request header
  - 서버가 응답을 맞춤화할 수 있도록 HTTP 요청에서 사용할 수 있는 헤더
- representation header
  - HTTP 메서지 본문으로 전송된 리소스의 표현을 설명하는 헤더

HTTP 요청 메시지



GET /home.html HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/testpage.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 18 Jul 2016 02:36:04 GMT
If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"
Cache-Control: max-age=0

HTTP 요청 메시지

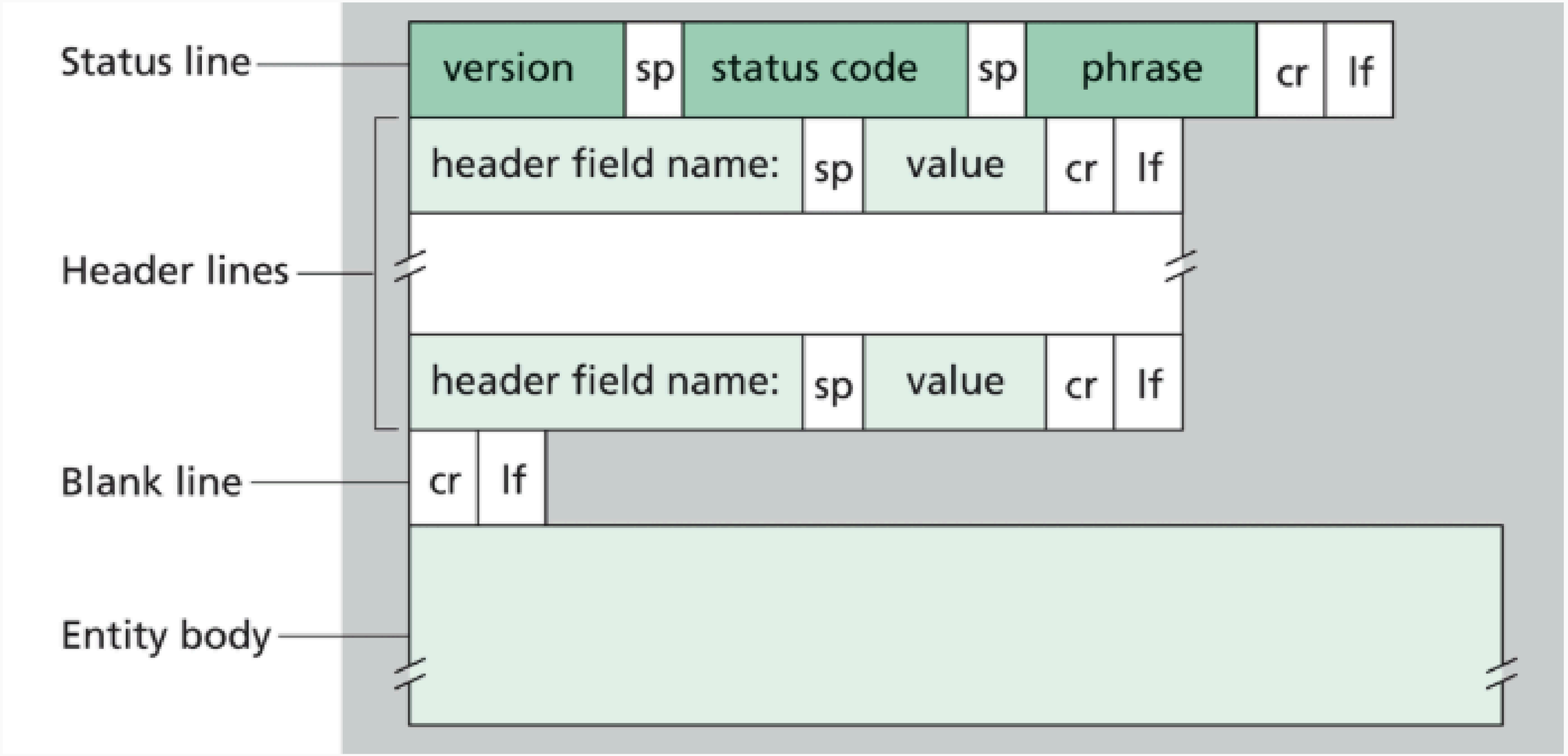


body

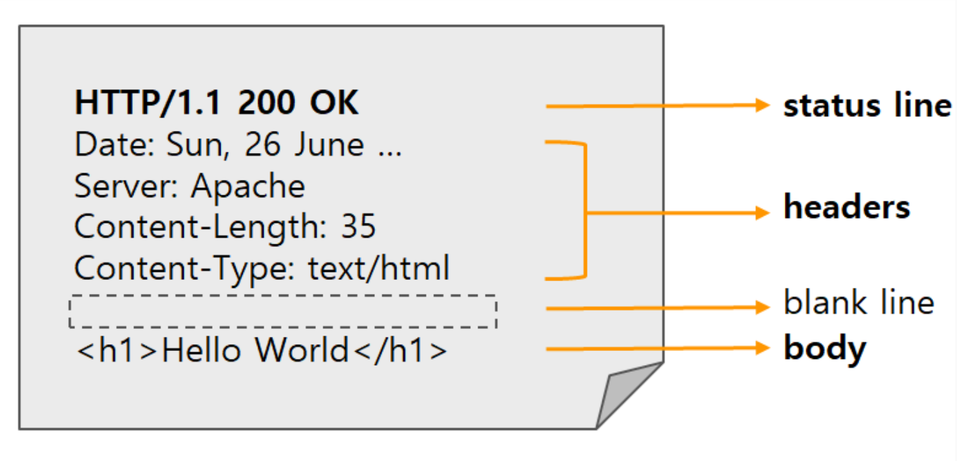
- 모든 요청에 본문이 들어가지는 않음
- GET, HEAD, DELETE, OPTION처럼 리소스를 가져오는 요청은 본문이 필요 없음

- 단일 리소스 본문 : ‘Content-Type’와 ‘Content-Length’로 정의된 단일 파일로 구성
- 다중 리소스 본문 : 각각 서로 다른 정보를 담고 있는 멀티파트 본문으로 구성

HTTP 응답 메시지


Status line
version
sp
status code
sp
phrase
cr
lf
Header lines
header field name:
sp
value
cr
lf
header field name:
sp
value
cr
lf
Blank line
cr
lf
Entity body

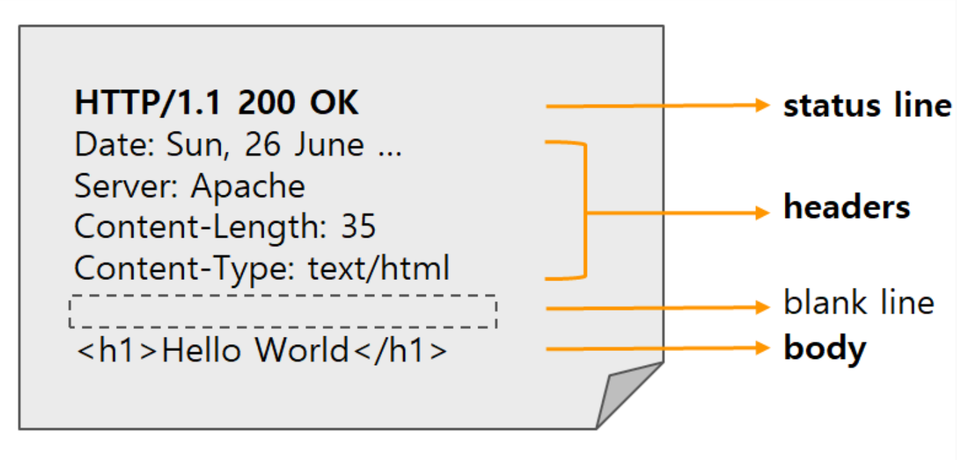
HTTP 응답 메시지



status line

- HTTP version
  - 응답 메시지에서 사용 할 HTTP 버전을 알려줌
- status code
  - 요청의 성공 여부를 나타냄
- status text
  - 상태 코드에 대한 짧고, 정보 제공 목적의 텍스트

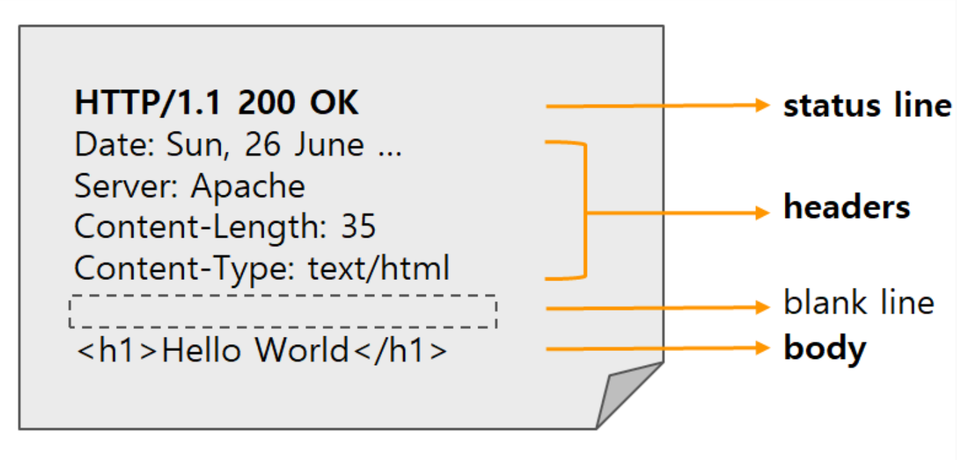
HTTP 응답 메시지



headers

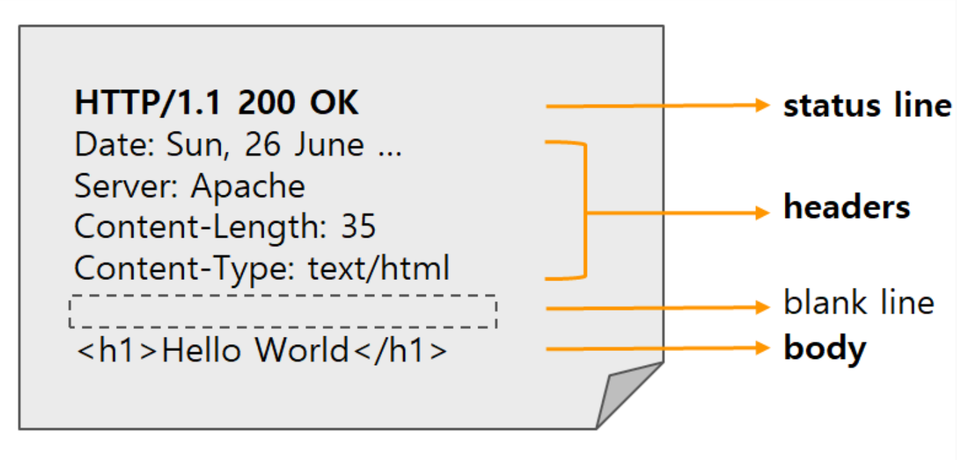
- general header
  - HTTP 헤더를 가리키는 데 사용되는 구식 용어
- request header
  - http 응답에서 사용될 수 있는 헤더
- representation header
  - HTTP 메서지 본문으로 전송된 리소스의 표현을 설명하는 헤더

HTTP 응답 메시지



200 OK
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Mon, 18 Jul 2016 16:06:00 GMT
Etag: "c561c68d0ba92bbeb8b0f612a9199f722e3a621a"
Keep-Alive: timeout=5, max=997
Last-Modified: Mon, 18 Jul 2016 02:36:04 GMT
Server: Apache
Set-Cookie: mykey=myvalue; expires=Mon, 17-Jul-2017 16:06:00 GMT; Max-Age=31449600; Path=/; secure
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
X-Backend-Server: developer2.webapp.scl3.mozilla.com
X-Cache-Info: not cacheable; meta data too large
X-kuma-revision: 1085259
x-frame-options: DENY

HTTP 응답 메시지



body

- 모든 요청에 본문이 들어가지는 않음
- ‘201 (Created)’, ‘204 (No Content)’처럼 해당 페이로드 없이도 요청에 응답하는 상태 코드는 본문 X

- 단일 리소스 본문 : ‘Content-Type’와 ‘Content-Length’로 정의된 단일 파일로 구성
- 단일 리소스 본문 : ‘Transfer-Encoding’가 ‘chunked’로 설정된 청크로 나뉘어 인코딩되는 길이를 모르는 하나의 파일로 구성
- 다중 리소스 본문 : 각각 서로 다른 정보를 담고 있는 멀티파트 본문으로 구성

05

HTTP method

HTTP method

GET

- 기본 역할
  - 대상 리소스의 현재 표현을 가져오는 데 사용되는 주요 정보 조회 방식
- 동작 방식
  - URI로 식별된 리소스를 요청하고 성공하면 ‘200 (OK)’로 반환
- 캐싱
  - GET 요청에 대한 서버의 응답은 캐시 가능
  - 서버는 이 응답을 저장해 두었다가 동일한 요청이 다시 올 경우 저장된 응답을 대신 전달할 수 있음
- 보안 고려사항
  - 사용자 입력이 URI에 노출될 수 있음

HTTP method

HEAD

- 기본 역할
  - GET과 동일하지만 서버가 응답에 본문을 전송하지 않음
- 사용 목적
  - 리소스가 실제 데이터를 다운로드하지 않고, 리소스에 대한 메타데이터를 얻기 위해 사용
  - 하이퍼링크 유효성 검사, 최근 수정 여부 확인 등
- 캐싱
  - HEAD 요청에 대한 서버의 응답은 캐시 가능
- 응답 헤더
  - 본문을 생성해야만 알 수 있는 헤더는 생략 가능
- 제한 사항
  - HEAD 요청에 본문을 담는 건 보안상 위험

HTTP method

POST

- 기본 역할
  - 요청 본문을 서버 리소스에 전달하여 서버가 정의된 방식대로 처리하도록 요청
- 주요 용도
  - HTML 폼 데이터 전송, 게시판/블로그 글 작성, 새로운 리소스 생성, 기존 리소스에 데이터 추가
- 응답 상태 코드
  - 요청 성공하여 새로운 리소스가 생성된 경우 서버는 ‘201 (Created)’ 응답으로 반환
  - 이미 존재하는 리소스를 보여주는 것과 같다면 서버는 ‘303 (See Other)’ 응답
- 캐싱
  - POST 요청에 대한 응답은 일반적으로 캐시되지 않음

HTTP method

PUT

- 기본 역할
  - 대상 리소스를 요청 본문에 담긴 표현으로 생성하거나 교체
- POST와의 차이
  - POST : 서버가 리소스를 생성하거나 처리함
  - PUT : 클라이언트가 리소스를 정의함
- 캐싱
  - PUT 요청에 대한 응답은 캐시되지 않음
  - 서버는 이 응답을 저장해 두었다가 동일한 요청이 다시 올 경우 저장된 응답을 대신 전달할 수 있음
- 응답 상태 코드
  - 새로운 리소스 생성 성공 : ‘201 (Created)’
  - 기존 리소스 수정 성공 : ‘200 (OK)’, ‘204 (NO Content)’
  - 데이터 불일치/제약 조건 위반 : ‘403 (Conflict)’, ‘415 (Unsupported Media Type)’

HTTP method

PATCH

- PUT과의 비교
  - PUT : 요청 본문의 내용으로 리소스 전체를 완전히 교체
  - PATCH : 제공된 지침에 따라 리소스의 일부만 수정
- 멱등성
  - 안전하지도 않고 멱등성을 보장하지도 않음
- 원자성
  - 서버는 모든 변경 사항을 하나의 단위로 적용해야 함
  - 일부만 적용되어서는 안되며, 전체 변경이 불가능한 경우 어떤 변경도 적용되서는 안됨
- 캐싱
  - PATCH에 대한 응답은 캐시되지 않음
  - 해당 URI의 기존 캐시된 데이터를 오래된 것으로 처리

HTTP method

PATCH

- 충돌 위험
  - 여러 PATCH 요청이 동시에 들어오면 리소스가 손상될 수 있음
- 패치 문서 형식
  - 모든 리소스에 적합한 단일 표준 패치 문서 형식은 없음
  - 서버는 요청 받은 패치 문서가 해당 리소스 유형에 적합한지 확인해야 함
- PATCH 사용 시점
  - 패치 문서가 PUT으로 보낼 전체 리소스 데이터보다 크다면 PUT이 더 효율적
  - 예측 불가능한 방식으로 리소스를 수정하는 경우에는 POST 고려

HTTP method

DELETE

- 기본 역할
  - 서버에 대상 리소스와의 연결을 제거하도록 요청
- 주요 용도
  - 원격 저장 환경(버전 관리, 리소스 취소/삭제)에 사용
  - PUT이나 POST로 생성된 리소스를 제거할 때 자주 사용
- 응답 상태 코드
  - ‘202 (Accepted)’ : 요청은 접수되었으나, 아직 삭제 작업이 실행되지 않은 상태
  - ‘204 (No Content)’ : 삭제 작업이 성공적으로 완료되었으며, 본문에 추가 정보가 없음
  - ‘200 (OK)’ : 삭제 작업이 성공적으로 완료되었으며, 본문에 처리 상태를 정보를 포함하여 전달
- 캐싱
  - DELETE에 대한 응답은 캐시되지 않음
- 제한 사항
  - DELETE 요청에 본문을 포함 하는 것은 안됨

HTTP method

CONNECT

- 기본 역할
  - 수신자에게 요청된 목적지 서버로 가는 터널을 생성하도록 요청
  - 터널 생성 시 그 이후는 TCP 통신으로 전환
  - 프록시 서버에 요청하기 위해 사용
- 요청 형식
  - 목적지 서버의 호스트와 포트 번호로만 구성
  - ‘CONNECT server.example.com:443 HTTP/1.1’
- 응답 상태 코드
  - ‘2xx (Successful)’ 응답 시 터널 모드로 전환
- 캐싱
  - CONNECT에 대한 응답은 캐시되지 않음
- 제한 사항
  - 악의적인 사용자가 프록시를 속여 웹 트래픽이 아닌 포트로 터널을 생성하도록 요청할 수 있음

HTTP method

OPTIONS

- 기본 역할
  - 특정 리소스 또는 서버 전체에 대해 사용 가능한 통신 옵션/요구사항을 확인
- 주요 용도
  - ‘*’ : 서버 전체의 능력 테스트
  - 특정 리소스 URI : 해당 리소스와의 통신에서 가능한 메서드/옵션 확인
- 서버 응답
  - ‘Allow’ : 서버가 지원하는 HTTP 메서드 목록을 알려줌
- 캐싱
  - OPTIONS에 대한 응답은 캐시되지 않음
- 본문
  - 일반적으로 OPTIONS 요청에는 본문을 보내지 않음

HTTP method

TRACE

- 기본 역할
  - 요청 메시지가 서버까지 가는 과정에서 어떤 변경이 있었는지 확인하기 위한 loop-back 테스트 요청
- 활용 사례
  - 클라이언트가 실제로 서버에 도달하는 요청 메시지 확인 가능
- 보안 주의 사항
  - 클라이언트는 절대로 민감한 정보(쿠키, 인증 정보)를 포함해서는 안됨
  - 서버는 응답을 생성할 때, 민감한 정보가 포함될 가능성이 있는 요청 필드는 제외해야 함
- 캐싱
  - TRACE에 대한 응답은 캐시되지 않음
- 본문
  - 클라이언트는 TRACE 요청에 본문을 보내서는 안됨

06

HTTP 상태 코드

HTTP 상태 코드

Informational Requests (1XX)

서버가 요청을 수신하고 이해했음을 나타냄

HTTP 상태 코드

Informational Requests (1XX)

- 100 Continue
  - 클라이언트가 서버로 보낸 요청에 문제가 없으니 다음 요청을 이어서 보내도 된다는 것을 의미
- 101 Switching Protocols
  - 클라이언트가 서버에 프로토콜 전환을 요청했고, 서버가 전환하는 프로토콜을 나타냄
- 102 Processing
  - 서버가 요청을 수신하였으며 이를 처리하고 있지만, 아직 제대로 된 응답을 알려줄 수 없음을 알려줌
  - 더 이상 사용되지 않음
- 103 Early Hints
  - 서버가 응답을 준비하는 동안에도 사용자 에이전트가 리소스를 미리 읽어들일 수 있도록 함
  - 실험적인 기능

HTTP 상태 코드

Successful Requests (2XX)

요청이 성공했음을 나타냄

- GET : 리소스를 가져왔고 메시지 본문으로 전송되었다는 것을 의미
- HEAD : 메시지 본문 없이 표현 헤더가 응답에 포함되어 있다는 것을 의미
- POST : 리소스가 명시하는 행동의 결과가 메시지 본문에 전송되었다는 것을 의미
- TRACE : 서버가 요청받은 메시지가 메시지 본문에 포함되어 있다는 것을 의미

HTTP 상태 코드

Successful Requests (2XX)

- 201 Created
  - 요청이 성공적으로 처리되었으며, 자원이 생성되었음을 나타냄
  - 응답이 반환되기 이전에 새로운 리소스가 생성됨
- 202 Accepted
  - 요청이 처리를 위해 수락되었으나, 아직 해당 요청에 대해 처리 중이거나 처리 시작되지 않았을 수도 있다는 것을 의미
  - 요청이 처리 중 실패할 수도 있기 때문에 요청은 실행될 수도 실행되지 않았을 수도 있음
- 203 Non-Authoritative Information
  - 돌려 받은 메타 정보 세트가 오리진 서버의 것과 일치하지 않지만 로컬이나 서드 파티 복사본에서 모아졌음을 의미
  - 200 OK` 응답을 반드시 우선
- 204 No Content
  - 요청이 성공했으나 요청에 대해서 보내줄 수 있는 콘텐츠가 없음

HTTP 상태 코드

Successful Requests (2XX)

- 205 Reset Content
  - 요청을 완수한 이후에 클라이언트에게 요청을 보낸 문서를 리셋하라고 알려줌
- 206 Partial Content
  - 클라이언트에서 복수의 스트림을 분할 다운로드 하고자 범위 헤더를 전송했기 때문에 사용됨
- 207 Multi-Status
  - 여러 리소스가 여러 상태 코드인 상황이 적절한 경우에 해당되는 정보를 전달함
- 208 Already Reported
  - DAV에서 사용됨
  - propstat 응답 속성으로 동일 컬렉션으로 바인드된 복수의 내부 멤버를 반복적으로 열거하는 것을 피하기 위해 사용
- 226 IM Used
  - 서버가 GET 요청에 대한 리소스 요청을 완료했고, 응답은 하나 이상의 인스턴스 조작이 현재 인스턴스에 적용 되었음을 알려줌

HTTP 상태 코드

Redirects (3XX)

클라이언트가 리다이렉션되었음을 나타냄

HTTP 상태 코드

Redirects (3XX)

- 300 Multiple Choice
  - 요청에 대해 여러 가지 옵션이 있으며, 클라이언트가 그중에 하나를 반드시 선택해야 함
- 301 Moved Permanently
  - 요청한 리소스의 URI가 영구적으로 이동되었으며, 서버는 새 URI로 응답함
- 302 Found
  - 요청한 리소스의 URI가 일시적으로 변경되었으며, 클라이언트는 다른 URL을 탐색해야 함
- 303 See Other
  - 클라이언트가 요청한 리소스를 다른 URI에서 GET 요청을 통해 얻어야 할 때, 서버가 클라이언트로 직접 보내는 응답
- 304 Not Modified
  - 클라이언트에게 응답이 수정되지 않았음을 알려주며, 동일한 캐시 버전의 응답을 사용할 수 있음

HTTP 상태 코드

Redirects (3XX)

- 305 Use Proxy
  - 요청한 응답은 반드시 프록시를 통해서 접속해야 하는 것을 알려줌
  - 보안 상의 문제로 사용되지 않음
- 306 unused
  - 추후 사용을 위해 예약되어 있음
- 307 Temporary Redirect
- 서버는 클라이언트가 이전 요청에서 사용된 동일한 방법으로 다른 URI에서 리소스를 요청하도록 지시
  - 이전 요청에서 사용한 HTTP 메서드를 변경해서는 안 됨
  - ‘302 Found’와 동일한 의미를 가짐
- 308 Permanent Redirect
  - 리소스는 영구적으로 다른 URI로 이동되며, 향후 모든 요청은 주어진 URI로 전달되어야 함
  - 이전 요청에서 사용한 HTTP 메서드를 변경해서는 안 됨
  - ‘301 Moved Permanently’와 동일한 의미를 가짐

HTTP 상태 코드

Client Errors (4XX)

클라이언트 측에 문제가 있음을 나타냄

HTTP 상태 코드

Client Errors (4XX)

- 400 Bad Request
  - 잘못된 문법으로 인하여 서버가 요청을 이해할 수 없음
- 401 Unauthorized
  - 요청된 응답을 받기 전에 클라이언트가 자신을 인증해야 함
- 402 Payment Required
  - 디지털 결제 시스템에 사용될 예정
  - 추후 사용을 위해 예약되어 있음
- 403 Forbidden
  - 클라이언트는 요청된 리소스에 액세스할 권한이 없음
  - ‘401 Unauthorized’과 달리 클라이언트가 누구인지 알려져 있음

HTTP 상태 코드

Client Errors (4XX)

- 404 NOT Found
  - 서버는 요청받은 리소스를 찾을 수 없음
  - 브라우저에서는 알려지지 않은 URL을 의미
  - 서버는 인증받지 않은 클라이언트로부터 리소스를 숨기기 위하여 이 응답을 403 대신에 전송할 수 있음
- 405 Method Not Allowed
  - 요청 메소드는 서버에서 알고 있지만, 제거되었고 사용할 수 없음
  - 예를 들어, 클라이언트가 POST를 통해 데이터를 제출해야 하는 양식을 요청하는 경우
- 406 Not Acceptable
  - 요청된 리소스는 요청에 전송된 수락 헤더에 허용되지 않는 콘텐츠만 생성할 수 있음
- 407 Proxy Authentication Required
  - 클라이언트는 프록시를 통해 자신을 인증해야 함

HTTP 상태 코드

Client Errors (4XX)

- 408 Request Timeout
  - 서버가 클라이언트의 요청을 기다리는 동안 시간 초과됨
- 409 Conflict
  - 서버의 현재 리소스 상태와 충돌하여 요청을 처리할 수 없음을 나타냄
- 410 Gone
  - 요청한 콘텐츠가 서버에서 영구적으로 삭제되어 전달해 줄 수 있는 주소 역시 존재하지 않음
- 411 Length Required
  - ‘Content-Length’ 헤더 필드가 정의되지 않은 요청이 들어와 서버가 요청을 거부함
- 412 Precondition Failed
  - 클라이언트의 헤더에 있는 전체조건은 서버의 전제조건을 충족시키지 못함
- 413 Payload Too Large
  - 요청이 너무 커서 서버가 처리할 수 없음을 나타냄

HTTP 상태 코드

Client Errors (4XX)

- 414 URI Too Long
  - 클라이언트가 요청한 URI는 서버에서 처리하지 않기로 한 길이보다 김
- 415 Unsupported Media Type
  - 요청된 미디어 포맷이 서버에서 지원하지 않아 서버가 요청을 거부함
- 416 Requested Range Not Satisfiable
  - ‘Range’ 헤더 필드에 요청된 지정 범위를 만족시킬 수 없음
- 417 Expectation Failed
  - ‘Export’ 헤더 필드로 요청한 예상이 서버에서는 적당하지 않음을 알려줌
- 418 I'm a teapot
  - 서버는 커피를 찻 주전자에 끓이는 것을 거절함
  - 일부 웹 사이트에서 만우절 농담으로 정의됨

HTTP 상태 코드

Client Errors (4XX)

- 421 Misdirected Request
  - 요청이 응답을 생성할 수 없는 서버로 전달됨
- 422 Unprocessable Entity
  - 요청이 잘 만들어졌지만, 문법 오류로 인하여 따를 수 없음
- 423 Locked
  - 클라이언트가 접근하려고 하는 리소스가 잠겨 있음
- 424 Failed Dependency
  - 이전 요청이 실패하였기에 지금의 요청도 실패함
- 426 Upgrade Required
  - 서버는 현재 프로토콜을 사용하여 요청을 처리하는 것을 거절하였지만, 클라이언트가 다른 프로토콜로 업그레이드하면 요청을 처리할 수 있음

HTTP 상태 코드

Client Errors (4XX)

- 428 Precondition Required
  - 오리진 서버는 요청이 조건적이여야 함
- 429 Too Many Requests
  - 클라이언트가 짧은 시간 안에 너무 많은 요청을 보냄
- 431 Request Header Fields Too Large
  - 요청한 헤더 필드가 너무 크기 때문에 서버는 요청을 처리하지 않음
- 451 Unavailable For Legal Reasons
  - 클라이언트가 정부에 의해 검열된 웹 페이지와 같은 불법적인 리소스를 요청함

HTTP 상태 코드

Server Errors (5XX)

서버 측에 문제가 있음을 나타냄

HTTP 상태 코드

Server Errors (5XX)

- 500 Internal Server Error
  - 서버가 처리 방법을 모르는 상황이 발생함
  - 서버는 아직 처리 방법을 알 수 없음
- 501 Not Implemented
  - 요청 메소드가 서버에서 지원되지 않기 때문에 서버가 요청을 처리할 수 없음
- 502 Bad Gateway
  - 서버가 요청을 처리하는 데 필요한 응답을 얻기 위해 게이트웨이로 작업하는 동안 잘못된 응답을 수신함
- 503 Service Unavailable
  - 서버가 요청을 처리할 준비가 되지 않음
  - 유지보수를 위해 작동이 중단되거나 과부화가 걸린 상태
- 504 Gateway Timeout
  - 서버가 게이트웨이 역할을 하고 있으며 적시에 응답을 받을 수 없음

HTTP 상태 코드

Server Errors (5XX)

- 505 HTTP Version Not Supported
  - 요청에 사용된 HTTP 버전은 서버에서 지원되지 않음
- 506 Variant Also Negotiates
  - 서버의 내부 구성 오류가 있음
- 507 Insufficient Storage
  - 서버의 저장 공간이 부족해 요청을 처리할 수 없음
- 508 Loop Detected
  - 서버가 요청을 처리하는 동안 무한 루프를 감지함
- 510 Not Extended
  - 서버가 요청을 이행하기 위해 추가 확장이 필요함
- 511 Network Authentication Required
  - 클라이언트가 네트워크에 접속하려면 인증을 받아야 할 필요가 있음

07

HTTP 헤더

HTTP 헤더

Host

- 요청하려는 서버의 도메인 이름과 포트 번호를 나타냄
- 포트가 주어지지 않으면 요청된 서버의 기본 포트를 의미함
- HOST 헤더의 필드는 HTTP/1.1 요청 메시지 내에 포함되어 전송되어야 함
- 역할 : 하나의 IP 주소로 여러 개의 웹사이트를 운영하는 서버에서, 클라이언트가 어떤 사이트에 접속하고 싶은지 알려줌
- 예시 : ‘Host: developer.mozilla.org’

HTTP 헤더

Accept

- 클라이언트가 이해할 수 있는 콘텐츠 유형을 알려줌
- 서버는 제안 중 하나를 선택하고, ‘Content-Type’ 헤더로 클라이언트에게 선택된 타입을 알려줌
- 역할 : HTML, XHTML, XML 문서를 받을 수 있음을 서버에 알려줌
- 예시 : ‘Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8’

Content-Type

- 응답 본문의 미디어 타입을 알려줌
- 예시 : ‘Content-Type: text/html; charset=utf-8’

HTTP 헤더

Accept-Language

- 클라이언트가 선호하는 자연어를 알려줌
- 서버는 제안 중 하나를 선택하고, ‘Content-Language’ 헤더로 클라이언트에게 선택된 내용을 알려줌
- 역할 : 다국어 서비스를 제공하는 웹사이트에서 클라이언트에게 맞는 연어(프랑스어, 영어, 한국어)로 페이지를 보여주도록 요청
- 예시 : ‘Accept-Language: fr-CH, fr;q=0.9, en;q=0.8, ko;q=0.7, *;q=0.5’

Content-Language

- 응답 본문의 사용된 언어를 알려줌
- 예시 : ‘Content-Language: de-DE’

HTTP 헤더

Accept-Encoding

- 클라이언트가 이해할 수 있는 압축 방식을 알려줌
- 서버는 제안 중 하나를 선택하고, ‘Content-Encoding’ 헤더로 클라이언트에게 선택된 내용을 알려줌
- 역할 : 서버에게 데이터를 압축해서 보내달라고 요청
- 예시 : ‘Accept-Encoding: deflate, gzip;q=1.0, *;q=0.5’
-

Content-Encoding

- 응답 본문이 어떻게 압축되었는지 알려줌
- - 예시 : ‘Content-Encoding: gzip’

HTTP 헤더

User-Agent

- 요청을 보낸 클라이언트의 정보를 담고 있음
- 역할 : 서버는 클라이언트의 정보를 보고 사용자에게 보여줄 페이지를 결정할 수 있음
- 사용자 예시
  - Firefox 사용자 : ‘Mozilla/5.0 (platform; rv:geckoversion) Gecko/geckotrail Firefox/firefoxversion’
  - Chrome 사용자 : ‘Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36’
  - Safari 사용자 : ‘Mozilla/5.0 (iPhone; CPU iPhone OS 13_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.1 Mobile/15E148 Safari/604.1’
  - 크롤러 : ‘Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)’

HTTP 헤더

Authorization

- 사용자의 인증 정보를 담고 있음
- - 역할 : 서버는 로그인이 허가된 사용자임을 판단할 수 있음
- 예시 : ‘Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l’

Cookie

- 서버가 이전에 클라이언트에 저장해 둔 작은 데이터 조각을 다시 서버로 보냄
- 역할 : 로그인 상태 유지, 장바구니 정보 저장 등 사용자의 상태를 기억하는 데 사용됨
- 예시 : ‘Cookie: lang=ko; theme=dark;’

HTTP 헤더

Origin

- 요청이 시작된 출처를 나타냄
- 주로 CORS 요청에 사용됨
- 역할 : 다른 도메인에서 온 요청의 경우 서버가 허용된 사이트인지 판단할 근거를 제공함
- 예시 : ‘Origin: https://developer.mozilla.org’

Referer

- 현재 요청을 보낸 페이지의 이전 페이지 주소를 담음
- 역할 : 사용자가 어떤 경로를 통해 현재 페이지로 유입되었는지 분석하는 데 사용
- 예시 : ‘Referer: https://developer.mozilla.org/ko/docs/Web/JavaScript’

HTTP 헤더

Content-Length

- 응답 본문의 크기를 바이트 단위로 알려줌
- 예시 : ‘Content-Length: 1523’

Location

- 페이지를 다른 주소로 리다이렉션 시킬 때 사용
- 예시 : ‘Location: /index.html’

HTTP 헤더

Cache-Control

- 캐시의 동작 방식을 지정함
- 역할 : 캐시의 유효 기간을 설정하거나, 캐시를 사용하지 못하게 하는 등 다양한 정책을 설정할 수 있음
- 예시 : ‘Cache-Control: no-cache, no-store, must-revalidate’

Expires

- 캐시의 만료 날짜를 지정함
- 예시 : ‘Expires: Sun, 26 Oct 2025 00:00:00 GMT’

HTTP 헤더

Set-Cookie

- 클라이언트에 쿠키를 저장하도록 지시함
- 예시 : ‘Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT’

Strict-Transport-Security

- 앞으로 이 사이트에 접속할 때는 무조건 HTTPS만 사용하도록 브라우저에 강제함
- 예시 : ‘Strict-Transport-Security: max-age=31536000; includeSubDomains’

HTTP 헤더

Content-Security-Policy

- 신뢰할 수 있는 외부 스크립트나 리소스만 로드하도록 허용하여 XSS를 방어함
- 예시 : ‘Content-Security-Policy: default-src 'self';’

X-Frame-Options

- 다른 사이트에서 `<iframe>` 등을 통해 현재 페이지를 삽입하는 것을 방지하여 클랙재킹을 막음
- 예시 : ‘X-Frame-Options: DENY’

HTTP 헤더

Server

- 응답을 생성한 웹 서버 소프트웨어의 정보를 알려줌
- 예시 : ‘Server: Apache/2.4.1 (Unix)’

Date

- 서버가 응답을 생성한 시각을 나타냄
- 예시 : ‘Date: Wed, 21 Oct 2015 07:28:00 GMT’

HTTP 헤더

Connection

- 현재의 요청과 응답이 완료된 후 네트워크 연결을 계속 유지할 지 아니면 닫을 지를 결정함
- 예시 : ‘Connection: keep-alive’, ‘Connection: close’

Keep-Alive

- ‘Connection: keep-alive’가 사용될 때, 연결을 얼마나 오래 유지할지에 대한 세부 설정을 전달함
- 예시 : ‘Keep-Alive: timeout=5, max=100’

HTTP 헤더

If-None-Match

- 클라이언트가 가지고 있는 리소스의 ‘ETag’ 값을 서버로 보냄
- 서버는 자신이 가진 리소스의 ‘ETag’ 값과 비교하여 값이 동일하면 본문 없는 ‘304 Not Modified’ 응답을 보냄
- 예시 : ‘If-None-Match: "abcdefg12345"’
- 서버 응답
  - 변경 없음 : ‘HTTP/1.1 304 Not Modified’
  - 변경됨 : ‘HTTP/1.1 200 OK’와 함께 새로운 데이터 전송

ETag

- 특정 버전의 리소스를 식별하는 고유한 문자열 검사기를 제공함
- 파일의 해시값 등으로 생성되면, 내용이 조금이라도 바뀌면 ETag 값도 변경됨
- 예시 : ‘ETag: "abcdefg12345"’

HTTP 헤더

If-Modified-Since

- 클라이언트가 가지고 있는 리소스의 ‘Last-Modified’ 값을 서버로 보냄
- 서버는 이 날짜 이후에 리소스가 수정되었는지 확인하고 수정되지 않았으면 ‘304 Not Modified’ 응답을 보냄
- 예시 : ‘If-Modified-Since: Tue, 15 Nov 2022 08:12:31 GMT’

Last-Modified

- 리소스가 서버에서 마지막으로 수정된 날짜와 시간을 알려줌
- 예시 : ‘Last-Modified: Tue, 15 Nov 2022 08:12:31 GMT’

HTTP 헤더

Forwarded

- 프록시를 거치면서 손실될 수 있는 클라이언트의 원래 IP 주소, 호스트, 프로토콜 등의 정보를 담는 헤더
- 예시 : ‘Forwarded: for=192.0.2.43; proto=https; host=example.com’

X-Forwarded-For

- 요청이 여러 프록시를 거칠 때, 각 프록시의 IP를 포함하여 최초 클라이언트의 IP 주소를 식별하는 데 사용됨
- 예시 : ‘X-Forwarded-For: 203.0.113.195, 70.41.3.18, 150.172.238.178’

HTTP 헤더

Via

- 요청과 응답이 어떤 프록시들을 거쳐 갔는지 그 경로를 기록함
- 주로 요청 흐름을 디버깅하거나 프록시 루프를 방지하는 데 사용됨
- 예시 : ‘Via: 1.0 fred, 1.1 p.example.net’

Access-Control-Allow-Origin

- 해당 리소스에 접근할 수 있는 외부 도메인을 지정함
- CORS 정책의 핵심 헤더
- 역할 : 모든 사이트 접근 허용
- 예시 : ‘Access-Control-Allow-Origin: *’

HTTP 헤더

Permissions-Policy

- 웹사이트가 특정 브라우저 기능을 사용할 수 있는지, 또는 사용하지 못하게 할지를 제어함
- 역할 : 카메라는 허용, 마이크 및 위치 정보 기능 금지
- 예시 : ‘Permissions-Policy: camera=*, microphone=(), geolocation=()’

2025.09.19

감사합니다

network