

2. HTTP Message

2.1 HTTP 개요

2.2 웹 서비스 제공 언어

2.3 HTTP Request/Response Message

2.4 일반 request 에디 & response 에디

2.5 보안 에디

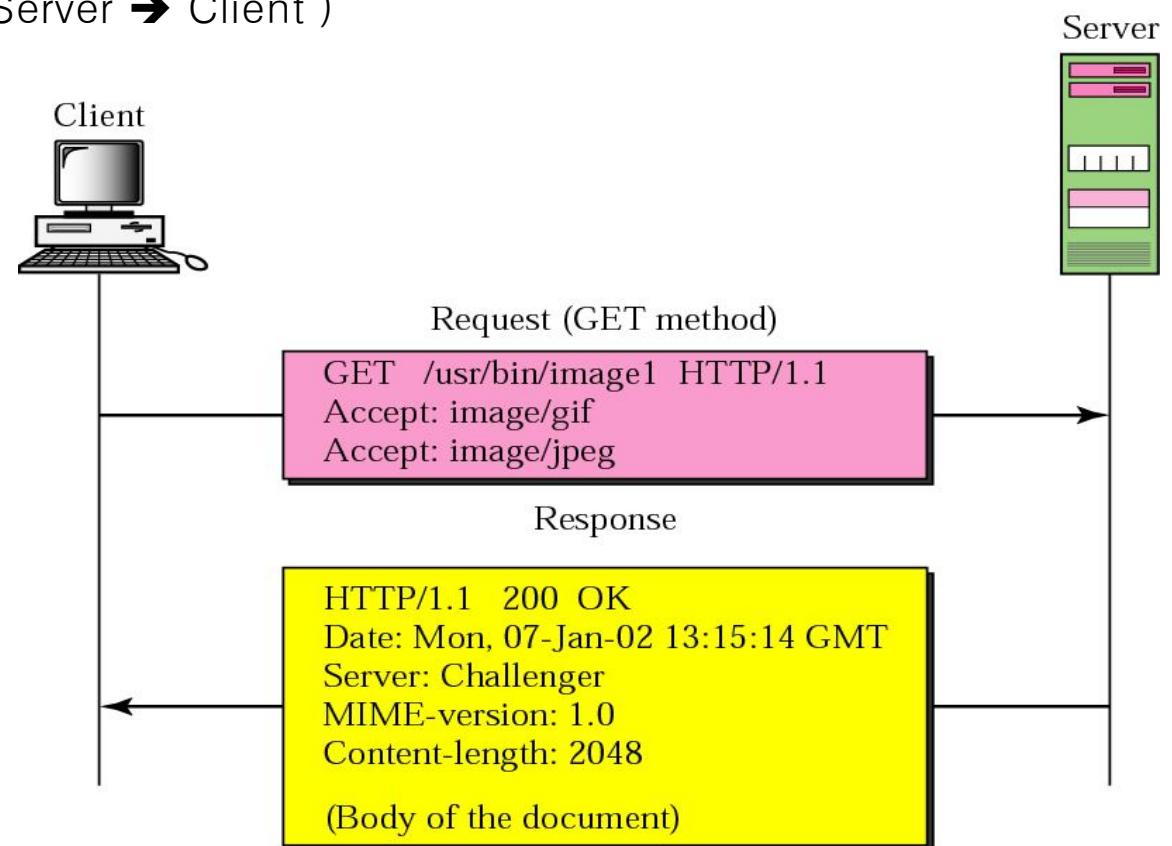
2.6 Cookie & Session

2.1 HTTP 개요

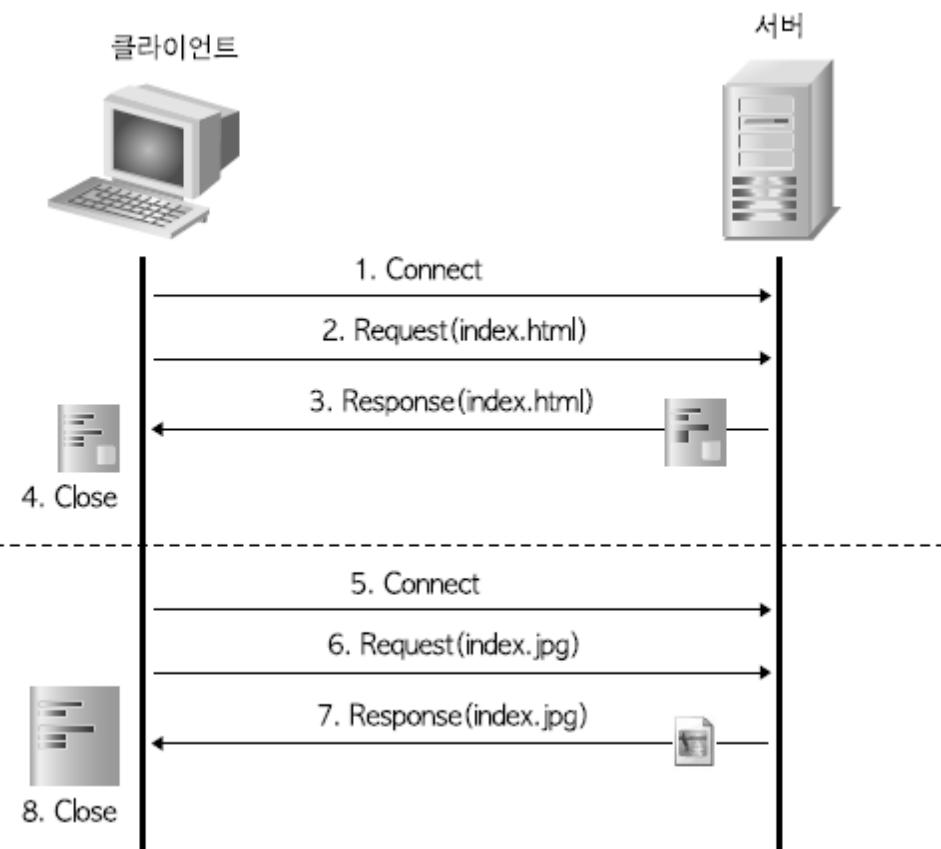
- WWW 상에서 HTML 문서를 주고 받는 응용 계층 프로토콜
- TCP 계열로 80번 포트 사용
- 웹에서 텍스트, 그래픽, 애니메이션을 보거나 사운드 재생 가능
- 클라이언트와 서버 사이에 이뤄지는 요청(request) / 응답(response) 프로토콜
- 현재 인터넷에서 사용되는 HTTP 버전은 HTTP/1.0, HTTP/1.1, HTTP/2.0 임
 - 1997년 1월에 공개된 HTTP/1.1 버전이 현재 가장 많이 사용
- 무상태(stateless) 프로토콜

1) Message 종류

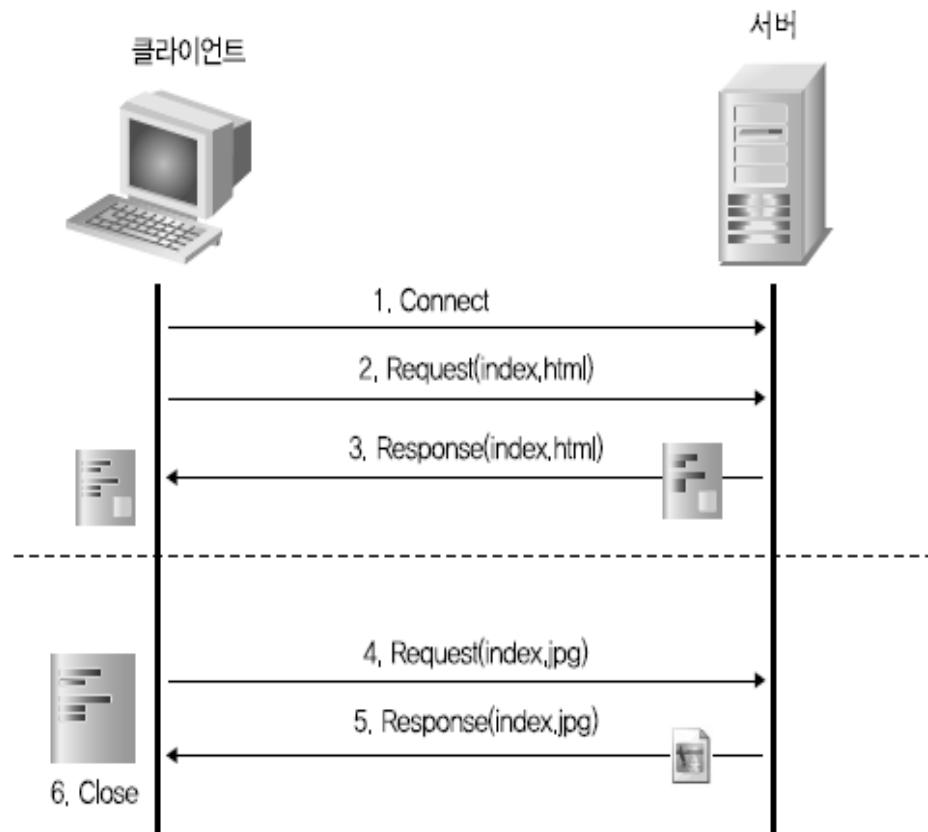
- Request : 요청 Message (Client → Server)
- Response : 응답 Message (Server → Client)



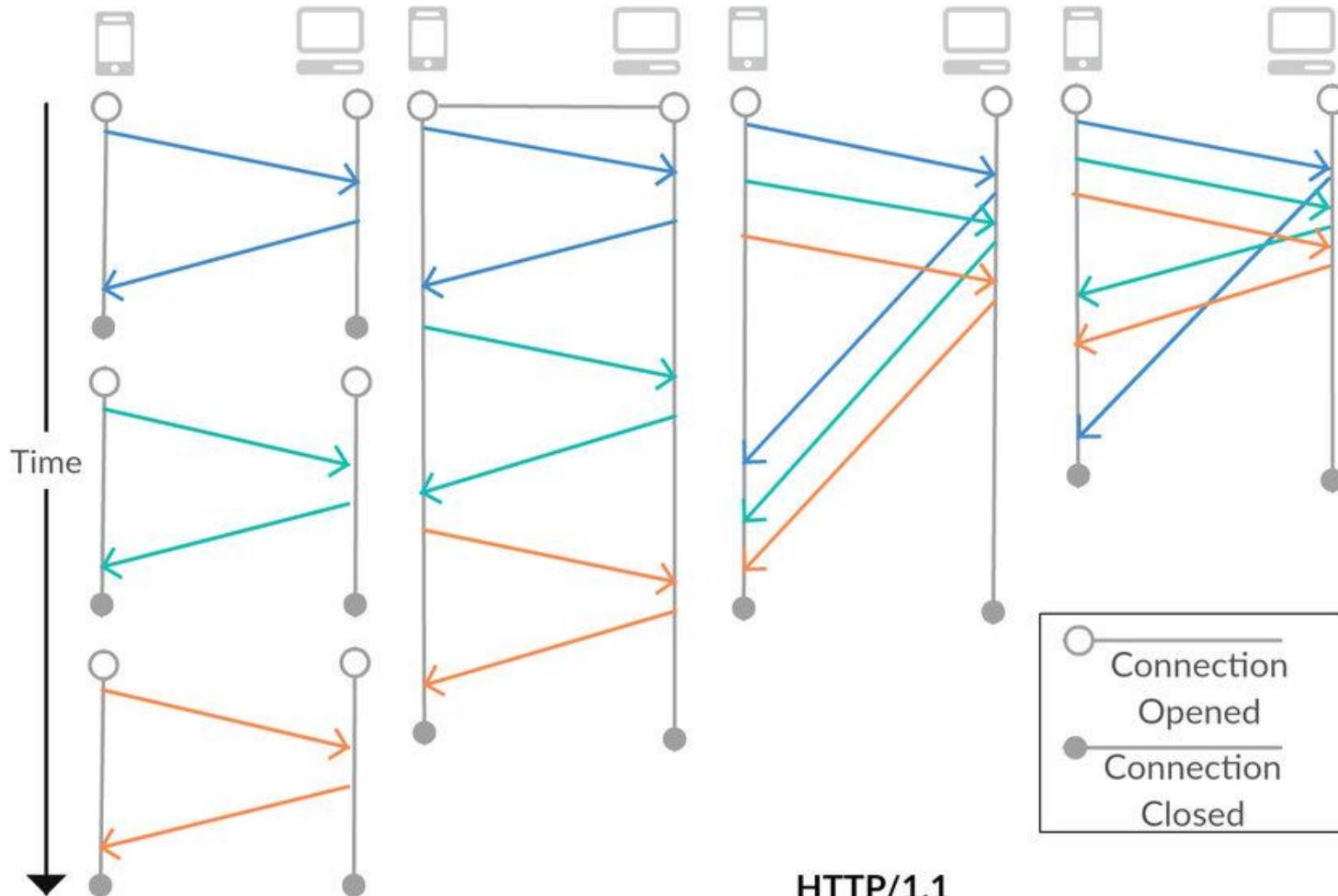
2) HTTP 버전



HTTP/1.0을 이용하여 index.html 읽기



HTTP/1.1을 이용하여 index.html 읽기



HTTP/1.0
Separate Connections

HTTP/1.1
Persistent Connections,
Request Queuing

HTTP/1.1
Pipelining,
Response Queuing,
Head-of-line
blocking

HTTP/2
Multiplexing over
single connection

2.2 웹 페이지 구성

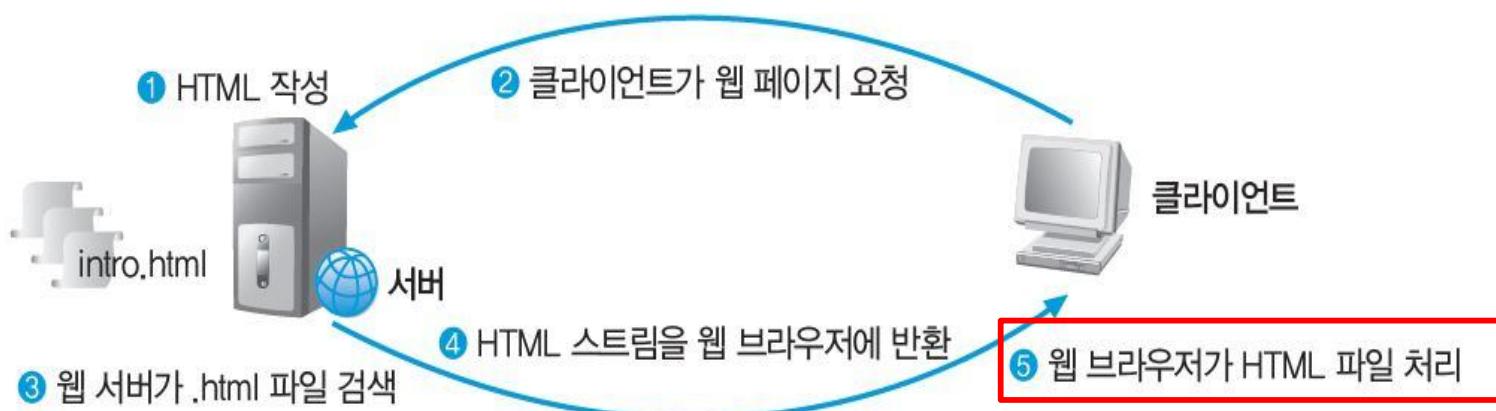
	구현 언어	구현 환경	실행 위치
정적 페이지 구성	HTML	Web Browser	클라이언트 측
동적 페이지 구성	JavaScript	JVM(Java Virtual Machine)	클라이언트 측(CSS)
	ASP, JSP, PHP	CGI WAS	서버 측(SSS)

Contents 종류

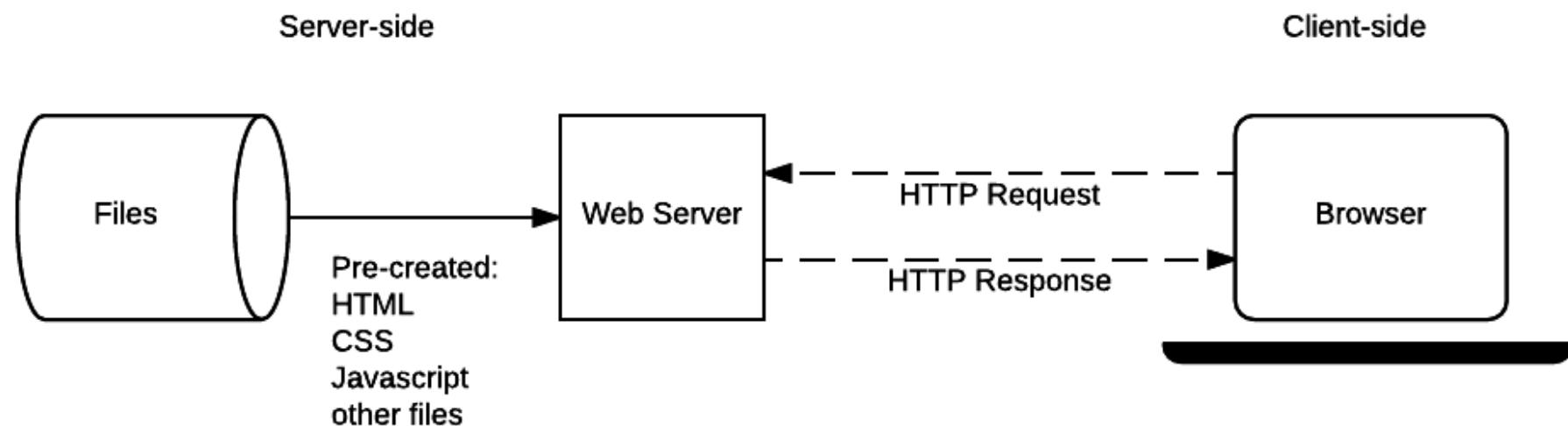
	설명	예제
정적 콘텐츠	<ul style="list-style-type: none">이미지, 음악, 문서파일(hwp,doc)와 같이 웹페이지에 업로드되어 변경되지 않은 콘텐츠웹 서버는 정적인 컨텐츠(html, css, js)를 제공하는 서버	확장자들이 jpg, png, hwp, docs, mp3 html, css, js 등을 가진 파일들
동적 콘텐츠	<ul style="list-style-type: none">게시판과 같이 데이터베이스에 저장되어 수시로 변경되는 정보들WAS는 DB 조회나, 어떤 로직을 처리해야 하는 동적인 컨텐츠를 제공하는 서버	확장자가 php, asp, jsp 등을 가진 파일들

1) HTML(Hyper Text Markup Language)

- 웹 서버에 HTML 문서를 저장하고 있다가 클라이언트가 요청하면 해당 페이지를 전송
- 클라이언트는 웹 페이지 해석 후 웹 브라우저로 나타냄
- 이를 정적인 웹 페이지라고 함.
- 정적인 웹 페이지는 변화를 반영하거나 새로운 것을 추가하기에 많은 시간이 걸림
- 웹 페이지는 바꿀 수 있는 가능성이 매우 낮다는 것이 보안상 장점

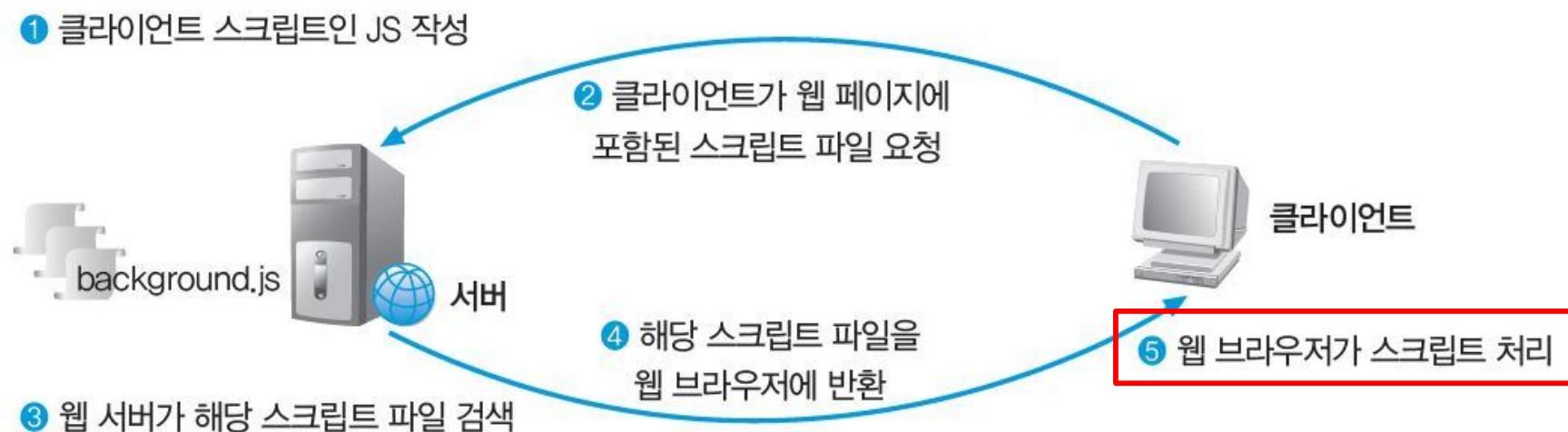


- Static Page (HTML)

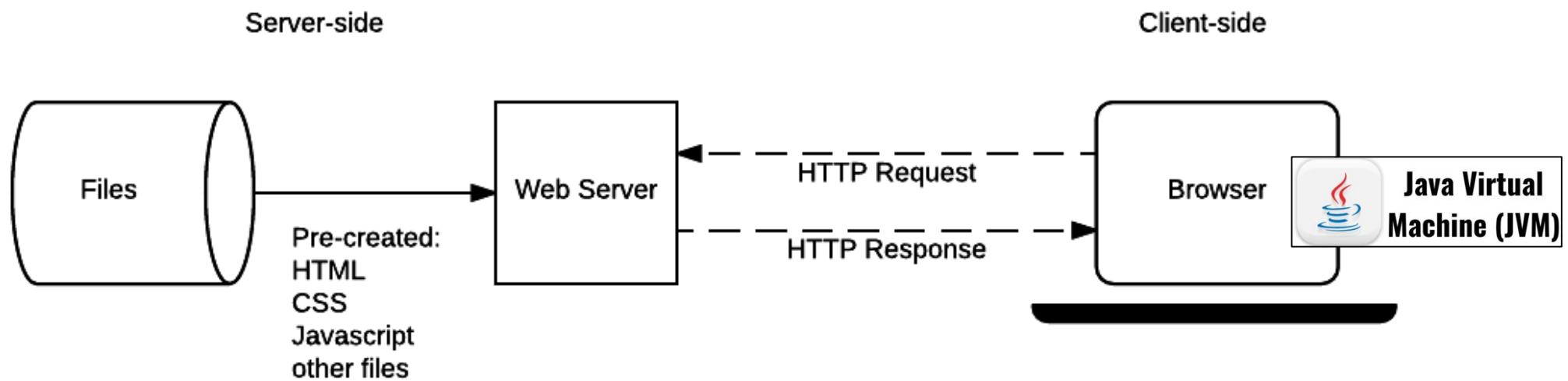


2) CSS(Client Side Script)

- 웹 서비스에 사용되는 스크립트, 클라이언트 측의 웹 브라우저에 의해 해석 및 적용
- 서버가 아닌 웹 브라우저에서 해석되어 화면에 적용
- 웹 서버의 부담을 줄이면서 다양한 기능을 수행할 수 있음
- JavaScript(객체지향언어), VB Script(주로 ASP와 연동)

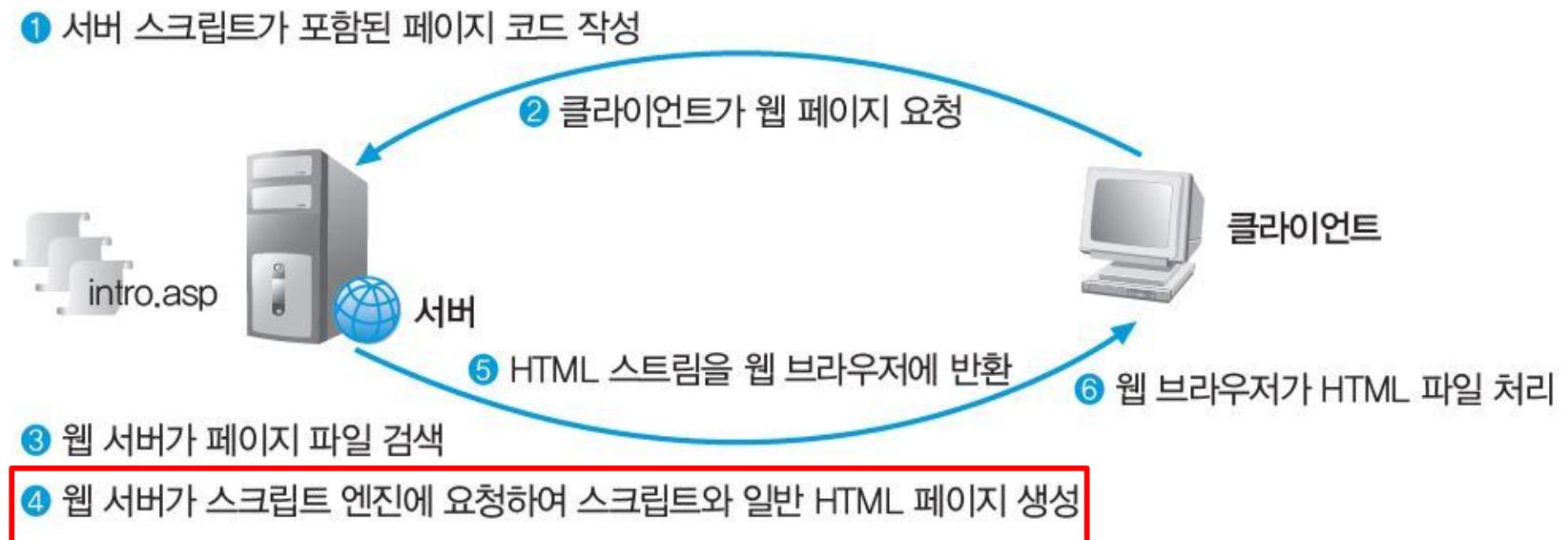


- Static sites (CSS)

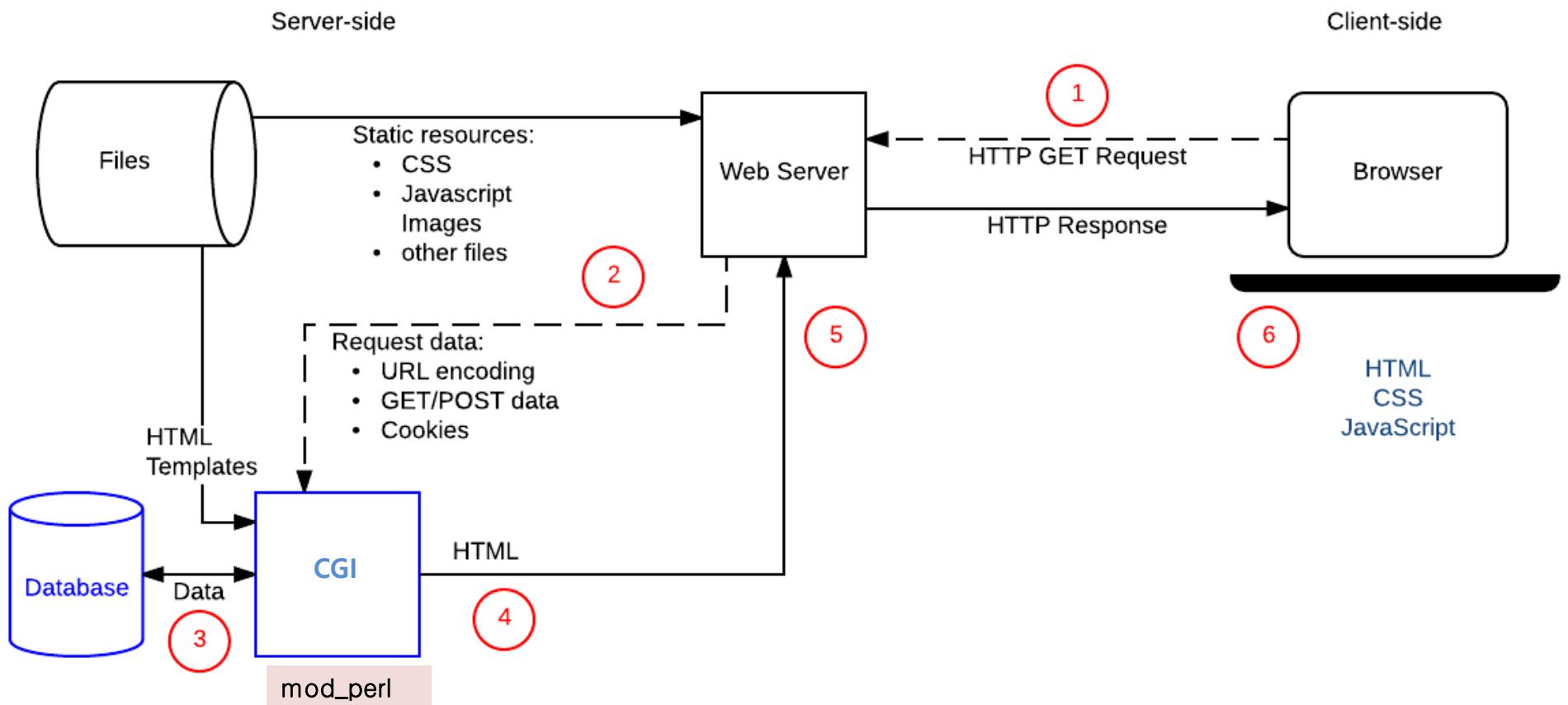


3) SSS(Server Side Script)

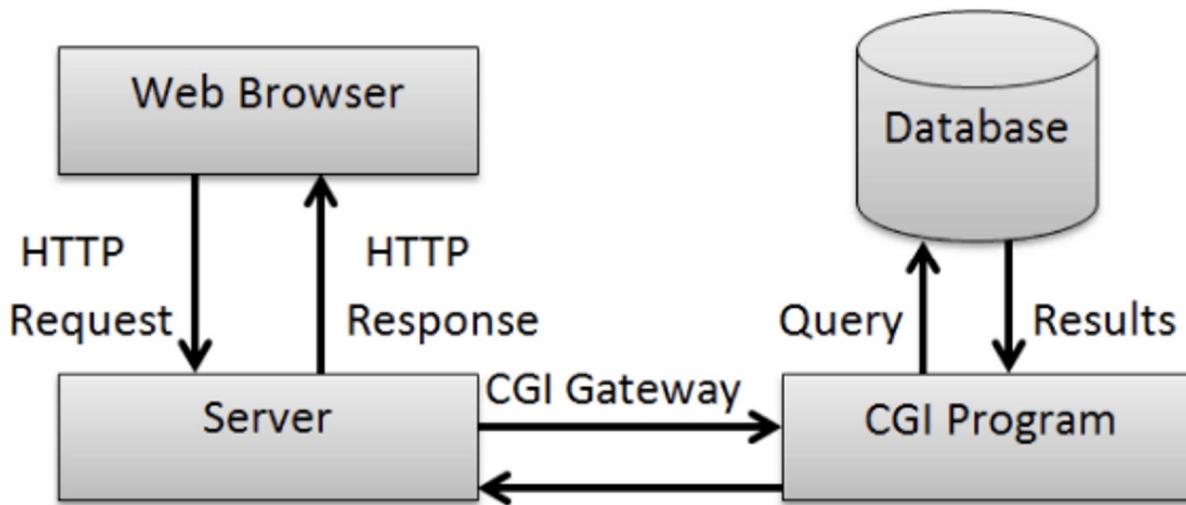
- CSS에서 받아온 정보를 처리하는 역할
- CSS에서 사용자가 어떠한 입력을 하게 되면 해당하는 정보를 SSS에서 처리하고 DB에 저장하거나 삭제
- 서버에 따라서 다른 언어를 사용하기 때문에 동작하는 웹 플랫폼이 다르기 때문에 맞춰서 사용



① CGI로 Dynamic sites 구현

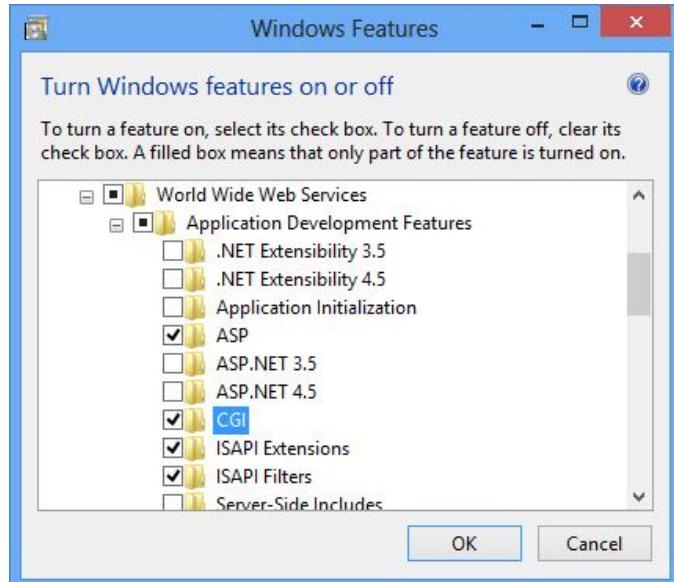


CGI(Common Gateway Interface)



- 웹 서버에서 프로그램을 동작시키기 위한 프로토콜
- CGI는 웹 서버에 미리 저장된 HTML을 전달하는 것 뿐만 아니라, 사용자의 요청을 CGI 규격을 준수한 프로그램에서 적절하게 처리하고, 처리 결과를 HTML로 생성하여 웹 서버에 전달
- Python, Java, PHP 등의 프로그래밍 언어로 CGI 규격을 준수한 CGI Program을 만들면, 웹 서버는 CGI Program을 호출하고, 클라이언트의 요청에 대해 개별 프로세스를 생성

CGI interface 설치

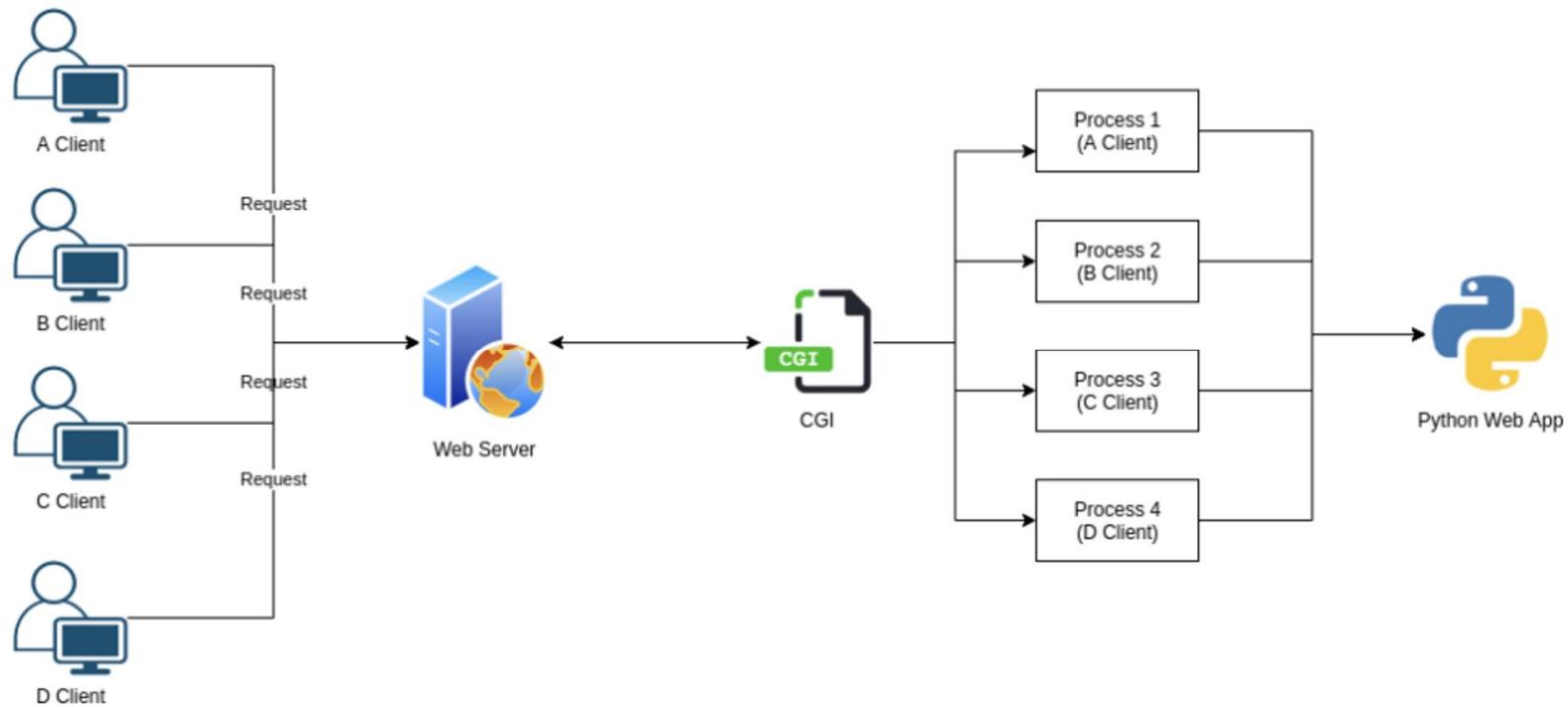


<< IIS 설치 시 >>

```
76
77 #cgiconfiguration
78 LoadModule cgi_module /usr/lib/apache2/modules/mod_cgi.so
79 ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
80 <Directory /usr/local/apache2/cgi-bin>
81     AllowOverride None
82     AddHandler cgi-script .cgi .pl
83     Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
84     Order allow,deny
85     Allow from all
86     Require all granted
87 </Directory>
88
```

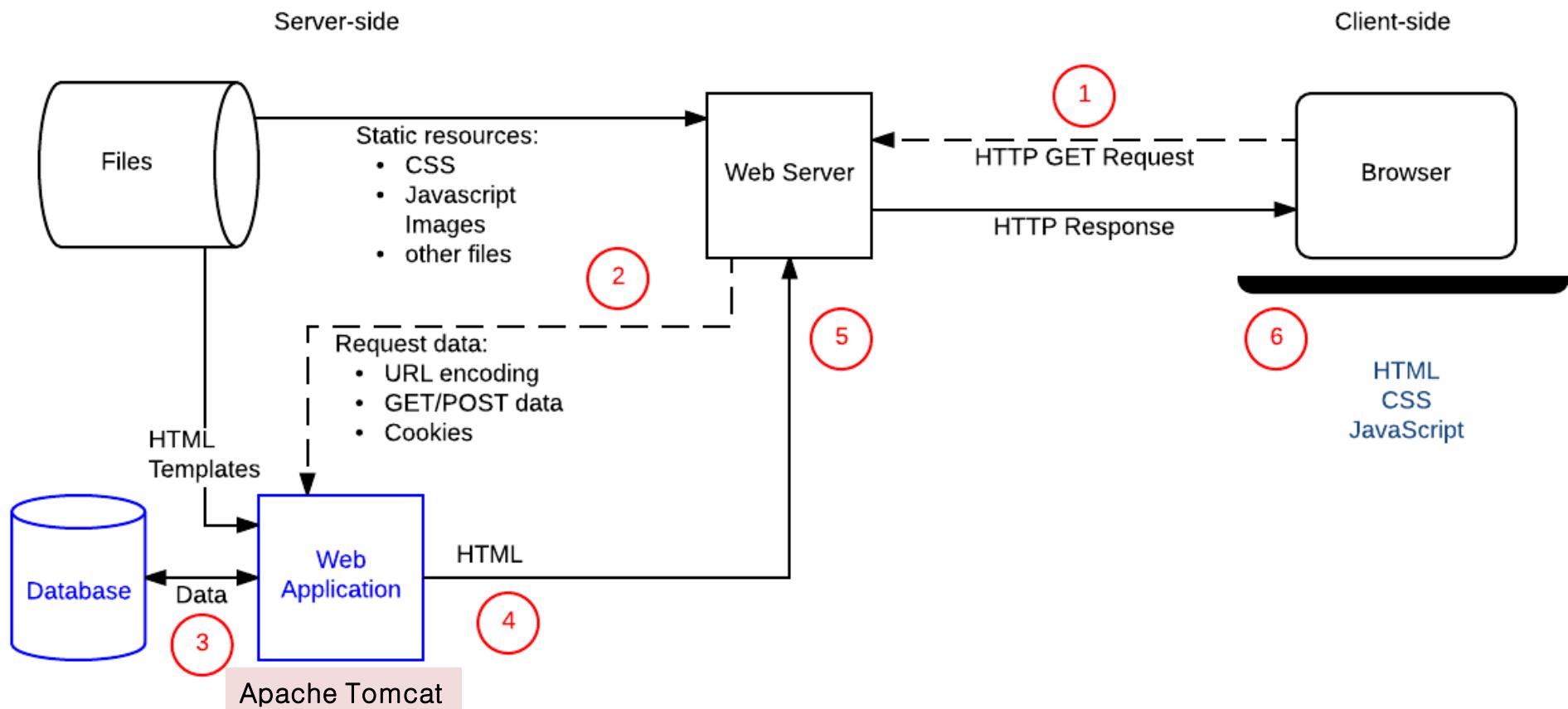
<< Apache 후 CGI 설정 >>

CGI(Common Gateway Interface) 한계

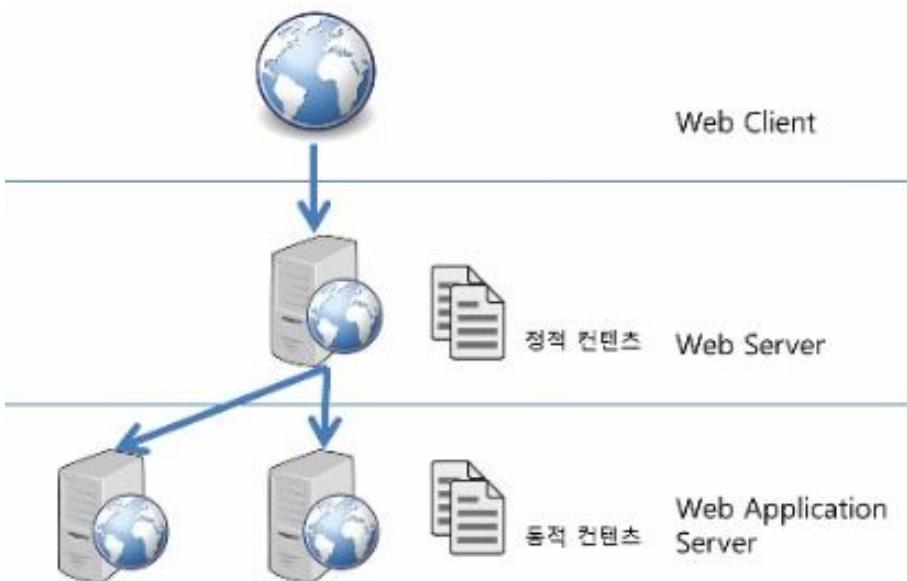


- 클라이언트의 요청이 많아지면 서버에 가해지는 부하를 적절하게 분배하지 못함
 - CGI가 클라이언트의 요청이 들어올 때마다 독립적인 프로세스를 생성하기 때문 (멀티 프로세싱)

2 WAS로 Dynamic sites 구현



웹서버와 WAS를 분리하는 이유



WAS가 해야 할 일의 부담을 줄이기 위함

- 웹 서버에서는 정적인 문서만 처리, WAS는 애플리케이션의 로직만 수행하도록 기능을 분배
- WAS앞에 웹 서버를 둠으로써 서버의 부담을 줄일 수 있음
- 웹 서버에서는 플러그인 형태로 WAS를 연결하면 일 처리를 나눌 수 있음

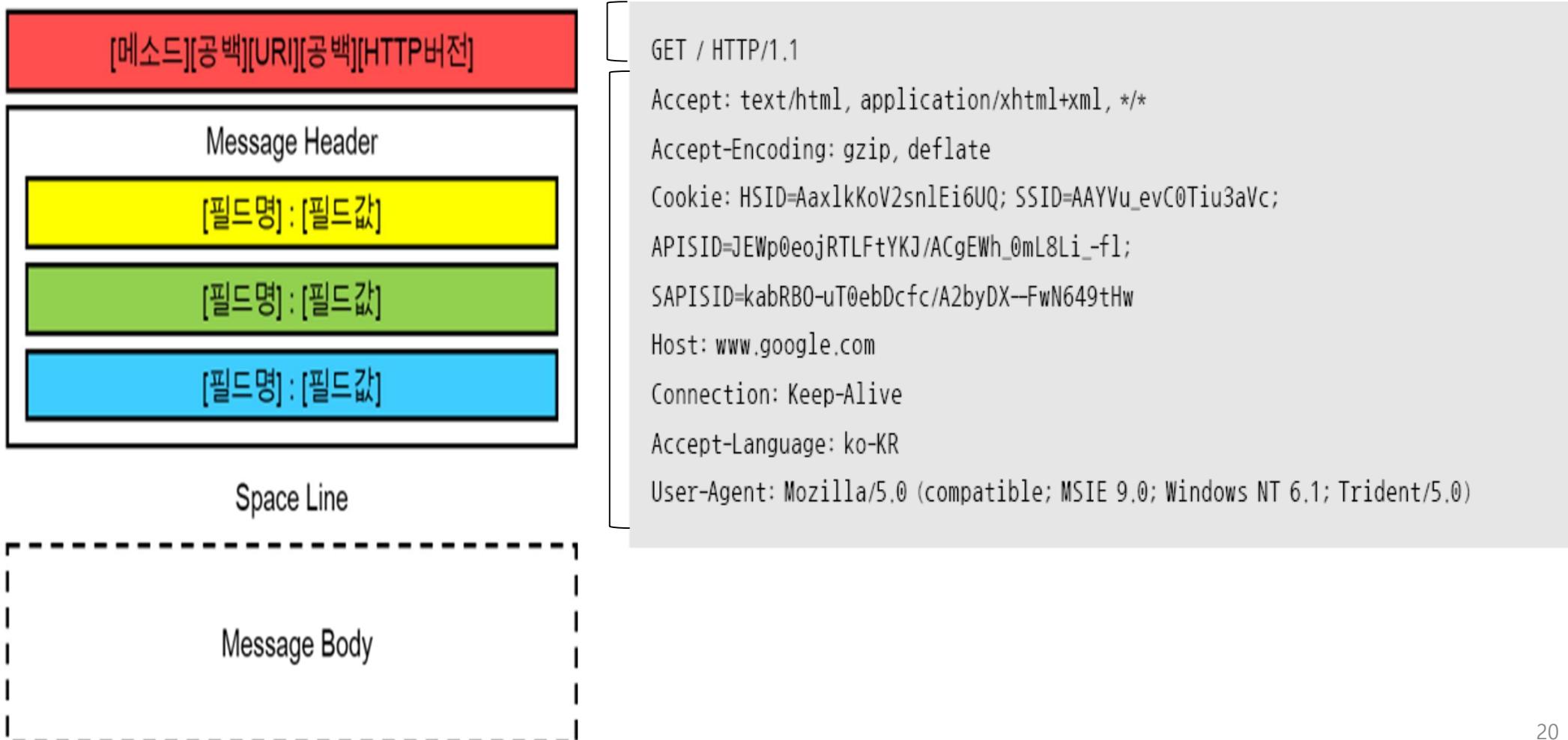
WAS의 환경설정 파일을 외부에 노출시키지 않도록 하기 위함

- 클라이언트와 연결하는 포트가 직접 WAS에 연결이 되어 있다면 중요한 설정 파일들이 노출될 수 있음
- 웹 서버와 WAS에 접근하는 포트가 다르기 때문에, WAS에 들어오는 포트에는 방화벽을 둬서 보안을 강화할 수도 있음

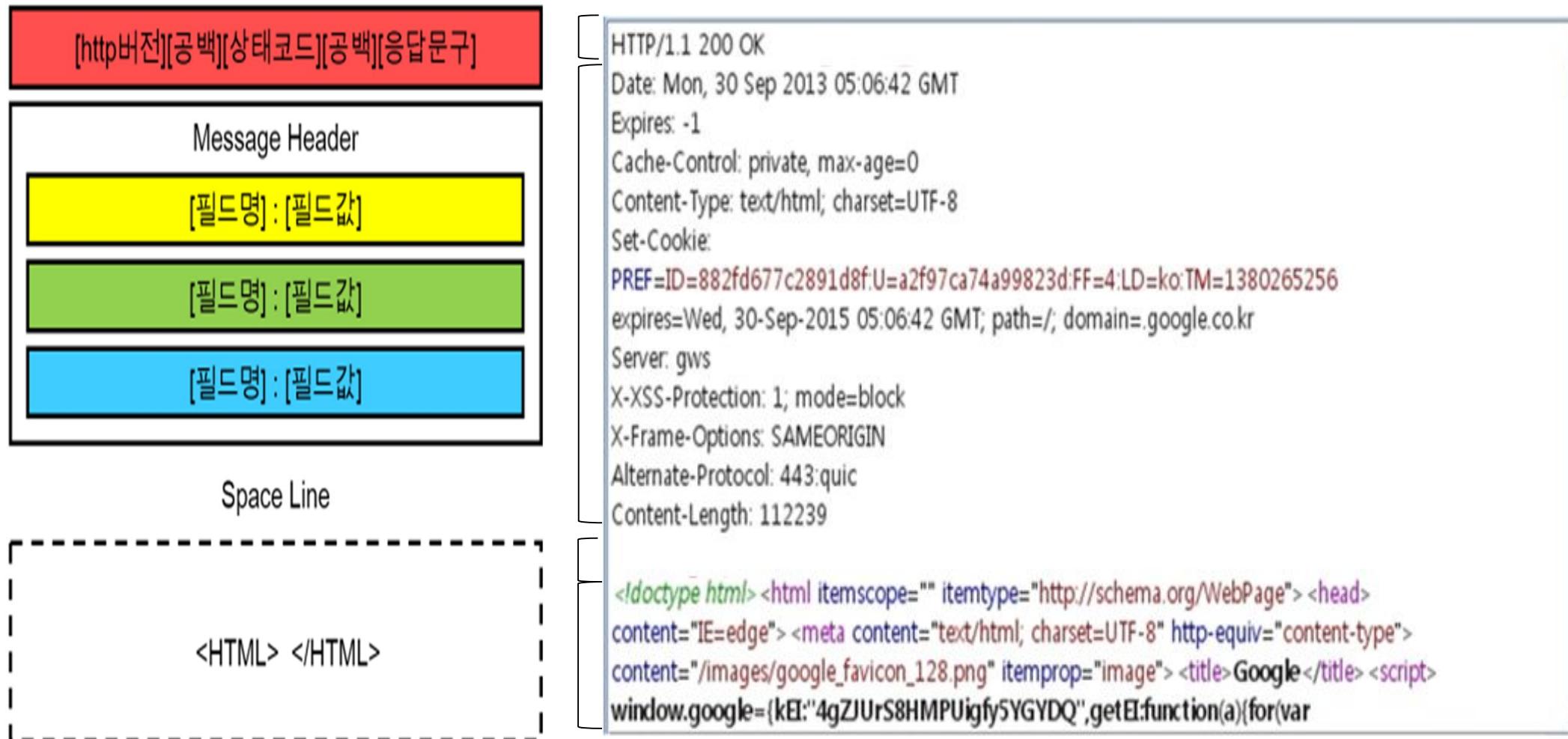
2.3 HTTP Request/Response Message

- HTTP Request Message
- HTTP Response Message

1) HTTP Request Message Format



2) HTTP Response Message Format



3) HTTP Request Method

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE

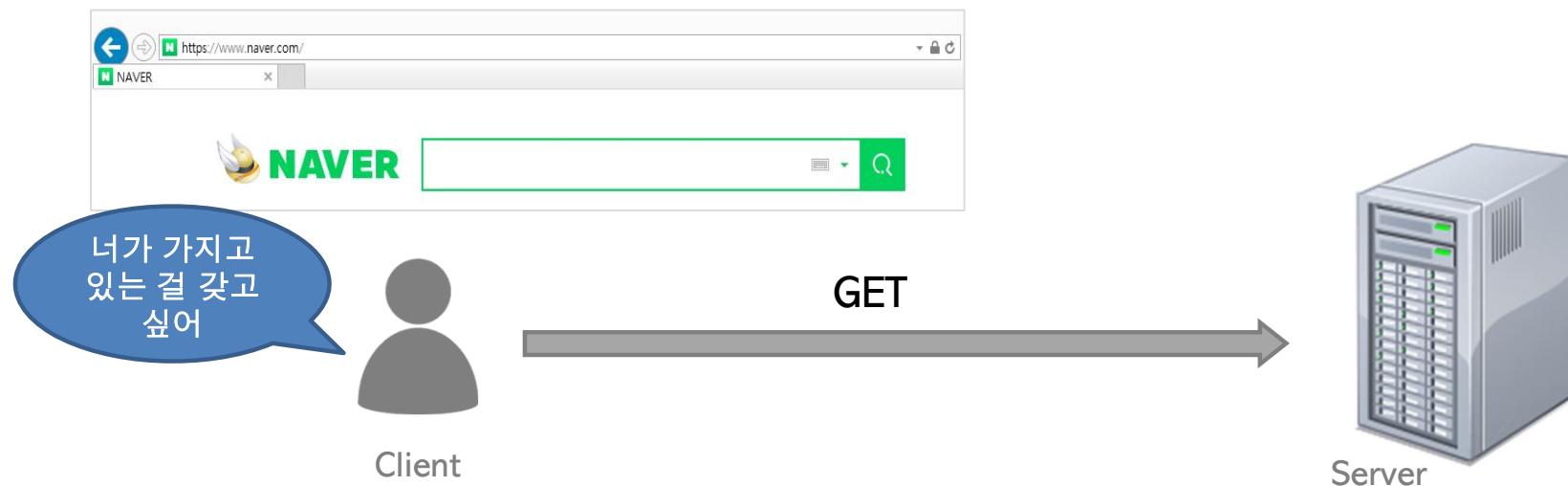
POST /form/entry HTTP/1.1

① 메소드 ② 요청URI ③ 프로토콜 버전

GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Encoding: gzip, deflate
Cookie: HSID=AaxlkKoV2snlEi6UQ; SSID=AAYVu_evCOTiu3aVc;
APSID=JEWp0ejRTLFtYKJ/ACgEWh_0mL8Li_-f1;
SAPISID=kabRB0-uT0ebDcfc/A2byDX-FwN649tHw
Host: www.google.com
Connection: Keep-Alive
Accept-Language: ko-KR
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)

① HTTP Request : Get 방식

- 요청 데이터의 인수를 웹 브라우저의 URL로 전송함
 - 데이터가 주소 입력란에 표시 (최소한의 보안도 유지되지 않는 취약한 방식)
- Request URI로 식별된 리소스를 가져올 수 있도록 요구



```
#curl -v 192.168.10.20
```

```
[root@kali) ~]
# curl -v 192.168.10.20
* Trying 192.168.10.20:80 ...
* Connected to 192.168.10.20 (192.168.10.20) port 80 (#0)
> GET / HTTP/1.1
> Host: 192.168.10.20
> User-Agent: curl/7.85.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 22 May 2024 07:15:10 GMT
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< X-Powered-By: PHP/5.2.4-2ubuntu5.10
< Content-Length: 891
< Content-Type: text/html
```

② HTTP Request : POST 방식

- 엔티티를 전송하기 위해 사용
- URL에 요청 데이터를 기록하지 않고 HTTP 헤더에 데이터를 전송.
- 엔티티를 URL로 전송하지 않으므로 다른 사용자가 링크로 해당 페이지를 볼 수 없음

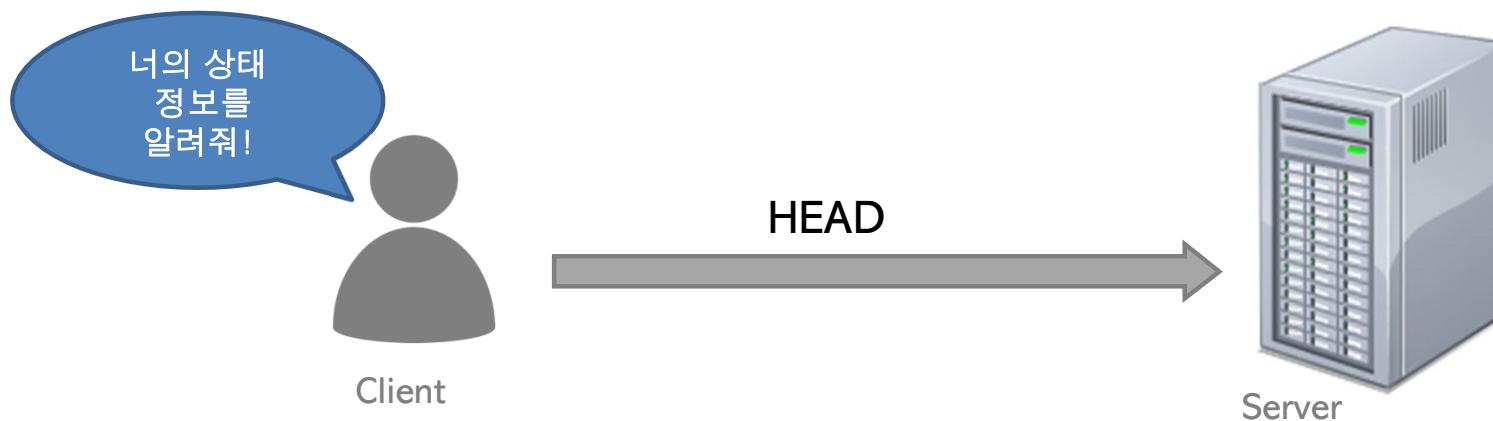


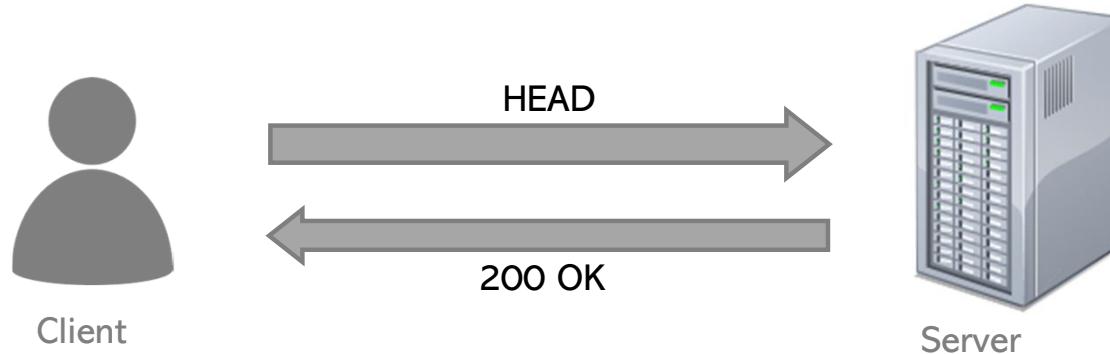
```
#curl -v -X POST 192.168.10.20
```

```
[root@kali] ~
# curl -v -X POST 192.168.10.20
* Trying 192.168.10.20:80 ...
* Connected to 192.168.10.20 (192.168.10.20) port 80 (#0)
> POST / HTTP/1.1
> Host: 192.168.10.20
> User-Agent: curl/7.85.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 22 May 2024 07:16:56 GMT
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< X-Powered-By: PHP/5.2.4-2ubuntu5.10
< Content-Length: 891
< Content-Type: text/html
```

3 HTTP Request : Head 방식

- GET방식과 동일
- 서버 측 데이터를 검색하고 요청하는 데 사용
- BODY가 없고 응답코드와 HEAD로만 응답
 - 서버의 응답 헤더를 봄으로써 Resource가 수정 되었는지 확인
- URI 유효성과 리소스 갱신 시간을 확인하는 목적을 사용
 - 웹 서버 정보확인, 헬스체크, 버전확인, 최종 수정일자 확인 용도





```

└──(root㉿kali)-[~]
└─# curl -I 192.168.10.30
HTTP/1.1 200 OK
Date: Wed, 29 May 2024 12:05:34 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g
Last-Modified: Sun, 02 Nov 2014 18:20:24 GMT
ETag: "ccb16-24c-506e4489b4a00"
Accept-Ranges: bytes
Content-Length: 588
Content-Type: text/html

```

```

└──(kali㉿kali)-[~]
└─$ curl -I 192.168.10.129
HTTP/1.1 200 OK
Date: Tue, 17 Jan 2023 11:30:02 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html

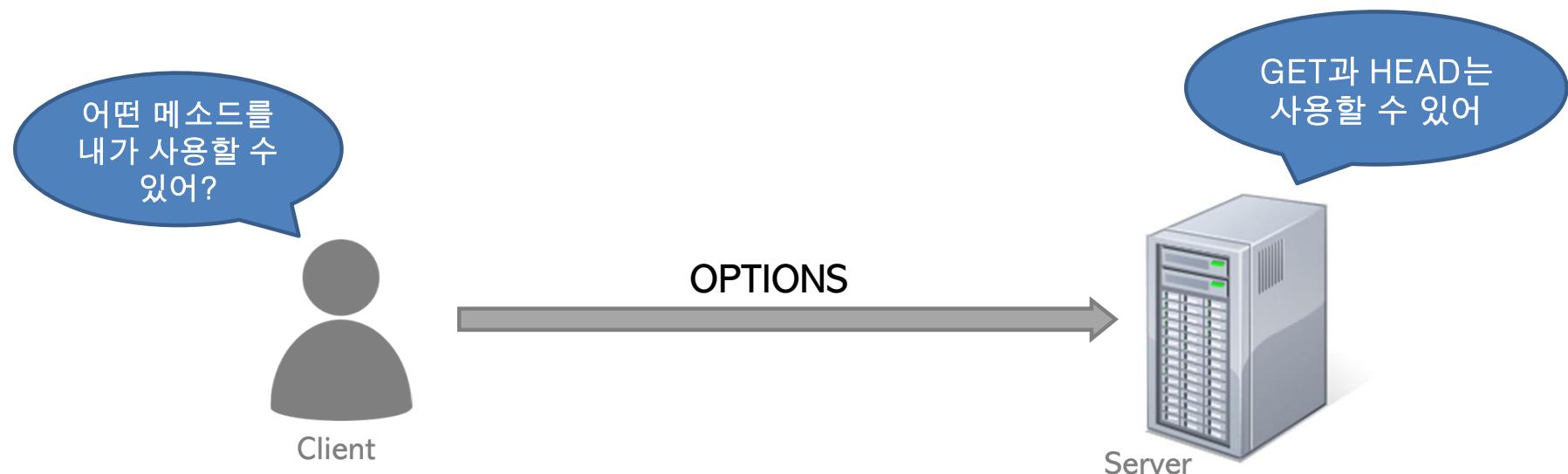
```

#curl -I 192.168.10.20

#curl -I http://192.168.10.20/dvwa/setup.php

4 HTTP Request : Option Method

- 자원에 대한 요구/응답 관계와 관련된 선택 사항 정보를 요청할 때 사용
- 시스템에서 지원되는 메소드 종류를 확인 할 수 있음
- 웹서버에서 지원하는 HTTP 요청방식을 확인하고자 할 때 사용



```
[root@kali]~]
# service apache2 start

[root@kali]~]
# curl -v -X OPTIONS localhost
* Trying 127.0.0.1:80...
* Connected to localhost (127.0.0.1) port 80 (#0)
> OPTIONS / HTTP/1.1
> Host: localhost
> User-Agent: curl/7.85.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 28 Mar 2023 22:59:04 GMT
< Server: Apache/2.4.54 (Debian)
< Allow: POST,OPTIONS,HEAD,GET
< Content-Length: 0
< Content-Type: text/html
<
* Connection #0 to host localhost left intact
```

```
#curl -v -X OPTIONS 192.168.10.10
#curl -v -X OPTIONS 192.168.10.20
#curl -v -X OPTIONS 192.168.10.30
```

```
[root@kali]~]
# nc 192.168.10.20 80
OPTIONS /dav/ HTTP/1.1
HOST: 192.168.10.20

HTTP/1.1 200 OK
Date: Wed, 22 May 2024 02:03:25 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Content-Type: text/plain
```

5 HTTP Request : Trace Method

- Trace method는 Client–Server 간에 loopback test를 진행
 - 통신 진행 상 어느 지점에서 에러가 났는지 확인하기 위한 용도

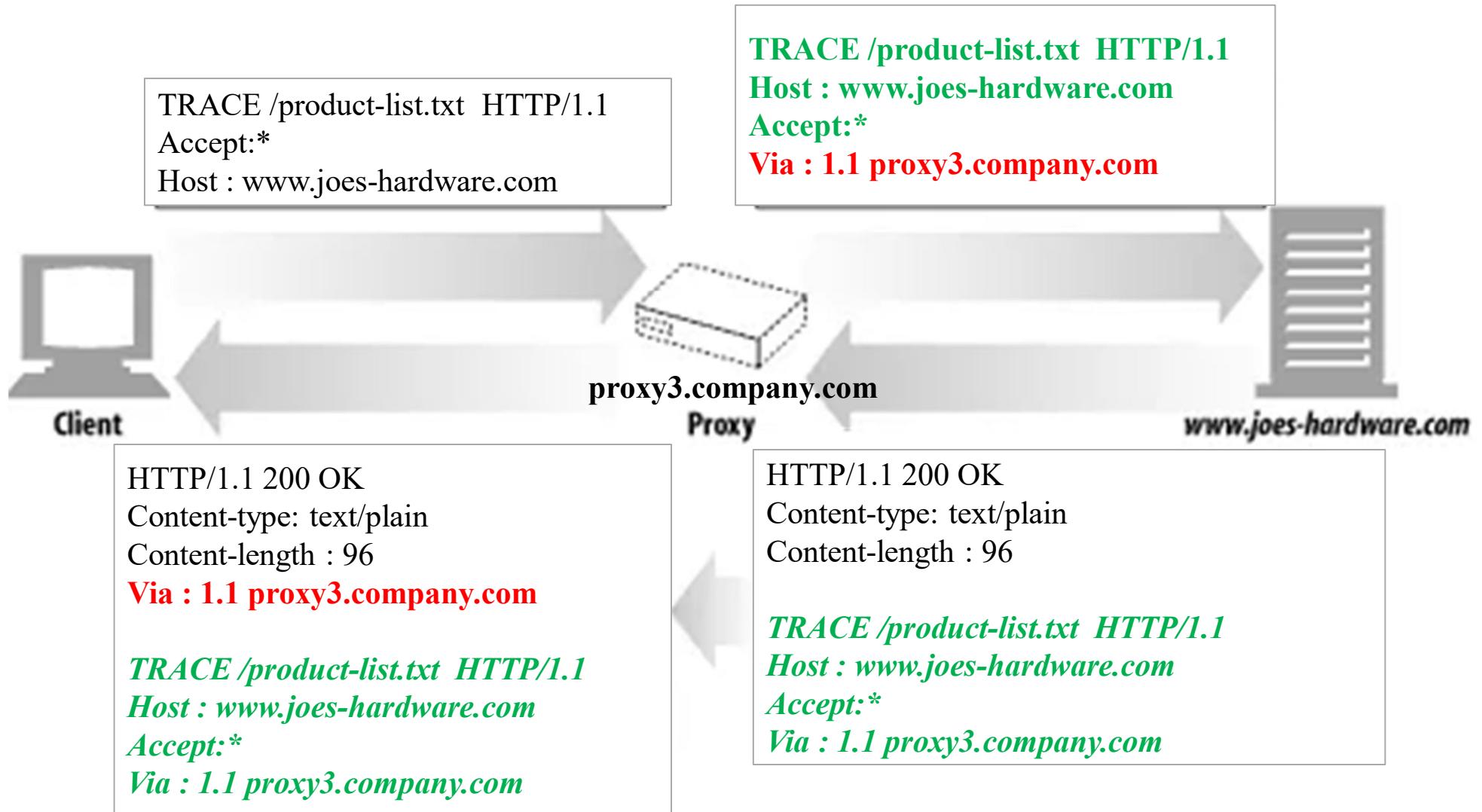
```
(root㉿kali)-[~]
# curl -v -X TRACE http://192.168.10.129
* Trying 192.168.10.129:80 ...
* Connected to 192.168.10.129 (192.168.10.129) port 80 (#0)
> TRACE / HTTP/1.1
> Host: 192.168.10.129
> User-Agent: curl/7.85.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 28 Mar 2023 23:14:00 GMT
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< Transfer-Encoding: chunked
< Content-Type: message/http
<
TRACE / HTTP/1.1
Host: 192.168.10.129
User-Agent: curl/7.85.0
Accept: */*

* Connection #0 to host 192.168.10.129 left intact
```

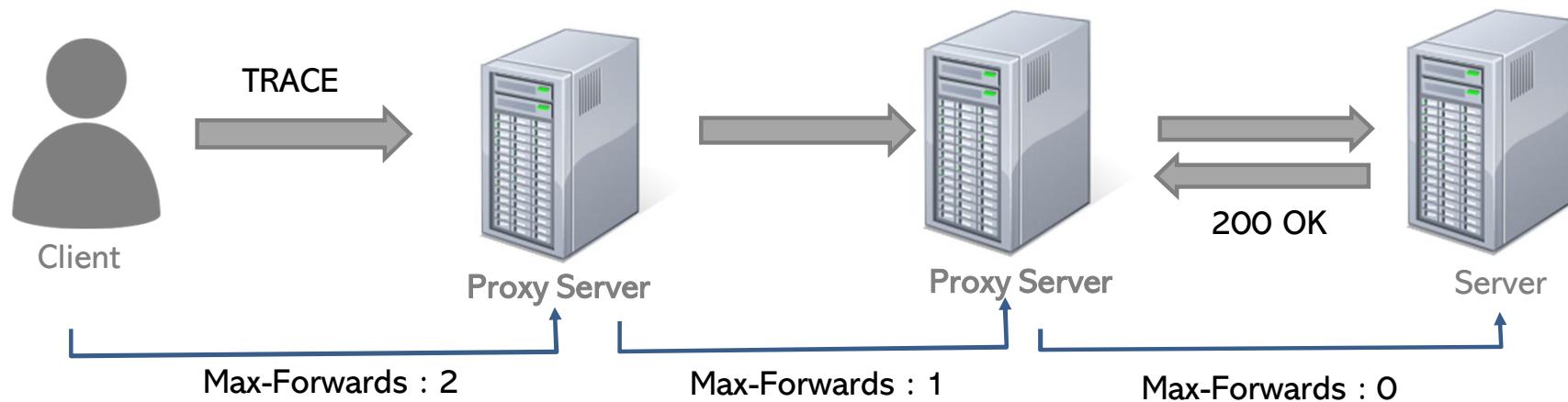
```
#curl -v -X TRACE 192.168.10.20
```

- 웹서버로 가는 네트워크 경로를 체크
- 웹서버와 클라이언트 사이에 중간서버
(웹 프록시 또는 웹캐시 서버 등)가
존재할 경우 trace 방식을 이용하여 확인



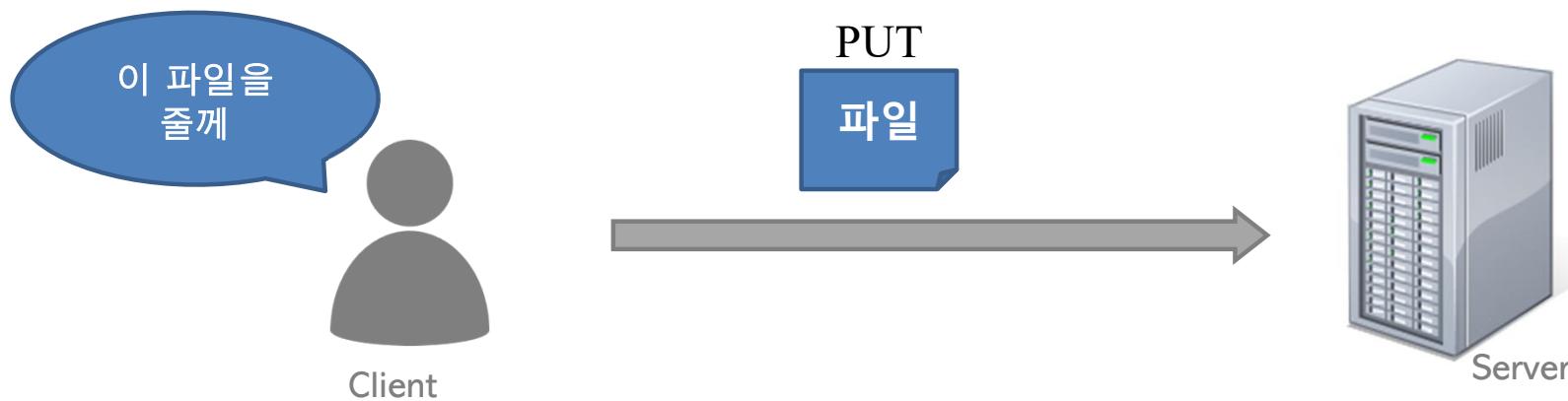


- 웹서버가 웹브라우저(클라이언트)에서 보낸 내용을 받아 이를 다시 웹 브라우저로 되돌려 주는 것
→ 클라이언트로 Loopback을 발생시킴
- 웹 서버로 가는 네트워크 경로를 체크하는 메소드
- Request를 보낼 때 “Max-Forwards”라는 헤더 필드에 수치를 포함
 - 서버를 통과 할 때마다 수치가 감소, 0이 된 곳을 끝으로 200 OK response를 되돌려줌



⑥ HTTP Request : PUT 방식

- POST와 유사한 전송 구조를 가짐
- 클라이언트가 지정한 위치에 특정 파일을 가져다 놓으려고 할 때 PUT 방식 사용
- 인증 기능 없이 누구든지 파일을 업로드 가능하다는 보안 상의 문제가 있음



```
#nano /etc/apache2/httpd.conf
```

```
<Directory />
```

```
<LimitExcept GET HEAD POST PUT DELETE OPTIONS>
```

```
    Order deny,allow
```

```
    Deny from all
```

```
</LimitExcept>
```

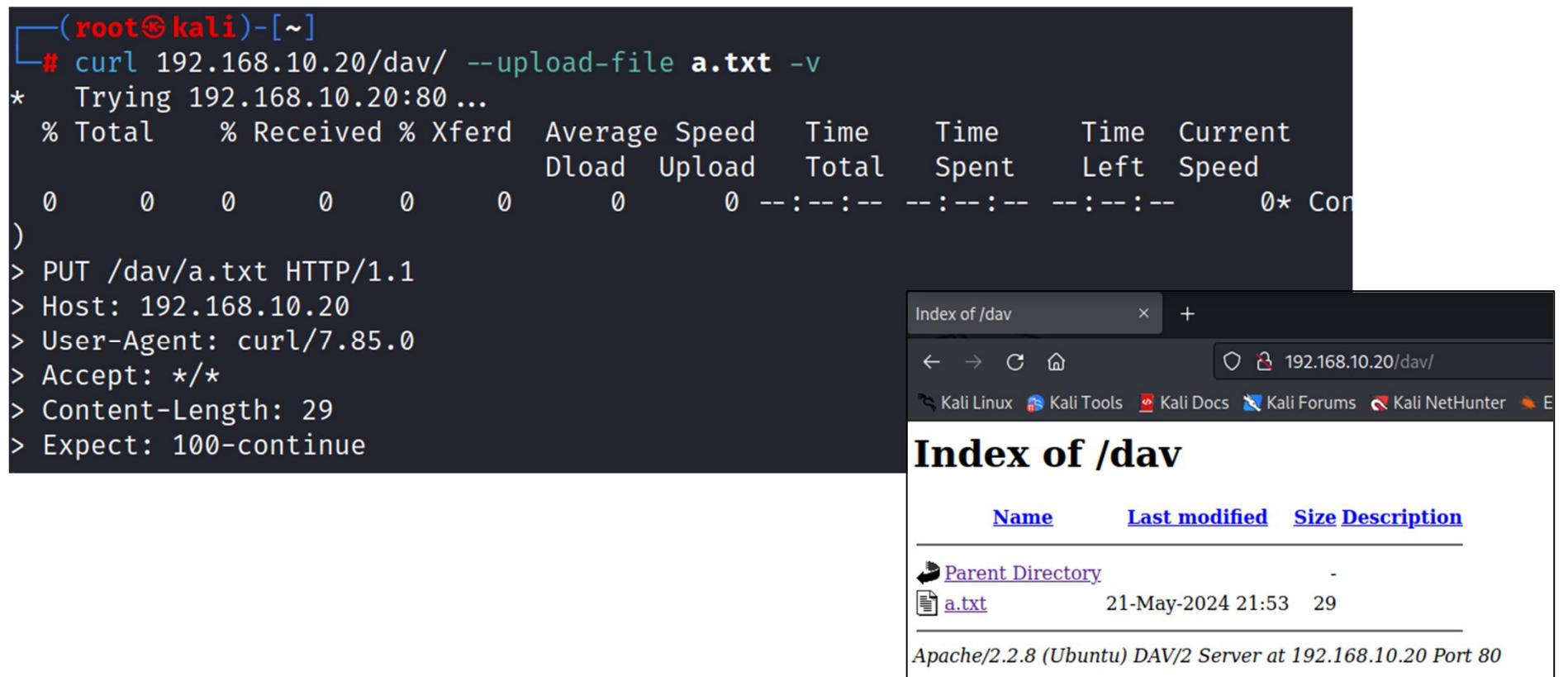
```
</Directory>
```

```
root@metasploitable:/var/www/dav# cd /etc/apache2
root@metasploitable:/etc/apache2# ls
apache2.conf  envvars  mods-available  ports.conf      sites-enabled
conf.d        httpd.conf  mods-enabled   sites-available
root@metasploitable:/etc/apache2# cat httpd.conf
<Directory />
    <LimitExcept GET HEAD POST PUT DELETE OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>

root@metasploitable:/etc/apache2#
```

```
#cat > a.txt
```

```
#curl http://192.168.10.20/dav/ --upload-file a.txt -v
```



The terminal session shows the command `curl 192.168.10.20/dav/ --upload-file a.txt -v` being run from a root shell on Kali Linux. The curl output shows the progress of the upload, including headers and the file content.

```
(root㉿kali)-[~]
# curl 192.168.10.20/dav/ --upload-file a.txt -v
* Trying 192.168.10.20:80 ...
% Total    % Received % Xferd  Average Speed   Time      Time      Current
          Dload  Upload   Total   Spent    Left  Speed
0       0     0      0      0        0      0 --:--:-- --:--:-- --:--:-- 0* Con
)
> PUT /dav/a.txt HTTP/1.1
> Host: 192.168.10.20
> User-Agent: curl/7.85.0
> Accept: */*
> Content-Length: 29
> Expect: 100-continue
```

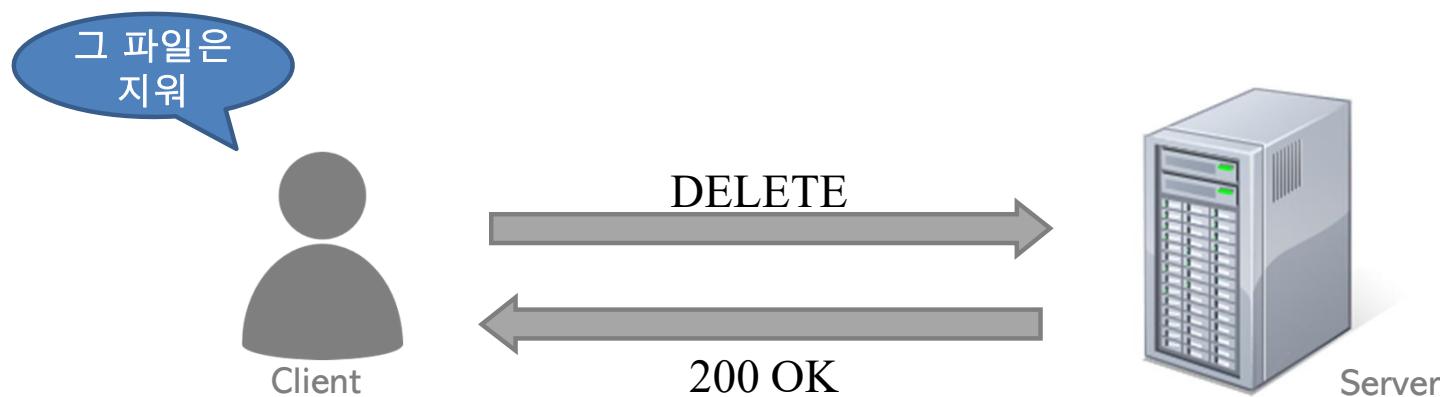
The browser screenshot shows the Apache DAV index page at `http://192.168.10.20/dav/`. It lists a single file, `a.txt`, with details such as name, last modified date, and size.

Name	Last modified	Size	Description
Parent Directory		-	
a.txt	21-May-2024 21:53	29	

Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.10.20 Port 80

7 HTTP Request : Delete 방식

- URI에 지정된 자원을 서버에서 지울 수 있게 함
- 원격지 웹 서버에 파일을 삭제하기 위해 사용되며 PUT과는 반대 메소드



```
(root㉿kali)-[~]
# nc 192.168.10.20 80
DELETE /dav/a.txt HTTP/1.1
HOST: 192.168.10.20

HTTP/1.1 204 No Content
Date: Wed, 22 May 2024 06:58:06 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
Content-Length: 0
Content-Type: text/plain
```

The screenshot shows a web browser window with the URL `192.168.10.20/dav/`. The title bar says "Index of /dav". The page content includes:

- A header with links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, and Exp.
- A main heading "Index of /dav".
- A table with columns: Name, Last modified, Size, and Description.
- A single entry: "Parent Directory" with a back arrow icon.
- A footer message: "Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.10.20 Port 80".

`http://192.168.10.20/dav`

```
#nc 192.168.10.20 80
DELETE /dav/a.txt HTTP/1.1
HOST: 192.168.10.20
```

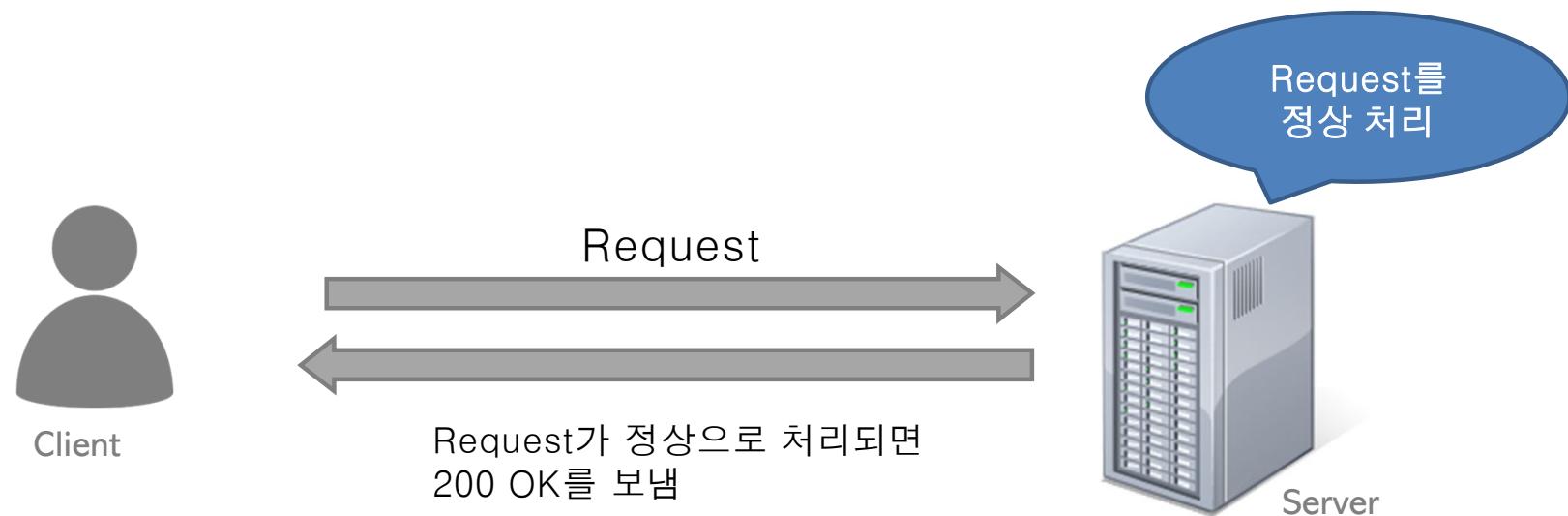
```
root@metasploitable:/var/www# ls
bad.php  dav  index.php  phpinfo.php  test      tikiwiki-old
dav     html  multilidæ  phpMyAdmin  tikiwiki  twiki
root@metasploitable:/var/www# cd dav
root@metasploitable:/var/www/dav# ls -l
total 0
root@metasploitable:/var/www/dav#
```

```
#ls /var/www/dav
```

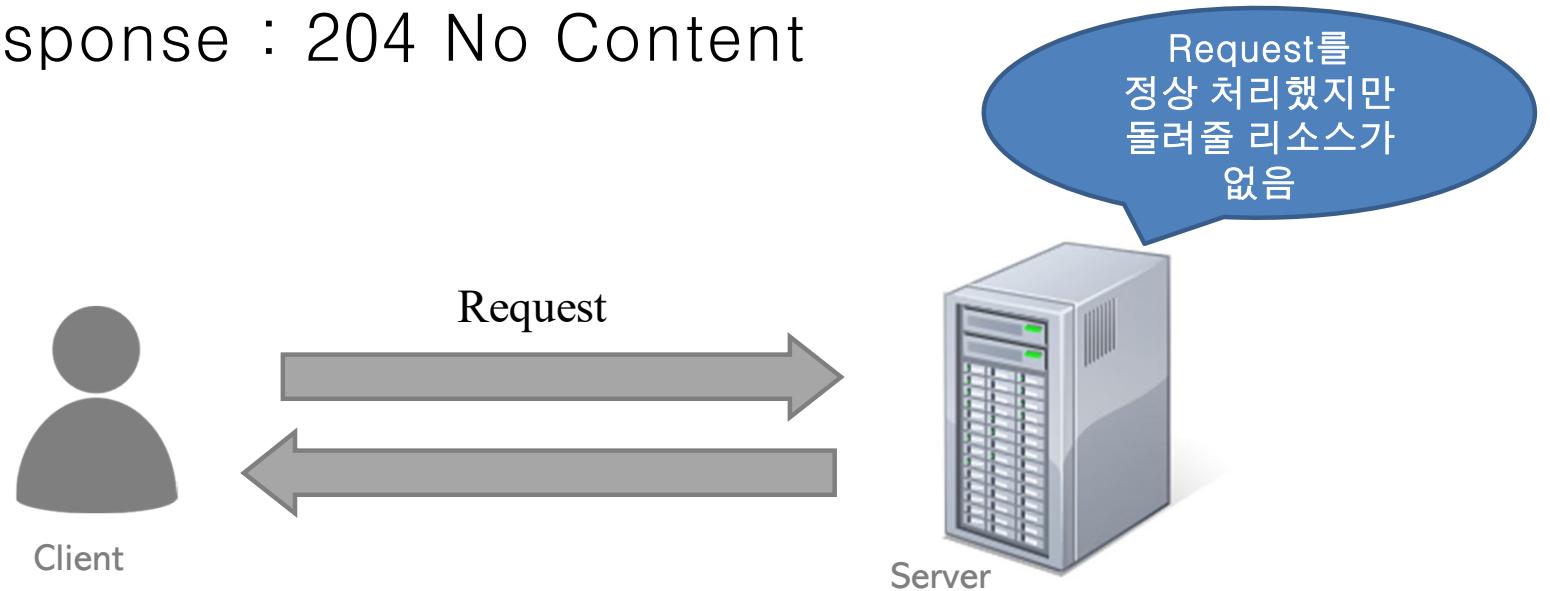
4) HTTP Response Code

실행 결과 코드	내용	설명
100번대	정보 전송	HTTP 1.0까지는 계열에 대한 정의가 이루어지지 않았기 때문에 실험 용도 외에는 100번대 서버 측의 응답이 없다.
200번대	성공	클라이언트의 요구가 성공적으로 수신 및 처리되었음을 의미한다.
300번대	리다이렉션	해당 요구 사항을 처리하기 위해 사용자 에이전트가 수행해야 할 추가적인 동작이 있음을 의미한다.
400번대	클라이언트 측 에러	클라이언트에 오류가 발생했을 때 사용한다. 예를 들면 클라이언트가 서버에 보내는 요구 메시지를 완전히 처리하지 못한 경우 등이다.
500번대	서버 측 에러	서버 자체에서 발생한 오류 상황이나 요구 사항을 제대로 처리할 수 없을 때 사용한다.

① HTTP Response : 2XX 성공(Success)



② HTTP Response : 204 No Content

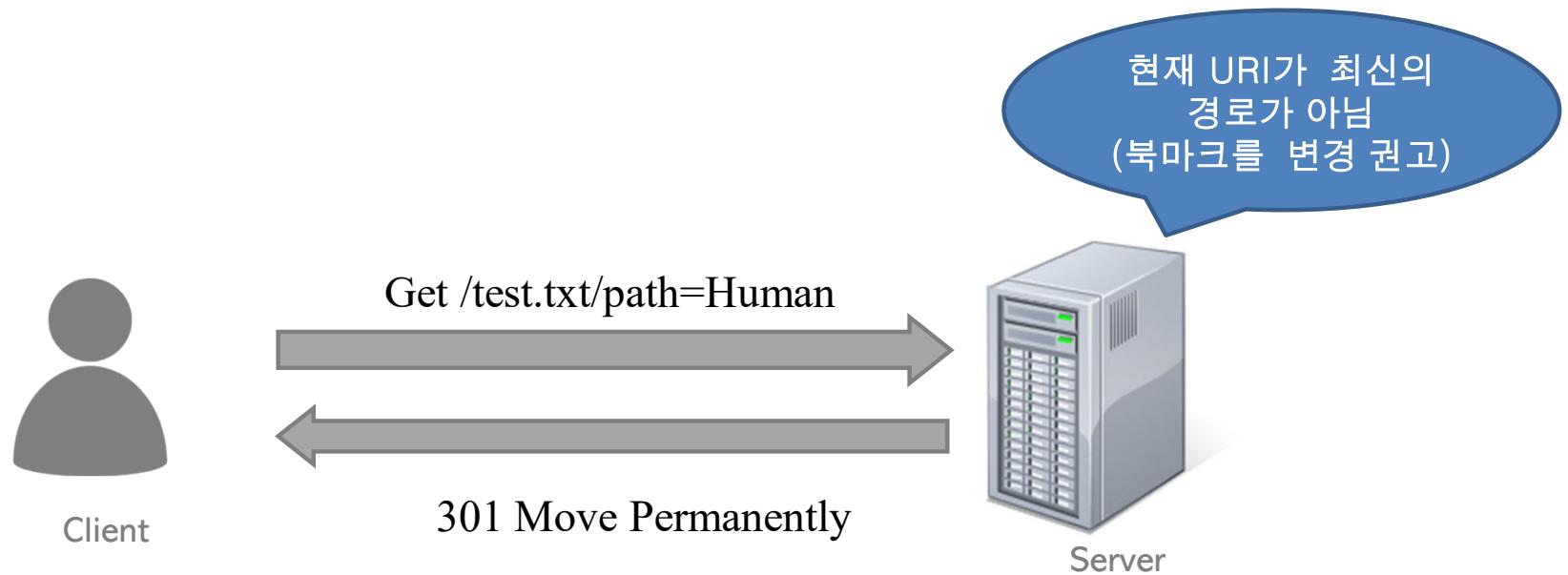


- Request를 받아서 처리하는데 성공했지만 Response에 엔티티 바디를 포함하지 않음
 - 브라우저에서 Request를 보낸 후 204 Response를 수신했어도 표시되어 있는 화면은 변하지 않음
- 클라이언트에 대해 새로운 정보를 보낼 필요가 없는 경우 사용
 - 서버에서 정상적인 변경 또는 삭제 처리가 이루어졌지만 새롭게 보일 정보가 없다는 것을 의미

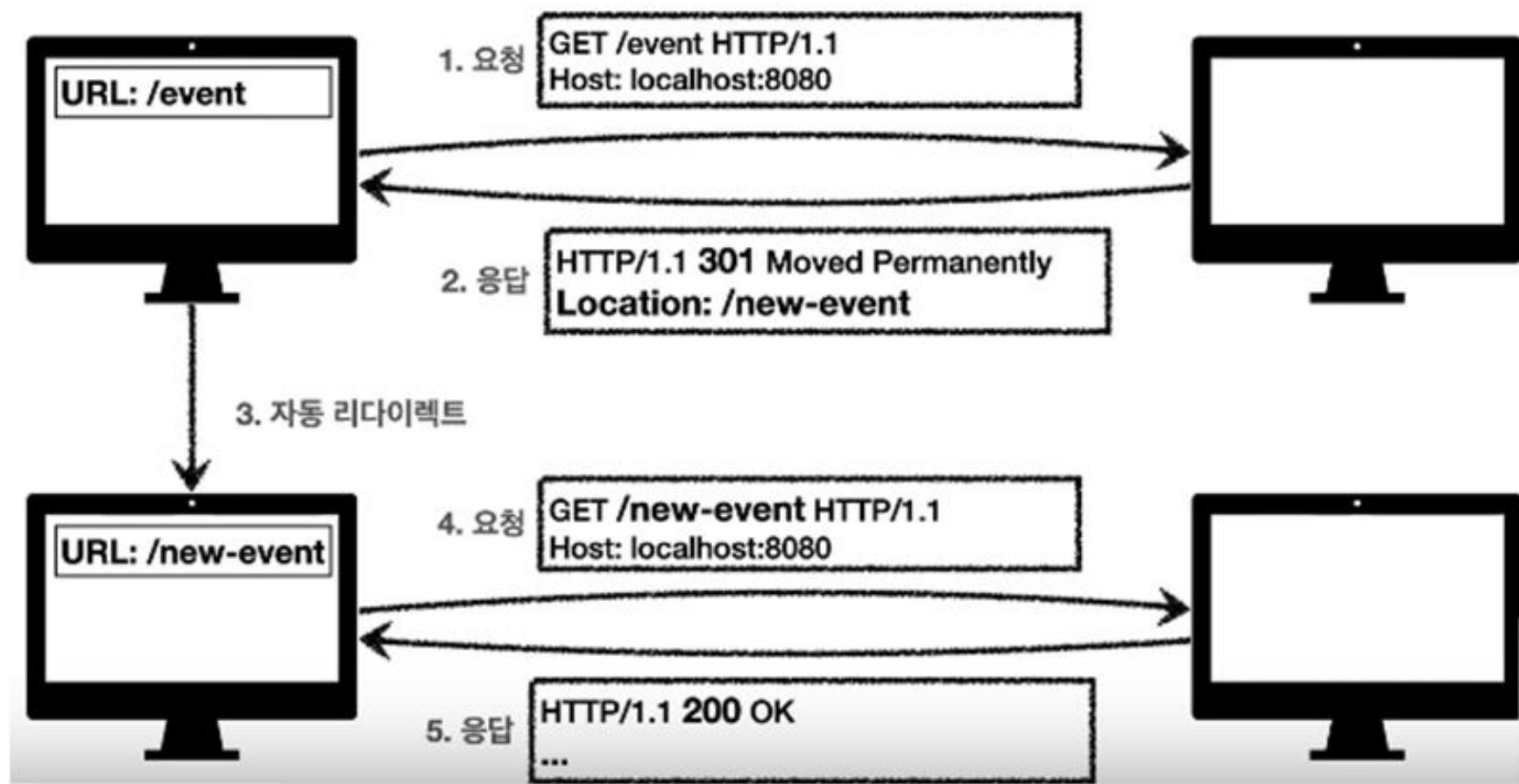
HTTP Response : 3XX Redirection

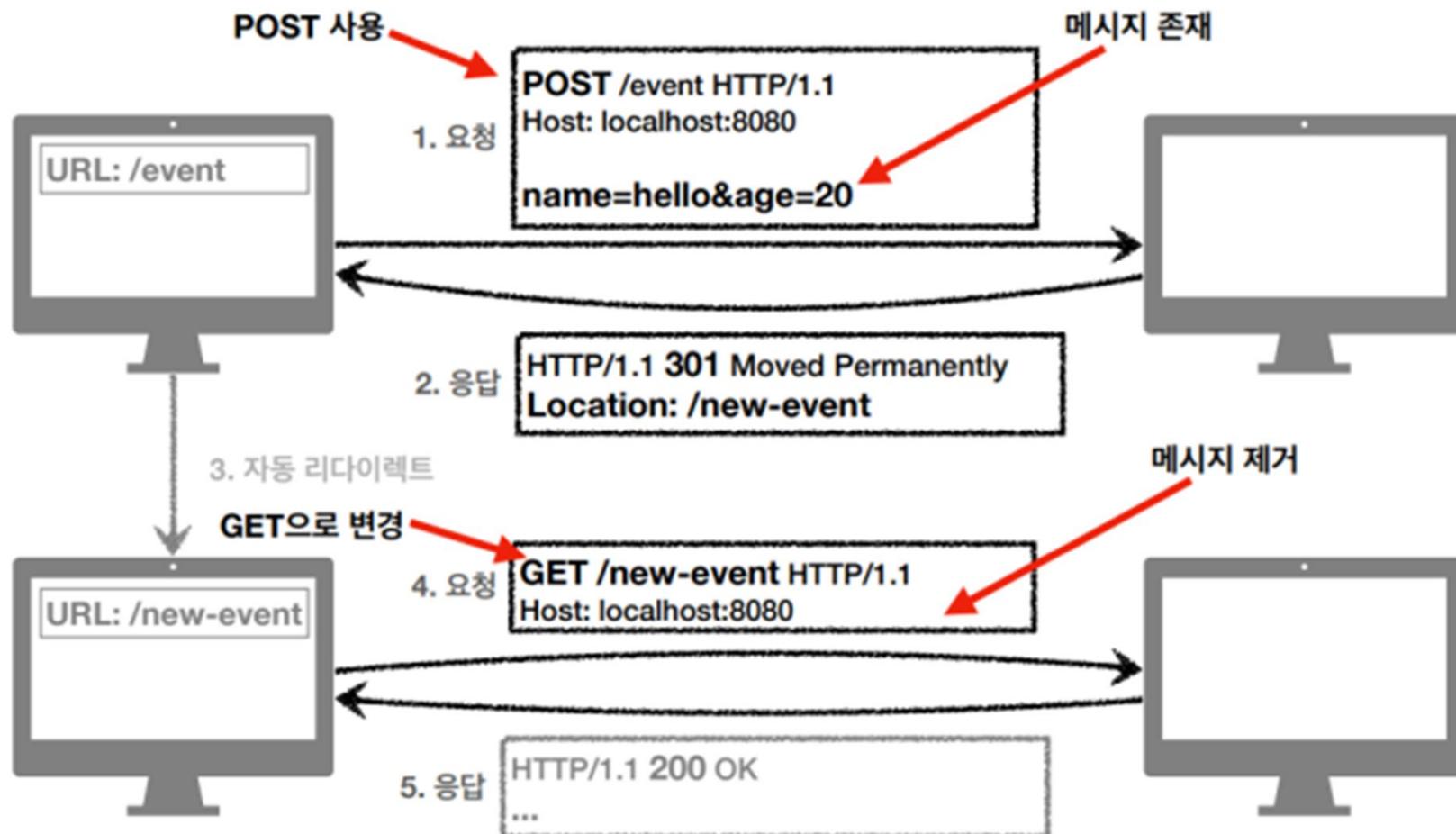
- 3XX Response는 Request가 정상적으로 처리를 종료하기 위해 브라우저 측에서 특별한 처리를 수행해야 함을 나타냄
- 웹 브라우저는 3xx 응답의 결과에 Location 헤더가 있으면, Location 위치로 redirect

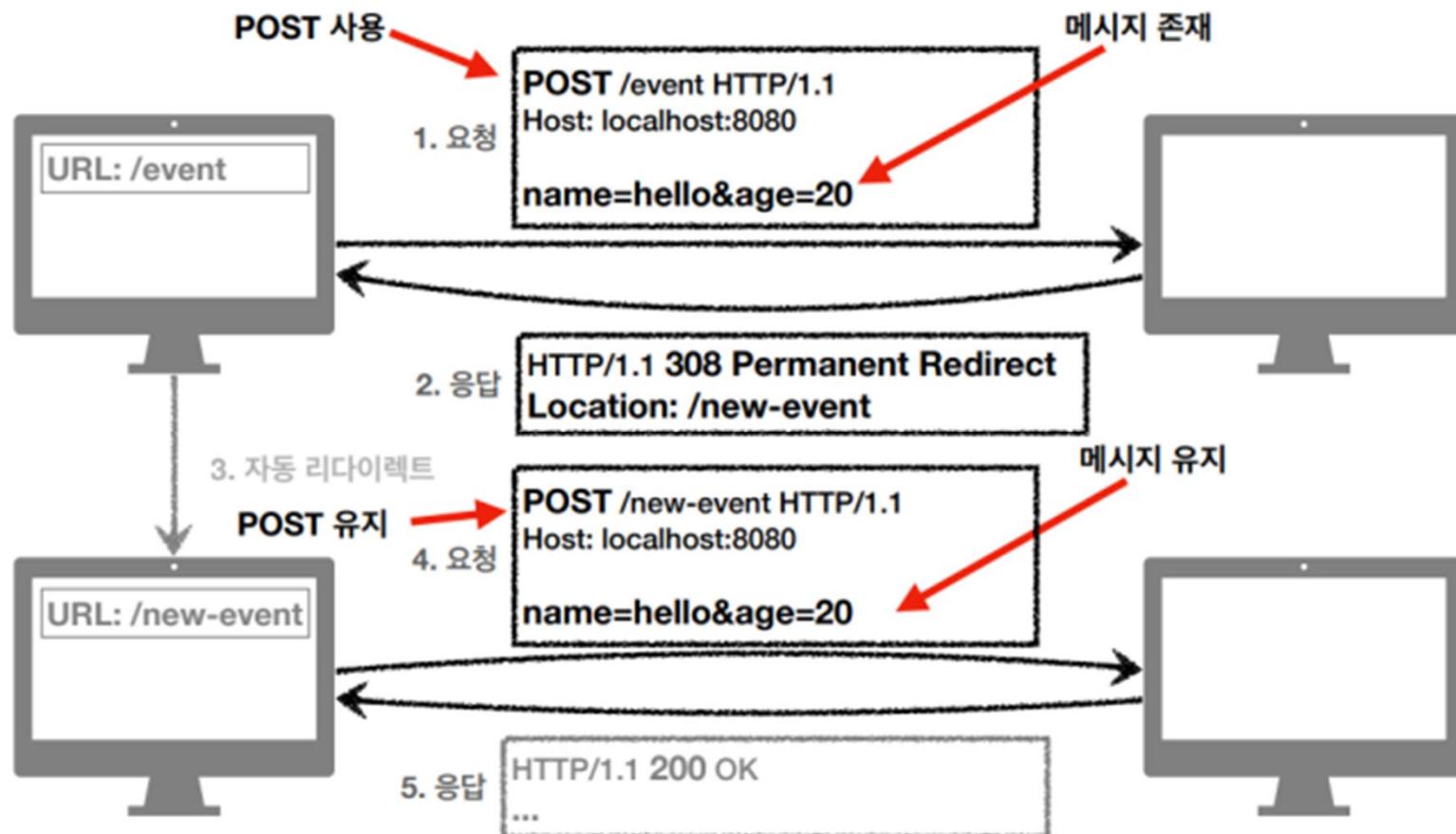
③ HTTP Response : 301 Moved Permanently



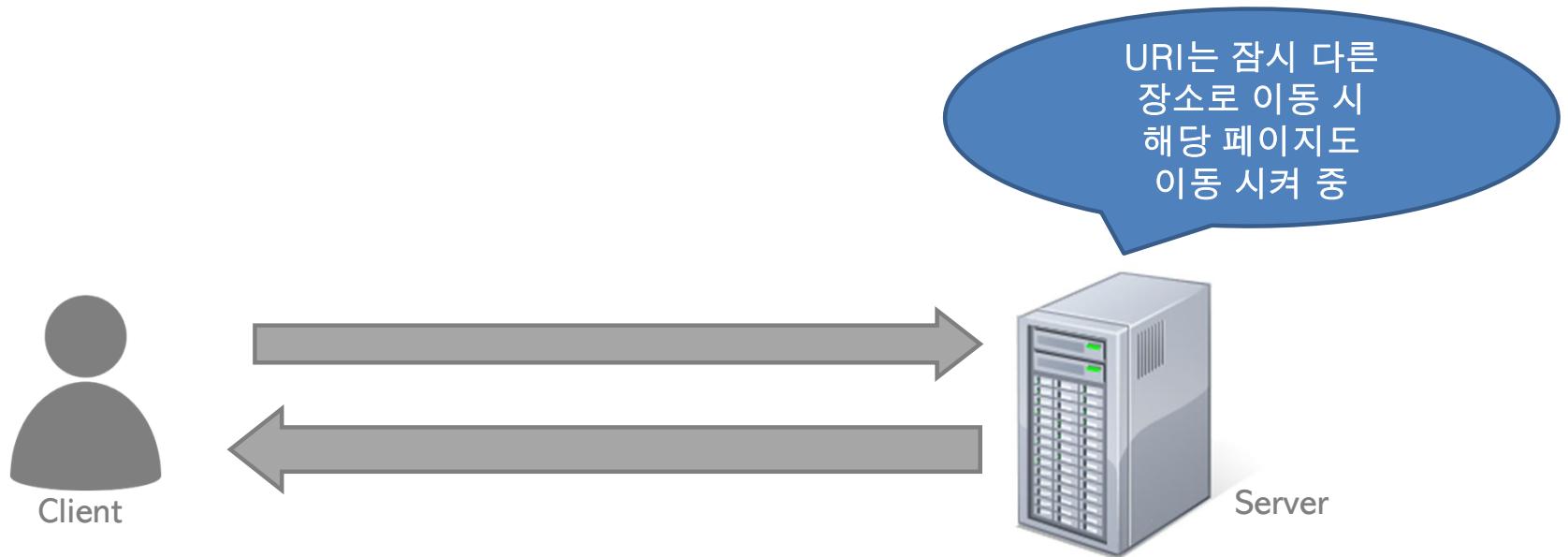
- Request된 리소스에는 새로운 URI가 부여되어 있기 때문에 새로운 URI를 사용해야 한다는 것을 알림
- 북마크하고 있는경우 Location 헤더 필드에서 가리키고 있는 URI로 북마크 권고



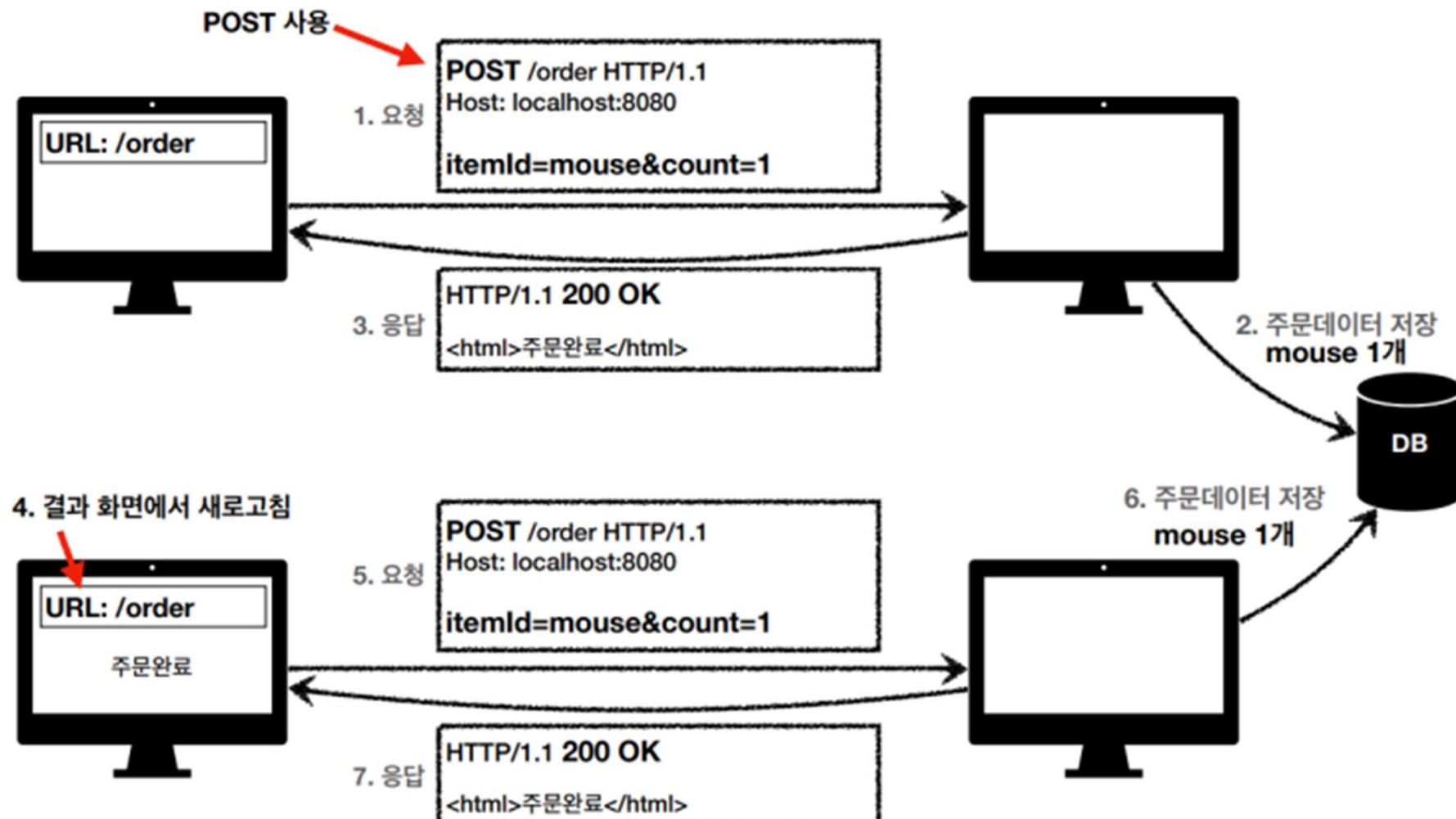




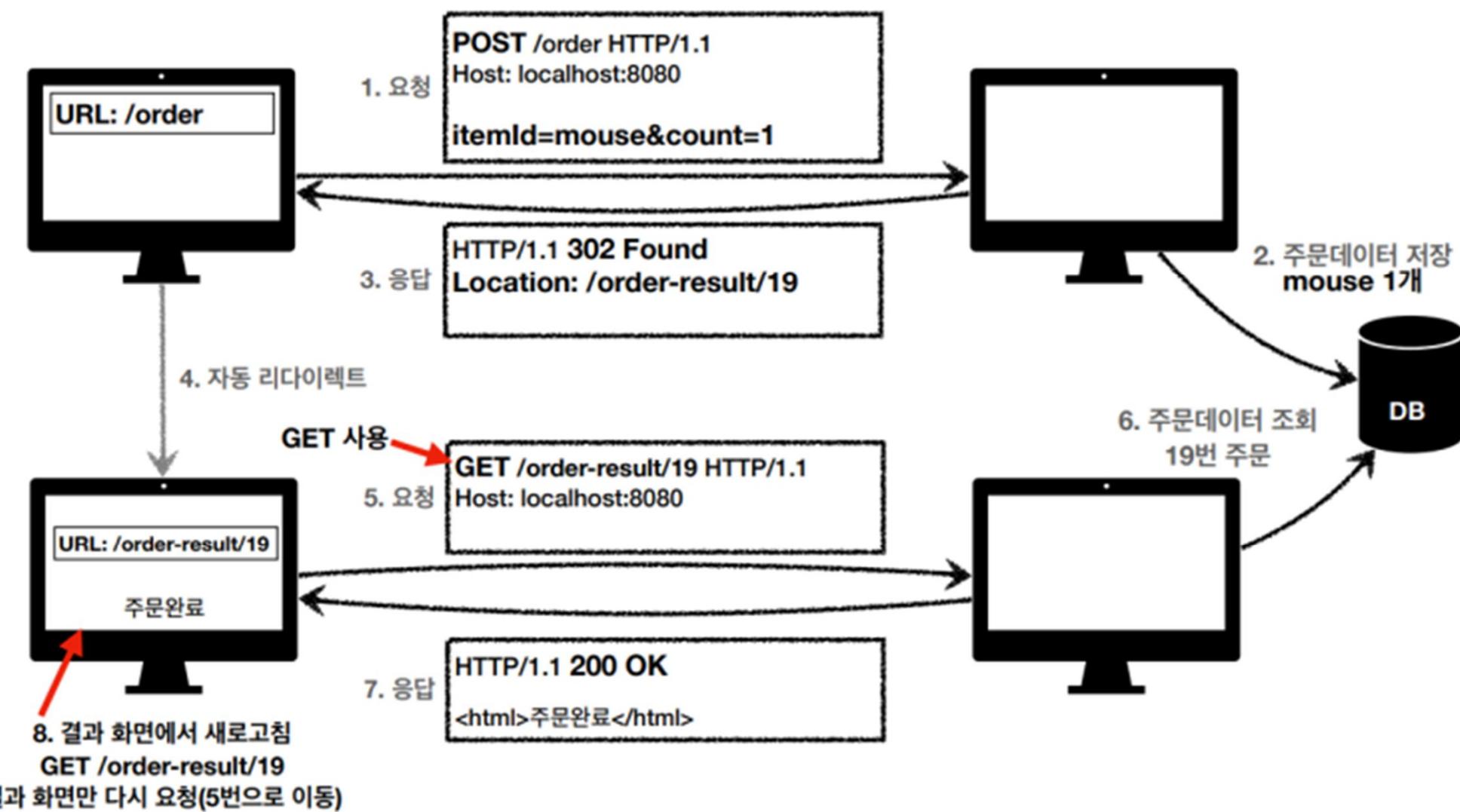
④ HTTP Response : 302 Found



- 새로운 URI가 할당되어 있기 때문에 그 URI를 사용하도록 권고
- 301과 비슷하지만 302는 일시적인 이동



<< redirection 302 사용 전 >>



302 Found

- 리다이렉트 시 요청 메서드가 GET으로 변하고, 본문이 제거될 수 있음

307 Temporary Redirect

- 302와 기능 동일
- 리다이렉트 시 요청 메서드와 본문 유지(요청 메서드를 변경하면 안 됨)

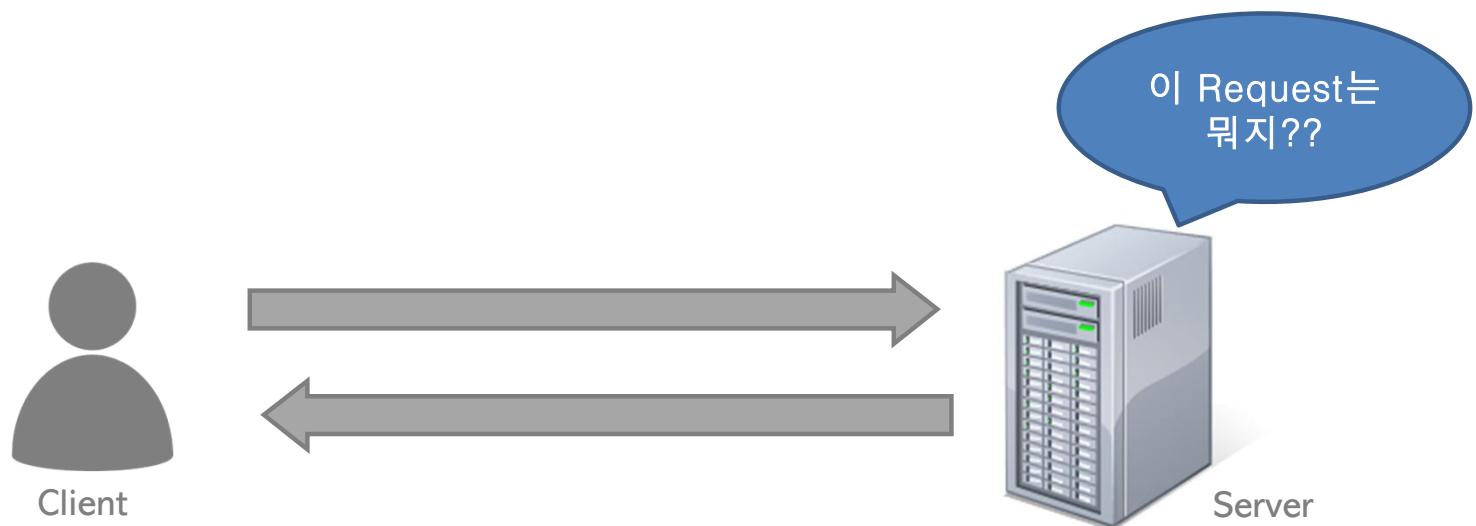
303 See Other

- 302와 기능 동일
- 리다이렉트 시 요청 메서드가 무조건 GET으로 변경

304 Not Modified

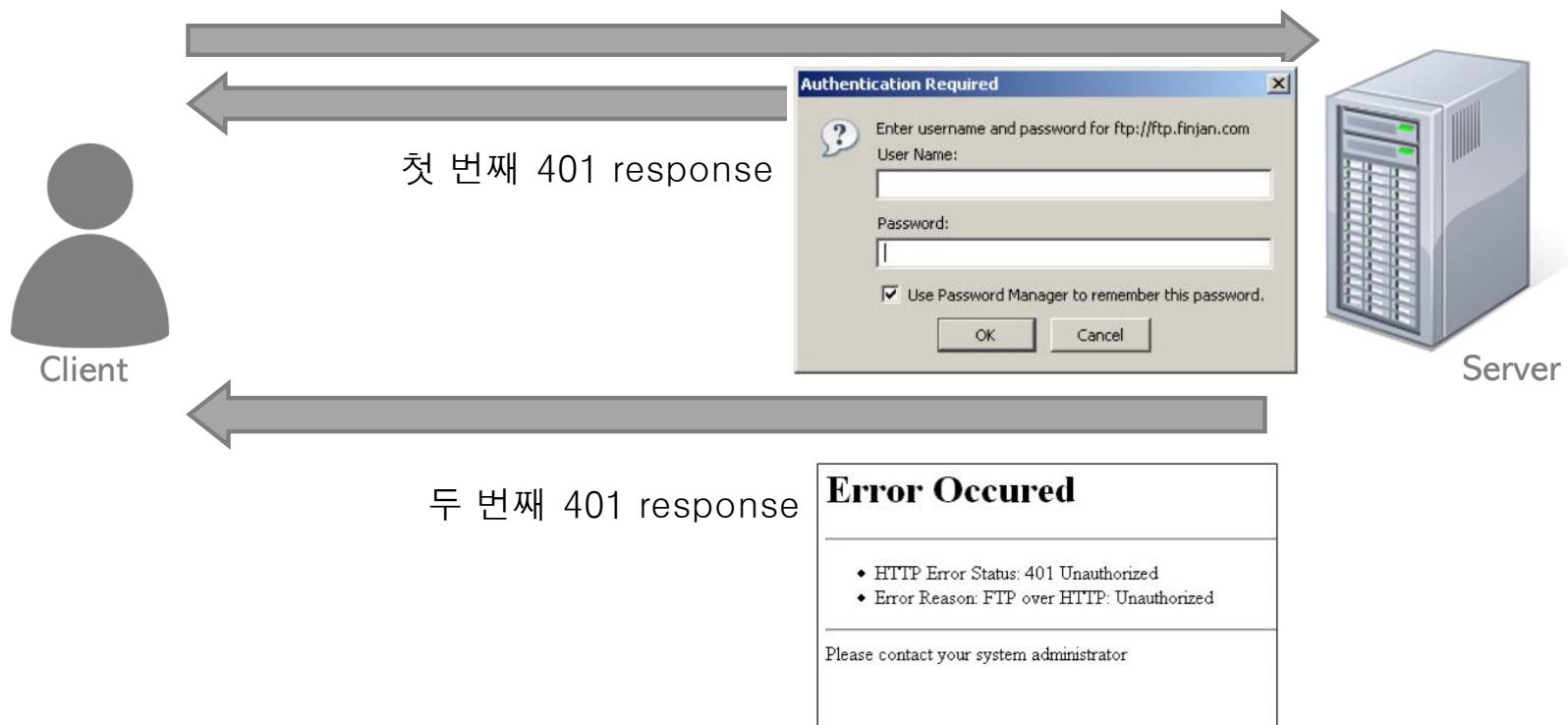
- 캐시를 목적으로 사용
- 클라이언트에게 리소스가 수정되지 않았음을 알려줌
- 따라서 클라이언트는 로컬 PC에 저장된 캐시를 재사용
- 304 응답은 응답에 메시지 바디를 포함하면 안 됨
- 조건부 GET, HEAD 요청 시 사용

⑤ HTTP Response : 400 Bad Request



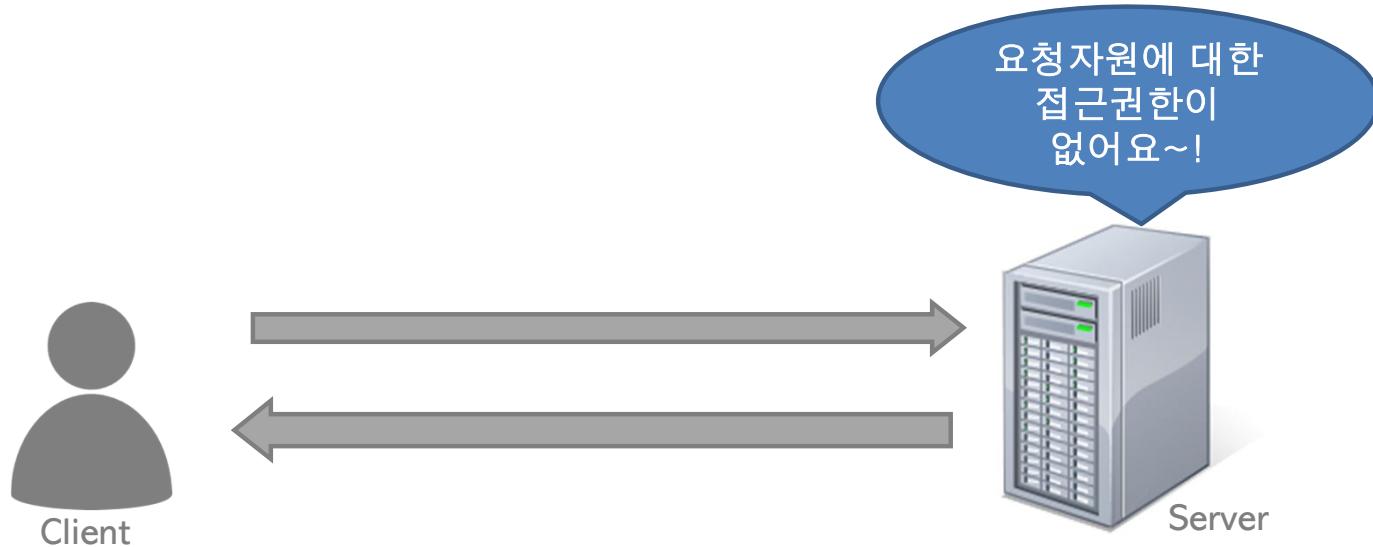
- Request 구문이 잘못되었음 나타냄
- 이 오류가 발생 시 request 내용을 재검토하고 재송신할 필요가 있음

⑥ HTTP Response : 401 Unauthorized



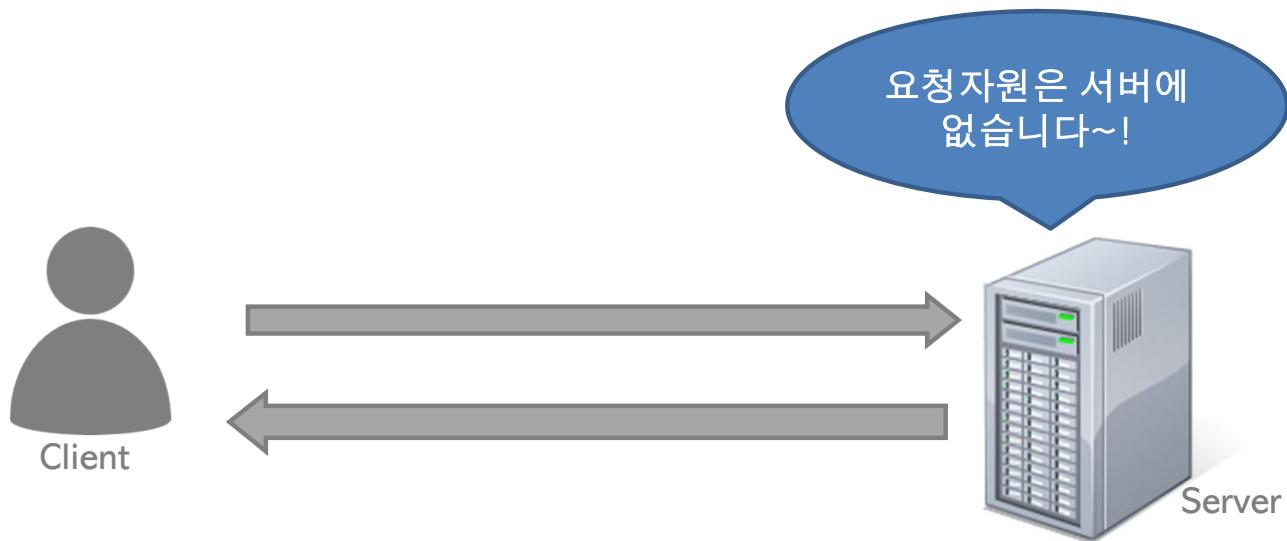
- 401 는 송신한 request에 인증 정보가 필요하다는 것을 나타냄

7 HTTP Response : 403 Forbidden



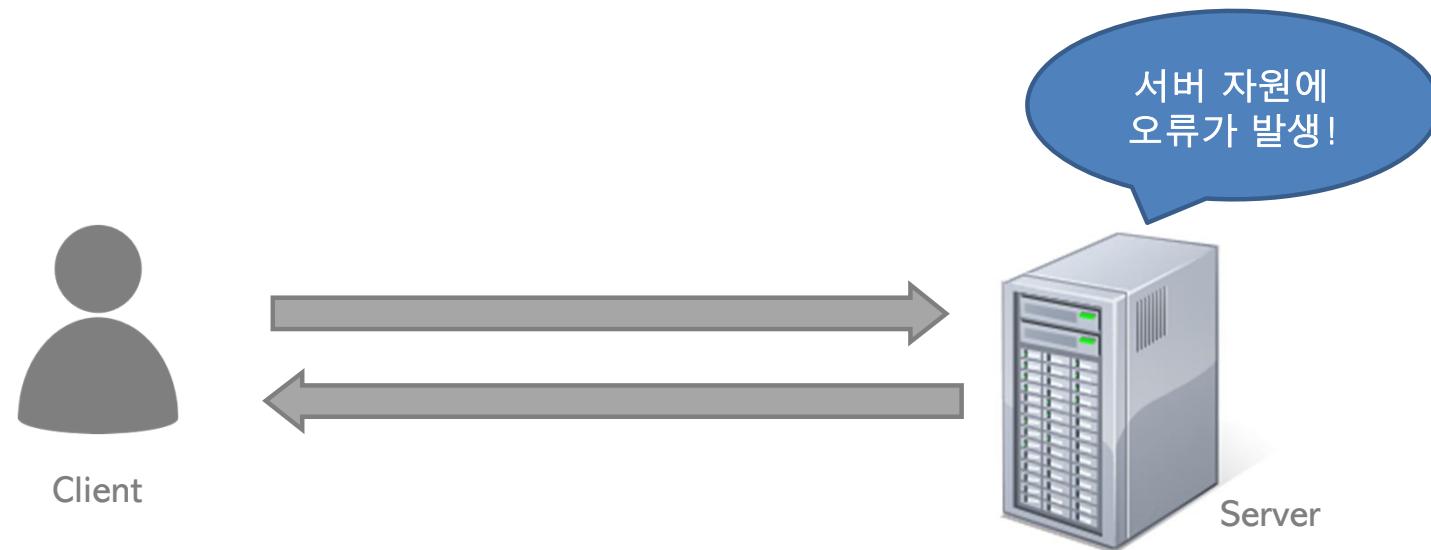
- 403 은 요청된 resource의 접근 거부되었음을 의미함
 - 파일 시스템의 권한이 부여되지 않은 경우
 - 액세스 권한(허가되지 않은 송신지 IP주소의 접근 등) 제한이 될 경우
- 서버 측은 거부의 이유를 분명히 할 필요가 있음
- 이유를 명확하게 할 경우에는 body에 기재해서 클라이언트측에 표시

8 HTTP Response : 404 Not Found



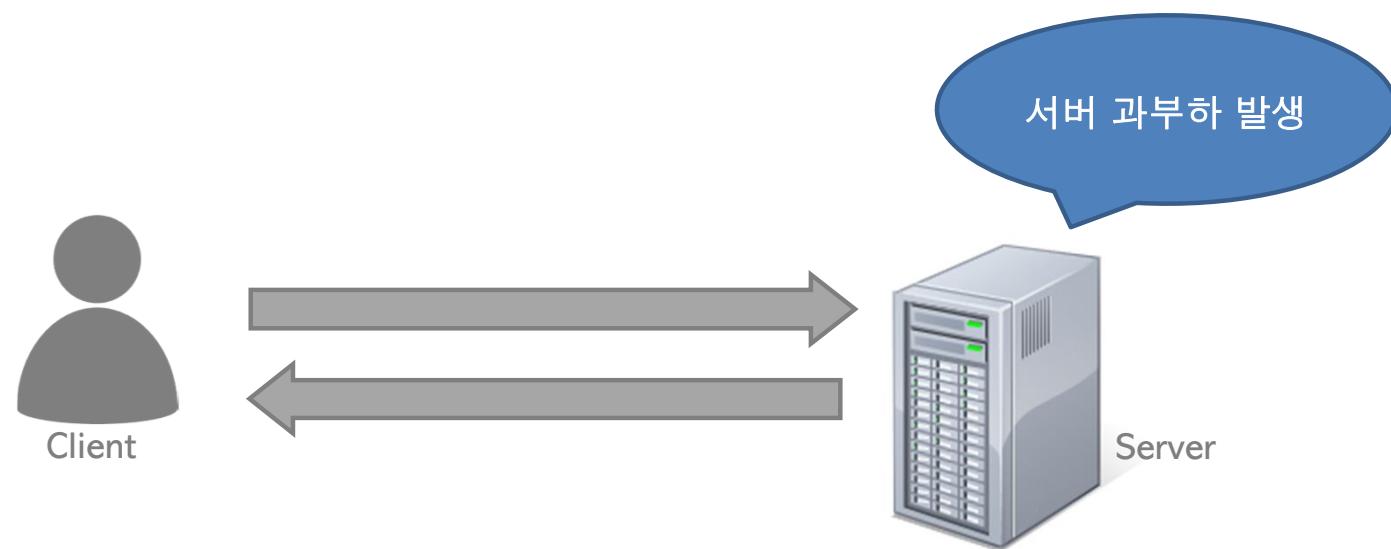
- Request한 resource가 서버 상에 없음을 나타냄

⑨ HTTP Response : 500 Internal Server Error



- Request 처리 도중에 오류가 발생했음을 나타냄
- 웹 애플리케이션에 에러가 발생한 경우나 일시적인 경우도 있음

⑩ HTTP Response : 503 Service Unavailable



- 일시적으로 서버가 과부하 상태이거나 점검 중임을 나타냄

2.4 일반 request 헤더 & response 헤더

① User-Agent

- 클라이언트 소프트웨어를 식별할 수 있는 header
(예) 웹브라우저의 이름과 버전이 표시, 웹 브라우저가 실행 중인 시스템 환경 등
- 웹 요청을 수행하는 클라이언트들은 user-agent에 자신의 정보를 보내어, 웹 애플리케이션이 header의 내용을 확인하고 header에 따라 다르게 동작 할 수 있도록 해 줌
- User-Agent header의 값을 확인하여 검색 엔진을 제어 할 수 있음
- User-Agent header에 따라 서버의 동작이 달라짐
 - 어떤 앱들은 호환성을 위해 다른 User-agent 정보를 사용하기도 함
- 악성 봇들이나 자동화 프로그램도 가짜 user-agent header를 사용하는 경우가 많음
- User agent spoofing : User-Agent를 조작하는 것

② Accept: text/html, application/xhtml+xml

- 클라이언트가 어떤 컨텐츠 타입을 처리할 수 있는지 서버에게 알려줌
- 여러 개의 값이 전달, 서버는 이 헤더에 저장된 타입 중에서 하나는 정하여 응답

③ Accept-Language:en-US,

- 클라이언트가 처리할 수 있는 언어 정보를 서버에게 알려줌

④ Accept-Encoding

- 클라이언트가 처리할 수 있는 인코딩 방식이나 압축 알고리즘에 대한 정보를 알려줌

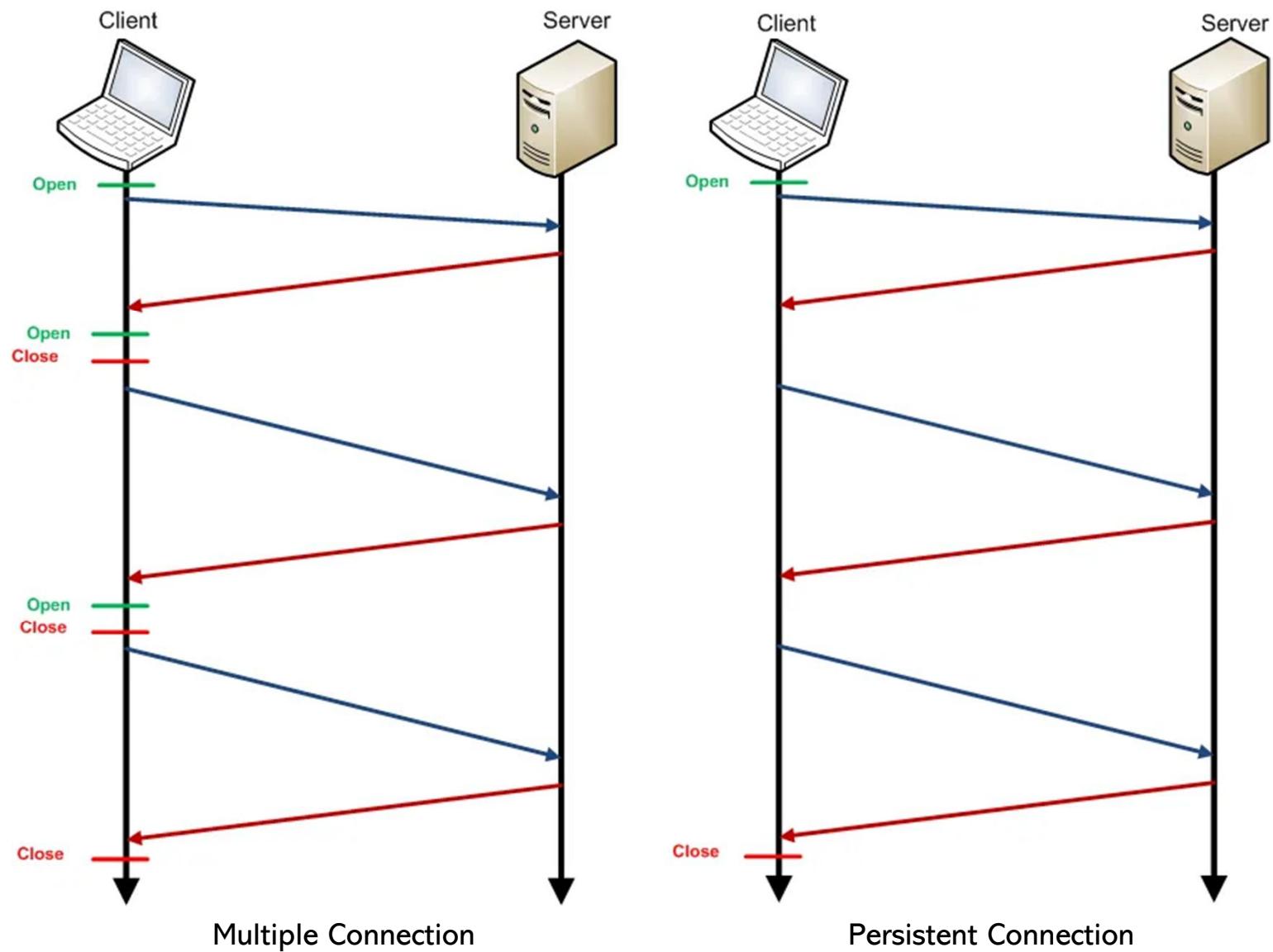
⑤ Content-Type, Content-Length

- Body가 존재하는 경우 Body의 종류와 길이를 알려주는 header
- 요청 메시지와 응답 메시지 둘 다 사용할 수 있음

Application/x-www-form-urlencoded	아스키 문자로 표현 할 수 있는 데이터 전송 시 사용 여러 개의 문자를 보낼 때 &로 구분 가장 많이 쓰이는 형태 (예) username=admin&password=password &user_token=4b39
Multipart/form-data	파일 등의 바이너리 데이터를 전송 시 사용
Application/json	JSON 데이터를 전송 시 사용
Text/xml	XML 데이터를 전송 시 사용

⑥ Connection: HTTP 1.1 Keep-Alive

- HTTP는 connectionless 방식으로 연결을 매번 종료하고 새로운 연결하는 구조
- 네트워크 비용측면에서 큰 비용을 소비하는 구조
- HTTP 1.1부터는 keep-alive 기능을 지원
- Keep-Alive
 - 연결된 HTTP socket에 IN/OUT 의 access가 마지막으로 종료된 시점부터 정의된 시간까지 access가 없더라도 대기하는 방식
 - 정의된 시간 내에 access가 이뤄진다면 계속 연결된 상태를 유지할 수 있음



HTTP/1.1 200 OK

Connection : Keep-Alive

Keep-Alive : timeout=5, max=1000

- max(max keepalive requests)

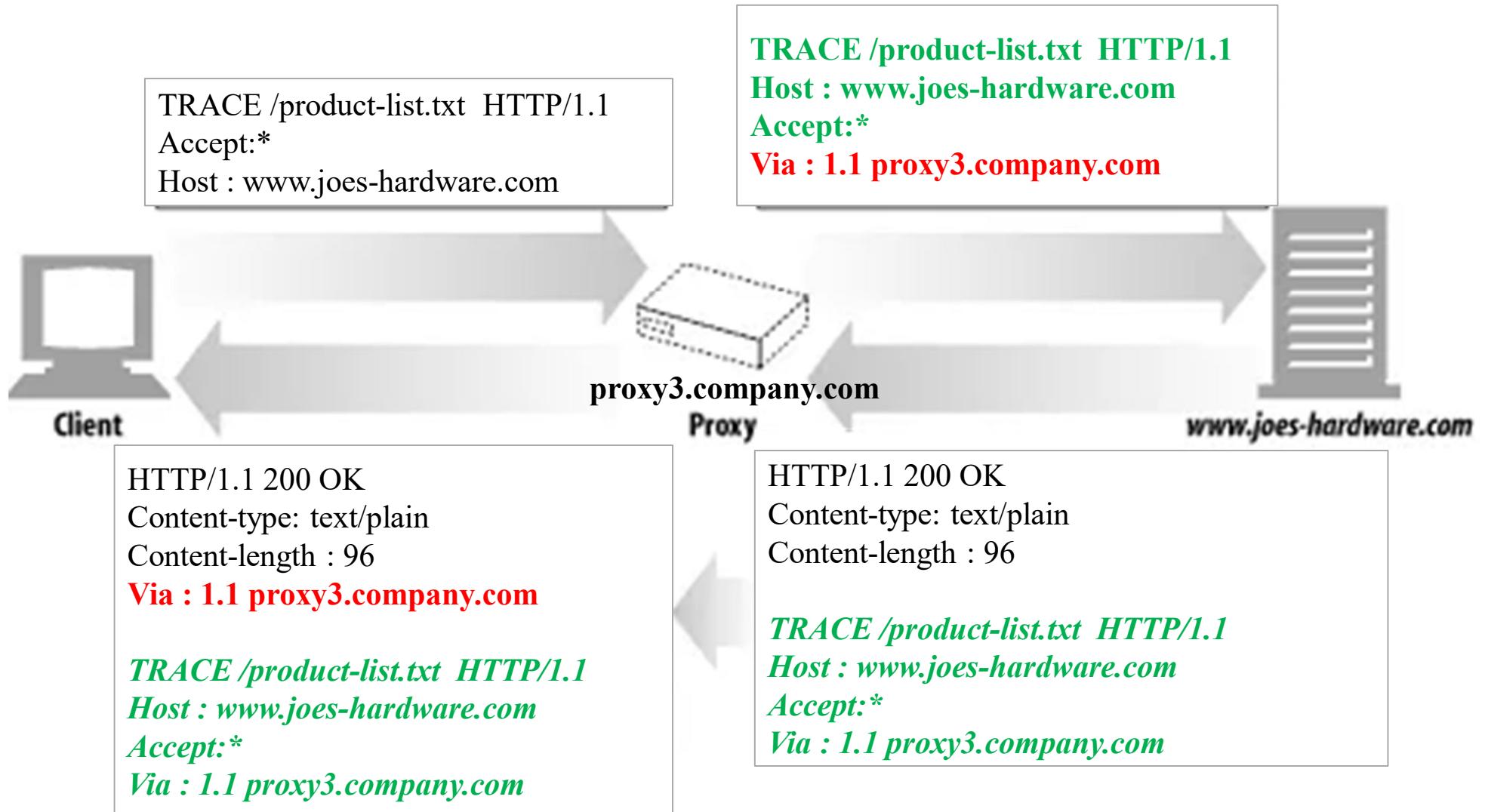
- keepalive connection을 통해서 주고 받을 수 있는 request의 최대 개수
 - 이 수보다 더 많은 요청을 주고 받을 경우에는 connection이 close

- timeout(keepalive timeout)

- connection이 idle 한 채로 얼마동안 유지 될 것인가를 의미
 - 이 시간이 지날 동안 request가 없을 경우 connection close

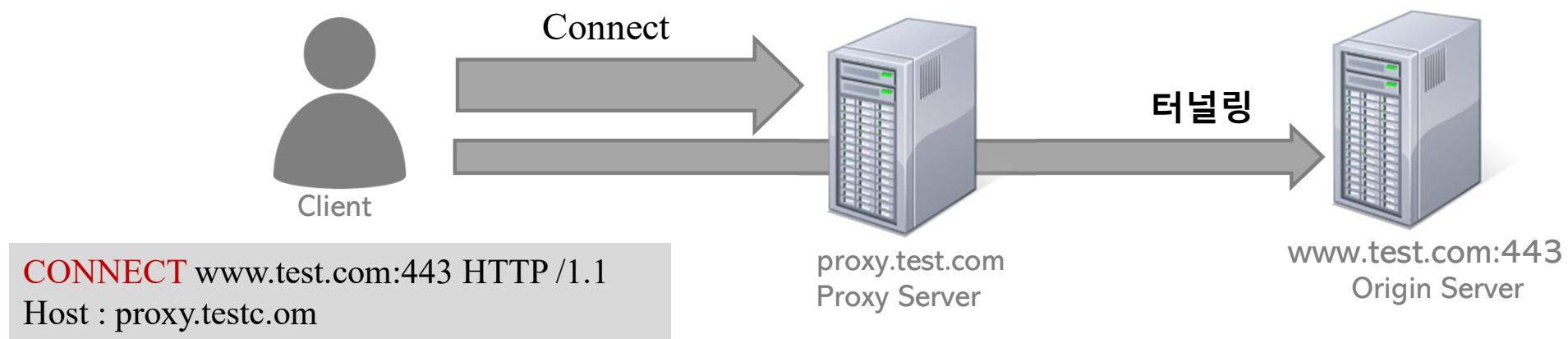
7 HTTP Via

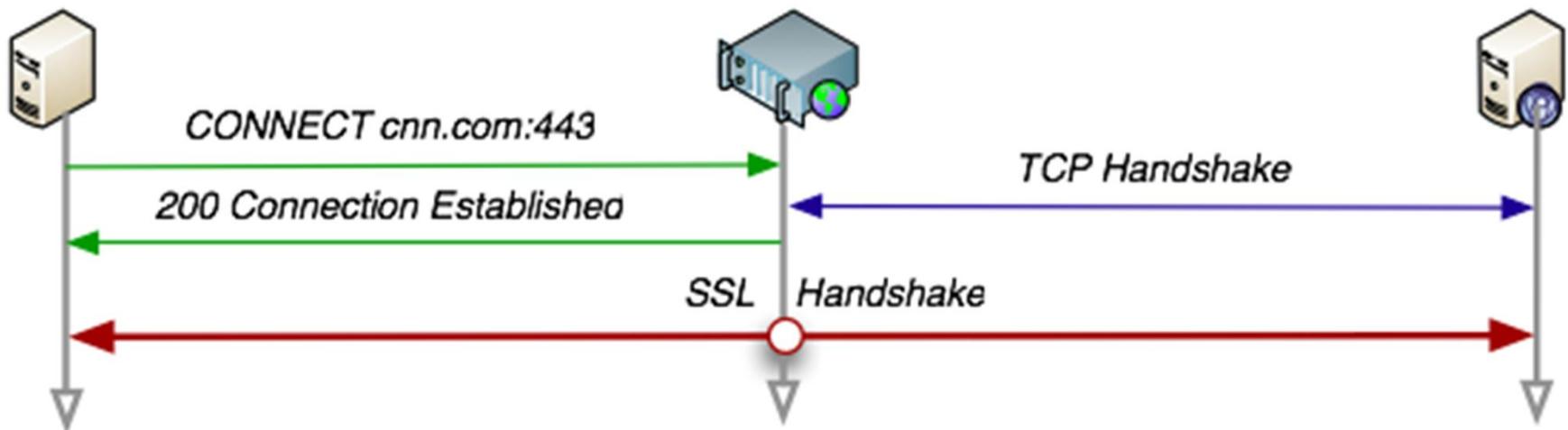
- 정방향 및 역방향 프록시에 의해 추가
 - 요청헤더와 응답 헤더에 나타날 수 있음
- Forward 메시지를 추적하거나, 요청 루프 방지, 요청과 응답 체인에 따라 송신자의 프로토콜 정보를 식별

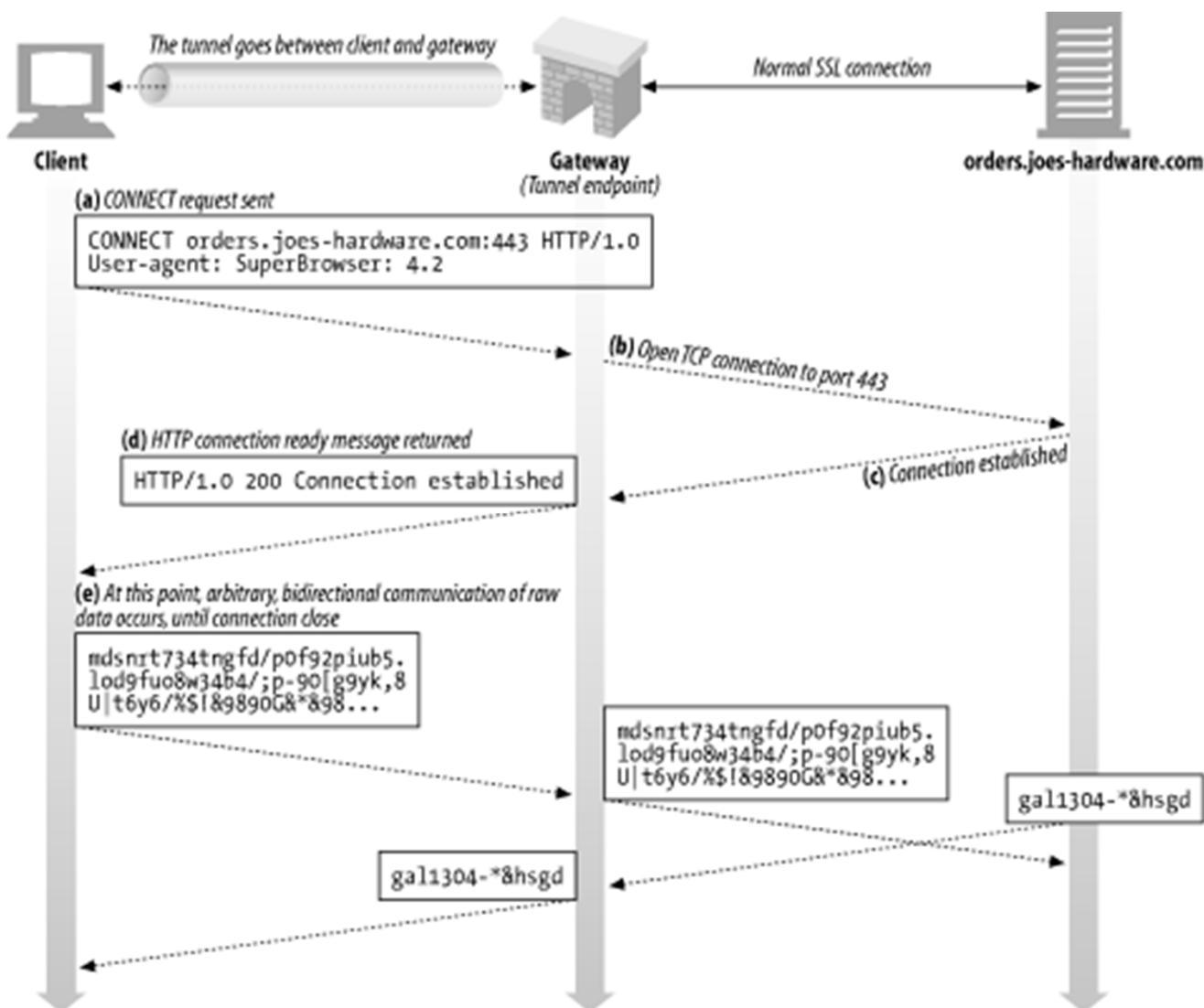


8 HTTP Request : Connect 방식

- Proxy와 같은 중계 서버와 사용 시 사용
 - Client가 Proxy를 통해 Server와 SSL 통신 시 사용
- 프록시에 터널 접속 확립을 요청함으로써 TCP 통신을 터널링 하기 위해 사용
- SSL 또는 TLS 등의 프로토콜로 암호화 된 것을 터널링 시키기 위해 사용
- 형식 : CONNECT 프록시서버:포트 HTTP버전





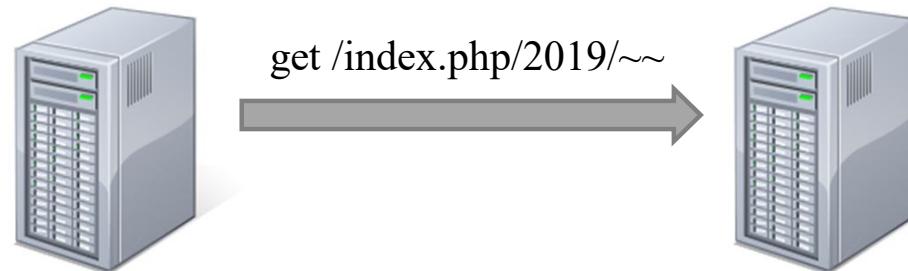


9 HTTP Referer

- 현재 사이트에 유입된 경로 표시 (이전 페이지의 URI 정보)
- 현재 표시하는 웹 페이지가 어떤 웹 페이지에서 요청되었는지 알 수 있음
 - 어떤 웹사이트나 웹 서버에서 방문자가 왔는지를 파악 할 수 있는 기능
- 페이지를 요청한 이전 페이지가 무엇인지를 알려주는 정보
 - 우리 사이트로의 유입이 어떤 검색서비스를 이용한 것인지 알고자 할 때 사용



(예) `http://www.A.com/1.html`이라는 웹페이지에 있는 **링크를 클릭하여**
`http://www.B.com/2.html`으로 이동했을 때 referer는 `http://www.A.com/1.html` 가 됨



```
GET /index.php/2019/12/06/who-wore-it-best-test/ HTTP/1.1
Host: 192.168.78.157
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
Referer: http://192.168.78.157/index.php/2019/12/
Connection: close
Upgrade-Insecure-Requests: 1
```

2.5 Cache Header

- 브라우저에 응답으로 온 HTML이나 JSON같은 데이터가 저장되어 나중에 서버에 요청을 보내지 않고도 브라우저에 저장된 응답을 사용할 수 있음
- 캐싱은 GET 요청에만 함
- Cache control

Cache-Control: no-store	
Cache-Control: no-cache	캐시를 쓰기 전에 서버에 캐시 사용 유무 확인 작업을 걸침
Cache-Control: must-revalidate	만료된 캐시만 서버에 확인
Cache-Control: public 또는 private	<ul style="list-style-type: none">- public이면 공유 캐시(또는 중개 서버)에 저장해도 됨- private이면 브라우저 같은 특정 사용자 환경에만 저장
Cache-Control: public, max-age=3600	<ul style="list-style-type: none">- 캐시 유효시간 (초 단위이므로 위 예제에서는 1시간)- 1시간이 지나면 이 응답 캐시는 만료된 것으로 간주

Age

- 캐시 응답 때 나타냄
- max-age 시간 내에서 얼마나 흘렀는지 초 단위로 알려줌
(예) 1분이 지나면 Age:60 이 캐시 응답 헤더에 포함

Expires

- 응답 컨텐츠가 언제 만료되는지를 나타냄
- max-age 시간 내에서 얼마나 흘렀는지 초 단위로 알려줌
(예) 1분이 지나면 Age:60 이 캐시 응답 헤더에 포함

ETag

- HTTP 컨텐츠가 바뀌었는지를 검사할 수 있는 태그
- 같은 주소의 자원이더라도 컨텐츠가 달라졌다면 ETag가 다름
- ETag 가 달라지면 서버가 클라이언트의 응답 내용이 변함을 감지하게 되어 캐시를 지우고 새로 컨텐츠를 내려 받을 수 있게 됨

If-None-Match

- 서버에게 Etag가 달라졌는지 검사해서 Etag가 다를 경우에만 컨텐츠를 새로 전달해 달라는 것
- ETag가 같다면 서버는 **304 Not Modified**를 응답해서 캐시를 그대로 사용하게 함

2.6 보안 헤더

① Max-Forwards header

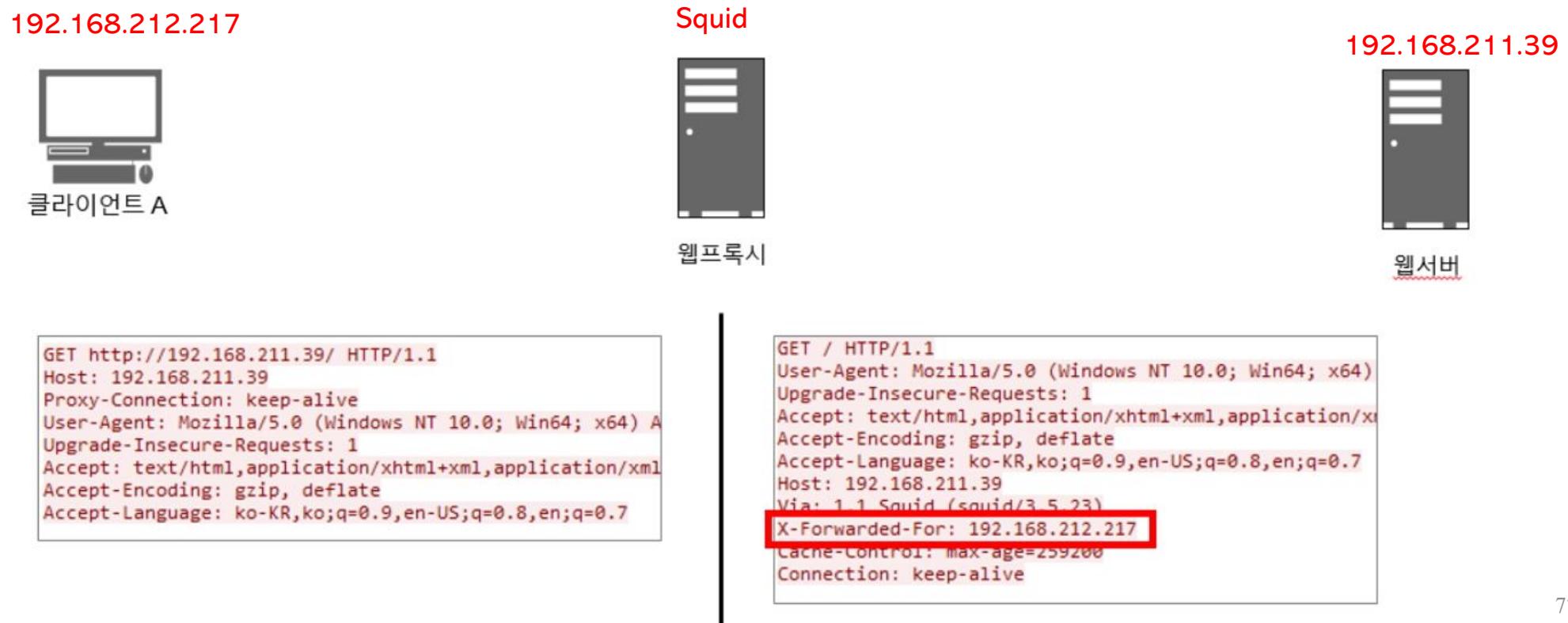
- Max-Forwards 요청 HTTP 헤더는 요청이 **통과하는 노드**(일반적으로 프록시)의 수를 제한하기 위해 TRACE 메서드와 함께 사용
- 이 값은 방문해야 하는 최대 노드 수를 나타내는 정수 값
- 노드에서 이 값은 감소하고 목적지에 도달하거나 Max-Forwards의 수신 값이 0이 될 때까지 TRACE 요청이 다음 노드로 전달
- TRACE 요청에 Max-Forwards 헤더가 없으면 노드는 최대 포워드 수가 없는 것으로 간주함

② X-Forwarded-For(XFF)

- HTTP Proxy나 로드 밸런서를 통해 웹서버에 접속하는 클라이언트의 IP주소를 식별하는 표준 헤더
- 클라이언트와 서버 중간에서 트래픽이 Proxy나 로드밸런서를 걸치면 서버 접근 로그에는 Proxy나 로드 밸런서의 IP주소만기록
- 클라이언트의 IP주소를 보기 위해 XFF요청헤더를 사용
 - 클라이언트의 IP 노출은 민감한 개인 정보 노출과 연관됨
 - XFF를 사용시 프라이버시를 주의해야 함
- XFF 헤더는 디버깅, 통계, 위치 종속적인 컨텐츠를 위해 사용
- X-Forward-For : <Client>, <Proxy1>, <Proxy2>

X-Forwarded-For(XFF)

- HTTP Server 에 요청한 client의 IP 를 식별하기 위한 표준
- Proxy(프록시) 환경에서 client IP 를 얻기



X-Forwarded-For(XFF)

- Apache httpd Web Server에서 XFF 사용방법

```
##수정 전
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

```
## 수정 후
```

```
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b" common
```

5 X-XSS-Protection

- Reflected XSS 공격이 탐지되었을 때, 웹 페이지가 로딩되는 것을 막아줌
- Internet Explorer, Chrome 및 Safari에서 제공하는 기능

X-XSS-Protection: 0

X-XSS-Protection: 1

X-XSS-Protection: 1; mode=block

X-XSS-Protection: 1; report=<reporting-uri>

- XSS 필터링을 비활성화
- XSS 필터링을 활성화 (디폴트) : XSS 공격이 감지되면 안전하지 않은 영역을 제거 후 랜더링
- 필터링을 사용 : 공격이 탐지되면 페이지 렌더링 중지
- 필터링을 사용 : 공격을 탐지하면 페이지 렌더링을 차단 후 위반 사황을 보고

⑥ X-Frame-Options

- <frame>이나 </iframe> 등을 사용하여 웹페이지가 출력되는 것을 제어
- 프레임을 이용하여 웹페이지가 표시될 수 있는 경우 클릭재킹(clickjacking) 공격에 노출 될 수 있음

⑦ X-XSS-Protection

- Reflected XSS 공격이 탐지되었을 때, 웹페이지가 로딩되는 것을 막아 줌

⑧ X-Content-Type-Options : nosniff

- MIME 스니핑을 차단하기 위해 사용하는 헤더
 - MIME 스니핑 (Content Sniffing)
 - 파일의 내용과 Content-type에 설정이 서로 다른 경우 파일의 내용으로부터 파일의 형식을 추측하여 실행하는 것
- (예) 공격자가 HTML 코드를 jpg, png 등 이미지 파일로 만들어 업로드
사용자의 웹 브라우저는 이미지 파일을 요청하더라도, **실제 파일의 내용이 HTML 형식인 것을 보고
HTML 코드를 실행**
- ➔ 공격자가 HTML 코드 내에 악성 자바스크립트를 삽입해두면 XSS 공격을 당할 위협이 있음

X-Content-Type-Options: nosniff

2.6 Cookie & Session

HTTP의 Stateless

- HTTP 프로토콜은 클라이언트의 상태정보를 유지 않음
- 동일 클라이언트의 현재 요청과 이전 요청을 식별하지 못함
- 클라이언트의 상태 정보 유지가 필요한 경우(로그인 상태 정보 유지, 장바구니 등) 클라이언트 기술로는 쿠키(cookie), 서버 기술로 세션(session)이 있음

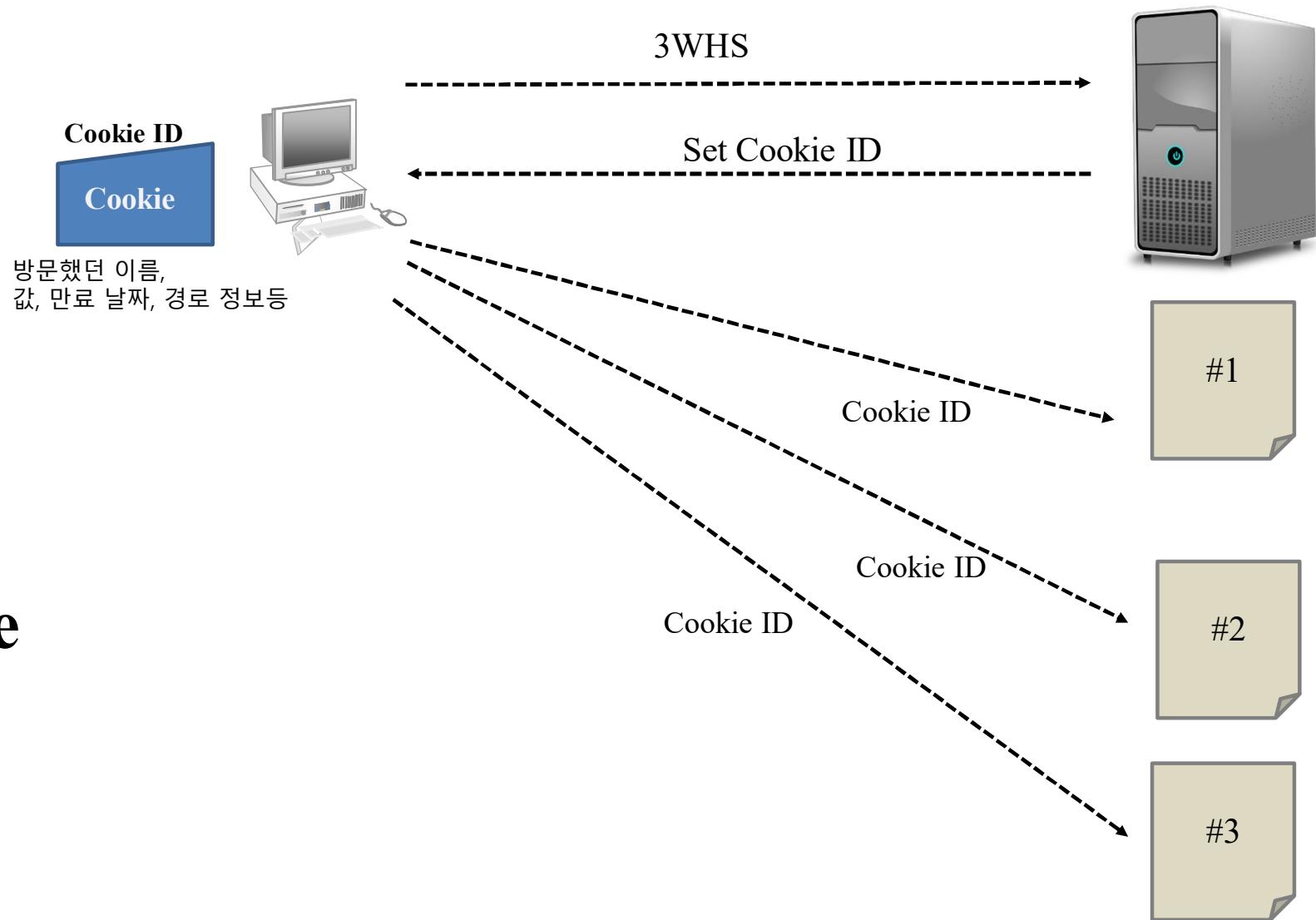
- 쿠키(cookie)

- ❶ 사용자 컴퓨터에 설치되는 작은 데이터 기록 정보 파일
- ❷ 쿠키에는 해당 웹 사이트에 방문했던 이름, 값, 만료 날짜, 경로 정보등이 담겨 있음
- ❸ 일정시간 동안 데이터를 저장할 수 있어 로그인 상태를 유지하거나 사용자 정보를 일정 시간동안 유지해야 할 때 주로 사용

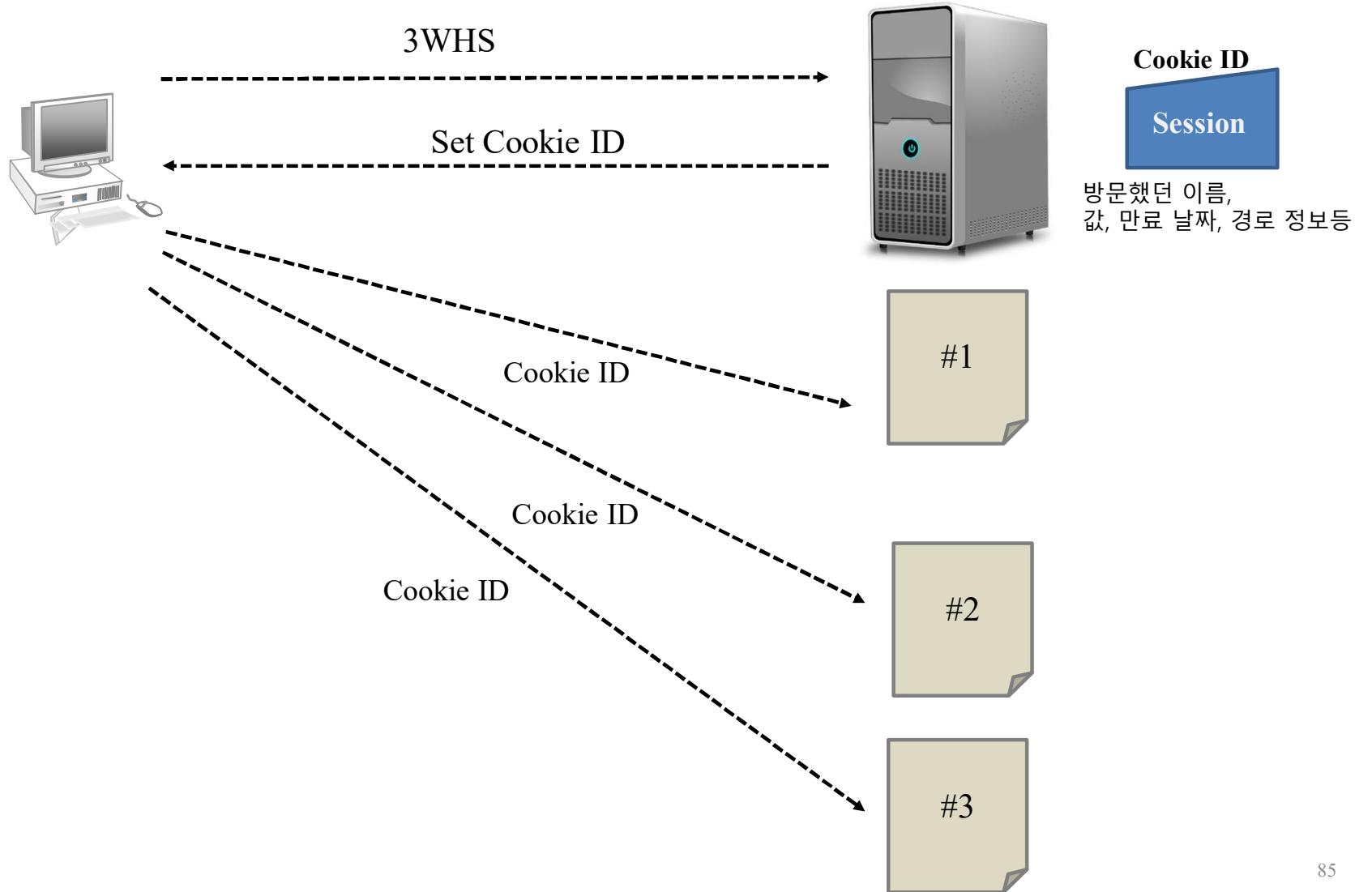
- 세션(session)

- ❶ 서버 메모리에 저장되는 정보
- ❷ 서버에 저장되기 때문에 쿠키와 달리 사용자 정보가 노출되지 않음
- ❸ 세션을 사용하더라도 클라이언트의 세션ID를 공격자가 가로채어 변조가 가능

Cookie

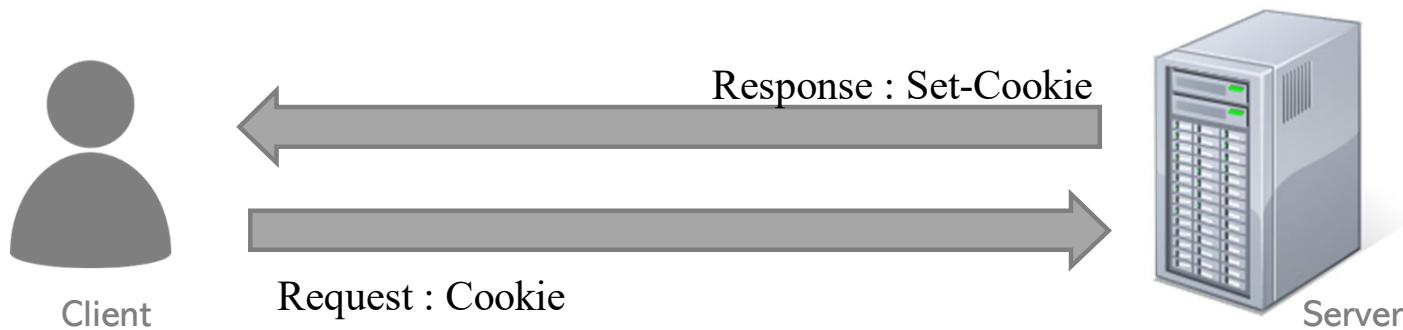


Session



Cookie를 위한 헤더 필더

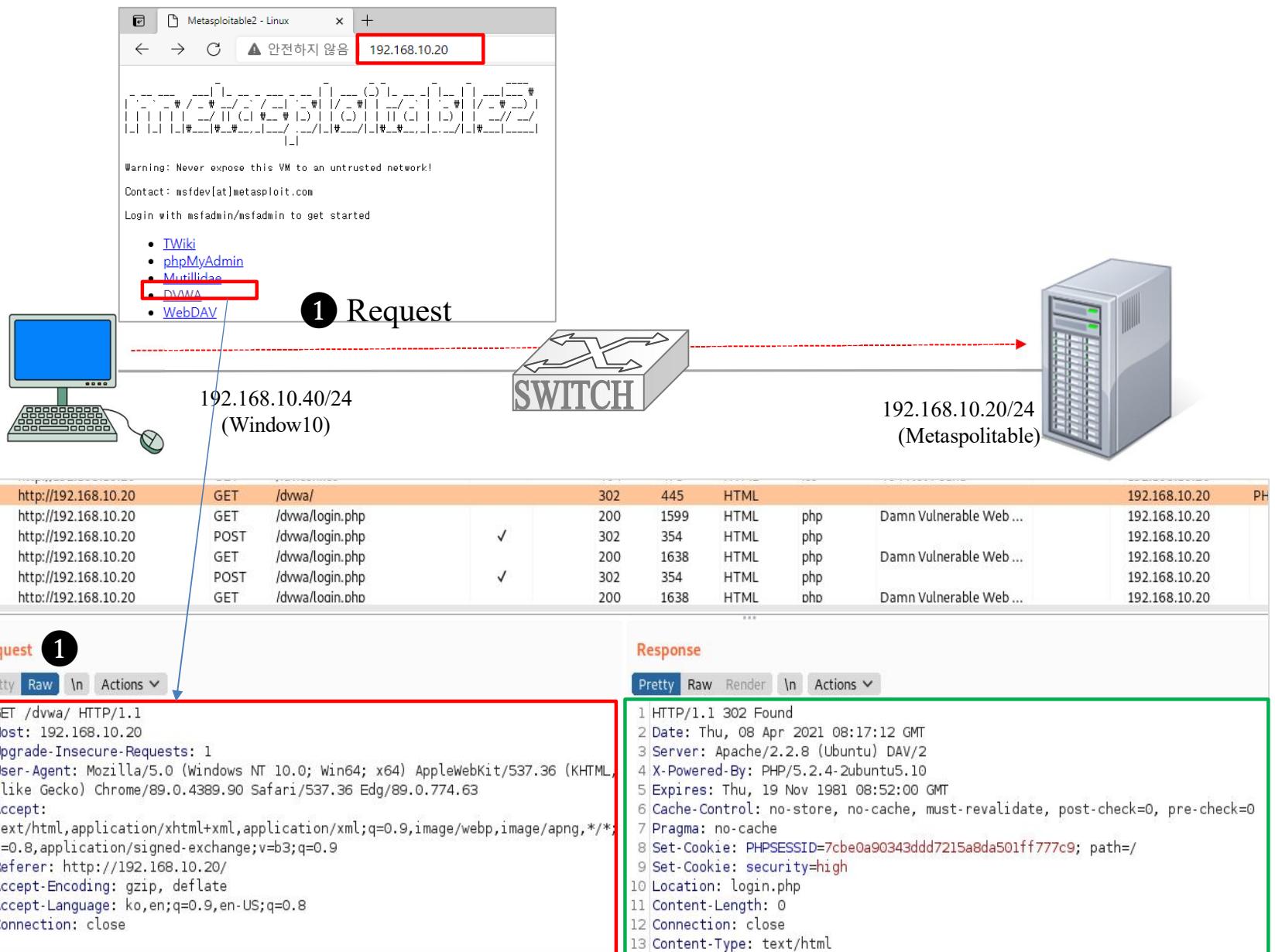
- Cookie는 유저 식별과 상태 관리에 사용되고 있는 기능
 - Set-Cookie : 상태 관리 개시를 위한 쿠키 정보
 - Cookie : 서버에서 수신한 쿠키 정보



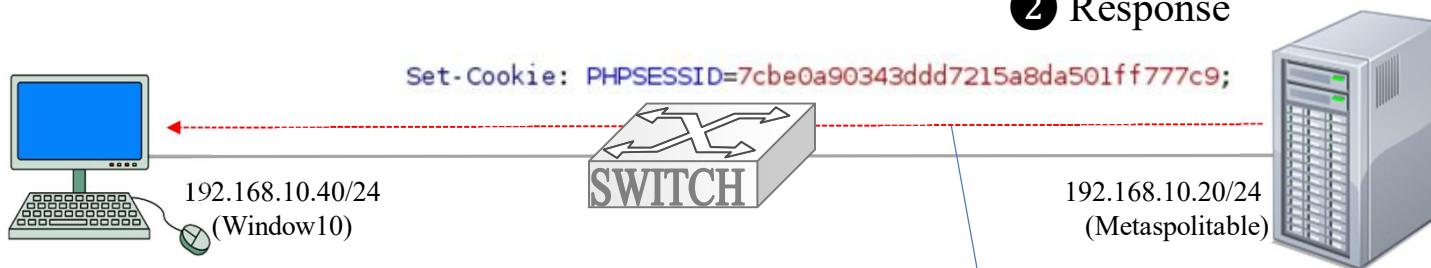
Set-Cookie

```
Set-Cookie : status-enable; expire=Tue,05 Jul 2020 07:26:31 GMT;  
Path=/; domain=.test.com;
```

NAME=VALUE	쿠키에 부여된 이름과 값
Expires=DATE	쿠키 유효기간(지정되지 않은 경우 브라우저 닫을때까지)
Path=PATH	쿠키 적용 대상이 도는 서버 상의 디렉터리
Domain=도메인명	쿠키 적용 대상이 도는 도메인명(지정되지 않을 경우 쿠키를 생성한 서버의 도메인)
Secure	HTTPS로 통신하고 있는 경우에만 쿠키 송신
HttpOnly	쿠키를 JAVAscript에서 액세스 하지 못하도록 제한



② Response



55	http://192.168.10.20	GET	/dwva/		302	445	HTML		192.168.10.20	PH
56	http://192.168.10.20	GET	/dwva/login.php		200	1599	HTML	php	Damn Vulnerable Web ...	192.168.10.20
73	http://192.168.10.20	POST	/dwva/login.php	✓	302	354	HTML	php	Damn Vulnerable Web ...	192.168.10.20
74	http://192.168.10.20	GET	/dwva/login.php		200	1638	HTML	php	Damn Vulnerable Web ...	192.168.10.20
101	http://192.168.10.20	POST	/dwva/login.php	✓	302	354	HTML	php	Damn Vulnerable Web ...	192.168.10.20
102	http://192.168.10.20	GET	/dwva/login.php		200	1638	HTML	php	Damn Vulnerable Web ...	192.168.10.20

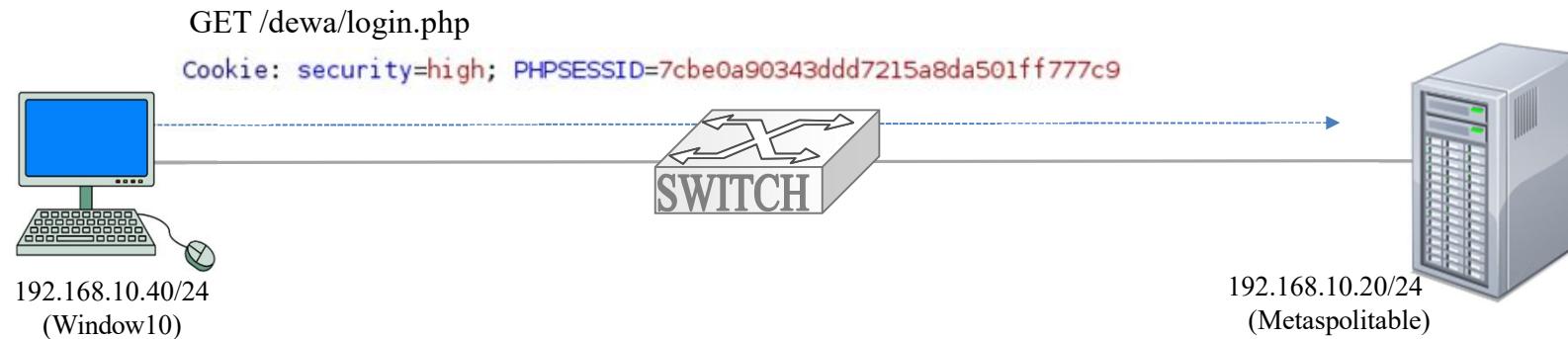
Request

```
Pretty Raw \n Actions ▾
1 GET /dwva/ HTTP/1.1
2 Host: 192.168.10.20
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36 Edg/89.0.774.63
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://192.168.10.20/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: ko,en;q=0.9,en-US;q=0.8
9 Connection: close
10
```

Response

```
Pretty Raw Render \n Actions ▾
1 HTTP/1.1 302 Found
2 Date: Thu, 08 Apr 2021 08:17:12 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
7 Pragma: no-cache
8 Set-Cookie: PHPSESSID=7cbe0a90343ddd7215a8da501ff777c9; path=/
9 Set-Cookie: security=high
10 Location: login.php
11 Content-Length: 0
12 Connection: close
13 Content-Type: text/html
```

3 Request



	Request	Response
55	http://192.168.10.20 GET /dewa/	302 445 HTML
56	http://192.168.10.20 GET /dewa/login.php	200 1599 HTML php Damn Vulnerable Web ...
73	http://192.168.10.20 POST /dewa/login.php	302 354 HTML php ✓
74	http://192.168.10.20 GET /dewa/login.php	200 1638 HTML php ✓ Damn Vulnerable Web ...
101	http://192.168.10.20 POST /dewa/login.php	302 354 HTML php ✓
102	http://192.168.10.20 GET /dewa/login.php	200 1638 HTML pho Damn Vulnerable Web ...

Request		3
Pretty Raw \n Actions ▾		Pretty Raw Render \n Actions ▾
<pre> GET /dewa/login.php HTTP/1.1 Host: 192.168.10.20 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36 Edg/89.0.774.63 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 Referer: http://192.168.10.20/ Accept-Encoding: gzip, deflate Accept-Language: en,en;q=0.9,es;q=0.8 Cookie: security=high; PHPSESSID=7cbe0a90343ddd7215a8da501ff777c9 Connection: close </pre>		<pre> 1 HTTP/1.1 200 OK 2 Date: Thu, 08 Apr 2021 08:17:12 GMT 3 Server: Apache/2.2.8 (Ubuntu) DAV/2 4 X-Powered-By: PHP/5.2.4-2ubuntu5.10 5 Pragma: no-cache 6 Cache-Control: no-cache, must-revalidate 7 Expires: Tue, 23 Jun 2009 12:00:00 GMT 8 Content-Length: 1289 9 Connection: close 10 Content-Type: text/html; charset=utf-8 11 12 13 14 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/1999/xhtml"> 15 16 <html xmlns="http://www.w3.org/1999/xhtml"> 17 18 <head> 19 20 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /></pre>