

2025년 상반기 K-디지털 트레이닝

# JWT 자바 라이브러리

---

[KB] IT's Your Life

기본 라이브러리에 없어서 외부 라이브러리 설치 필요

jjwt

\* KB 국민은행

## ✓ 의존성

```
implementation("io.jsonwebtoken:jjwt-api:0.11.5")
```

```
runtimeOnly("io.jsonwebtoken:jjwt-impl:0.11.5")
```

```
implementation("io.jsonwebtoken:jjwt-jackson:0.11.5")
```

자바가 필요로 하는 라이브러리.

ObjectMapper 사용하기 위해서

## ✓ Secret Key 준비 서버만 알고 있는 첨가물

- 암호화에 사용할 임의의 문자열
- 개발 시에는 직접 지정

```
private String secretKey = "아주 긴 임의의 문자열 지정";  
secretKey = Base64.getEncoder().encodeToString(secretKey.getBytes());           // BASE64 인코딩
```

- 운영 시에는 자동 생성

```
private Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);
```

→ 서버가 재기동 되면 key문자열이 갱신되므로 기존 발급 토큰은 사용불가 예전에 발급한 jwt 사용못함

✓ 유효 기간(밀리 초 단위)

- 기본 토큰 유효시간
  - JWT의 유효 시간 설정

```
static final long TOKEN_PERIOD = 1000L * 60L * 5L; // 테스트용 5분 - 만기 확인용
```

## ✓ Payload 정보 구성

### ○ Claims 객체

Date now = new Date();

// registered claim ✓

Claims claims = Jwts.claims().setSubject(userPk); // sub

claims.setIssuedAt(now)

claims.setExpiration(new Date(date.getTime() + TOKEN\_PERIOD)) // exp

// public claim ✓

claims.put("role", role); 키, 값

유저네임

// iat

## ✓ JWT 토큰 생성

```
String token = Jwts.builder() ✓  
    .setSubject(subject) ✓ 대상  
    .setIssuedAt(new Date())  
    .setExpiration(new Date(new Date().getTime() + expire)) ✓  
    .signWith(key) ✓  
    .compact();
```

## ✓ JWT 검증

- 유효시간 이전이면 true 리턴
- 토큰이 해석되지 않는 경우 또는 유효 시간 만료인 경우 예외 발생
- 예외
  - ExpiredJwtException: 유효 시간 만기 → refresh 토큰을 통한 갱신 과정.
  - UnsupportedJwtException: 지원하지 않은 JWT
  - MalformedJwtException: 잘못된 JWT 포맷 예외
  - SignatureException: 서명 불일치 예외
  - IllegalArgumentException: 잘못된 정보 포함→ 400 401 응답

```
Jws<Claims> claims = Jwts.parserBuilder()    토큰 문자열에서 클레임들 꺼내기 (파싱)
    .setSigningKey(key)
    .build()
    .parseClaimsJws(jwtToken); // 예외 발생 가능
                        토큰문자열
```

## ✓ JWT 정보 추출

### ○ subject 추출

```
Jwts.parserBuilder()  
    .setSigningKey(key)  
    .build()  
    .parseClaimsJws(token)  
    .getBody()  
    .getSubject();           // username 추출
```



## ✓ JWT 정보 추출

### ○ Claim에서 정보 추출

```
Claims claims = parseClaims(accessToken);
```

```
if (claims.get(AUTHORITIES_KEY) == null) {  
    throw new RuntimeException("권한 정보가 없는 토큰입니다.");  
}
```

```
final String username = claims.getSubject();  
final CurrentUser userDetails = (CurrentUser) userService.loadUserByUsername(username);
```

사용자 관련된 정보들

한가지 방법이 있는게 아니라 JWT를 활용한 방법들은 많다.

- 1) 권한정보를 token 생성시 토큰에 담기
- 2) 권한정보를 토큰 생성시 토큰에 안담음, 권한 정보는 매번 디비에 물어보기.

- 1) 보안 약 2) 보안 굳 but 매번 디비 호출 오버헤드

우리는 2번으로 진행

## ✓ JwtProcessor

### ○ Jwt 작업을 위한 Helper 클래스

- 주요 작업을 캡슐화

String generateToken(String subject)	subject(username)에 대한 관련 토큰 생성 ✓
String getUsername(String token)	token에서 username 추출 ✓
boolean validateToken(String token)	token의 유효성 검증 ✓

## security.util.JwtProcessor.java

```
package org.scoula.security.util;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jws;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import org.springframework.stereotype.Component;
```

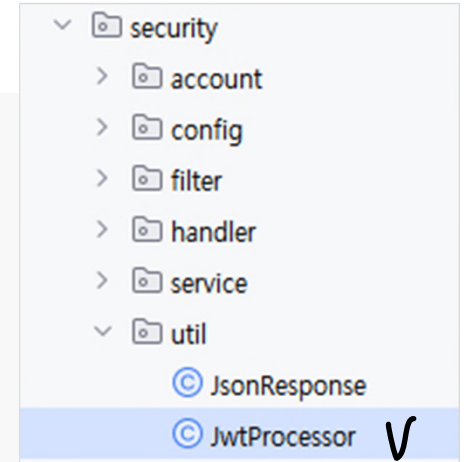
```
import java.nio.charset.StandardCharsets;
import java.security.Key;
import java.util.Date;
```

```
@Component 활요을 위해 di되게끔
```

```
public class JwtProcessor {
    static private final long TOKEN_VALID_MILISECOND = 1000L * 60 * 5; // 5 분 ✓
```

```
    private String secretKey = "충분히 긴 임의의(랜덤한) 비밀키 문자열 배정"; ✓
    private Key key = Keys.hmacShaKeyFor(secretKey.getBytes(StandardCharsets.UTF_8)); ✓
```

```
// private Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256); -- 운영시 사용 ✓
```



## security.util.JwtProcessor.java

// JWT 생성

public String generateToken(String subject) {

return Jwts.builder()

.setSubject(subject) ✓

.setIssuedAt(new Date())

.setExpiration(new Date(new Date().getTime() + TOKEN\_VALID\_MILLISECOND)) 만기시간 지정

.signWith(key) 서명 붙이기

.compact();

}

// JWT Subject(username) 추출 - 해석 불가인 경우 예외 발생

// 예외 ExpiredJwtException, UnsupportedJwtException, MalformedJwtException, SignatureException,

// IllegalArgumentException

public String getUsername(String token) {

return Jwts.parserBuilder()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

유저네임을 인자로

디비를 통한 정보 조회를 하지 않는 방법이라면  
claim 구성을 통한 추가정보를 넣어줘야한다.  
아까 앞에서 다 봤쥬?

두 메서드  
런타임 예외  
발생 가능.

특히 아까 말한  
유효 만기 예외가  
제일 많이 발생함

토큰에서 유저 네임 뽑기

유저 네임만 토큰에 넣었고 다른 정보는 디비에 넣어서 보관  
하게끔 함.

## security.util.JwtProcessor.java

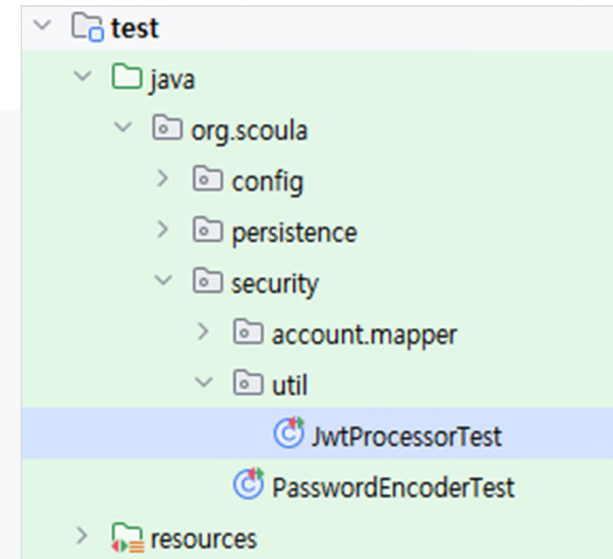
```
// JWT 검증(유효 기간 검증) - 해석 불가인 경우 예외 발생 //
```

```
public boolean validateToken(String token) {  
    Jws<Claims> claims = Jwts.parserBuilder()  
        .setSigningKey(key)  
        .build()  
        .parseClaimsJws(token);  
    return true;  
}
```

이 메소드는 refresh 토큰 검증할때 주로 쓰임

```
@ExtendWith(SpringExtension.class)
@SpringBootTest(classes = { RootConfig.class, SecurityConfig.class })
@Log4j2
class JwtProcessorTest {
    @Autowired
    JwtProcessor jwtProcessor;

    @Test
    void generateToken() {
        String username = "user0";
        String token = jwtProcessor.generateToken(username);
        log.info(token);
        assertNotNull(token);
    }
}
```



INFO : org.scoula.security.util.JwtProcessorTest - eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTt0OwN3io\_c JWT

복사하여 다음 테스트에 사용

## test :: JwtProcessorTest.java

```
@Test
void getUsername() {
    String token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWx0IiwiaWF0IjE5ODAzMDg0fQ.nwD4rlroYL6hr_-Esav8KIsHw573MbAiTT-Nz_yYHI8bMcyGZMOEjMt0Own3io_c"; ~

    String username = jwtProcessor.getUsername(token); ✓
    log.info(username);
    assertNotNull(username);
}
```

INFO : org.scoula.security.util.JwtProcessorTest - **user0**

5분전이면 성공,

## test :: JwtProcessorTest.java

```
@Test
void validateToken() {
    // 5분 경과 후 테스트
    String token = "eyJhbGciOiJIUzI1NiIsInR5cCI6ImlhdCI6MTcyMTgwMjc4NCwiZXhwIjoxODAzMDg0fQ.nwD4rlroYL6hr_-Esav8KIsHw573MbAiTT-Nz_yYHI8bMcyGZMOEjMt0Own3io_c";

    boolean isValid = jwtProcessor.validateToken(token); // 5분 경과 후면 예외 발생
    log.info(isValid);
    assertTrue(isValid); // 5분전이면 true
}
}
```

**JWT expired at** 2024-07-24T06:38:04Z. Current time: 2024-07-24T06:38:45Z, a difference of 41480 milliseconds. Allowed clock skew: 0 milliseconds.

**io.jsonwebtoken.ExpiredJwtException:** JWT expired at 2024-07-24T06:38:04Z. ...

jwt는 위조 변조에 강하다. 서명 때문에.

도청당해서 통째로 읽히는 거는 약하다.

다른 사용자에게 통째로 읽은 걸 넣어도 통과 가능하다.

도청을 막기 위해서는 https 프로토콜을 써야한다.

==> https+jwt 보안적으로 좋아! == jwt가 http 헤더에 노출되지 않는다.

하지만 실습은 일단  
http+jwt로 할 예정