6장 - HTTP 헤더

HTTP 메시지 헤더

HTTP 프로토콜의 `리퀘스트와 리스폰스에는 반드시 메시지 헤더가 포함`

- 메시지 헤더란 ?
  - 리퀘스트나 리스폰스를 처리하기 위한 `정보가 포함`
  - 몇 가지의 요소로 구성
    - 리퀘스트의 HTTP 메시지
      - 메소드, URI, HTTP 버전, HTTP 헤더 필드 등으로 구성
    - 리스폰스의 HTTP 메시지
      - HTTP의 메시지와 HTTP 버전, 상태 코드(코드와 설명), HTTP 헤더 필드 등으로 구성

HTTP 헤더 필드

`요소 중 가장 다양한 정보`를 가지고 있다.

`리퀘스트와 리스폰스 양쪽 모두 존재`

`HTTP 메시지에 관한 정보`를 가지고 있다.

구조

- `필드 명`과 `필드 값`으로 구성
- ":" 으로 분류
  - ex) 헤더 필드 명 : 필드 값
    - Content-Type : text/html

헤더 필드는 용도에 따라 `4종류`로 분류

- **일반적 헤더 필드 (General Header Fields)**
  - 리퀘스트, 리스폰스 메시지 둘 다 사용되는 헤더
- **리퀘스트 헤더 필드 (Request Header Fields)**

    - 리퀘스트 메시지에 사용되는 헤더
    - 리퀘스트의 부가적 정보와 클아이언트의 정보, 리스폰스의 콘텐츠에 관한 우선 순위 등을 부가한다.
- **리스폰스 헤더 필드 (Response Header Fields)**
    - 리스폰스 메시지에 사용되는 헤더
    - 리스폰스의 정보와 서버의 정보, 클라이언트의 추가 정보 요구 등을 부가한다.
- **엔티티 헤더 필드 (Entity Header Fields)**
    - 리퀘스트와 리스폰스 메시지에 포함된 엔티티에 사용되는 헤더
    - 콘텐츠 갱신 시간 등의 엔티티에 관한 정보를 부가한다.
- **쿠키를 위한 헤더 필드**
- **그 이외의 헤더 필드**

End-to-end 헤더 & Hop-by-hop 헤더

`HTTP 헤더 필드`는 `캐시와 비캐시 프록시의 동작을 정의`하기 위해 `두 가지 카테고리`로 분류

- End-to-end 헤더
    - 이 카테고리에 분류된 헤더는 `리퀘스트나 리스폰스의 최종 수신자에게 전송`된다.
    - `캐시에서 구축된 리스폰스 중 보존`되야 하고, `다시 전송되지 않으면 안되도록`되어 있다.
- Hop-by-hop 헤더
    - 이 카테고리에 분류된 헤더는 `한 번 전송에 대해서만 유효`하고 캐시와 프록시에 의해서 전송되지 않는 것도 있다.
    - HTTP/1.1과 그 이후에서 사용되는 헤더는 Connection 헤더 필드에 열거해야 한다.

HTTP/1.1 일반 헤더 필드

Cache-Control

일반 헤더 필드는 리퀘스트 메시지와 리스폰스 메시지 양쪽에서 사용되는 헤더

- Cache-Control 헤더
    - 디렉티브로 불리는 명령을 사용하여 캐싱 동작을 지정한다.

    - 지정한 디렉티브에는 파라미터가 있는 것과 없는 것도 있다.
    - 여러개의 디렉티브를 지정하는 경우에는 콤마 “,”로 구분
        - ex) Cache-Control : private, max-age=0, no-cache
- Cache-Control 디렉티브 일람
    - 사용 가능한 디렉티브를 리퀘스트와 리스폰스로 나눠서 나타낸다.

캐시가 가능한지 여부를 나타내는 디렉티브

- public 디렉티브
    - ex) Cache-control : public
    - Public 디렉티브가 사용되는 경우, 다른 유저에게도 돌려줄 수 있는 캐시를 해도 좋다는 것을 명시적으로 나타낸다.
- private 디렉티브
    - Cache-Control : private
    - private 디렉티브가 사용되는 경우, 리스폰스는 특정 유저만을 대상으로 하고 있다는 것을 나타낸다.
    - public 디렉티브와 기능이 반대
- no-cache 디렉티브
    - Cache-Control : no-cache
    - 캐시로부터 오래된 리소스가 반휜되는 것을 막기 위해 사용된다.
    - 클라이언트의 리퀘스트로 사용된 경우
        - 캐시된 리스폰스를 클라이언트가 받아 들이지 않음을 나타낸다.
        - 즉, 중간 캐시 서버가 오리진 서버까지 리퀘스트를 전송해야 한다.
    - 서버의 리스폰스에 사용된 경우
        - 캐시 서버는 리소스를 저장할 수 없다.
        - 오리진 서버는 캐시 서버가 이후의 리퀘스트에서 리소스의 유효성을 재확인하지 않고는 그 리스폰스를 사용하지 못하도록 한다.
        - Cache-Control : no-cache=Location
        - 서버의 리스폰스로 no-cache의 필드 값에 헤더 필드 명이 지정된 경우에는 이 지정된 헤더 필드만 캐시할 수 없다.

        - 즉, 지정된 헤더 필드 이외에는 캐시하는 것이 가능하다.
        - 이 파라미터는 리스폰스 디렉티브만 사용할 수 있다.

캐시로 보존 가능한 것을 제어하는 디렉티브

- no-store 디렉티브
    - Cache-Control : no-store
    - no-store 디렉티브가 사용된 경우
        - 리퀘스트(그와 대응되는 리스폰스) 혹은 리스폰스에 기밀 정보가 포함되어 있음을 나타낸다.
        - 그렇기 때문에 로컬 스토리지에 보존해서는 안되도록 지정한다.

캐시 기한이나 검증을 지정하는 디렉티브

- s-maxage 디렉티브
    - Cache-Control : s-maxage=604800 (단위 : 초)
    - 기능은 max-age 디렉티브와 동일
    - 다른점 : 여러 유저가 이용할 수 있는 공유 캐시 서버에만 적용
    - 즉, 같은 유저에 반복해서 리스폰스를 반환하는 캐시 서버는 무효한 디렉티브이다.
    - s-maxage 디렉티브가 사용되는 경우
        - Expires 헤더 필드와 max-age 디렉티브는 무시된다.
- max-age 디렉티브
    - Cache-Control : max-age=604800 (단위 : 초)
    - 클라이언트의 리퀘스트로 max-age 디렉티브가 사용된 경우
        - 지정되었던 값보다 새로운 경우에는 캐시되었던 리소스를 받아드릴 수 있다.
        - 지정한 값이 0이면 캐시 서버는 리퀘스트를 항상 오리진 서버에 넘길 필요가 있다.
    - 서버의 리스폰스에서 max-age 디렉티브가 사용된 경우
        - 캐시 서버가 유효성의 재확인을 하지 않고 리소스를 캐시에 보존해 두는 최대 시간을 나타낸다.
    - HTTP/1.1 캐시 서버는 동시에 Expires 헤더 필드가 달린 경우에는 max-age 디렉티브의 지정을 우선하고 Expires 헤더 필드를 무시한다.

    - HTTP/1.0 캐시 서버는 반대로 max-age 디렉티브가 무시된다.
- min-fresh 디렉티브
    - Cache-Control: min-fresh=60 (단위 : 초)
    - min-fresh 디렉티브가 사용되는 경우
        - 캐시된 리소스가 적어도 지정된 시간은 최신 상태의 것을 반환하도록 캐시 서버에 요구
        - ex)
            - 60초로 지정되어 있는 경우에는 60초 이내에 유효 기한이 끝나는 리소스를 리스폰스로 반환하면 안된다.
- max-stale 디렉티브
    - Cache-Control: max-stale=3600 (단위 : 초)
    - max-stale가 사용되는 경우
        - 캐시된 리소스의 유효 기한이 끝났더라도 받아들일 수 있음을 나타냄
    - 디렉티브에 값이 지정되어 있지 않는 경우
        - 클라이언트는 아무리 시간이 경과했더라도 리스폰스를 받아 들인다.
    - 디렉티브 값이 지정되어 있는 경우
        - 유효 기한이 지난 후로부터 지정 시간 내라면 받아 들인다는 뜻을 서버에 전달
- only-if-cached 디렉티브
    - Cache-Control: only-if-cached
    - 사용되는 경우
        - 클라이언트는 캐시 서버에 대해서 목적한 리소스가 로컬 캐시에 있는 경우만 리스폰스를 반환하도록 요구

      → 즉, 캐시 서버에서 리스폰스의 리로드와 유효성을 재확인하지 않도록 요구

      → 캐시 서버가 로컬 캐시로부터 응답할 수 없는 경우에는 “504 Gateway Timeout” 상태를 반환
- must-revalidate 디렉티브
    - Cache-Control: must-revalidate
    - 사용되는 경우

        - 리스폰스의 캐시가 현재도 유효한지 아닌지의 여부를 오리진 서버에 조회를 요구
        - 리퀘스트에서 max-stale 디렉티브를 사용하고 있더라도 무시
    - 프록시가 오리진 서버에 도달할 수 없고, 리소스를 다시 요구할 수 없는 경우에는 캐시는 클라이언트에 504(Gateway Timeout)를 반환

Connection

커넥션 헤더 필드는 두 가지 역할

- 프록시에 더 이상 전송하지 않는 헤더 필드를 지정
    - Connection : 더 이상 전송하지 않는 헤더 필드 명
- 지속적 접속 관리
    - Connection : Close

Date

Date 헤더 필드는 HTTP 메시지를 생성한 날짜를 나타낸다.

- Date: Tue, 03 Jul 2012 04:40:59 GMT
- Date: Tue, 03-Jul-12 04:40:59 GMT
- Date: Tue Jul 03 04:40:59 2012

Pragma

Pragma 헤더 필드는 HTTP/1.1 보다 오래된 버전의 흔적으로 HTTP/1.0 와의 `후방 호환성만을 위해서 정의` 되어 있는 헤더 필드이다.

지정할 수 있는 형식은 다음과 같이 1개뿐이다.

- Pragma: no-cache
- 일반 헤더 필드이지만 `클라이언트의 리퀘스트에서만 사용`
- 클라이언트는 캐시된 리소스의 리스폰스를 원하지 않음을 `모든 중간 서버에 알리기 위해 사용`

Trailer

메시지 바디의 뒤에 기술되어 있는 헤더 필드를 미리 전달할 수 있다.

- HTTP/1.1에 구현되어 있는 `청크 전송 인코딩을 사용하고 있는 경우에 사용 가능`

Transfer-Encoding

메시지 바디의 전송 코딩 형식을 지정하는 경우에 사용

HTTP/1.1에서 전송 코딩 형식으로 `청크 전송`만이 정의

Upgrade

HTTP 및 다른 프로토콜의 새로운 버전이 통신에 이용되는 경우에 사용된다.

지정하는 대상이 전혀 다른 통신 프로토콜이라고 하더라도 문제가 없다.

Via

클라이언트와 서버 간의 리퀘스트 혹은 리스폰스 메시지의 경로를 알기 위해서 사용된다.

- 프록시 혹은 게이트웨이는 자신의 서버 정보를 Via 헤더 필드에 추가한 뒤에 메세지를 전송
- traceroute와 메일의 Received 헤더의 기능과 유사
- 전송된 메시지의 추적과 리퀘스트 루프의 회피 등에 사용되기 때문에 프록시를 경유하는 경우에는 반드시 부가할 필요가 있다.

Warning

HTTP/1.0 리스폰스 헤더(Retry-After)가 HTTP/1.1에서 변경된 것으로, 리스폰스에 관한 추가 정보를 전달한다.

- Warning: [경고 코드][경고한 호스트 : 포트 번호]"[경고문]"([날짜])

리퀘스트 헤더 필드

리퀘스트 헤더 필드는 클라이언트 측에서 서버 측으로 송신된 리퀘스트 메시지에 사용되는 헤더

리퀘스트의 부가 정보와 클라이언트의 정보, 리스폰스의 콘텐츠에 관한 우선 순위 등으로 추가

Accept

유저 에이전트에 처리할 수 있는 미디어 타입과 미디어 타입의 `상대적인 우선 순위를 전달`하기 위해서 사용

미디어 타입의 지정은 “타입/서브 타입”으로서 한 번에 여러 번 설정할 수도 있다.

- 미디어 타입
  - 텍스트 파일
    - text/html, text/css, …
  - 이미지 파일
    - jpeg, gif, png, …
  - 동영상 파일
    - mpeg, quicktime, …
  - 애플리케이션용 바이너리 파일
    - octet-stream, zip, …

Accept-Charset

유저 에이전트에서 처리할 수 있는 문자셋으로, 문자셋의 상대적인 우선 순위를 전달하기 위해서 사용

- 문자셋은 한번에 여러 개를 지정할 수 있다.

Accept-Encoding

유저 에이전트가 처리할 수 있는 콘텐츠 코딩과 콘텐츠 코딩의 상대적인 우선 순위를 전달하기 위해서 사용

콘텐츠 코딩의 저징은 한 번에 여러 개를 지정할 수 있다.

- gzip
- compress
- deflate
- identity

Accept-Language

유저 에이전트가 처리할 수 있는 자연어의 세트와 자연어 세트의 상대적인 우선 순위를 전달하기 위해 사용

- ex) 한국어 리소스가 있는 경우 한국어, 없을 경우 영어 리소스로 리스폰스를 받고 싶다는 것

Authorization

유저 에이전트의 인증 정보(크리덴셜 값)을 전달하기 위해서 사용

통상, 서버에 인증을 받으려 하는 유저 에이전트는 상태 코드 401 리스폰스의 뒤에 리퀘스트에 Authorization 헤더 필드를 포함한다.

공유 캐시가 Authorization 헤더 필드를 포함하는 리퀘스트를 받은 경우에는 조금은 다른 동작을 한다.

Expect

클라이언트가 서버에 특정 동작 요구를 전달한다.

기대하고 있는 요구에 서버가 응답하지 못해서 에러가 발생하는 경우에는 상태 코드 417 Expectation Failed를 반환한다.

- Expect: 100-continue

From

유저 에이전트를 사용하고 있는 유저의 메일 주소를 전달한다.

- 기본적으로는 검색 엔진 등의 에이전트 책임자에게 연락처 메일 주소를 나타내는 목적으로 사용

Host

리퀘스트한 리소스의 인터넷 호스트와 포트 번호를 전달한다.

Host 헤더는 HTTP/1.1에서 유일한 필수 헤더 필드

- 존재하는 이유
    - 1대의 서버에서 복수의 도메인을 할당할 수 있는 가상 호스트의 구조와 매우 깊은 관련
    - 리퀘스트가 서버에 오면 호스트 명을 IP 주소로 해결해 리퀘스트가 처리된다.
    - 이 때 같은 IP 주소로 복수의 도메인이 적용되어 있다고 한다면 어느 도메인에 대한 리퀘스트인지 확인이 불가
    - Host 헤더 필드에 리퀘스트를 받을 호스트 명을 명확하게 해 둘 필요가 있다.
- Host: www.domain.com

If-XXX 라는 서식의 리퀘스트 헤더 필드는 조건부 리퀘스트
조건부 리퀘스트를 받은 서버는 지정된 조건에 맞는 경우에만 리퀘스트를 받는다.

If-Match

조건부 리퀘스트의 하나로 서버 상의 리소스를 특정하기 위해서 엔티티 태그(ETag) 값을 전달

이 때 서버는 약한 ETag 값을 사용할 수 없다.

서버는 If-Match의 필드 값과 리소스의 ETag 값이 일치하는 경우에만 리퀘스트를 받아들일 수 있다.

일치하지 않으면 412 precondition Failed 리스폰스를 반환

If-Modified-Since

조건부 리퀘스트의 하나로 리소스가 갱신 날짜가 필드 값보다 새롭지 않다면 리퀘스트를 받아들이겠다는 뜻을 전달

If-None-Match

If-Match 헤더 필드와는 반대로 동작

지정된 엔티티 태그(ETag) 값이 지정된 리소스의 ETag 값과 일치하지 않으면 리퀘스트를 받아들이겠다는 뜻으로 전달.

If-Range

지정한 필드값(ETag)과 지정한 리소스의 ETag 값 혹은 날짜가 일치하면 Range 리퀘스트로서 처리하고 싶다는 것을 전달한다.

If-Unmodified-Since

지정된 리소스가 필드 값에 지정된 날짜 이후에 갱신되어 있지 않는 경우에만 리퀘스트를 받아들이도록 전달한다.

지정된 날짜 이후에 갱신된 경우에는 상태 코드 412 Precondition Failed 리스폰스를 반환

Max-Forwards

TRACE or OPTIONS 메소드에 의한 리퀘스트를 할 때에 전송해도 좋은 서버 수의 최대치를 10진수 정수로 지정한다.

서버는 다음 서버에 리퀘스트를 전송할 때는 Max-Forwards 값에서 1을 빼서 다시 세트한다.

Max-Forwards 값이 0인 리퀘스트를 받는 경우에는 전송하지 않고 리스폰스를 반환할 필요가 있다.

Proxy-Authorization

프록시 서버에서의 인증 요구를 받아들인 때에 인증에 필요한 클라이언트의 정보를 전달한다.

HTTP 액세스 인증과 비슷하나 다른 점은 클라이언트와 프록시 사이에 인증이 이루어진다는 것이다.

클라이언트와 서버의 경우, Authorization 헤더 필드와 같은 역할을 한다.

Range

리소스의 일부분만 취득하는 Range 리퀘스트를 할 때 지정 범위를 전달한다.

Referer

리퀘스트가 발생한 본래 리소스의 URI를 전달한다.

TE

리스폰스로 받을 수 있는 전송 코딩의 형식과 상대적인 우선 순위를 전달한다.

Accept-Encoding 헤더 필드와 유사하지만 여기선 전송 코딩에 적용된다.

User-Agent

리퀘스트를 생성한 브라우저와 유저 에이전트의 이름 등을 전달하기 위한 필드이다.

리스폰스 헤더 필드

서버 측으로부터 클라이언트 측으로 송신되는 리스폰스 메시지에 적용된 헤더

리스폰스의 부가 정보나 서버의 정보, 클라이언트에 부가 정보 요구 등을 나타낸다.

Accept-Ranges

서버가 리소스의 일부분만 지정해서 취득할 수 있는 Range 리퀘스트를 접수할 수 있는지 여부를 전달

Age

얼마나 오래 전에 오리진 서버에서 리스폰스가 생성되었는지를 전달

필드값의 단위는 초

ETag

앤티티 태그

일의적으로 리소스를 특정하기 위한 문자열을 전달한다.

서브는 리소스마다 ETag값을 할당

강력한 ETag 값과 약한 ETag 값

강한 ETag 값

- ETag: “Usagi-1234”
- 엔티티가 아주 조금 다르더라도 반드시 값은 변화한다.

약한 ETag 값

- ETag: W/”usagi-1234”
- 리소스가 같다는 것만을 나타낸다.
- 의미가 다른 리소스로 그 차이가 있는 경우에만 ETag 값이 변화한다.
- 값의 앞부분에 W/가 붙는다.

Location

리스폰스의 수신자에 대해서 Request-URI 이외의 리소스 액세스를 유도하는 경우에 사용

Porxy-Authenticate

프록시 서버에서의 인증 요구를 클라이언트에 전달한다.

클라이언트와 프록시 사이에서 인증이 이루어진다.

Retry-After

클라이언트가 일정 시간 후에 리퀘스트를 다시 시행해야 하는지를 전달한다.

주로 상태 코드 503 Service Unavilable 리스폰스를 사용한다.

Server

서버에 설치되어 있는 HTTP 서버의 소프트웨어를 전달한다.

Vary

캐시를 컨트롤하기 위해서 사용한다.

오리진 서버가 프록시 서버에 로컬 캐시를 사용하는 방법에 대한 지시를 전달한다.

오리진 서버로부터 Vary에 지정되었던 리스폰스를 받아들인 프록시 서버는 이후 캐시된 때의 리퀘스트와 같은 Vary에 지정되어 있는 헤더 필드를 가진 리퀘스트에 대해서만 캐시를 반환할 수 있다.

WWW-Authenticate

HTTP 액세스 인증에 사용되는데 Reuqest-URI에 지정했던 리소스에 저굥ㅇ할 수 있는 인증 스키마와 파라미터를 나타내는 Challenge를 전달한다.

엔티티 헤더 필드

리퀘스트 메시지와 리스폰스 메시지에 포함된 엔티티에 사용되는 헤더

콘텐츠의 갱신 시간 같은 엔티티에 관한 정보를 포함

Allow

Reuqest-URI에 지정된 리소스가 제공하는 메소드의 일람을 전달

서버가 받을 수 없는 메소드를 수신한 경우에는 405 리스폰스를 반화

Content-Encoding

엔티티 바디에 대해서 실시한 콘텐츠 코딩 형식을 전달

콘텐츠 코딩은 엔티티의 정보가 누락되지 않도록 압축할 것을 지시

Content-Language

엔티티 바디에 사용된 자연어를 전달

Content-Length

엔티티 바디의 크기를 전달

Content-Location

메시지 바디에 대응하는 URI를 전달

Content-MD5

메시지 바디가 변경되지 않고 도착했는지 확인하기 위해 MD5 알고리즘에 의해서 생성된 값을 전달

Content-Range

범위를 지정해서 일부분만을 리퀘스트 하는 Range 리퀘스트에 대해서 리스폰스를 할 때에 사용

Content-Type

엔티티 바디에 포함되는 오브젝트의 미디어 타입을 전달

Expires

리소스의 유효 기한 날짜를 전달

캐시 서버가 Expires 헤더 필드를 포함한 리소스를 수신한 경우 필드 값으로 지정된 날짜까지 리스폰스의 복사본을 유지하고 리퀘스트에는 캐시로 응답한다.

Last-Modified

리소스가 마지막으로 갱신되었던 날짜 정보를 전달

쿠키를 위한 헤더 필드

- 서버와 클라이언트 간의 상태를 관리하는 쿠키는 HTTP/1.1의 사양인 RFC2616에 포함된 것은 아니지만 웹 사이트에서 널리 사용
- 쿠키는 유저 식별과 상태 관리에 사용되고 있는 기능
- 웹 브라우저 경유로 유저의 컴퓨터 상에 일시적으로 데이터를 기록해 두고, 다음에 그 유저가 웹 사이트에 액세스 해 왔을 때 지난 번에 발행한 쿠키를 송신 받을 수 있다.
- 쿠키가 호출되었을 때는 쿠키의 유효 기한과 송신지의 도메인, 경로, 프로토콜 등을 체크하는 것이 가능하기 때문에, 적절하게 발행된 쿠키는 다른 웹 사이트와 공격자의 공격

에 의해 데이터가 도난 당하는 일은 없다.

1. 넷스케이프사에 의한 사양
2. RFC2109
3. FRC2965
4. FRC6265

Set-Cookie

서버가 클라이언트에 대해서 상태 관리를 시작할 때 다양한 정보를 전달

1. Expires 속성
2. Path 속성
3. Domain 속성
4. Secure 속성
5. HttpOnly 속성

Cookie

클라이언트가 HTTP의 상태 관리 지원을 원할 때 서버로부터 수신한 쿠키를 이후의 리퀘스트에 포함해서 전달
