서비스 로직 작성

User 서비스 로그인

User 프로젝트 의존성 추가


```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-loadbalancer'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    runtimeOnly 'com.mysql:mysql-connector-j:8.4.0'

    implementation 'io.jsonwebtoken:jjwt-api:0.12.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-gson:0.12.5'

    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}
```

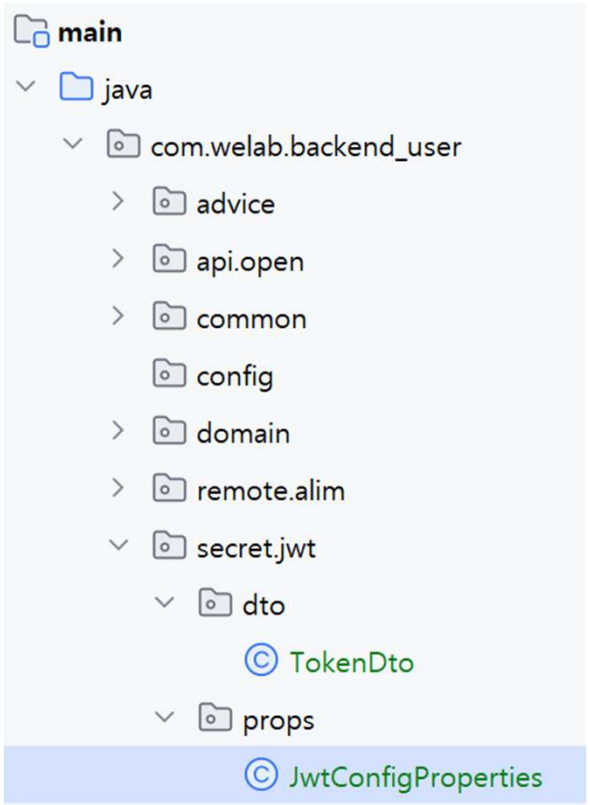
User 프로젝트 JWT 설정

tablet

```
jwt:
  expires-in: 86400 # 1day
  mobile-expires-in: 31536000
        -expires-in: 31536000
  secret-key: AADfaskllew32dsfasdTG764Gdslkj298GsWg86G
```

application-local.yml

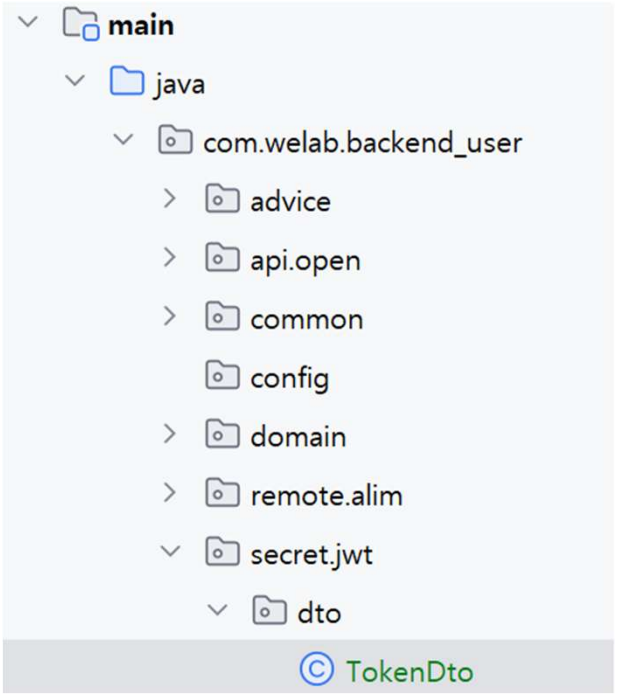
User 프로젝트 코드 작성 – JwtConfigProperties



application - local.yml에 작성된 jwt 속성 값 가져오기

```
@Component
@ConfigurationProperties(value = "jwt", ignoreUnknownFields = true)
@Getter
@Setter
public class JwtConfigProperties {
    private Integer expiresIn;
    private Integer mobileExpiresIn;
    private Integer tabletExpiresIn;
    private String secretKey;
}
```

User 프로젝트 코드 작성 – TokenDto




```java
@NoArgsConstructor(access = AccessLevel.PRIVATE)
public class TokenDto {
    @Getter
    @Setter
    @NoArgsConstructor
    @AllArgsConstructor
    public static class JwtToken {
        private String token;
        private Integer expiresIn;
    }

    @Getter
    @RequiredArgsConstructor
    public static class AccessToken {
        private final JwtToken access;
    }

    @Getter
    @Setter
    @RequiredArgsConstructor
    public static class AccessRefreshToken {
        private final JwtToken access;
        private final JwtToken refresh;
    }
}
```

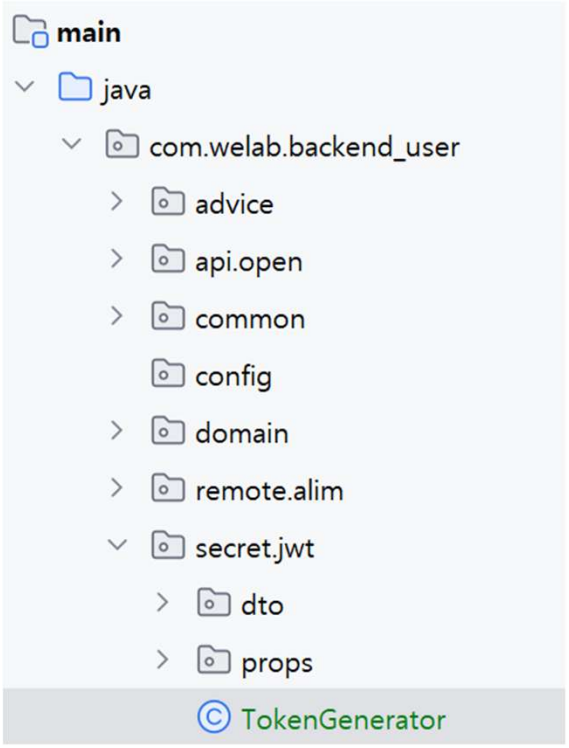
JWT의 기본 정보를 담는 역할

오직 Access Token만 필요한 경우에 사용
ex. Access Token만 사용하여 인증이 필요한 API를 호출 시

Access Token과 Refresh Token이 모두 필요한 경우에 사용
ex. 사용자 로그인이나 토큰 재발급 시에 이 두 가지 토큰을 함께 클라이언트에 전달해야 할 때 유용

API 응답 시점에 따라 필요한 토큰 정보만 담아서 보낼 수 있습니다.
예를 들어, 로그인 시에는 AccessRefreshToken을,
Access Token 만료 후 재발급 시에는 AccessToken만 보낼 수 있습니다.
이를 통해 불필요한 데이터를 전송하는 것을 방지하고, 클라이언트와 서버 간의 통신 효율성을 높일 수 있습니다.

User 프로젝트 코드 작성 – TokenGenerator



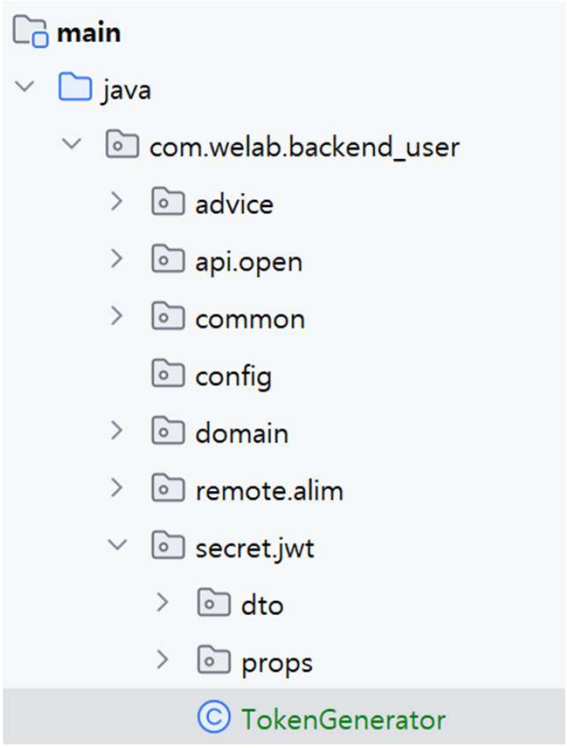
+ Java 변수를 Main Memory에 저장하겠다라는 것을 명시
+ 다른 쓰레드도 메모리에 저장된 같은 값을 사용

```
@Component
@RequiredArgsConstructor
public class TokenGenerator {
    private final JwtConfigProperties configProperties;

    private volatile SecretKey secretKey;

    private SecretKey getSecretKey() {
        if (secretKey == null) {
            synchronized (this) {
                if (secretKey == null) {
                    secretKey = Keys.hmacShaKeyFor(Decoders.BASE64.decode(configProperties.getSecretKey()));
                }
            }
        }
        return secretKey;
    }
```

synchronized() 진입 필요 여부 확인용

+ Critical Section과 유사하게 해당 블럭은 한 명만 수행 가능
+ 두 명이 들어오면 한 명이 대기

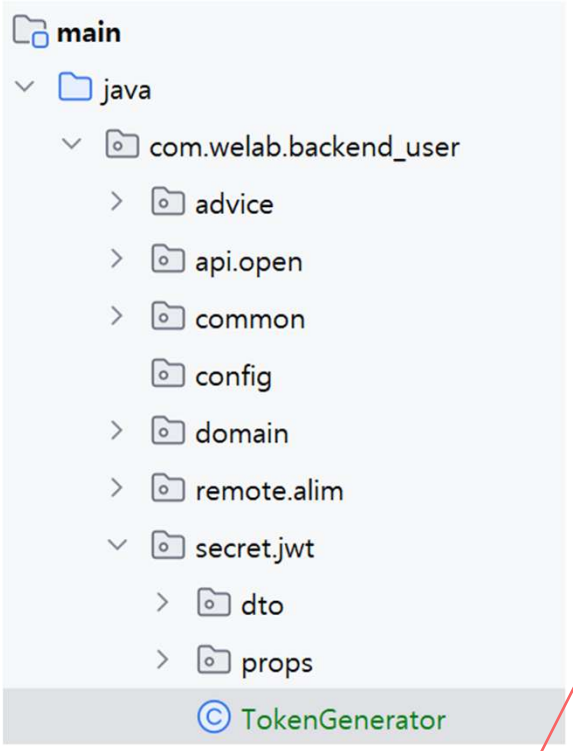
User 프로젝트 코드 작성 – TokenGenerator (계속)




```
public TokenDto.AccessToken generateAccessToken(String userId, String deviceType) {
    TokenDto.JwtToken jwtToken = this.generateJwtToken(userId, deviceType, false);
    return new TokenDto.AccessToken(jwtToken);
}
```

```
public TokenDto.AccessRefreshToken generateAccessRefreshToken(String userId, String deviceType) {
    TokenDto.JwtToken accessJwtToken = this.generateJwtToken(userId, deviceType, false);
    TokenDto.JwtToken refreshJwtToken = this.generateJwtToken(userId, deviceType, true);
    return new TokenDto.AccessRefreshToken(accessJwtToken, refreshJwtToken);
}
```

User 프로젝트 코드 작성 – TokenGenerator (계속)

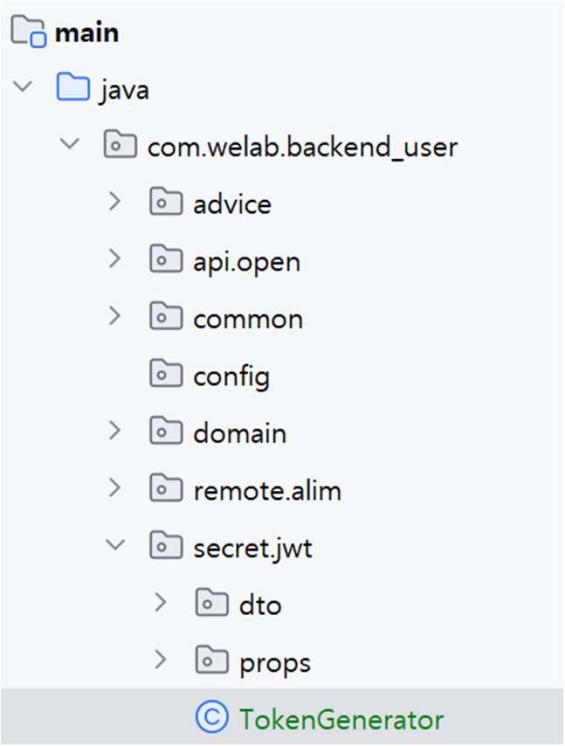



```
public TokenDto.JwtToken generateJwtToken(String userId,
                                          String deviceType,
                                          boolean refreshToken
) {
    int tokenExpiresIn = tokenExpiresIn(refreshToken, deviceType);
    String tokenType = refreshToken ? "refresh" : "access";

    String token = Jwts.builder()
            .issuer("welab")
            .subject(userId)
            .claim("userId", userId)
            .claim("deviceType", deviceType)
            .claim("tokenType", tokenType)
            .issuedAt(new Date())
            .expiration(new Date(System.currentTimeMillis() + tokenExpiresIn * 1000L))
            .signWith(getSecretKey())
            .header().add("typ", "JWT")
            .and()
            .compact();

    return new TokenDto.JwtToken(token, tokenExpiresIn);
}
```

정보 저장

String 값 대신 enum을 사용하여
오타로 인한 오류 최소화로 리팩토링 가능
ex. "refresh", "access", "userId", etc

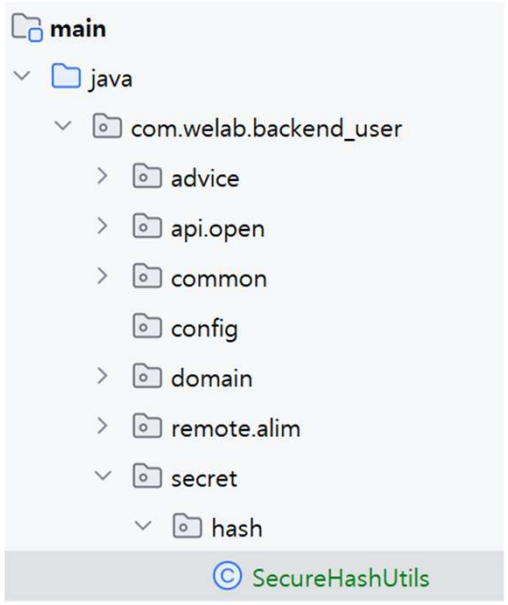
User 프로젝트 코드 작성 – TokenGenerator (계속)




```
    private int tokenExpiresIn(boolean refreshToken, String deviceType) {

        int expiresIn = 60 * 15;

        if (refreshToken) {
            if (deviceType != null) {
                if (deviceType.equals("WEB")) {
                    expiresIn = configProperties.getExpiresIn();
                } else if (deviceType.equals("MOBILE")) {
                    expiresIn = configProperties.getMobileExpiresIn();
                }
            } else {
                expiresIn = configProperties.getExpiresIn();
            }
        }

        return expiresIn;
    }
}
```

중복되는 코드이므로 리팩토리 가능

User 프로젝트 코드 작성 – SecureHashUtils (Hash는 복호화 불필요)

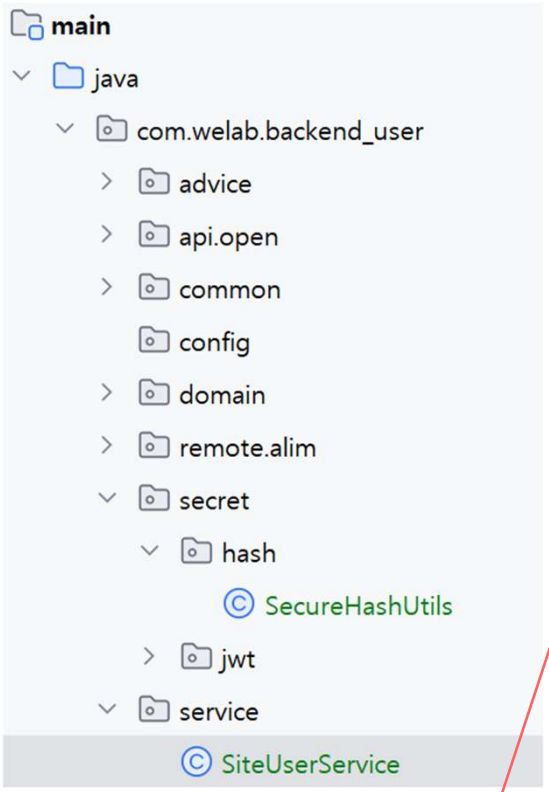



```
public class SecureHashUtils {
    public static String hash(String message) {
        // TODO: message -> SHA-1 또는 SHA-256 hash
        return message;
    }

    public static boolean matches(String message, String hashedMessage) {
        String hashed = hash(message);

        return hashed.equals(hashedMessage);
    }
}
```

해싱 기능

해싱 값 비교

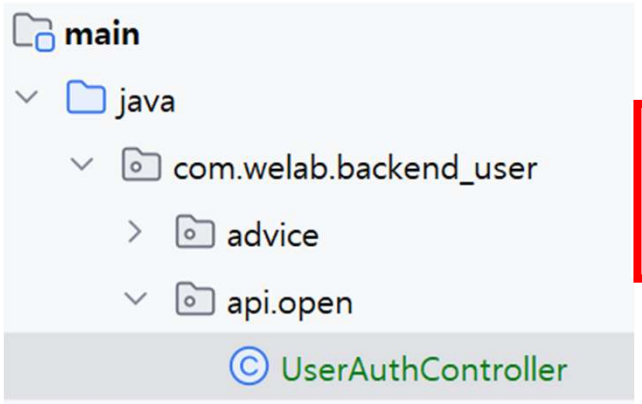
User 프로젝트 코드 작성 – SiteUserService 수정




```
@Transactional(readOnly = true)
public TokenDto.AccessRefreshToken login(SiteUserLoginDto loginDto) {
    SiteUser user = siteUserRepository.findByUserId(loginDto.getUserId());
    if (user == null) {
        throw new NotFound("사용자를 찾을 수 없습니다.");
    }

    if( !SecureHashUtils.matches(loginDto.getPassword(), user.getPassword())){
        throw new BadParameter("비밀번호가 맞지 않습니다.");
    }

    return tokenGenerator.generateAccessRefreshToken(loginDto.getUserId(), "WEB");
}
```

로그인 실패 원인을
추측하지 못하도록
에러 메시지 통일 必

User 프로젝트 코드 작성 – UserAuthController API 추가




```
@PostMapping(value = "/login")
public ApiResponseDto<TokenDto.AccessRefreshToken> login(@RequestBody @Valid SiteUserLoginDto loginDto) {
    TokenDto.AccessRefreshToken token = siteUserService.login(loginDto);
    return ApiResponseDto.createOk(token);
}
```

로그인 성공 시, TokenDto.AccessRefreshToken 반환