

삼성 청년 SW 아카데미

VueJS

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

목차

1. Cookie & Session
2. JWT (JSON Web Tokens)

Cookie & Session

삼성 청년 SW 아카데미

JWT (JSON Web Tokens).

|| Cookie를 활용한 인증.

✓ Cookie.

- 클라이언트가 웹사이트에 접속할 때 사용하게 되는 기록 파일 (client의 browse에 저장).
- Key, Value 형식의 문자열 형태로 저장.

✓ 로그인 과정 (with Cookie).

1. 서버가 클라이언트에 정보를 전달할 때 저장하고자 하는 정보를 응답 헤더(Cookie)에 저장하여 전달.
2. 해당 클라이언트는 서버에 요청을 보낼 때마다, 매번 저장된 쿠키를 요청 헤더의 Cookie에 담아 전달.
3. 서버는 쿠키의 정보를 이용하여 해당 요청의 클라이언트 식별이 가능.

Body	Cookies (3)	Headers (6)	Test Results	Status: 200 OK	Time: 56 ms	Size: 1.27 KB	Save Response
Name	Value	Domain	Path	Expires	HttpOnly	Secure	
JSESSIONID	7A1B99B15FE...	localhost	/guestbook	Session	true	false	
ssafy_id	ssafy	localhost	/	Sun, 28 May 2023 10:20:00 UTC	false	false	
ssafy_pwd	ssafy	localhost	/	Sun, 28 May 2023 10:20:00 UTC	false	false	

JWT (JSON Web Tokens).

|| Cookie를 활용한 인증.

✓ Cookie 단점.

- Cookie는 노출이 되기 때문에 id, password등 민감한 정보를 저장할 경우 보안에 좋지 않음.
- Client에 file로 저장되기 때문에 조작이 가능하고, 사이즈가 제한되어 있기 때문에 충분한 데이터 저장이 불가능.
- 브라우저별 Cookie에 대한 지원 형태문제로 브라우저간 공유 불가능.
- 쿠키의 사이즈가 커질수록 네트워크의 부하가 가중됨.

JWT (JSON Web Tokens).

|| Session을 활용한 인증.

✓ Session.

- 민감한 정보를 노출 했을 때의 단점을 막기위해 나온 것.
- 클라이언트의 인증 정보를 Cookie가 아닌 서버 측에 저장하고 관리함.

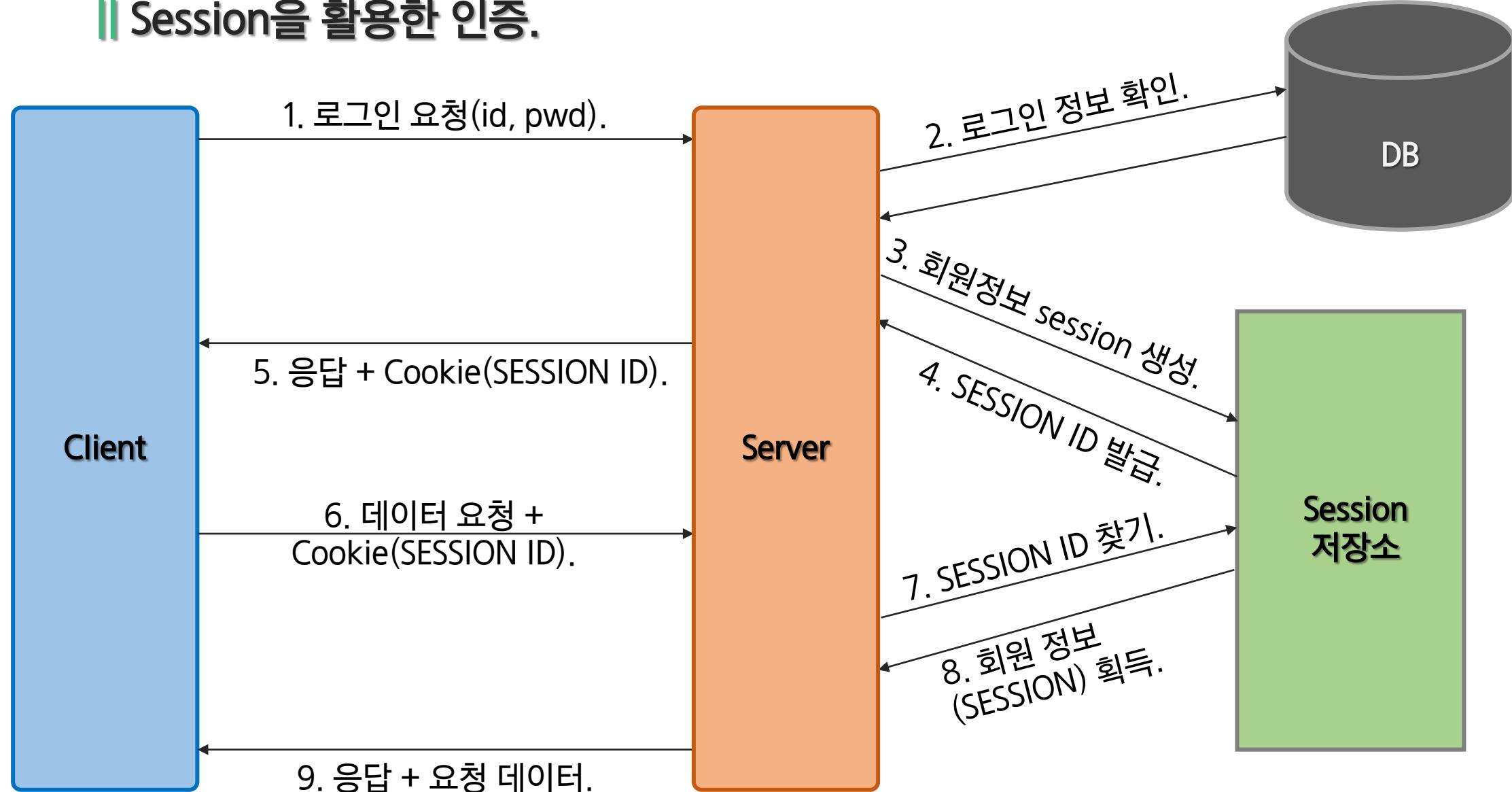
✓ 로그인 과정 (with Session).

1. ID, PASSWORD로 서버에 로그인 요청.
2. ID, PASSWORD로 인증 후 사용자를 구분할 수 있는 유니크한 ID를 만들어 세션 저장소에 저장한 후 SESSION ID를 발행.
3. SESSION ID를 클라이언트에 반환.
4. 인증 정보 요청 시마다 이 SESSION ID를 Cookie에 담아 Header에 담아 서버에 전달.
5. 인증이 필요한 요청의 경우 서버는 클라이언트가 보낸 SESSION ID와 서버의 세션 저장소에 있는 SESSION ID를 비교 확인.

Body	Cookies (2)	Headers (6)	Test Results		Status: 200 OK	Time: 9 ms	Size: 3.25 KB	Save Response	▼
Name	Value	Domain	Path	Expires	HttpOnly	Secure			
JSESSIONID	4AC7AC5448 ...	localhost	/guestbook	Session	true	false			

JWT (JSON Web Tokens).

Session을 활용한 인증.



JWT (JSON Web Tokens).

|| Session을 활용한 인증.

✓ Session 단점.

- User의 Session에 대한 정보를 서버 메모리에 저장하는 부담.
- 이로 인한 Scale out/up이 요구되며 Scale out 시 로드 밸런싱등 신경 써야 할 부분이 많아짐.
- Cookie에 저장되는 Session ID의 경우 중요 정보가 없기 때문에 탈취 당해도 안전하지만, 그 Cookie를 사용하여 클라이언트인척 위장 할 수 있는 보안의 약점이 생김.
- 매번 요청 시 세션 저장소를 조회해야 함.

이어서..

삼성 청년 SW 아카데미

JWT (JSON Web Tokens)

삼성 청년 SW 아카데미

JWT (JSON Web Tokens).

|| Token을 활용한 인증.

✓ Token.

- 클라이언트가 서버에 접속하여 사용자 인증을 했을 때 유일 값인 '[토큰\(token\)](#)'을 발급.
- 서버에 요청을 보낼 때 요청의 헤더에 토큰을 넣어서 보냄.
- 클라이언트가 보낸 토큰이 서버에서 발급한 토큰과 같은지를 체크하여 인증처리.
- 토큰은 앱과 서버가 통신 및 인증할 때 많이 사용.

✓ 서버(session)기반 VS 토큰(token)기반 인증.

서버기반	토큰기반
서버에 사용자 인증 정보를 관리.	클라이언트에 토큰 발급.
클라이언트로부터 요청을 받으면 클라이언트의 상태를 계속 유지 (Stateful).	로그인이 필요한 작업일 경우 헤더에 토큰을 삽입하여 보내고 인증 (Stateless)
클라이언트가 많아지면 성능 문제 발생.	

JWT (JSON Web Tokens).

|| Token을 활용한 인증.

✓ Token 단점.

- 토큰 자체의 데이터 길이가 길다.
- 이로 인한 인증 요청이 많아질수록 네트워크 부하가 심해질 수 있다.
- Payload 자체는 암호화되지 않기 때문에 중요 정보 저장 불가.
- 네트워크 전송방식 이기 때문에 토큰을 탈취 당할 우려가 있다. [expire 설정으로 해결?]

JWT (JSON Web Tokens).

|| JWT (JSON Web Token).

✓ JWT

- 인증에 필요한 정보를 암호화 시킨 JSON 토큰.
- JSON 데이터를 Base64 URL-safe-Encode를 통해 인코딩하여 직렬화.
- 토큰 내부에 위변조 방지를 위한 개인키를 통한 전자서명 포함.

JWT (JSON Web Tokens).

|| JWT (JSON Web Token).

✓ JWT 구성

- <https://jwt.io/>

The screenshot shows the jwt.io Debugger interface. At the top, there's a warning message: "Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side." Below this, the algorithm is set to "HS256".

Encoded: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiBPc3Npbm5vLCJjbGFzcycI6MTB9.MQdLZ6PiL0Mw40J aGlq6xslwo7oc4t5JpvNRT9n-PVk

Decoded:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
{  
  "userid": "ssafy",  
  "name": "안호인",  
  "class": 10  
}
```

VERIFY SIGNATURE:

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  ssafySecret  
)
```

secret base64 encoded

Signature Verified (with a checkmark icon)

SHARE JWT button

JWT (JSON Web Tokens).

|| JWT (JSON Web Token).

✓ JWT 구성

- <https://jwt.io/>

XXXXXX.XYYYYYY.ZZZZZZZZ

헤더[Header]

내용[Payload]

서명[Signature]

eyJhbGciOiJIUzI1NiI<input type="text">eyJ1c2VyaWQiOiJzc2FmeSIsIm5hbWUiOiLsIYjtmqjsnbgiLCJjbGFzcyI6MTB9.I7QT9PhfNI3f-JvDvvq33JpRcKliLUOUXFbTLv4w-ow

Header	Payload	Signature
{ "alg": "HS256", "typ": "JWT" }	{ "userid": "ssafy", "name": "안효인", "class": 10 }	HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret)

JWT (JSON Web Tokens).

|| JWT (JSON Web Token).

✓ JWT 구성

- Header.
 - JWT에서 사용할 토큰의 타입과 암호화 알고리즘 정보로 구성.
 - key - value 형태로 구성.
- Payload.
 - 서버로 보낼 사용자 권한 정보와 데이터로 구성. (key - value 형태)
 - 토큰에 담을 클레임(Claim) 정보 포함.
 - Claim? : key - value 형식으로 이루어진 한 쌍의 데이터.
 - 토큰에는 여러 개의 claim을 넣을 수 있음.
 - payload에는 민감한 정보를 넣으면 안됨 (암호화 X)
- Signature.
 - Secret Key(서버의 개인키)를 포함하여 암호화.
 - 토큰의 유효성을 검증하기 위한 문자열로 구성.

JWT (JSON Web Tokens).

|| JWT (JSON Web Token).

✓ Refresh Token

- AccessToken을 탈취 당했을 경우에 대한 최소한의 대비.(해결책 X)
- AccessToken의 유효기간을 짧게 설정하여 탈취 되어도 사용기간을 줄이는 효과.
- RefreshToken을 통해 다시 AccessToken을 발급받아 사용.
- RefreshToken은 인증 정보를 담고 있지 않고 오로지 AccessToken 재발급 용도로만 사용함.

JWT (JSON Web Tokens).

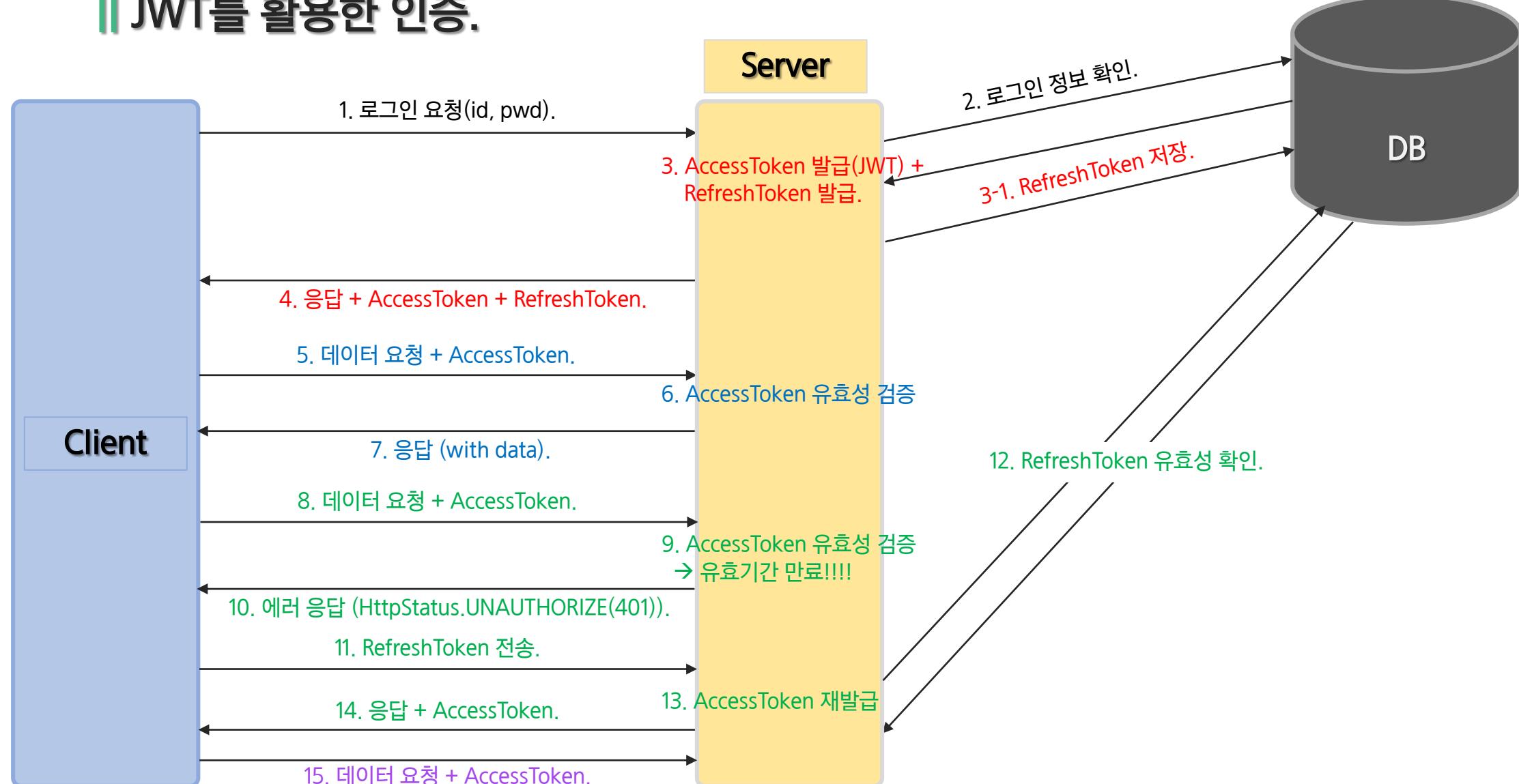
|| JWT (JSON Web Token).

✓ AccessToken & Refresh Token 사용한 Login Process

1. 클라이언트가 ID, PW로 서버에게 인증을 요청하고 서버는 이를 확인하여 Access Token과 Refresh Token을 발급.
2. 클라이언트는 이를 받아 Refresh Token을 저장하고 Access Token을 가지고 서버에 자유롭게 요청.
3. 요청을 하던 도중 Access Token이 만료되어 더 이상 사용할 수 없다는 오류를 서버로부터 전달받음.
4. 클라이언트는 본인이 사용한 Access Token이 만료되었다는 사실을 인지하고 본인이 가지고 있던 Refresh Token을 서버로 전달하여 새로운 Access Token의 발급을 요청.
5. 서버는 Refresh Token을 받아 서버의 Refresh Token Storage에 해당 토큰이 있는지 확인하고, 있다면 Access Token을 생성하여 전달.
6. 이후 2로 돌아가서 동일한 작업을 진행합니다.

JWT (JSON Web Tokens).

|| JWT를 활용한 인증.



내일
방송에서
만나요!

삼성 청년 SW 아카데미