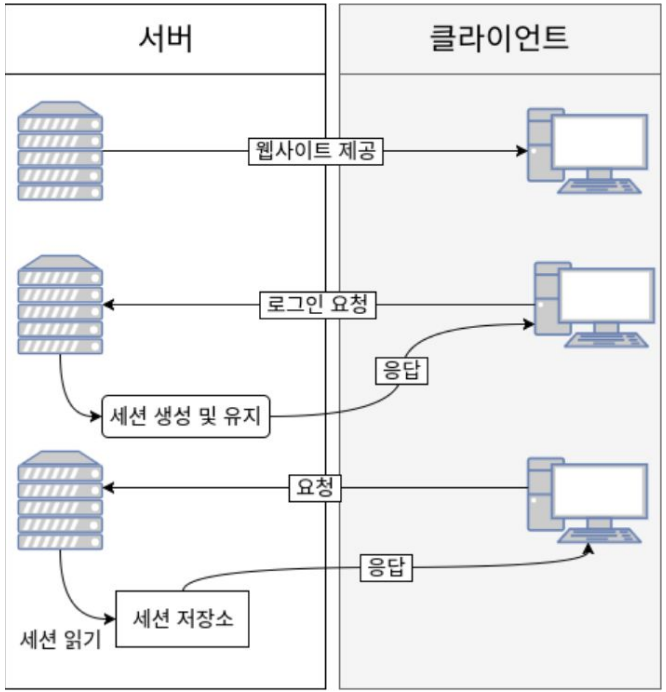
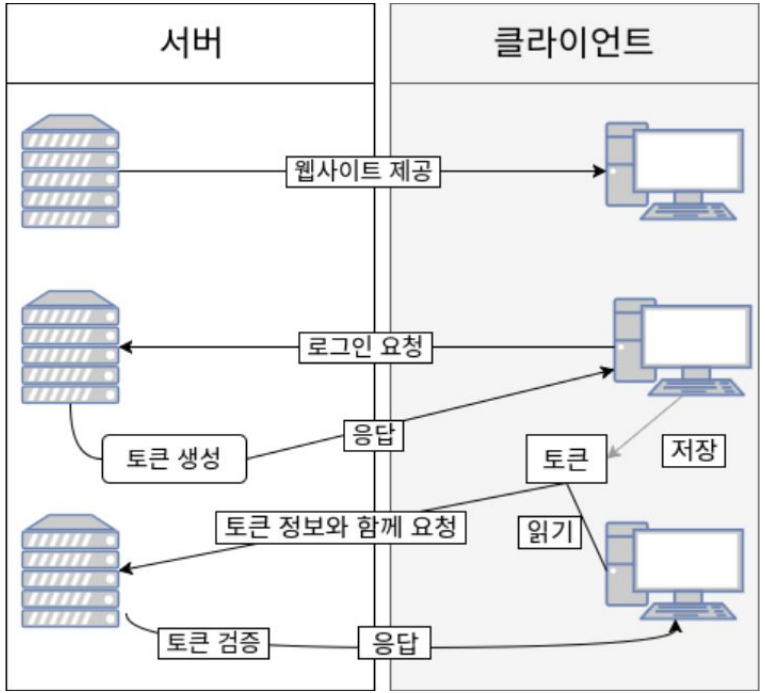
JWT(JSON Web Token)

목차

1. JWT 정의

- 토큰 기반 인증 시스템의 구현체이다.





1-1 토큰의 장점

1. 무상태(Stateless)

2. 확장성(Scalability)

3. 보안성

4. 확장성(Extensibility)

2. JWT 특징

1. JSON 객체를 사용

2. 주류 프로그래밍 언어

   (C,JAVA,PYTHON,C++,PHP,JS,,,) 등

   대부분 지원

3. 자가 수용적이다.

4. 쉽게 전달될 수 있다.

3. 어떤 상황에서 사용하는 것이 좋을까?

1. 회원 인증

2. 정보 교류

4. 생김새

aaaaaa.bbbbbb.cccccc

헤더(header) 내용(payload) 서명(signature)

1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o 2 3

1 Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2 Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

3 Signature

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

4-1. 헤더

- typ:토큰의 타입 지정JWT
- alg:해싱 알고리즘 지정sha256,RSA

```
# Node.js에서의 인코딩
const header = {
  "typ": "JWT",
  "alg": "HS256"
};

// encode to base64
const encodedPayload = new Buffer(JSON.stringify(payload))
                           .toString('base64')
                           .replace('=', '');
console.log('payload: ',encodedPayload);

/* Result:
header: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
*/
```

4-2. 정보(payload)

- 클레임(claim)
  - 등록된 클레임 : iss, sub, aud, exp, nbf, iat, jti
  - 공개 클레임 : URL 형식으로 된 클레임
  - 비공개 클레임 : 충돌에 유의하여 사용

```
const payload = {
    "iss": "velopert.com",  // 등록된 클레임
    "exp": "1485270000000", // 등록된 클레임
    "https://velopert.com/jwt_claims/is_admin": true,   // 공개 클레임
    "userId": "11028373727102", // 비공개 클레임
    "username": "velopert"      // 비공개 클레임
};

// encode to base64
const encodedPayload = new Buffer(JSON.stringify(payload))
                           .toString('base64')
                           .replace('=', '');

console.log('payload: ',encodedPayload);

/* result
payload:
eyJpc3MiOiJ2ZWxvcGVydC5jb20iLCJleHAiOiIxNDg1MjcwMDAwMDAwIiwiaHR0cHM6Ly92ZWxvcGVydC
5jb20vandOX2NsYWltcy9pc19hZG1pbiI6dHJ1ZSwidXNlcklkIjoiMTEwMjgzNzM3MjcxMDIiLCJ1c2Vy
bmFtZSI6InZlbG9wZXJ0In0
*/
```

4-3. 서명(Signature)

- 헤더의 인코딩값, payload의 인코딩값을 합친다.(중간에 `.`을 넣어줍니다)
- 비밀키로 다시 해쉬하여 생성한다.

```
// 수도코드
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

```
const crypto = require('crypto');
const signature = crypto.createHmac('sha256', 'secret')
              .update(encodedHeader + '.' + encodedPayload)
              .digest('base64')
              .replace('=', '');

console.log('signature: ',signature);

/* result
Signature:  WE5fMufM0NDSVGJ8cAolXGkyB5RmYwCto1pQwDIqo2w
*/
```

4-4. 결과

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) secret base64 encoded
```

5. 실습

- jjwt를 이용하여 JWT를 만드는 코드.
- JWT의 헤더, 클래임, 암호 등의 필요한 정보를 넣고 직렬화(compact())
- 단순하게 Session처럼 정보를 넣어놓고 빼쓰기 위해서는 claim에 데이터 삽입

```java
@Service("jwtService")
public class JwtServiceImpl implements JwtService{

    private static final String SALT =  "luvookSecret";

    @Override
    public <T> String create(String key, T data, String subject){
        String jwt = Jwts.builder()
                        .setHeaderParam("typ", "JWT")
                        .setHeaderParam("regDate", System.currentTimeMillis())
                        .setSubject(subject)
                        .claim(key, data)
                        .signWith(SignatureAlgorithm.HS256, this.generateKey())
                        .compact();
        return jwt;
    }

    private byte[] generateKey(){
        byte[] key = null;
        try {
            key = SALT.getBytes("UTF-8");
        } catch (UnsupportedEncodingException e) {
            if(log.isInfoEnabled()){
                e.printStackTrace();
            }else{
                log.error("Making JWT Key Error ::: {}", e.getMessage());
            }
        }

        return key;
    }
}
```

5. 실습

- HTTP Header -> JWT -> Claim -> Key -> Value.

```
@Service("jwtService")
public class JwtServiceImpl implements JwtService{
    @Override
    public Map<String, Object> get(String key) {
        HttpServletRequest request = ((ServletRequestAttributes)
RequestContextHolder.currentRequestAttributes()).getRequest();
        String jwt = request.getHeader("Authorization");
        Jws<Claims> claims = null;
        try {
            claims = Jwts.parser()
                        .setSigningKey(SALT.getBytes("UTF-8"))
                        .parseClaimsJws(jwt);
        } catch (Exception e) {
            throw new UnauthorizedException();
        }
        @SuppressWarnings("unchecked")
        Map<String, Object> value = (LinkedHashMap<String,
Object>)claims.getBody().get(key);
        return value;
    }
}
```

- 전체 소스 : https://github.com/viviennes7/luvook

Q&A

감사합니다.