JWT 기반 인증 (2)

Access Token을 통한 로그인

20210463 박연종

park@duck.com

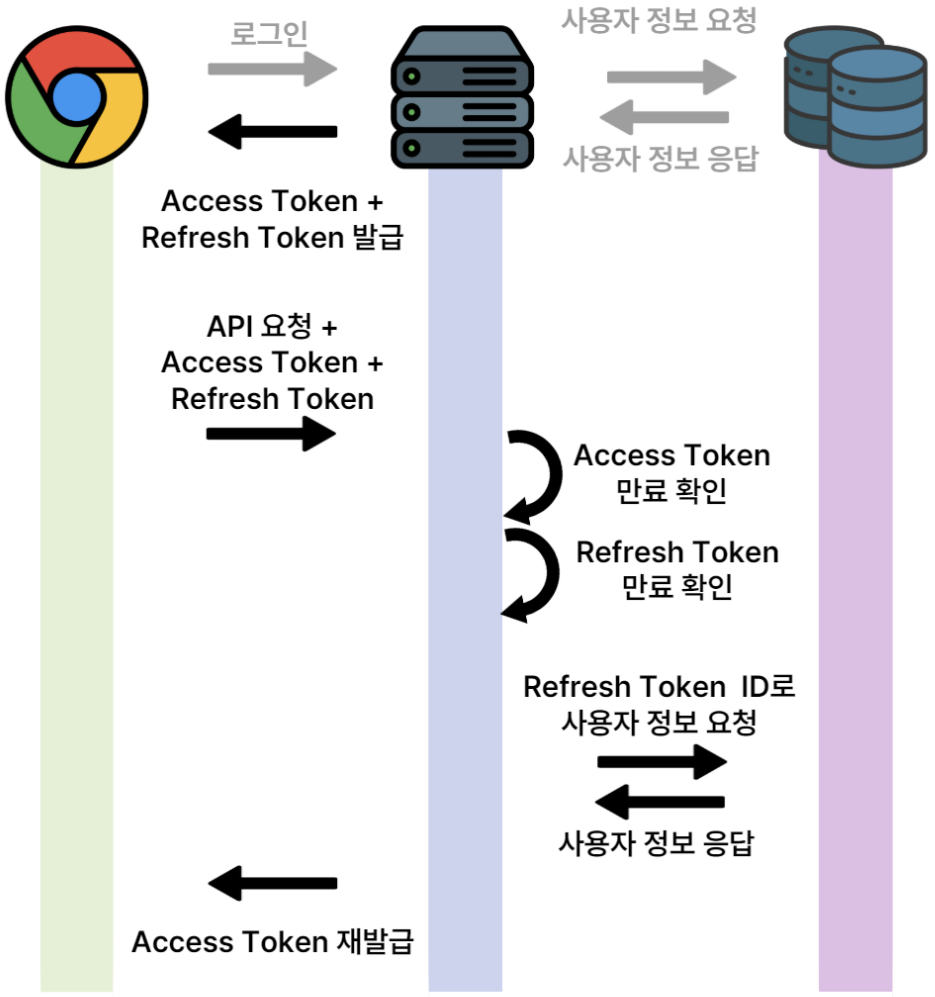
SSL SEMINAR

Photo by Pawel Czerwinski

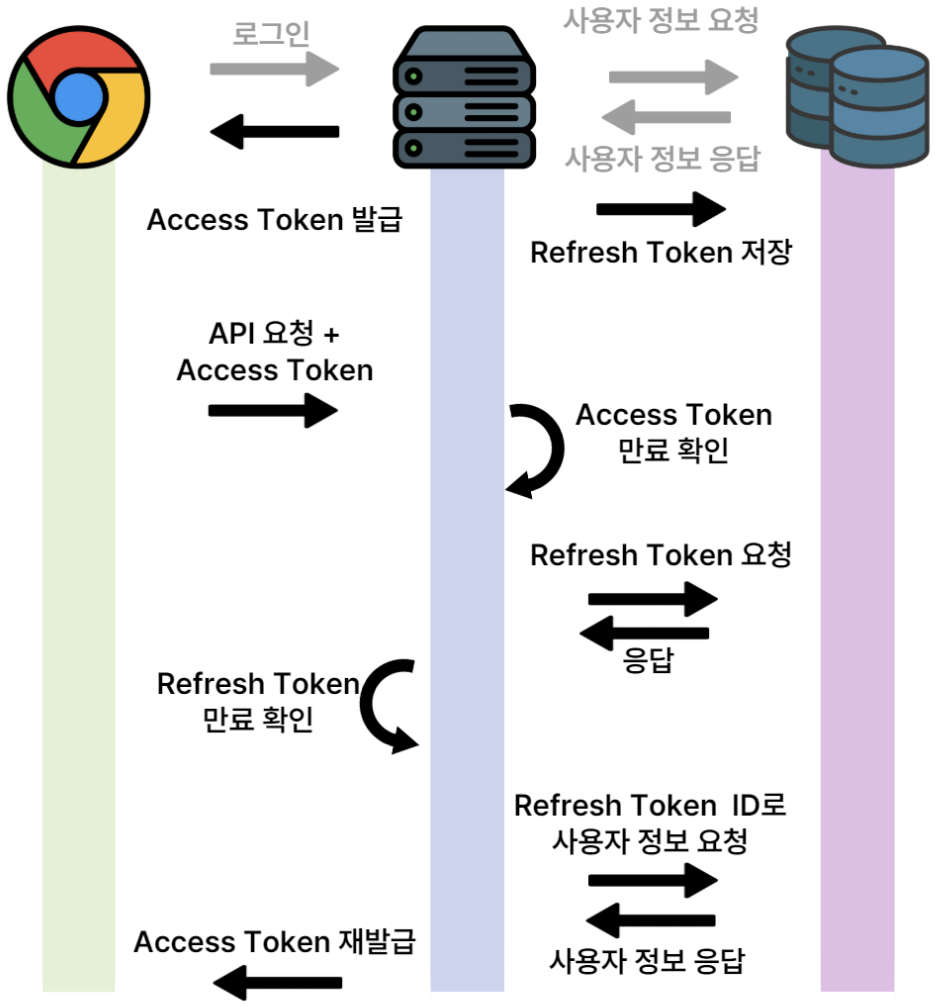
Spring Security와 JWT 기반 인증 흐름

System Software Lab.

Access Token + Refresh Token (쿠키)



Access Token + Refresh Token (서버)



01.
JWT 프로젝트 생성

build.gradle 의존성 추가

Spring Boot가 Spring Framework와 잘 어울리는 의존성을 자동으로 가져오는

Spring Boot Starter 지원

```
// 사용자 회원가입, 인증 관련 라이브러리 Spring Security
implementation 'org.springframework.boot:spring-boot-starter-security'
testImplementation 'org.springframework.security:spring-security-test'

//JWT 생성 및 파싱, 검증 관련 라이브러리 Java Json Web Token
implementation 'io.jsonwebtoken:jjwt-api:0.12.6'
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.6'
runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.12.6'
```

외부 라이브러리이므로 개발자가 직접 Spring Framework와 잘 어울리는 의존성을 추가해야 함

JWT 기반 인증에 필요한 프로젝트 기반 다지기

System Software Lab.

1. **JWT 설정 파일 만들기**
2. JWT 생성 및 파싱 컴포넌트 만들기
3. JWT 검증 필터 만들기 및 등록
4. 사용자 확인을 위한 서비스 만들기
5. 회원가입 및 로그인 엔드 포인트 및 로직 만들기
6. 권한이 필요한 엔드 포인트 만들기

설정 파일 만들기

System Software Lab.

/src/main/resources/application.yml 또는 application.properties

(별도의 설정 파일로 분리하려면 spring.config.import 설정으로 해당 설정 파일 경로를 추가)

```yaml
spring:
  application:
    name: demo

  config:
    import: # 설정 파일 분리 (공개 VCS 저장소에 올리지 않은)
      - properties/jwt.yml

jwt:
  access-token:
    issuer: d449    # 발행자 /등록 클레임
    audience: d449  # JWT 사용 대상 /등록 클레임
    expired: 600    # 10분 (600초)
    secret: d449    # HS256 암호화에 필요한 솔트(비밀 키)
                    # 최소 256비트의 비밀 키를 사용해야 안전
```

```java
package com.patulus.demo.global.jwt;

import lombok.Getter;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Getter
@Component
public class JwtProperties {

    private final String issuer;
    private final String audience;
    private final long expired;

    public JwtProperties(
            @Value("${jwt.access-token.issuer}") String issuer,
            @Value("${jwt.access-token.audience}") String audience,
            @Value("${jwt.access-token.expired}") long expired,
            @Value("${jwt.access-token.secret}") String secret,
    ) {
        this.issuer = issuer;
        this.audience = audience;
        this.expired = expired;
        this.secret = secret;
    }
}
```

JWT 기반 인증에 필요한 프로젝트 기반 다지기

1. JWT 설정 파일 만들기
2. **JWT 생성 및 파싱 컴포넌트 만들기**
3. JWT 검증 필터 만들기 및 등록
4. 사용자 확인을 위한 서비스 만들기
5. 회원가입 및 로그인 엔드 포인트 및 로직 만들기
6. 권한이 필요한 엔드 포인트 만들기

JWT 생성 및 파싱 컴포넌트 만들기 (1) : 생성 메서드

System Software Lab.

```java
package com.patulus.demo.global.jwt;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Date;
import java.util.Map;

@Component
@RequiredArgsConstructor
@Slf4j
public class JwtHandler {

    public static final String USER_ID = "USER_ID";
    public static final String USER_ROLE = "USER_ROLE";
    private static final long MILLI_SECOND = 1000L;

    private final JwtProperties jwtProperties;
    private final SecretKey secret = new
SecretKeySpec(jwtProperties.getSecret().getBytes(StandardCharsets.UTF_8),
Jwts.SIG.HS256.key().build().getAlgorithm());
```

```java
public TokenResponse createTokens(JwtUserClaim jwtUserClaim) {
        Map<String, Object> tokenClaims = this.createClaims(jwtUserClaim);

        Date now = new Date(System.currentTimeMillis());

        try {
            String accessToken = Jwts.builder()
                    .claims(tokenClaims)
                    .issuer(jwtProperties.getIssuer())
                    .audience().add(jwtProperties.getAudience()).and()
                    .issuedAt(now)
                    .expiration(new Date(now.getTime() +
jwtProperties.getExpired() * MILLI_SECOND))
                    .signWith(secret)
                    .compact();

            return TokenResponse.create(accessToken);
        } catch (Exception ex) {
            log.error("JWT 생성 중 오류가 발생했습니다.", ex);
            return null;
        }
    }


    public Map<String, Object> createClaims(JwtUserClaim jwtUserClaim) {
        return Map.of(
                USER_ID, jwtUserClaim.userId(),
                USER_ROLE, jwtUserClaim.role()
        );
    }

}
```

JWT 생성 및 파싱 컴포넌트 만들기 (2) : 파싱 메서드

System Software Lab.

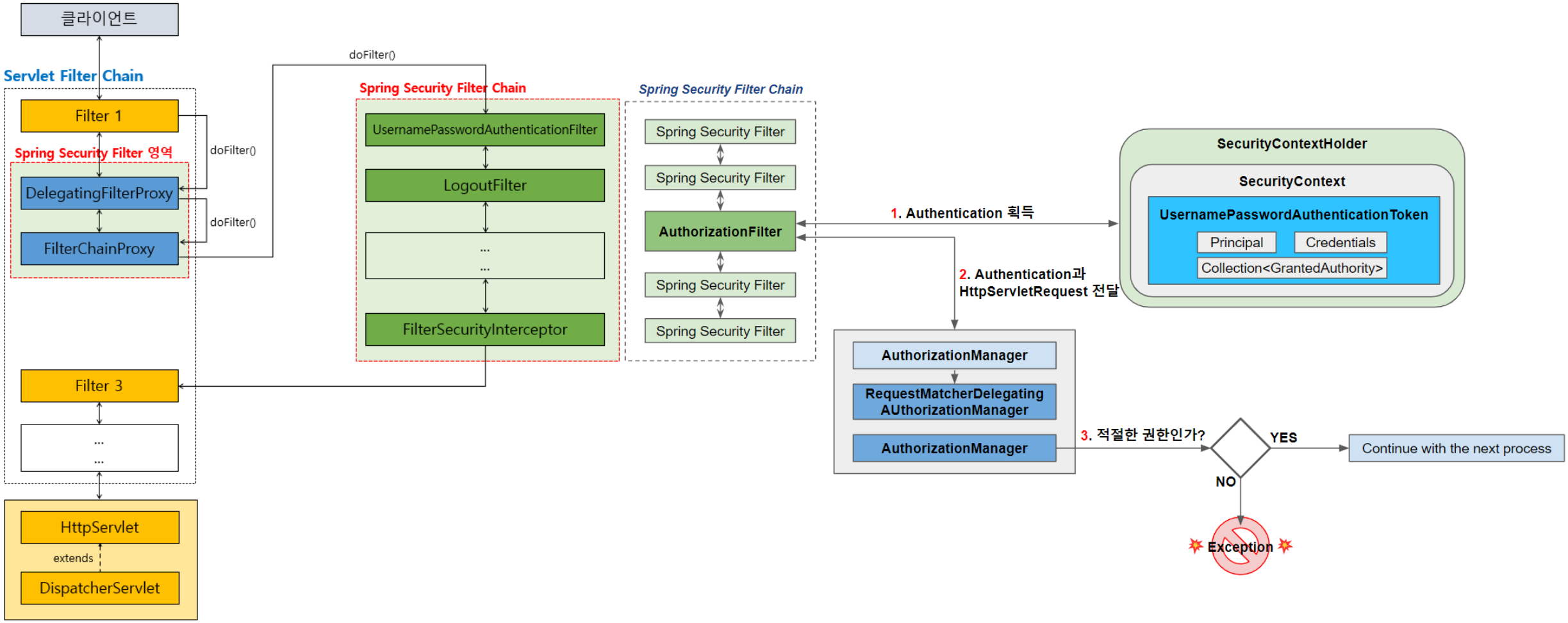
```java
package com.patulus.demo.global.jwt;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Date;
import java.util.Map;

@Component
@RequiredArgsConstructor
@Slf4j
public class JwtHandler {

    public static final String USER_ID = "USER_ID";
    public static final String USER_ROLE = "USER_ROLE";
    private static final long MILLI_SECOND = 1000L;

    private final JwtProperties jwtProperties;
    private final SecretKey secret = new
SecretKeySpec(jwtProperties.getSecret().getBytes(StandardCharsets.UTF_8),
Jwts.SIG.HS256.key().build().getAlgorithm());
```

```java
public JwtUserClaim parseToken(String token) {
        try {
            Claims claims = Jwts.parser()
                    .verifyWith(secret)
                    .build()
                    .parseSignedClaims(token)
                    .getPayload();

            return convert(claims);
        } catch (Exception ex) {
            log.error("JWT 검증 후 사용자 정보를 얻는 과정에서 오류가 발생했습
니다.", ex);
            return null;
        }
    }

    public JwtUserClaim convert(Claims claims) {
        return new JwtUserClaim(
                claims.get(USER_ID, Long.class),
                claims.get(USER_ROLE, String.class)
        );
    }
```

JWT 기반 인증에 필요한 프로젝트 기반 다지기

1. JWT 설정 파일 만들기
2. JWT 생성 및 파싱 컴포넌트 만들기
3. **JWT 검증 필터 만들기 및 등록**
4. 사용자 확인을 위한 서비스 만들기
5. 회원가입 및 로그인 엔드 포인트 및 로직 만들기
6. 권한이 필요한 엔드 포인트 만들기

Filter 생성 이유


클라이언트
Servlet Filter Chain
Filter 1
Spring Security Filter 영역
DelegatingFilterProxy
FilterChainProxy
doFilter()
doFilter()
doFilter()
Filter 3
...
...
HttpServlet
extends
DispatcherServlet
Spring Security Filter Chain
UsernamePasswordAuthenticationFilter
LogoutFilter
...
...
FilterSecurityInterceptor
Spring Security Filter Chain
Spring Security Filter
Spring Security Filter
AuthorizationFilter
Spring Security Filter
Spring Security Filter
1. Authentication 획득
2. Authentication과
HttpServletRequest 전달
SecurityContextHolder
SecurityContext
UsernamePasswordAuthenticationToken
Principal
Credentials
Collection<GrantedAuthority>
AuthorizationManager
RequestMatcherDelegating
AUthorizationManager
AuthorizationManager
3. 적절한 권한인가?
YES
Continue with the next process
NO
Exception

Spring Security 사용자 정의 Filter 만들기

System Software Lab.

```java
package com.patulus.demo.global.jwt;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.patulus.springBoardProjectAshurin.global.base.entity.exception.ErrorCode;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpHeaders;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.util.matcher.RequestHeaderRequestMatcher;
import org.springframework.security.web.util.matcher.RequestMatcher;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;
import java.util.Optional;

import static com.patulus.demo.global.base.entity.response.ResponseUtil.createFailureResponse;

@RequiredArgsConstructor
@Slf4j
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    public static final String BEARER_PREFIX = "Bearer ";

    private final AuthenticationManager authenticationManager;
    private final RequestMatcher requestMatcher = new RequestHeaderRequestMatcher(HttpHeaders.AUTHORIZATION);

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {
        if (!requestMatcher.matches(request)) {
            filterChain.doFilter(request, response);
            return;
        }

        try {
            String tokenValue = resolveToken(request)
                    .orElseThrow(() -> new
SecurityException(ErrorCode.JWT_NOT_EXIST.getMessage()));

            JwtAuthenticationToken token = new JwtAuthenticationToken(tokenValue);
            Authentication authentication = authenticationManager.authenticate(token);

            SecurityContextHolder.getContext().setAuthentication(authentication);
            filterChain.doFilter(request, response);
        } catch (Exception ex) {
            this.handleServiceException(response, ex);
        }
    }

    private Optional<String> resolveToken(HttpServletRequest request){
        String bearerToken = request.getHeader(HttpHeaders.AUTHORIZATION);

        if(bearerToken != null && bearerToken.startsWith(BEARER_PREFIX)){
            return Optional.of(bearerToken.substring(BEARER_PREFIX.length()));
        }

        return Optional.empty();
    }
}
```

토큰 인증 매니저 만들기

System Software Lab.

```java
@Slf4j
@RequiredArgsConstructor
@Component
public class TokenProvider implements AuthenticationProvider {

    private final JwtHandler jwtHandler;

    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        JwtAuthenticationToken jwtAuthenticationToken = (JwtAuthenticationToken) authentication;
        String tokenValue = jwtAuthenticationToken.token();
        if (tokenValue == null) {
            return null;
        }

        try {
            JwtUserClaim claims = jwtHandler.parseToken(tokenValue);

            return new JwtAuthentication(claims);
        } catch (io.jsonwebtoken.security.SecurityException | MalformedJwtException e) {
            log.info("잘못된 JWT 서명입니다.");
        } catch (ExpiredJwtException e) {
            log.info("만료된 JWT 토큰입니다.");
        } catch (UnsupportedJwtException e) {
            log.info("지원되지 않는 JWT 토큰입니다.");
        } catch (IllegalArgumentException e) {
            log.info("JWT 토큰이 잘못되었습니다.");
        }

        return null;
    }

    @Override
    public boolean supports(Class<?> authentication) {
        return JwtAuthenticationToken.class.isAssignableFrom(authentication);
    }

}
```

```java
public JwtUserClaim parseToken(String token) {
        try {
            Claims claims = Jwts.parser()
                    .verifyWith(secret)
                    .build()
                    .parseSignedClaims(token)
                    .getPayload();

            return convert(claims);
        } catch (Exception ex) {
            log.error("JWT 검증 후 사용자 정보를 얻는 과정에서 오류가 발생했습니다
            return null;
        }
    }

    public JwtUserClaim convert(Claims claims) {
        return new JwtUserClaim(
                claims.get(USER_ID, Long.class),
                claims.get(USER_ROLE, String.class)
        );
    }
```

SpringContextHolder와 Authentication 객체

```
import org.springframework.security.core.Authentication;
```

```
public record JwtAuthenticationToken(
        String token
) implements Authentication {
}
```

처음 접속 시 유용한 정보가 없으므로 토큰만

담는 객체를 생성

```
public record JwtAuthentication(
        Long userId,
        String role
) implements Authentication {
```

실제 유용한 정보를 SpringContextHolder에 저장할 수 있도록

토큰을 파싱하여 유용한 정보를 담는 객체를 생성

토큰은 토큰 인증 매니저에서 검증 및 파싱

SpringContextHolder

- 스프링 빈(객체 인스턴스)을 관리하는 ApplicationContext(Map)와 결합
- 클라이언트 요청당 하나의 인스턴스가 생성
- setAuthentication으로 인증 정보를 등록해 엔드 포인트의 권한 확인 용이

Authentication

- 사용자 인증 정보를 저장하는 객체
- SpringContextHolder에 인증 정보를 등록하려면

  Authentication*(인터페이스)* 타입이어야 함

JWT 기반 인증에 필요한 프로젝트 기반 다지기

1. JWT 설정 파일 만들기
2. JWT 생성 및 파싱 컴포넌트 만들기
3. JWT 검증 필터 만들기 및 등록
4. **사용자 확인을 위한 서비스 만들기**
5. 회원가입 및 로그인 엔드 포인트 및 로직 만들기
6. 권한이 필요한 엔드 포인트 만들기

실제 회원인지 확인하기 위한 UserDetailService 만들기

System Software Lab.

```java
@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {
    private final UserRepository userRepository;

    @Override
    @Transactional
    public UserDetails loadUserByUsername(final String username) {
        return userRepository.findByUsername(username)
                .map(user -> createUser(username, user))
                .orElseThrow(() -> new UsernameNotFoundException(username + " -> 데이터베이스에서 찾을 수 없습니다."));
    }

    private org.springframework.security.core.userdetails.User createUser(String username, User user) {
        List<GrantedAuthority> grantedAuthorities = List.of(new SimpleGrantedAuthority(user.getRole()));

        return new org.springframework.security.core.userdetails.User(user.getUsername(),
                user.getPassword(),
                grantedAuthorities);
    }
}
```

Spring Security가 제공하는 UserDetail이 아닌 DB와 매핑되는 엔티티

JWT 기반 인증에 필요한 프로젝트 기반 다지기

System Software Lab.

1. JWT 설정 파일 만들기
2. JWT 생성 및 파싱 컴포넌트 만들기
3. JWT 검증 필터 만들기 및 등록
4. 사용자 확인을 위한 서비스 만들기
5. **회원가입 및 로그인 엔드 포인트 및 로직 만들기**
6. 권한이 필요한 엔드 포인트 만들기

User 엔티티

System Software Lab.

```java
@Entity
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@Getter
public class User extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String password;

    private String role;

    @Builder
    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

}
```

회원가입, 로그인, 사용자 정보 찾기 로직 (비즈니스 로직)

System Software Lab.

```java
@Service
@RequiredArgsConstructor
@Transactional(readOnly = true)
public class UserService {
    private final UserRepository userRepository;
    private final AuthenticationManagerBuilder authenticationManagerBuilder;
    private final JwtHandler jwtHandler;
    private final PasswordEncoder passwordEncoder;

    @Transactional
    public User signUp(UserSignInRequest request) {
        if (userRepository.findByUsername(request.username()).orElse(null) != null) {
            throw new RuntimeException("이미 가입되어 있는 유저입니다.");
        }

        User user = User.builder()
                .username(request.username())
                .password(passwordEncoder.encode(request.password()))
                .role("ROLE_USER")
                .build();

        return userRepository.save(user);
    }
```

```java
    public TokenResponse signIn(UserSignInRequest request) {
        User user = userRepository.findByUsername(request.username())
                .orElseThrow(() -> new RuntimeException("해당 유저가 없습니다."));

        if (!passwordEncoder.matches(request.password(), user.getPassword())) {
            throw new RuntimeException("비밀번호가 다릅니다.");
        }

        return jwtHandler.createTokens(JwtUserClaim.create(user));
    }

    public String userInfo(Long userId) {
        User user = userRepository.findById(userId)
                .orElseThrow(() -> new RuntimeException("해당 유저가 없습니다."));

        return user.getUsername();
    }
}
```

회원가입, 로그인 엔드 포인트

System Software Lab.

```java
@RestController
@RequiredArgsConstructor
public class UserController {

    private final UserService userService;

    @PostMapping("/sign-up")
    public ResponseBody<Void> signUp(@RequestBody UserSignInRequest request) {
        userService.signUp(request);

        return ResponseUtil.createSuccessResponse();
    }

    @PostMapping("/sign-in")
    public ResponseEntity<ResponseBody<TokenResponse>> signIn(@RequestBody UserSignInRequest request) {
        TokenResponse tokenResponse = userService.signIn(request);

        HttpHeaders headers = new HttpHeaders();
        headers.set(HttpHeaders.AUTHORIZATION, "Barer " + tokenResponse.accessToken());

        return new ResponseEntity<>(ResponseUtil.createSuccessResponse(tokenResponse), headers, HttpStatus.OK);
    }

}
```

JWT 기반 인증에 필요한 프로젝트 기반 다지기

1. JWT 설정 파일 만들기
2. JWT 생성 및 파싱 컴포넌트 만들기
3. JWT 검증 필터 만들기 및 등록
4. 사용자 확인을 위한 서비스 만들기
5. 회원가입 및 로그인 엔드 포인트 및 로직 만들기
6. **권한이 필요한 엔드 포인트 만들기**

권한이 필요한 엔드 포인트

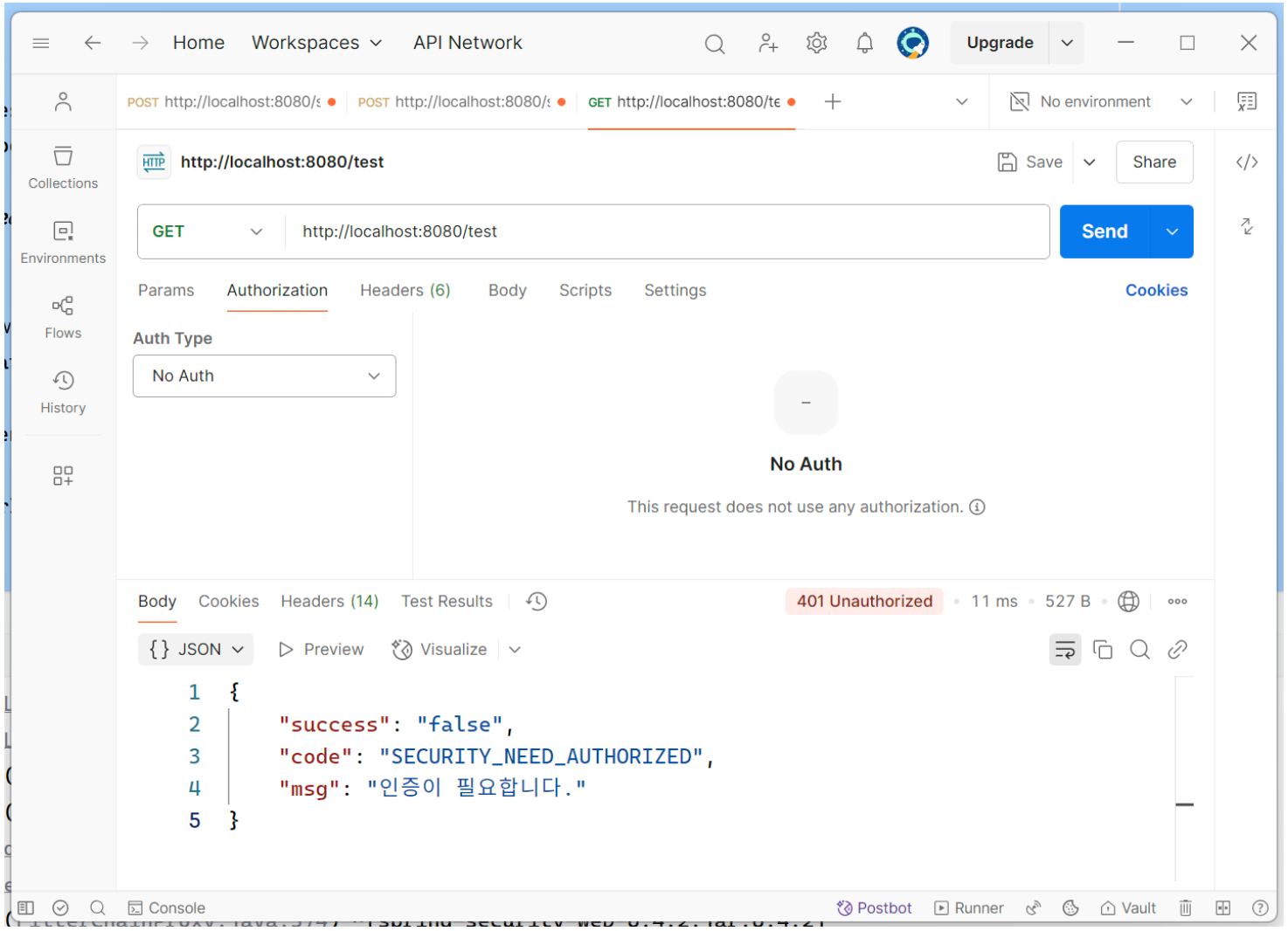
System Software Lab.

```java
@RestController
@RequiredArgsConstructor
public class UserController {

    private final UserService userService;

    @GetMapping("/test")
    @PreAuthorize("isAuthenticated() and hasRole('ROLE_USER')")
    public String test() {
        JwtAuthentication authentication = (JwtAuthentication) SecurityContextHolder.getContext().getAuthentication();

        return userService.userInfo(authentication.userId());
    }

}
```

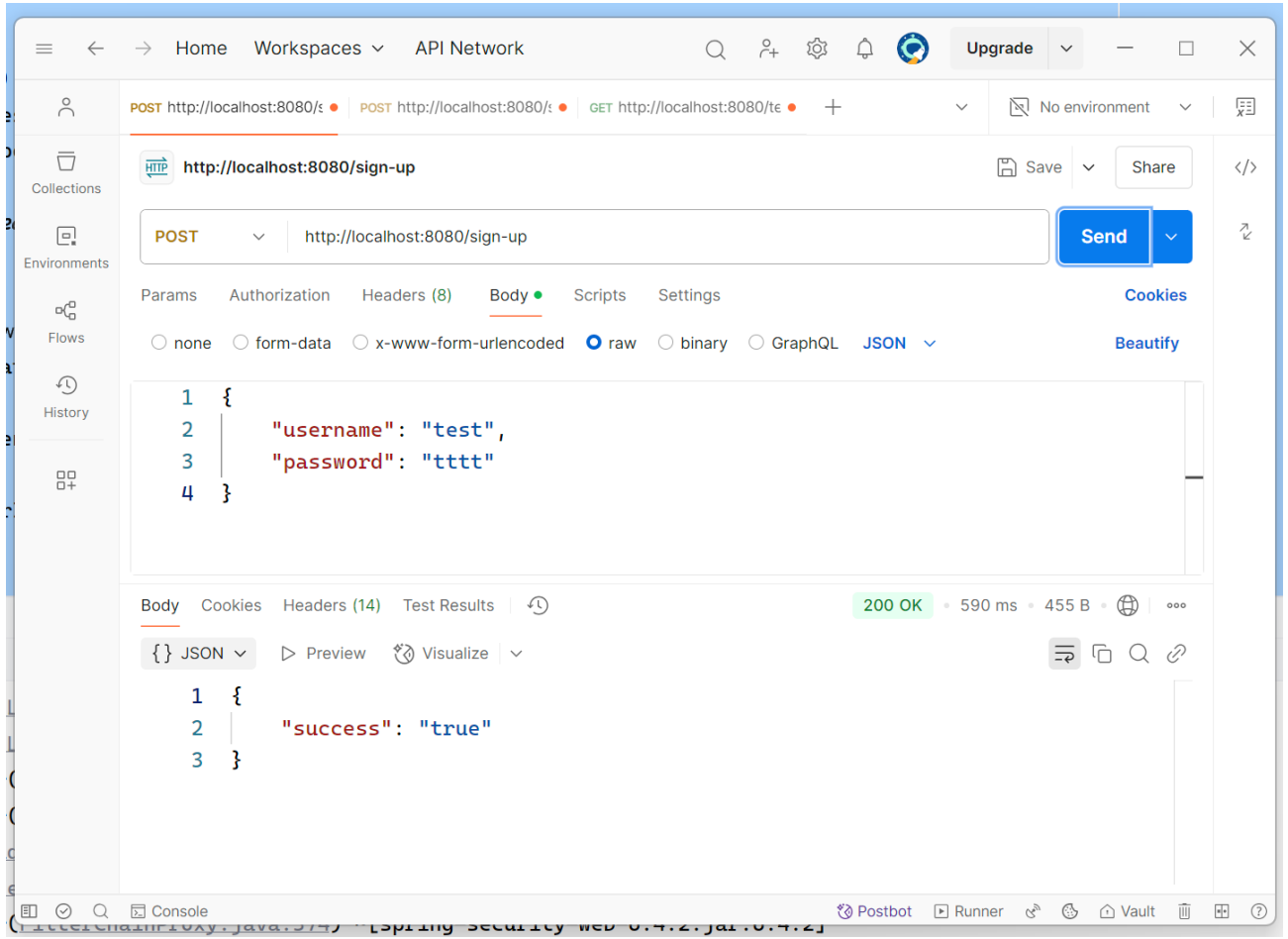
02.
테스트

테스트 (1) 로그인하지 않았을 때 권한이 필요한 엔드 포인트에 요청

System Software Lab.



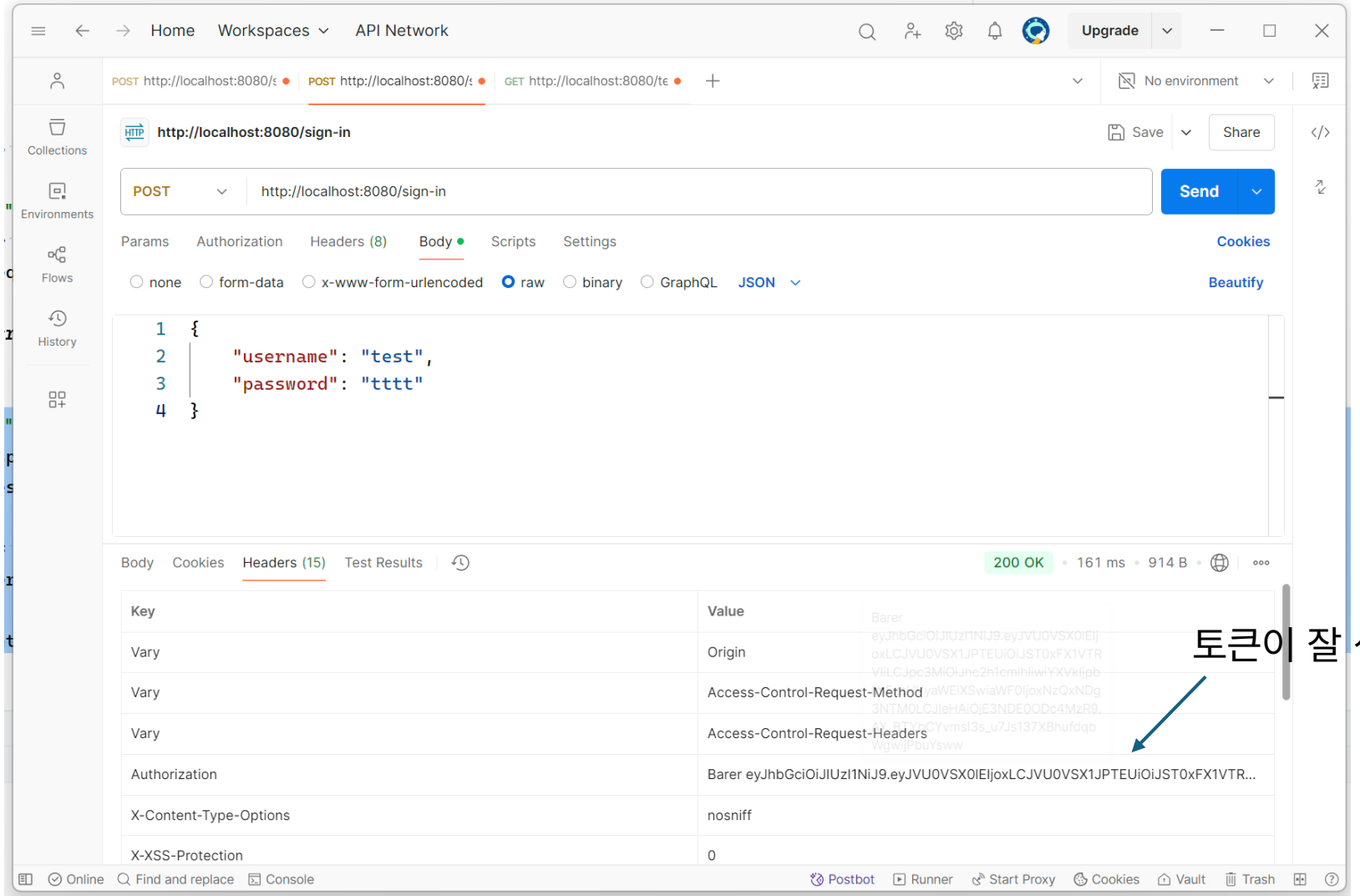
테스트 (2) 회원가입



password	role	username
$2a$10$Uj8rGAqZK.rvI3BtkArJU./BqzAwqJXji0IRX8umqsJGwAkYmiMfS	ROLE_USER	test

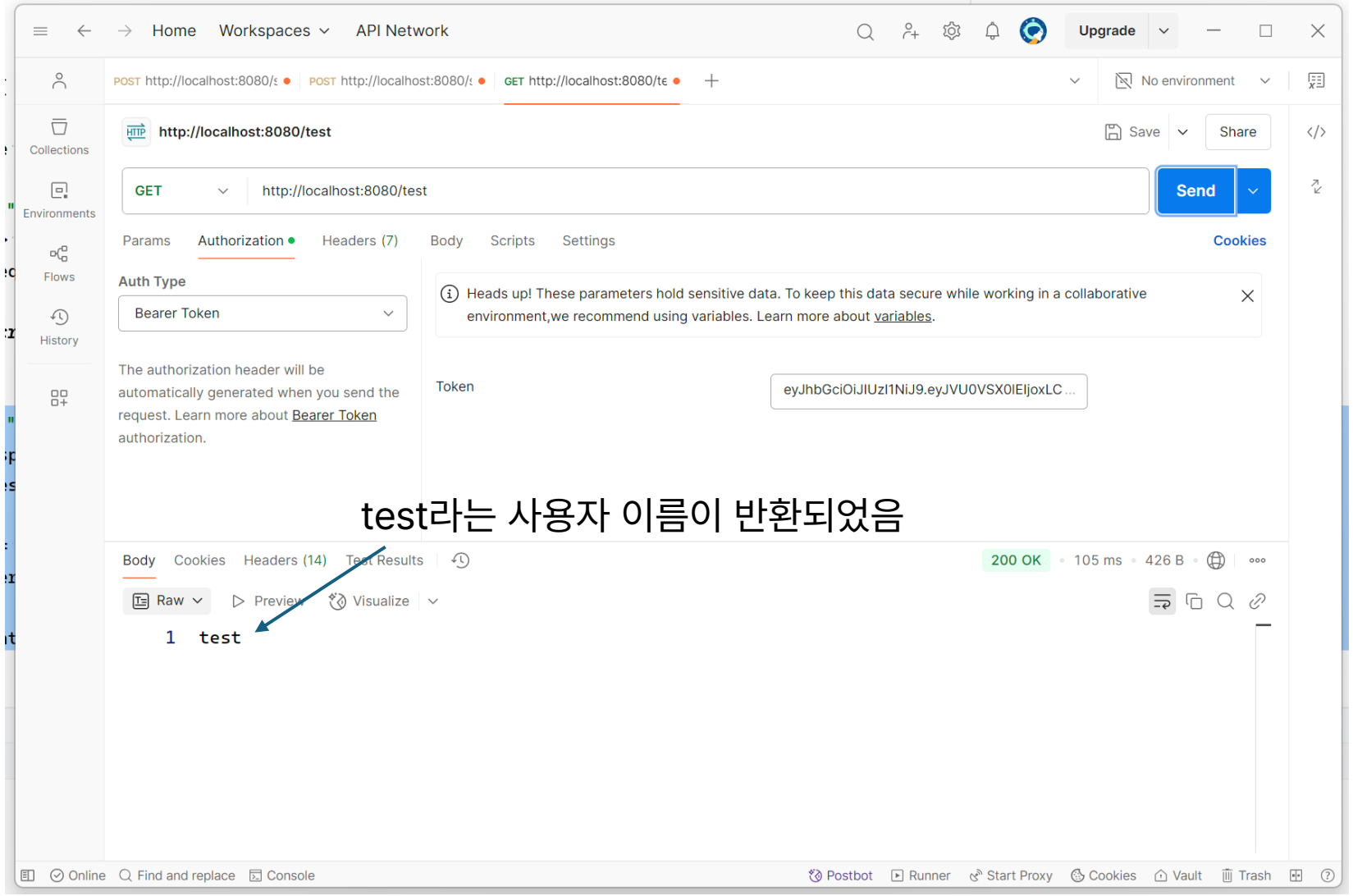
비밀번호와 솔트가 Bcrypt로 암호화되는 Spring Security 로직이 동작 및 반환 값 저장
String의 equals로 비교 불가하며 별도 제공되는 matches 메서드로 비교해야 함

테스트 (3) 로그인



토큰이 잘 생성되었다!

테스트 (4) 로그인 후 권한이 필요한 엔드 포인트에 요청



테스트 (5) 토큰 디코딩

System Software Lab.

JWT Decoder JWT Encoder

Paste a JWT below that you'd like to decode, validate, and verify.

Generate example

ENCODED VALUE

JSON WEB TOKEN (JWT) COPY CLEAR

Valid JWT

Signature Verified

```
eyJhbGciOiJIUzI1NiJ9.eyJVU0VSX0lEIjoxLCJVU0VSX1JPTEUiOiJST0xFX1VTRVIiLCJpc3MiOiJhc2h1c
mlhIiwiYXVkIjpbImFzaHVyaWEiXSwiaWF0IjoxNzQxNDg3NTM0LCJleHAiOjE3NDE0ODc4MzR9.AY_BTYbCYv
msI3s_u7Js137XBhufdqbWgwijPbuYsww
```

DECODED HEADER

JSON CLAIMS TABLE COPY

```
{
  "alg": "HS256"
}
```

DECODED PAYLOAD

JSON CLAIMS TABLE COPY

```
{
  "USER_ID": 1,
  "USER_ROLE": "ROLE_USER",
  "iss": "d449",
  "aud": [
    "d449"
  ],
  "iat": 1741487534,
  "exp": 1741487834
}
```

JWT SIGNATURE VERIFICATION (OPTIONAL)

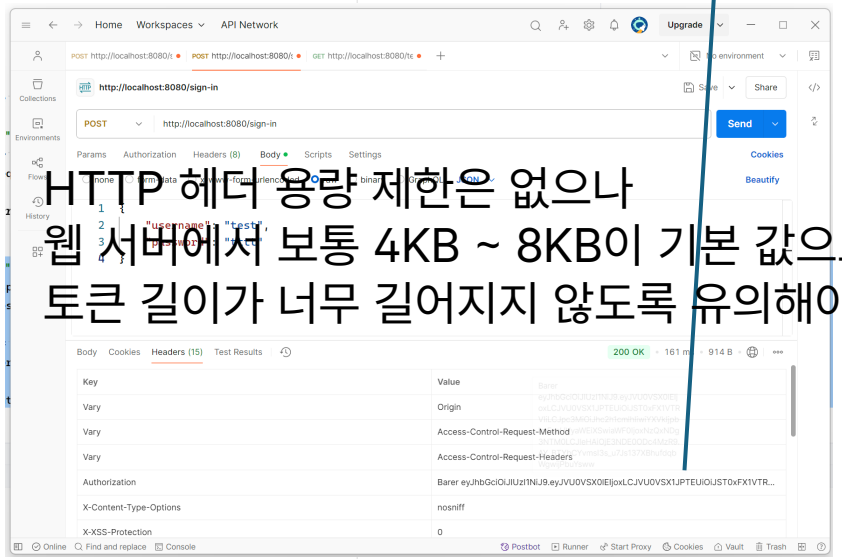
Enter the secret used to sign the JWT below:

SECRET COPY CLEAR

Valid secret

d449

Encoding Format UTF-8



HTTP 헤더 용량 제한은 없으나
웹 서버에서 보통 4KB ~ 8KB이 기본 값으로 지정돼 있어
토큰 길이가 너무 길어지지 않도록 유의해야 함

감사합니다

Email / park@duck.com
Insta / @yeonjong.park
GitHub / patulus

Photo by Pawel Czerwinski