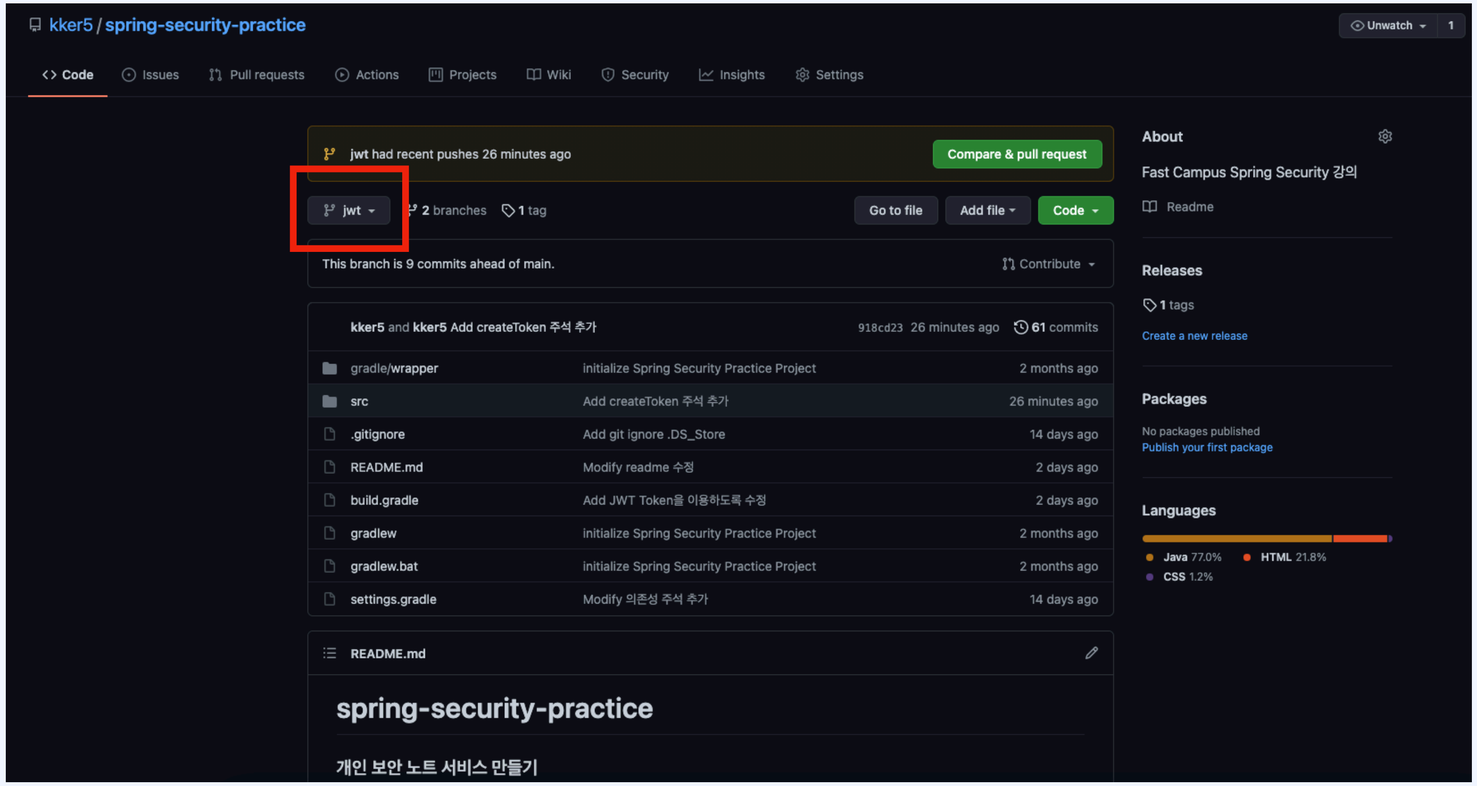
Spring Security

6 Spring Security + JWT 구현하기

Spring Security + JWT 구현하기

Github (개인보안노트서비스프로젝트에서 branch만 jwt로 변경하면 됩니다.)

https://github.com/kker5/spring-security-practice/tree/jwt



JWT Util 만들기

의존성 추가

```
// jjwt
implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.2'
runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.2'
```

JwtKey

JWT Secret Key 를 관리하고 제공합니다.

Key Rolling을 지원합니다.

JwtUtils

JWT 토큰을 생성하거나 Parsing하는 메소드를 제공합니다.

SigningKeyResolver

JWT의 헤더에서 kid를 찾아서 Key(SecretKey+알고리즘)를 찾아옵니다.

Signature를 검증할 때 사용합니다.

JWT Util 만들기

JwtKey.getKey()

kid로 Secret Key를 찾아옵니다.

```
/**
 * kid로 Key찾기
 *
 * @param kid kid
 * @return Key
 */
public static Key getKey(String kid) {
    String key = SECRET_KEY_SET.getOrDefault(kid, null);
    if (key == null)
        return null;
    return Keys.hmacShaKeyFor(key.getBytes(StandardCharsets.UTF_8));
}
```

JWT Util 만들기

JwtKey.getRandomKey()

여러개의 Secret Key 중에 랜덤으로 선택하여 kid와 SecretKey를 제공합니다.

```
/**
 * SECRET_KEY_SET 에서 랜덤한 KEY 가져오기
 *
 * @return kid와 key Pair
 */
public static Pair<String, String> getRandomKey() {
    String kid = KID_SET[randomIndex.nextInt(KID_SET.length)];
    return Pair.of(kid, SECRET_KEY_SET.get(kid));
}
```

JWT Util 만들기

JwtUtils.createToken()

User로 JWT Token을 만듭니다.

HEADER : alg(알고리즘종류), kid

PAYLOAD : sub(username), iat(토큰발행시간), exp(토큰만료시간)

SIGNATURE : JwtKey.getRandomKey로 구한 Secret Key로 HS512 해시

```
public static String createToken(User user) {
    Claims claims = Jwts.claims().setSubject(user.getUsername());// subject
    Date now = new Date(); // 현재 시간
    Pair<String, String> key = JwtKey.getRandomKey();
    // JWT Token 생성
    return Jwts.builder()
            .setClaims(claims) // 정보 저장
            .setIssuedAt(now) // 토큰 발행 시간 정보
            .setHeaderParam(JwsHeader.KEY_ID, key.getFirst()) // kid
            .setExpiration(new Date(now.getTime() + JwtProperties.EXPIRATION_TIME)) // 토큰 만료 시간 설정 (now + 10분)
            .signWith(Keys.hmacShaKeyFor(key.getSecond().getBytes(StandardCharsets.UTF_8))) // 알고리즘과 SecretKey
            .compact();
}
```

JWT Util 만들기

JwtUtils.getUsername()

JWT Token에서 username을 구합니다.

SigningKeyResolver로 Signature를 검증합니다.

이 과정중에 토큰이 만료가 되었거나 적합하지 않으면 Exception이 발생합니다.

```
/**
 * 토큰에서 username 찾기
 * @param token 토큰
 * @return username
 */
public static String getUsername(String token) {
    return Jwts.parserBuilder() // parser용 builder
            .setSigningKeyResolver(SigningKeyResolver.instance)
            .build()
            .parseClaimsJws(token)
            .getBody()
            .getSubject();
}
```

JWT Util 만들기

SigningKeyResolver

SigningKeyResolver는 헤더에 있는 kid를 조회해서 그 kid에 해당하는 비밀키를 가져옵니다.

```
/**
 * JwsHeader를 통해 Signature 복호화에 필요한 Key를 조회해옵니다.
 */
public class SigningKeyResolver extends SigningKeyResolverAdapter {
    public static SigningKeyResolver instance = new SigningKeyResolver();
    @Override
    public Key resolveSigningKey(JwsHeader jwsHeader, Claims claims) {
        String kid = jwsHeader.getKeyId();
        if (kid == null)
            return null;
        return JwtKey.getKey(kid);
    }
}
```

JWT Filter 만들기

JwtAuthenticationFilter

로그인을 하면 JWT 토큰을 응답 쿠키에 넣어줍니다.

UsernamePasswordAuthenticationFilter를 상속했기 때문에 기본동작은 거의 비슷합니다.

로그인에 성공하면 User 정보로 JWT Token을 생성하고 응답 쿠키에 값을 넣어줍니다.

```
@Override
protected void successfulAuthentication(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain chain,
        Authentication authResult
) throws IOException {
    User user = (User) authResult.getPrincipal();
    String token = JwtUtils.createToken(user);
    // 쿠키 생성
    Cookie cookie = new Cookie(JwtProperties.COOKIE_NAME, token);
    cookie.setMaxAge(JwtProperties.EXPIRATION_TIME); // 쿠키의 만료시간 설정
    cookie.setPath("/");
    response.addCookie(cookie);
    response.sendRedirect("/");
```

JWT Filter 만들기

JwtAuthorizationFilter

1. Cookie에서 JWT Token을 구합니다.
2. JWT Token을 파싱하여 username을 구합니다.
3. username으로 User를 구하고 Authentication을 생성합니다.
4. 생성된 Authentication을 SecurityContext에 넣습니다.
5. Exception이 발생하면 응답의 쿠키를 null로 변경합니다.

JWT Filter 만들기

SpringSecurityConfig

```
// basic authentication
http.httpBasic().disable(); // basic authentication filter 비활성화
// csrf
http.csrf().disable();
// remember-me
http.rememberMe().disable();
// stateless
http.sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
// jwt filter
http.addFilterBefore(
        new JwtAuthenticationFilter(authenticationManager()),
        UsernamePasswordAuthenticationFilter.class
).addFilterBefore(
        new JwtAuthorizationFilter(userRepository),
        BasicAuthenticationFilter.class
);
```

JWT Filter 만들기

결과화면