3장

HTTP 메시지란

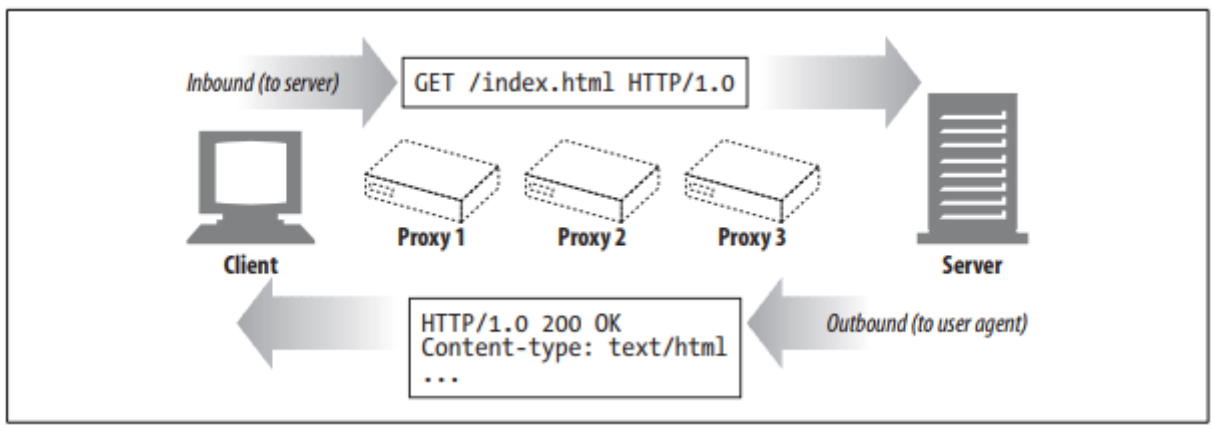
- HTTP 애플리케이션 간에 주고받는 데이터의 블록

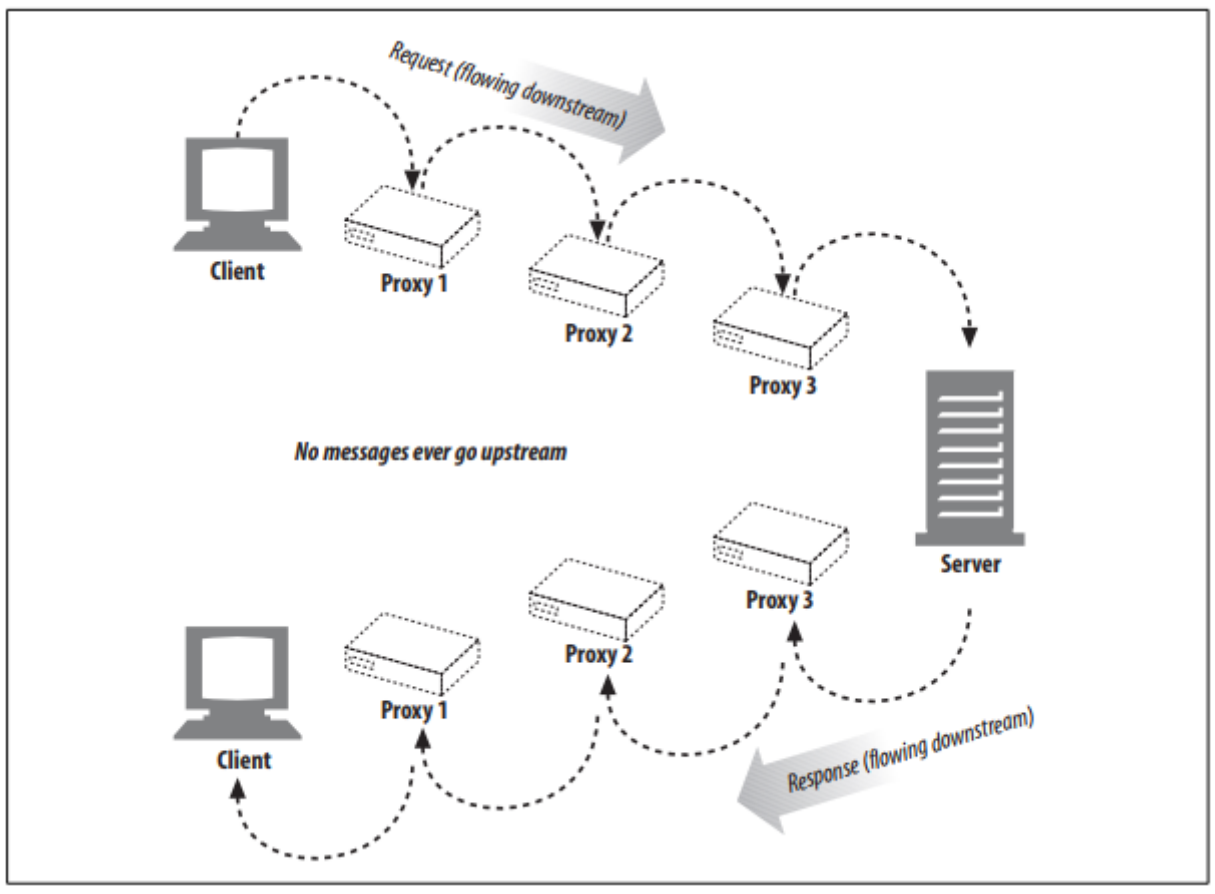
  요청 메시지, 응답 메시지로 구분

HTTP 메시지의 흐름

- 클라이언트, 서버, 프록시 사이를 흐름
- 인바운드, 아웃바운드, 업스트림, 다운스트림은 메시지의 방향 표현

1. 인바운드는 사용자에서 서버
2. 아웃바운드는 서버에서 사용자
3. 프락시 1은 프락시 3의 업스트림
4. 프락시 3은 프락시 1의 다운스트림
5. 모든 데이터는 다운스트림으로 흐름





HTTP 메시지 구성

- 시작줄, 헤더, 엔터티 본문으로 구성

  1. 시작줄

  - 요청은 <메서드><요청URL><버전>

    - 응답은 <버전><상태코드><사유구절>

메서드(GPPPD THO)	설명	본문	안전한 메서드
GET	리소스 조회	x	o
POST	리소스 등록	o	x
PUT	리소스 대체, 없으면 생성	o	x
PATCH	리소스 일부 변경	o	x
DELETE	리소스 삭제	x	x
TRACE	서버에서부터 루프백 테스트 보낸 데이터가 망가졌는지, 수정됐는지 확인	x	x
HEAD	리소스의 상태줄과 헤더만 조회	x	o
OPTIONS(=preflight)	웹 서버에서 지원하는 요청방식 확인	x	x

※ 루프백은 원래의 장치로 돌아가는 것

※ 안전한 메서드는 HTTP의 결과로 서버에 어떤 작용도 없음을 의미. 구매 버튼을 눌렀을 때, POST 요청이 발생하고, 신용카드로 대금이 청구되는 경우는 '작용'이 존재한다.

안전한 메서드가 서버에 작용을 유발하지 않는다는 보장은 없다(개발자에게 달려있음). 안전한 메서드의 목적은, 안전하지 않은 메서드가 사용될 때 그 사실을 알려주는 HTTP 애플리케이션을 만들 수 있도록 하는 것에 있다. 가령 앞선 예처럼 신용카드가 결제되는 상황에 대한 알림 등을 의미한다.

2. 헤더
    - 요청과 응답 메시지에 추가 정보
    - 이름:값 쌍의 리스트
    - 없는 경우, CRLF만 존재

3. 엔터티 본문
    - 텍스트, 이진 데이터 포함
    - 없어도 됨

- 헤더와 바디를 구분하기 위해 CRLF 사용
    - carriage return은 커서를 맨 앞으로,
    - line feed는 커서를 한 줄 아래로,
    - EOL(End-Of-Line)은 CRLF와 동일

HTTP 버전

- HTTP 버전 1.1과 대화하는 HTTP 버전 1.2는, HTTP 버전 1.2의 기능을 사용할 수 없음.
- 버전 번호는 분리된 숫자. 2.22는 2.3보다 크다. (22 > 3)

헤더 분류

1. 일반 헤더
    a. 일반 헤더: connection, date 등
    b. 캐시 헤더: cache-control, pragma
2. 요청 헤더
    a. 요청 헤더: User-Agent, Client-ip
    b. Accept 관련 헤더: Accept
    c. 조건부 요청 헤더: if-Modified-Since, if-Unmodified-Since, if-Match

    d. 요청 보안 헤더: Authrization, Cookie
    e. 프락시 요청 헤더
3. 응답 헤더
    a. 응답 헤더: Warning, Server
    b. 협상 헤더
    c. 보안 헤더
4. 엔터티 헤더
    a. 엔터티 헤더
    b. 콘텐츠 헤더: Content-Length, Content-Type
    c. 엔터티 캐싱 헤더: Etag, Expired, Last-Modified
5. 확장 헤더

1. 일반 헤더

- 요청과 응답 양쪽에 모두 나타남

헤더	설명
Connection	클라이언트와 서버가 요청/응답 연결에 대한 옵션을 정할 수 있게 해준다.
Date[a]	메시지가 언제 만들어졌는지에 대한 날짜와 시간을 제공한다.
MIME-Version	발송자가 사용한 MIME의 버전을 알려준다.
Trailer chunked transfer	인코딩으로 인코딩된 메시지의 끝 부분에 위치한 헤더들의 목록을 나열한다.[b]
Transfer-Encoding	수신자에게 안전한 전송을 위해 메시지에 어떤 인코딩이 적용되었는지 말해준다.
Upgrade	발송자가 '업그레이드'하길 원하는 새 버전이나 프로토콜을 알려준다.
Via	이 메시지가 어떤 중개자(프락시, 게이트웨이)를 거쳐 왔는지 보여준다.

a 부록 C는 Date 헤더에서 사용할 수 있는 날짜 포맷들을 나열하고 있다.
b 청크(chunk) 전송 코딩은 15장의 "청크와 지속 커넥션"에서 더 자세히 다룬다.

- HTTP 1.0부터는 캐시 헤더 도입

헤더	설명
Cache-Control	메시지와 함께 캐시 지시자를 전달하기 위해 사용한다.
Pragma[a]	메시지와 함께 지시자를 전달하는 또 다른 방법. 캐시에 국한되지 않는다.

a Pragma는 엄밀히 말하면 요청 헤더이며, 응답에 사용하기 위해 정의되지 않았다. 응답 헤더로 사용하는 흔한 오용 때문에 많은 클라이언트와 프락시는 Pragma를 응답 헤더로 해석하지만, 사실 엄밀한 의미는 잘 정의되어 있지 않다. 어쨌든 Pragma는 Cache-Control로 인해 더 이상 사용되지 않을 예정(deprecated)이다.

2. 요청 헤더

헤더	설명
Client-IP[a]	클라이언트가 실행된 컴퓨터의 IP를 제공한다.
From	클라이언트 사용자의 메일 주소를 제공한다.[b]
Host	요청의 대상이 되는 서버의 호스트 명과 포트를 준다.
Referer	현재의 요청 URI가 들어있었던 문서의 URL을 제공한다.
UA-Color	클라이언트 기기 디스플레이의 색상 능력에 대한 정보를 제공한다.
UA-CPU[c]	클라이언트 CPU의 종류나 제조사를 알려준다.
UA-Disp	클라이언트의 디스플레이(화면) 능력에 대한 정보를 제공한다.
UA-OS	클라이언트 기기에서 동작 중인 운영체제의 이름과 버전을 알려준다.
UA-Pixels	클라이언트 기기 디스플레이에 대한 픽셀 정보를 제공한다.
User-Agent	요청을 보낸 애플리케이션의 이름을 서버에게 말해준다.

a Client-IP와 UA-* 헤더는 RFC 2616에 정의되어 있지 않지만 많은 HTTP 클라이언트 애플리케이션에서 구현되어 있다.
b RFC 822 이메일 주소 포맷.
c 몇몇 클라이언트에서 구현되어 있기는 하지만, UA-* 헤더는 해로운 것일 수도 있다. 콘텐츠(특히 HTML)는 특정 클라이언트 설정에 맞추어져서는 안 된다.

- Accept 관련 헤더

헤더	설명
Accept	서버에게 서버가 보내도 되는 미디어 종류를 말해준다.
Accept-Charset	서버에게 서버가 보내도 되는 문자집합을 말해준다.
Accept-Encoding	서버에게 서버가 보내도 되는 인코딩을 말해준다.
Accept-Language	서버에게 서버가 보내도 되는 언어를 말해준다.
TE[a]	서버에게 서버가 보내도 되는 확장 전송 코딩을 말해준다.

a 자세한 것은 15장의 "Transfer-Encoding 헤더"를 보라.

- 조건부 요청 헤더

헤더	설명
Expect	클라이언트가 요청에 필요한 서버의 행동을 열거할 수 있게 해준다.
If-Match	문서의 엔터티 태그가 주어진 엔터티 태그와 일치하는 경우에만 문서를 가져온다.[a]
If-Modified-Since	주어진 날짜 이후에 리소스가 변경되지 않았다면 요청을 제한한다.
If-None-Match	문서의 엔터티 태그가 주어진 엔터티 태그와 일치하지 않는 경우에만 문서를 가져온다.
If-Range	문서의 특정 범위에 대한 요청을 할 수 있게 해준다.
If-Unmodified-Since	주어진 날짜 이후에 리소스가 변경되었다면 요청을 제한한다.
Range	서버가 범위 요청을 지원한다면, 리소스에 대한 특정 범위를 요청한다.[b]

a 엔터티 태그에 대해 더 자세한 것은 7장을 보라. 태그는 기본적으로 리소스의 버전에 대한 식별자이다.
b Range 헤더에 대해 더 자세한 것은 15장의 "범위 요청"을 보라.

- 요청 보안 헤더

헤더	요청
Authorization	클라이언트가 서버에게 제공하는 인증 그 자체에 대한 정보를 담고 있다.
Cookie	클라이언트가 서버에게 토큰을 전달할 때 사용한다. 진짜 보안 헤더는 아니지만, 보안에 영향을 줄 수 있다는 것은 확실하다.[a]
Cookie2	요청자가 지원하는 쿠키의 버전을 알려줄 때 사용한다. 11장의 "Version 1(RFC 2965) 쿠키"를 보라.

a Cookie 헤더는 RFC 2616에 정의되어 있지 않다. 이것에 대해서는 11장에서 자세히 다룰 것이다.

- 프락시 요청 헤더

헤더	설명
Max-Forwards	요청이 원 서버로 향하는 과정에서 다른 프락시나 게이트웨이로 전달될 수 있는 최대 횟수. TRACE 메서드와 함께 사용된다.[a]
Proxy-Authorization	Authorization과 같으나 프락시에서 인증을 할 때 쓰인다.
Proxy-Connection	Connection과 같으나 프락시에서 연결을 맺을 때 쓰인다.

a 6장의 "Max-Forwards"를 보라.

3. 응답 헤더

- Server: Tiki-Hut/1.0 (클라이언트에게 Tiki-Hut 서버 1.0버전과 대화하고 있음을 알려줌)

헤더	설명
Age	응답이 얼마나 오래되었는지[a]
Public[b]	서버가 특정 리소스에 대해 지원하는 요청 메서드의 목록
Retry-After	현재 리소스가 사용 불가능한 상태일 때, 언제 가능해지는지 날짜 혹은 시각
Server	서버 애플리케이션의 이름과 버전
Title[c]	HTML 문서에서 주어진 것과 같은 제목
Warning	사유 구절에 있는 것보다 더 자세한 경고 메시지

a 응답이 중개자(아마 프락시 캐시)를 통해서 왔음을 암시한다.
b Public 헤더는 RFC 2068에서 정의되었지만 이후의 HTTP 명세인 RFC 2616에서 빠졌으며, 최신 HTTP 명세인 RFC 7231에서도 정의되어 있지 않다.
c Title 헤더는 RFC 2616에는 정의되어 있지 않다. HTTP/1.0 초안을 보라(http://www.w3.org/Protocols/HTTP/HTTP2.html).

- 협상 헤더
  - 서버에 프랑스어와 독일어로 번역한 HTML 문서가 있는 경우와 같이, 여러 가지 표현이 가능한 상황이라면, HTTP/1.1은 서버와 클라이언트가 어떤 표현을 택할 것인가에 대한 협상을 할 수 있도록 지원. 17장 '협상'에서 추가 설명

헤더	설명
Accept-Ranges	서버가 자원에 대해 받아들일 수 있는 범위의 형태
Vary	서버가 확인해 보아야 하고 그렇기 때문에 응답에 영향을 줄 수 있는 헤더들의 목록. 예) 서버가 클라이언트에게 보내줄 리소스의 가장 적절한 버전을 선택하기 위해 살펴보아야 하는 헤더들의 목록.

- 응답 보안 헤더

헤더	설명
Proxy-Authenticate	프락시에서 클라이언트로 보낸 인증요구의 목록
Set-Cookie	진짜 보안 헤더는 아니지만, 보안에 영향은 줄 수 있다. 서버가 클라이언트를 인증할 수 있도록 클라이언트 측에 토큰을 설정하기 위해 사용한다.[a]
Set-Cookie2	Set-Cookie와 비슷하게 RFC 2965로 정의된 쿠키. 11장의 "Version 1(RFC 2965) 쿠키"를 보라.
WWW-Authenticate	서버에서 클라이언트로 보낸 인증요구의 목록

a Set-Cookie와 Set-Cookie2는 확장 헤더이며, 11장에서 다룰 것이다.

4. Entity 헤더

- 엔터티 본문의 정보에 대한 헤더
- 요청과 응답 양쪽에 모두 나타남
- 일반적으로 자신이 다루고 있는 것이 무엇인지 말해준다.

헤더	설명
Allow	이 엔터티에 대해 수행될 수 있는 요청 메서드들을 나열한다.
Location	클라이언트에게 엔터티가 실제로 어디에 위치하고 있는지 말해준다. 수신자에게 리소스에 대한 (아마도 새로운) 위치(URL)를 알려줄 때 사용한다.

- 콘텐츠 헤더

헤더	설명
Content-Base[a]	본문에서 사용된 상대 URL을 계산하기 위한 기저 URL
Content-Encoding	본문에 적용된 어떤 인코딩
Content-Language	본문을 이해하는데 가장 적절한 자연어
Content-Length	본문의 길이나 크기
Content-Location	리소스가 실제로 어디에 위치하는지
Content-MD5	본문의 MD5 체크섬(checksum)
Content-Range	전체 리소스에서 이 엔터티가 해당하는 범위를 바이트 단위로 표현
Content-Type	이 본문이 어떤 종류의 객체인지

a Content-Base 헤더는 RFC 2616에 정의되어 있지 않다.

- 엔터티 캐싱 헤더

헤더	설명
ETag[a]	이 엔터티에 대한 엔터티 태그
Expires	이 엔터티가 더 이상 유효하지 않아 원본을 다시 받아와야 하는 일시
Last-Modified	가장 최근 이 엔터티가 변경된 일시

a 엔터티 태그는 기본적으로 리소스의 특정 버전에 대한 식별자다.

5. 확장 헤더

- 명세에 정의되지 않은 헤더
- HTTP 프로그램은 의미를 몰라도 전달해야 함

헤더를 여러 줄로 나누기

- 헤더 줄을 쪼개려면 최소 하나의 스페이스 혹은 탭 필요

```
HTTP/1.0 200 OK
Content-Type: image/gif
Content-Length: 8572
Server: Test Server
    Version 1.0
```

상태 코드

번호 범위	의미
100 ~	정보
200 ~	성공

번호 범위	의미
300 ~	리다이렉트
400 ~	클라이언트 에러
500 ~ 599	서버 에러

100 ~ 199

- 100 Continue
  - 엔터티 본문을 전송하기 전에, 서버가 받을것인지 확인
  - 서버가 사용할 수 없는 큰 엔터티를 보내지 않으려는 목적

200 ~ 299

- 200 OK
  - 요청 정상 처리
- 201 Created
  - 서버 객체를 생성하는 요청(PUT과 같은)에 대한 응답
- 202 Accepted
  - 요청은 받아들여짐. 어떤 것도 수행하지 않음.

400 ~ 499

- 400 BadRequest
  - 잘못된 요청
- 401 Unauthorized
  - 인증 요구
- 403 Forbidden
  - 요청이 서버에 의해 거부.

500 ~ 599

- 500 Internal Server Error
  - 서버가 요청을 처리할 수 없게 만드는 에러
- 501 Not Implemeneted
  - 서버의 능력을 넘는 요청을 했을 때
- 502 Bad Gateway
  - 자신의 부모 게이트웨이 접근이 불가