클라이언트 식별과 쿠키

클라이언트를 인식하기위한 정보들은 뭐가있을까?

HTTP 헤더

클라이언트 IP 주소

사용자 로그인

쿠키

HTTP 헤더

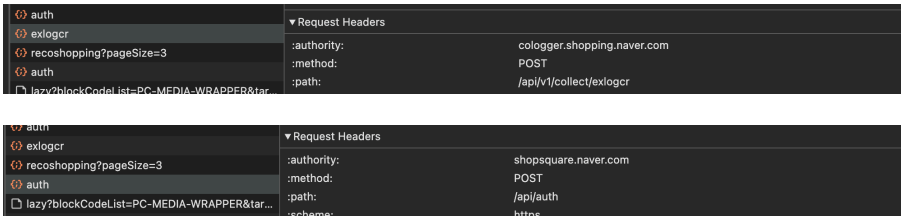
pseudo-header	설명
:method	HTTP 메소드 (GET 등)
:path	요청 경로
:scheme	http / https
:authority	호스트/도메인

authority

HTTP/2에서는 HTTP/1.1과 달리 :authority라는 특수한 헤더 필드가 추가되었습니다.

:authority는 요청한 리소스의 권한을 나타냅니다.

HTTP/2에서는 여러개의 요청을 하나의 TCP연결을 통해 동시에 처리히가 때문에. 각 요청이 어떤 호스트에 대한 것인지를 명시하는 것이 중요해서 이를 위해 :authority 헤더 필드가 추가되었다고합니다.



서로 다른 호스트에대한 주소를 call 한 것으로 보임

method

HTTP/1.1 에서는 실제 HTTP 요청 메소드를 사용했지만 HTTP/2에서는 :method 헤더 필드가 추가되었습니다.

1. HTTP/1.1에서의 요청 구조

```
GET /index.html HTTP/1.1
Host: example.com
```

- 요청 라인(Request Line): 메소드, 경로, 프로토콜 버전이 포함
- 헤더는 그 아래부터
- **텍스트 기반**이라서 사람이 읽고 쓰기 편했지만, 비효율적

2. HTTP/2에서 요청 구조의 변화

```
:method: GET
:path: /index.html
:scheme: https
:authority: example.com
```

- HTTP/2는 **텍스트 대신 바이너리 프레임** 기반
- 요청의 핵심 정보는 **pseudo-header field**로 전달됨
- 일반 헤더보다 **앞에 위치**, 콜론(:)으로 구분
- 요청 라인이 사라졌고, 정보는 구조적으로 분리됨

scheme

scheme은 HTTP 요청이 사용하는 URL scheme을 나타내는 헤더 필드입니다.

```
:scheme: https
```

2. 클라이언트 IP 주소

초기에는 클라이언트의 ip주소를 사용해서 사용자 식별을 하려고했지만

안타깝게도 클라이언트 IP 주소로 사용자를 식별하는 방식은 다음과 같은 약점을 가진다.

- 클라이언트 IP 주소는 사용자가 아닌 사용하는 컴퓨터를 가리킨다.
- 많은 인터넷서비스 제공자(ISP)는 사용자가 로그인하면 동적으로 IP 주소를 할당한다
- 보안을 강화하고 부족한 주소를 관리하려고 많은 사용자가 네트워크 주소 변환 방화벽을 통해 실제 ip 주소를 방화벽 뒤로 숨기고 하나의 방화벽 IP주소로 변환한다.
- 보통 HTTP 프락시와 게이트웨이는 원서버에 새로운 TCP 연결을해서 웹서버는 클라이언트의 IP 주소 대신 프락시 서버의 IP 주소를 본다.

3. 뚱뚱한 URL

특정 웹사이트는 사용자의 URL 마다 버전을 기술하여 사용자를 식별하고 추적하였다.

사용자가 그 사이트를 돌아다니면, 웹 서버는 URL 에 있는 상태 정보를 유지하는 하이퍼링크를 동적으로 생성한다.

1. 사용자 최초 접속
- 사용자가 웹사이트에 처음 접근하면, 서버가 고유 식별자(예: 세션ID)를 생성합니다.
- 서버는 이 식별자를 URL에 포함시켜 클라이언트를 리다이렉트합니다.
- 예: `/main/SESSION12345` 또는 `/main?session=SESSION12345`.

2. 클라이언트의 모든 요청에 식별자 포함
- 사용자는 서버가 제공한 뚱뚱한 URL을 사용해 사이트를 탐색합니다.
- 모든 하이퍼링크와 폼 액션 등도 이 식별자를 포함하도록 서버가 HTML을 생성합니다.
- 예: `<a href="/cart/SESSION12345">장바구니</a>`.

3. 서버는 URL에서 식별자 추출
- 서버는 요청받은 URL에서 세션ID 등 상태 정보를 추출해 해당 사용자를 식별합니다.
- 서버는 이 식별자를 기반으로 사용자의 세션 정보(로그인 상태, 장바구니 등)를 조회합니다.

4. 응답 및 상태 유지
- 서버는 응답을 보낼 때, 다시 모든 링크에 식별자를 포함시켜 반환합니다.
- 사용자가 다른 페이지로 이동해도 항상 식별자가 URL에 남아 있으므로, 서버는 지속적으로 사용자를 추적할 수 있습니다

뚱뚱한 URL 을사용하는 대표적인 사이트

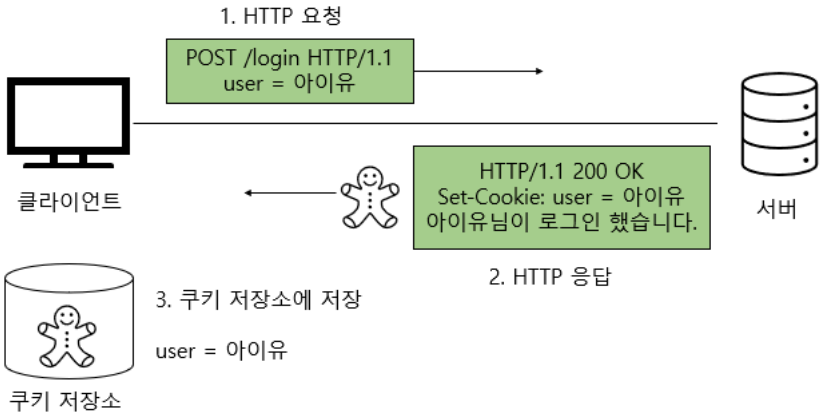
https://www.amazon.com/s?k=cleaning+tools&_encoding=UTF8&content-id=amzn1.sym.83009b1f-702c-4be7-814b-0240b8f687d2&pd_rd_r=f1c9bb95-31ee-47fd-b897-776b007c0e1d&pd_rd_w=FI1O0&pd_rd_wg=zbb4d&pf_rd_p=83009b1f-702c-4be7-814b-0240b8f687d2&pf_rd_r=E151C1A0BMH5GX2BXTPN&ref=pd_hp_d_atf_unk

못생긴 URL, 공유하지 못하는 URL, 캐시를 사용할 수 없음, 서버 부하가중 등 이런 문제로 쿠키를 사용해서 세션관리합니다.

4. 쿠키

쿠키는 사용자를 식별하고 세션을 유지하는 방식 중에서 현재까지 가장 널리사용하는 방식이다.

쿠키는 어떻게 동작하는가



1. **클라이언트 요청 → 서버 응답**

   서버는 클라이언트를 식별하기 위해 **쿠키를 생성**하고,

   `Set-Cookie` 헤더를 통해 응답에 포함시킴.
2. **클라이언트 저장**

   브라우저는 받은 쿠키를 **쿠키 저장소에 저장**함.
3. **다음 요청 시 자동 전송**

   클라이언트는 이후 요청에 **해당 쿠키를 포함**하여 서버에 전송함.
4. **서버는 쿠키 확인**

서버는 쿠키를 읽고, **클라이언트를 식별**함.

5. **쿠키 없으면 재로그인 필요**
   쿠키가 없으면 서버는 사용자를 식별할 수 없으므로,
   다시 **로그인 요청**을 하게 됨.

사이트마다 각기 다른 쿠키들

브라우저는 보통 각 사이트에 두개, 새개의 쿠키만을 보낸다.

또한 많은 웹 사이트는 광고를 관리하는 협력업체와 계약을 한다.

이광고들은 웹사이트 자체의 일부인 것처럼 제작되고 지속 쿠키를 만들어낸다

서드파티 광고 쿠키

네이버 같은 웹사이트로 예시를들면 광고 수익을 얻기 위해 **광고 플랫폼(A사)**과 계약을 해.

A사는 광고를 보여주는 스크립트를 네이버 페이지에 심어줌:

```
html
<script src="https://ads.adpartner.com/ad.js"></script>
```

광고가 "웹사이트 일부처럼 보이는" 이유

- 이 스크립트는 네이버 페이지에 **광고 콘텐츠를 DOM에 직접 삽입**함.
- 사용자 입장에서는 그냥 네이버의 일부분처럼 보이지만,
- 사실 이 광고 콘텐츠는 **ads.adpartner.com**에서 로딩된 거야.

그런데 쿠키는 어떻게 만들어지냐?

- 사용자가 페이지를 열면 `adpartner.com`에서 이미지나 자바스크립트 파일을 불러옴.
- 이때 광고서버(`adpartner.com`)는 **자기 도메인 기준으로 쿠키를 심을 수 있음**:

```
http
Set-Cookie: ad_id=xyz789; Domain=adpartner.com; Expires=Fri, 01 Jan 2038 00:00:00 GMT
```

- 이게 바로 **서드파티 쿠키!**
- 광고서버는 이 쿠키를 통해 "어떤 사용자가 어디서 어떤 광고를 봤는지" 추적 가능.

그래서 생기는 추적 메커니즘

- A사 광고를 사용하는 **다른 사이트(B, C, D)**에서도 동일한 쿠키가 전송됨
- 광고사는 이걸로 **"이 사용자는 네이버 > 다음 > 유튜브" 순으로 돌아다녔다"**는 걸 파악할 수 있어
- → **크로스사이트 추적(Cross-site tracking)**

그래서 최근엔?

- **크롬, 사파리, 파이어폭스 모두 서드파티 쿠키를 점점 막고 있음**
- 특히 **사파리는 거의 모든 서드파티 쿠키를 차단**함
- **구글도 2024~2025년 중 크롬에서 서드파티 쿠키 완전 제거 예정** (Privacy Sandbox 도입 중)

쿠키와 세션 추적

1. **사용자가 브라우저에서 example.com 검색/접속**

- 주소창에 입력 or 검색 결과 클릭
- 브라우저는 HTTP 요청을 보냄:

```
http
GET / HTTP/1.1
Host: example.com
```

2. **서버에서 리디렉트 응답**

- 서버는 클라이언트를 `www.example.com`으로 리디렉트 시킴

```
http
복사편집
HTTP/1.1 301 Moved Permanently
Location: https://www.example.com/
```

- 브라우저는 이 응답을 보고 **자동으로 리디렉션 대상 URL로 재요청**함

3. **리디렉션된 주소로 재요청**

```
http
GET / HTTP/1.1
Host: www.example.com
```

- 이때 서버는 응답하면서 **Set-Cookie** 헤더로 쿠키를 보낼 수 있어:

```
http
HTTP/1.1 200 OK
Set-Cookie: session_id=abc123; Path=/; Secure; HttpOnly
Set-Cookie: user_pref=dark; Max-Age=86400; Path=/
```

4. **브라우저가 쿠키 저장**

- `www.example.com`에 대해 저장된 쿠키 목록에 추가됨
- 쿠키는 **도메인 기준**으로 저장되며, 다음 요청부터 자동으로 포함됨:

```
http
복사편집
Cookie: session_id=abc123; user_pref=dark
```

5. **사용자가 페이지 이동 or 새로고침**

- 같은 도메인 내의 요청이면, 브라우저는 자동으로 쿠키를 포함시켜 요청함
- 서버는 이 쿠키로 사용자를 식별하고 로그인 유지, 장바구니 유지 같은 걸 해줄 수 있어

https://www.lush.co.kr/m?nbsrc=adwords_g&nbkw=lush&gad_source=1&gad_campaignid=22424207011&gbraid=0AAAAACRU4mb6m0NCvEMG9