

Http 03

2022.07.15

HTTP - HyperText Transfer Protocol 특징

- 클라이언트 서버 구조

클라이언트는 서버에 요청을 보내고 응답을 대기

클라이언트 - UI등 편의에 집중하고 서버는 내부 동작이나 속도등에 집중한다.

그러므로 서로 독립적으로 발전이 가능함.

- 무상태 프로토콜 (**stateless**)

stateful, stateless 차이

상태정보를 다 알려줘야 하는것 -> stateful

상태정보를 알려주지 않아도 클라이언트가 가지고 있는것 stateless

즉, 응답 서버를 마음껏 바꿀수 있다.

스케일 아웃 - 수평 확장에 유리하다.

-단점 : 서버에 무리가 갈 수 있다,

- 비 연결성

요청 응답이 끝나면 연결을 끊는다.

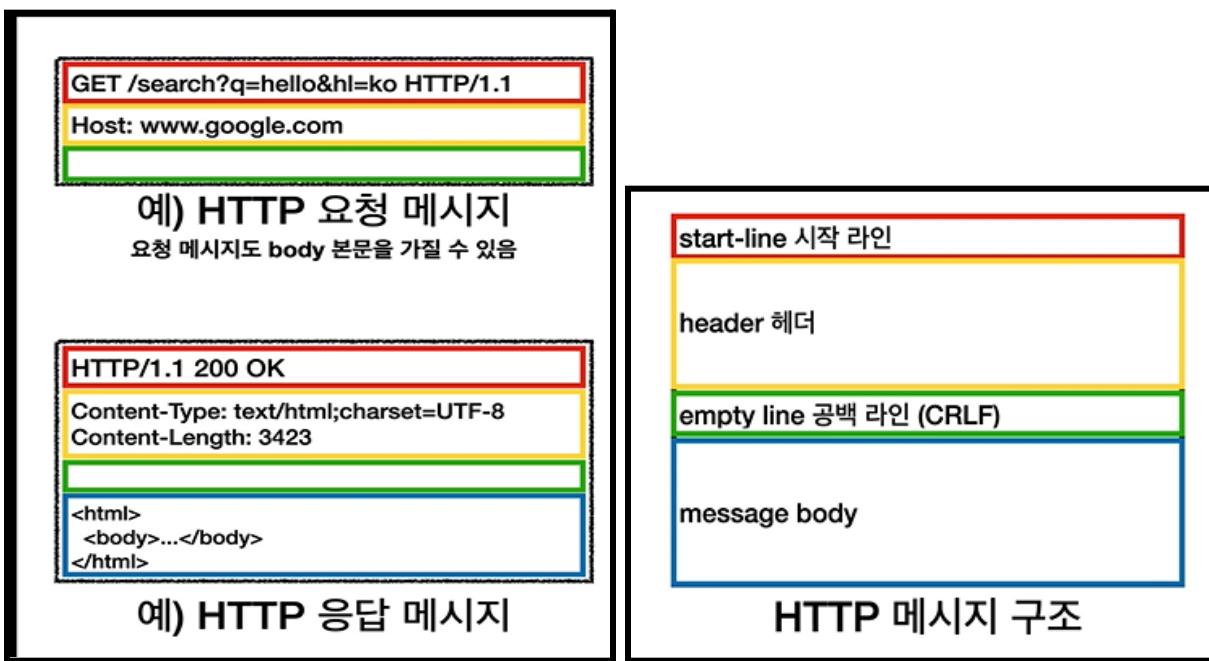
단점 - 연결을 계속 새로 맺어야함 (3 way handshake 추가)

지속연결로(Persistent Connections)로 문제 해결

[선착순 이벤트]

정적페이지 -> 스테이트리스로

HTTP 메시지



시작라인

요청 메시지

메소드 GET, POST, PUT, DELETE... / 요청 대상 / 요청내용 / 요청 버전

응답 메시지

status-line = 버전 status-code sp reason-phrase crlf

HTTP 헤더

HTTP 전송에 필요한 부가정보가 전부 들어있다.

필요한 메타데이터가 전부 들어있다.

HTTP 메시지 바디

- 실제 전송할 데이터

API URI 설계

리소스를 기준으로 식별!

회원 수정, 등록, 삭제 등에서 회원 자체가 리소스가 된다.

리소스와 행위를 분리 해야 한다!.

리소스 : 회원

행위 : 수정 등록 삭제

행위(메소드) 구분을 HTTP에서 진행

메서드 종류

HTTP 메서드 종류

주요 메서드

- GET: 리소스 조회
- POST: 요청 데이터 처리, 주로 등록에 사용
- PUT: 리소스를 대체, 해당 리소스가 없으면 생성
- PATCH: 리소스 부분 변경
- DELETE: 리소스 삭제

리소스 -> Representation

HTTP 메서드 종류

기타 메서드

- **HEAD:** GET과 동일하지만 메시지 부분을 제외하고, 상태 줄과 헤더만 반환
- **OPTIONS:** 대상 리소스에 대한 통신 가능 옵션(메서드)을 설명(주로 CORS에서 사용)
- **CONNECT:** 대상 자원으로 식별되는 서버에 대한 터널을 설정
- **TRACE:** 대상 리소스에 대한 경로를 따라 메시지 루프백 테스트를 수행

GET

- 리소스 조회 , 전달하고 싶은 데이터를 query를 통해 전달
- 메시지 바디를 전달할 수 있지만, 지원하지 않는 곳이 많아 권장 X

POST

- 데이터 줄테니까 서버한테 처리해달라고 해주는 것 .
- 신규 데이터 등록, 변경 등
- 대상 리소스가 리소스의 고유 한 의미 체계에 따라 요청에 포함 된 표현을 처리하도록 요청
- 요청 데이터를 처리하는데에도 사용 프로세스 상태가 변경될 때도 사용
- GET과의 차이 GET => 캐싱한다. POST => 캐싱이 어렵다.

PUT

- 리소스를 대체 있으면 대체, 없으면 덮어버린다.
- 클라이언트가 리소스를 식별 (클라이언트가 전체 위치를 알고 있다) 지정함
- 리소스 수정X 갈아치우는 것

PATCH

- 리소스를 수정 -부분 변경 가능

DELETE

- 리소스를 제거

HTTP 메서드의 속성

안전 (Safe)

- 호출해도 리소스를 변경하지 않는다.
- 계속 호출하면 로그로 장애가 발생하면? 무관, 우선 리소스 상태만 고려

멱등 (Idempotent)

- $f(f(x)) = f(x)$ 멱함수 호출시에도 호출 결과가 같다 .
- GET, PUT, DELETE 멱등
- POST 멱등 아님

활용

- 자동 복구 메커니즘
- 멱등한 요청은 중복요청할 때 사용 가능하다. 사용 가능한 판단 근거

캐시가능 (Cacheable)

- 응답 결과 리소스를 캐시해서 사용해도 되는가 ?
- GET HEAD POST PATCH 캐시 가능
- 실제로는 GET HEAD 정도만 캐시로 사용
- POST, PATCH는 본문 내용까지 캐시 키로 고려해야 하는데 구현이 힘듬
- 실무에서는 거의 GET만 캐시 사용