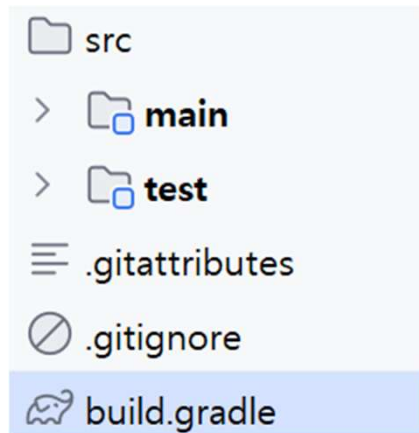


서비스 로직 작성

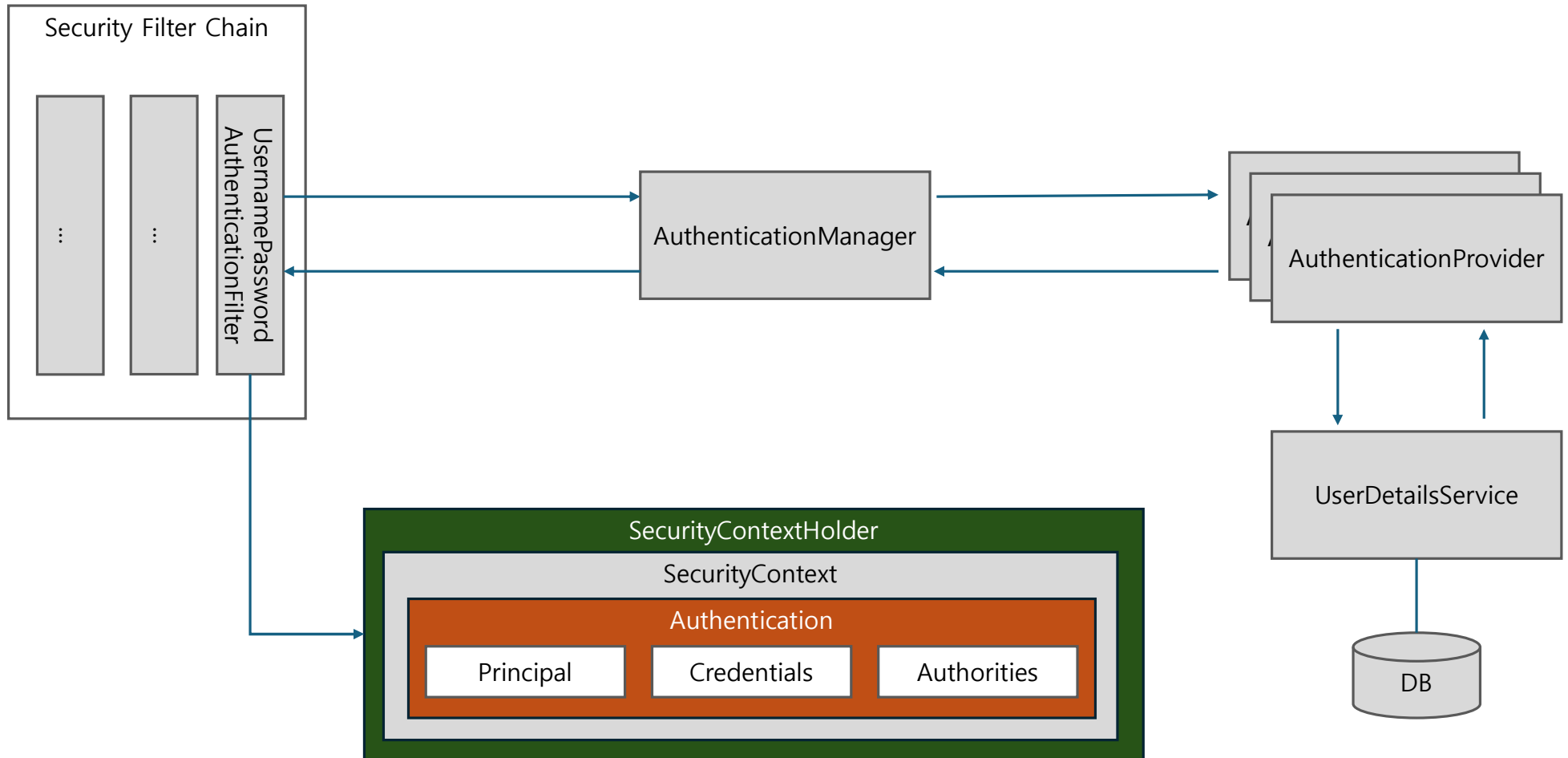
API Gateway 토큰 인증

Gateway 프로젝트 의존성 추가

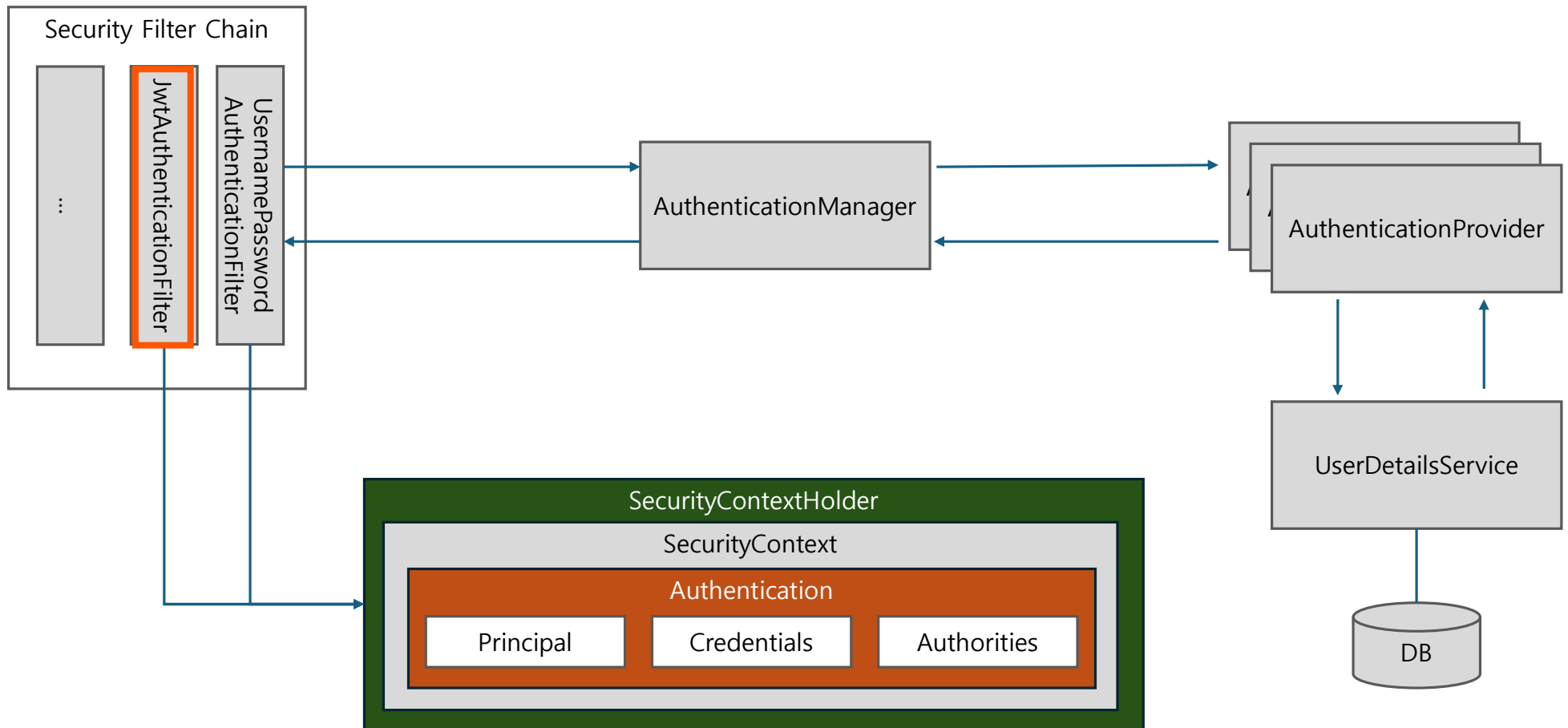


```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.cloud:spring-cloud-starter-gateway-server-webmvc'  
    implementation 'org.springframework.cloud:spring-cloud-starter-loadbalancer'  
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'  
    implementation 'org.springframework.boot:spring-boot-starter-actuator'  
  
    implementation 'org.springframework.boot:spring-boot-starter-security'  
    implementation 'io.jsonwebtoken:jjwt-api:0.12.5'  
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.5'  
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson: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'  
}
```

Spring Security 인증 프로세스



Spring Security 인증 프로세스 JwtAuthenticationFilter추가



Authentication interface

Spring Security에서 인증 정보를 담고 있는 가장 상위 개념
사용자의 인증 상태와 관련된 모든 정보 포함

```
public interface Authentication extends Principal, Serializable {  
    Collection<? extends GrantedAuthority> getAuthorities();  
    Object getCredentials();  
    Object getPrincipal();  
    boolean isAuthenticated();  
    ...  
}
```

→ 사용자의 권한 목록

→ 인증 자격 증명

→ 인증된 사용자에 대한 정보


→ 인증되었는지 여부

Principal interface

userId 등 인증된 사용자 정보 제공

```
public interface Principal {  
    public boolean equals(Object another);  
    public String toString();  
    public int hashCode();  
    public String getName();  
    ...  
}
```

```
public class UserPrincipal implements Principal {  
    private final String userId;  
  
    @Override  
    public String getName() {  
        return userId;  
    }  
  
    ...  
}
```




equals() 메소드에서 hashCode() 메소드를 사용하기 때문에 같이 Overriding 해야 함

Gateway 프로젝트 JWT 설정

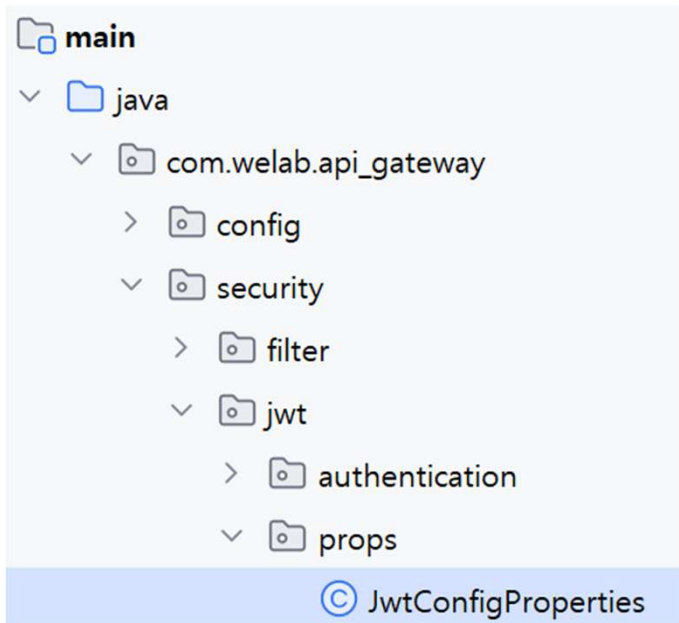
```
jwt:  
  header: Authorization  
  secret-key: AADfaskllew32dsfasdTG764Gdslkj298GsWg86Gkalsdjfalkdsjfkldsifklasdjfkwejqtklfksadnfklsadjmflk
```

application-local.yml

backend-user
JWT Secret Key가



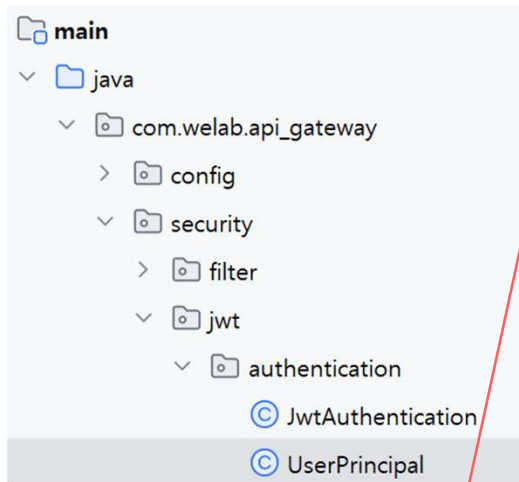
Gateway 프로젝트 코드 작성 - JwtConfigProperties



application - local.yml
jwt 가

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

Gateway 프로젝트 코드 작성 - UserPrincipal



```
@Getter
@RequiredArgsConstructor
public class UserPrincipal implements Principal {
    private final String userId;

    public boolean hasName() {
        return userId != null;
    }

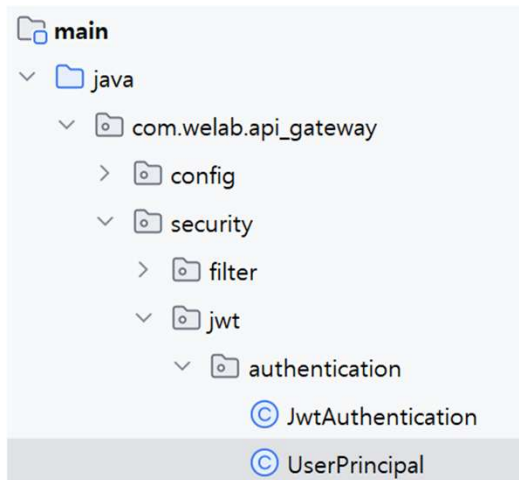
    public boolean hasMandatory() {
        return userId != null;
    }

    @Override
    public String toString() {
        return getName();
    }

    @Override
    public String getName() {
        return userId;
    }
}
```

@Component

Gateway 프로젝트 코드 작성 – UserPrincipal (계속)



`@Override`

```
public boolean equals(Object another) {
    if (this == another) return true;
    if (another == null) return false;
    if (!getClass().isAssignableFrom(another.getClass())) return false;
```

```
    UserPrincipal principal = (UserPrincipal) another;
```

```
    if (!Objects.equals(userId, principal.userId)) {
        return false;
    }
```

```
    return true;
```

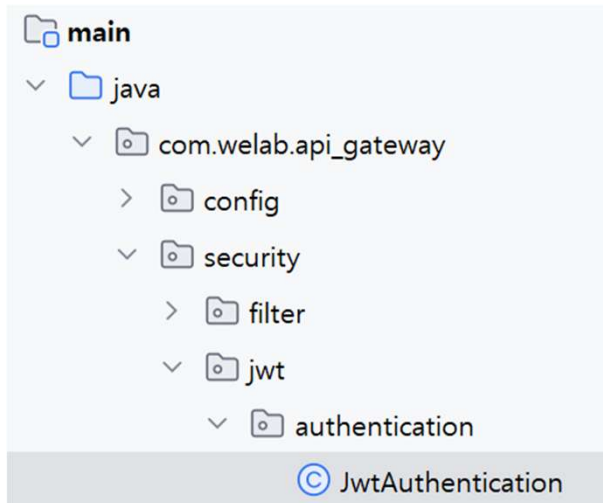
```
}
```

`@Override`

```
public int hashCode() {
    int result = userId != null ? userId.hashCode() : 0;
    return result;
}
```

```
}
```

Gateway 프로젝트 코드 작성 - JwtAuthentication



생성자

```
@Getter
public class JwtAuthentication extends AbstractAuthenticationToken {
    private final String token;
    private final UserPrincipal principal;
```

```
    public JwtAuthentication(UserPrincipal principal, String token,
                             Collection<? extends GrantedAuthority> authorities) {
        super(authorities);
        this.principal = principal;
        this.token = token;
        this.setDetails(principal);
        setAuthenticated(true);
    }
```

```
@Override
public boolean isAuthenticated() {
    return true;
}
```

클래스가 생성되면
인증이 성공된 것이기 때문에
항상 true를 반환

```
@Override
public String getCredentials() {
    return token;
}
```

```
@Override
public UserPrincipal getPrincipal() {
    return principal;
}
}
```

Gateway 프로젝트 코드 작성 - JwtTokenValidator

```
main
├── java
│   └── com.welab.api_gateway
│       ├── config
│       ├── security
│       │   ├── filter
│       │   └── jwt
│       │       ├── authentication
│       │       │   ├── JwtAuthentication
│       │       │   └── UserPrincipal
│       │       └── props
│       │           └── JwtConfigProperties
│       └── JwtTokenValidator
```

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

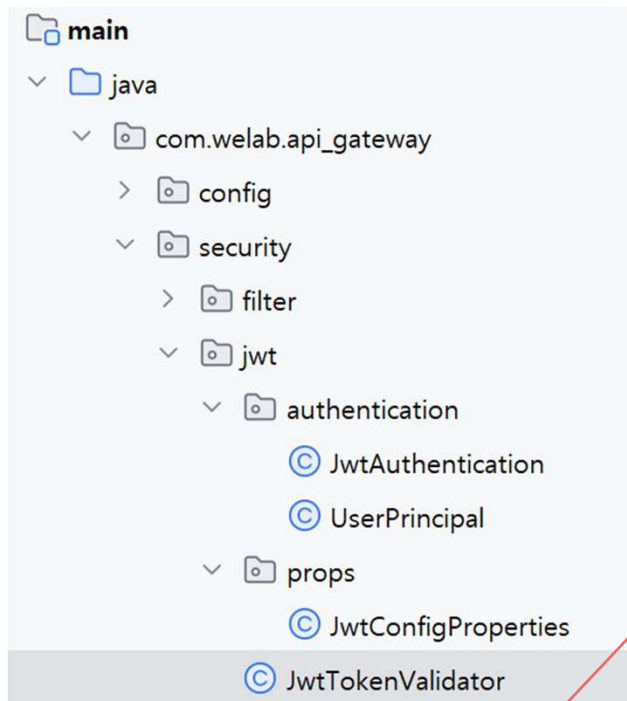
    private volatile SecretKey secretKey;
```

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

    return secretKey;
}
```

User 서비스와 동일

Gateway 프로젝트 코드 작성 – JwtTokenValidator (계속)



User

```
public JwtAuthentication validateToken(String token) {

    String userId = null;

    final Claims claims = this.verifyAndGetClaims(token);
    if (claims == null) {
        return null;
    }

    Date expirationDate = claims.getExpiration();
    if (expirationDate == null || expirationDate.before(new Date())) {
        return null;
    }

    userId = claims.get("userId", String.class);

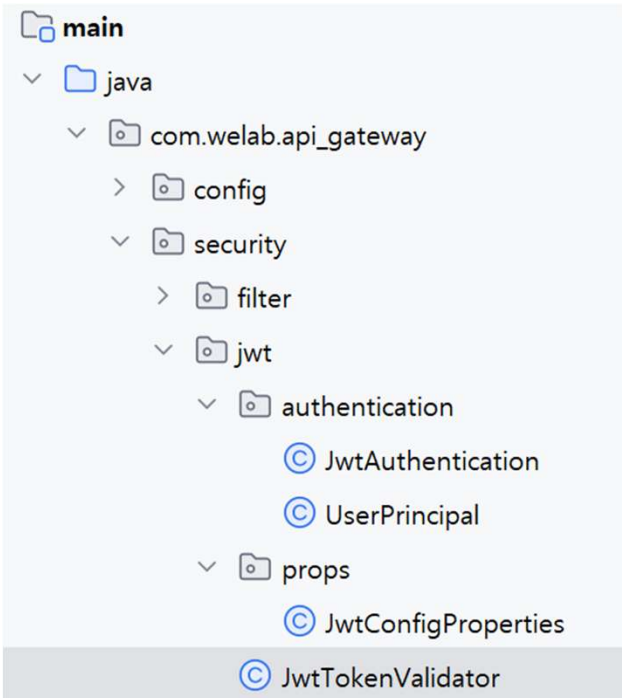
    String tokenType = claims.get("tokenType", String.class);
    if (!"access".equals(tokenType)) {
        return null;
    }

    UserPrincipal principal = new UserPrincipal(userId);

    return new JwtAuthentication(principal, token, getGrantedAuthorities("user"));
}
```

JwtAuthentication

Gateway 프로젝트 코드 작성 – JwtTokenValidator (계속)



```
private Claims verifyAndGetClaims(String token) {
    Claims claims;
    try {
        claims = Jwts.parser()
            .verifyWith(getSecretKey())
            .build()
            .parseSignedClaims(token)
            .getPayload();
    } catch (Exception e) {
        claims = null;
    }

    return claims;
}
```

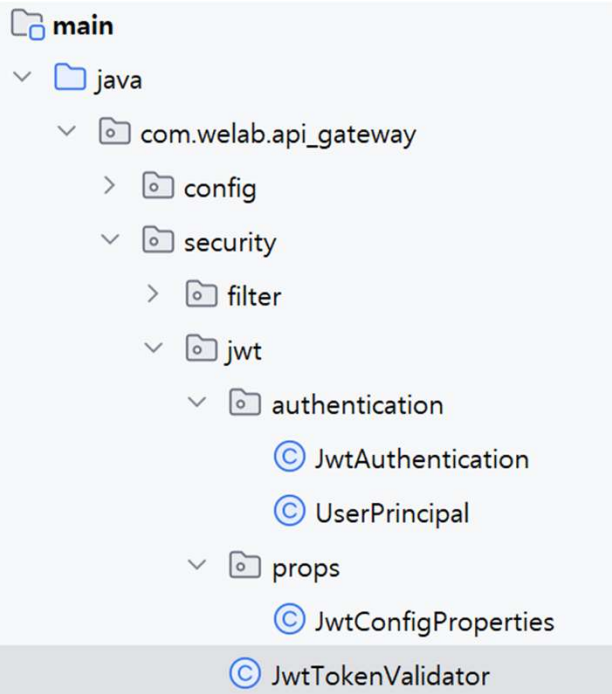
```
private List<GrantedAuthority> getGrantedAuthorities(String role) {
    ArrayList<GrantedAuthority> grantedAuthorities = new ArrayList<>();
    if (role != null) {
        grantedAuthorities.add(new SimpleGrantedAuthority(role));
    }

    return grantedAuthorities;
}
```

Spring Security

(=)

Gateway 프로젝트 코드 작성 – JwtTokenValidator (계속)



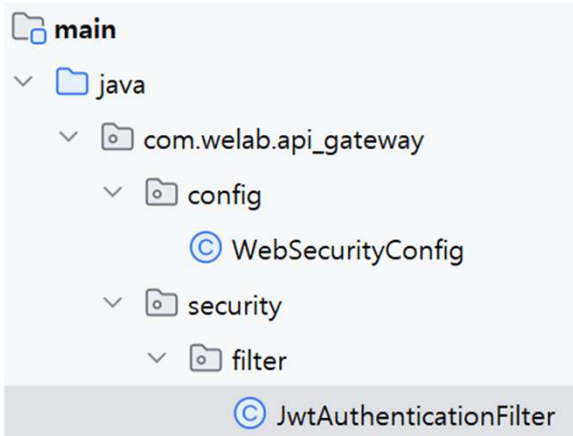
```
public String getToken(HttpServletRequest request) {
    String authHeader = getAuthHeaderFromHeader(request);
    if (authHeader != null && authHeader.startsWith("Bearer ")) {
        return authHeader.substring(7);
    }
    return null;
}
```

헤더에서
"Bearer"로
시작하는 값
찾아서 토큰 추출

```
private String getAuthHeaderFromHeader(HttpServletRequest request) {
    return request.getHeader(configProperties.getHeader());
}
```

헤더 추출

Gateway 프로젝트 코드 작성 – JwtAuthenticationFilter



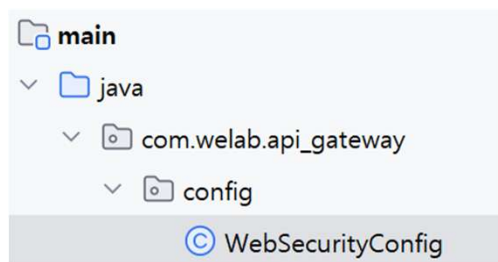
```
@Slf4j
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final JwtTokenValidator jwtTokenValidator;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {
        String jwtToken = jwtTokenValidator.getToken(request);

        if (jwtToken != null) {                                     (토큰에서 authentication추출)
            JwtAuthentication authentication = jwtTokenValidator.validateToken(jwtToken);
            if (authentication != null) {
                SecurityContextHolder.getContext().setAuthentication(authentication);    (인증 상태 등록)
            }
        }

        filterChain.doFilter(request, response);
    }
}
```


Gateway 프로젝트 코드 작성 – WebSecurityConfig



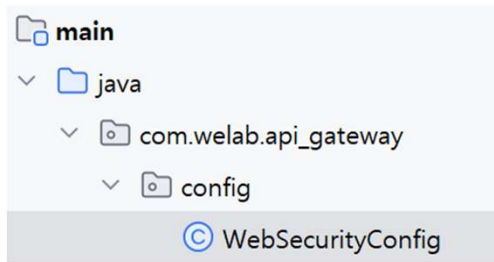
```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfig {
    private final JwtTokenValidator jwtTokenValidator;
```

```
@Bean
public SecurityFilterChain applicationSecurity(HttpSecurity http) throws Exception {
    http
        .cors(httpSecurityCorsConfigurer -> {
            httpSecurityCorsConfigurer.configurationSource(corsConfigurationSource());
        })
        .csrf(AbstractHttpConfigurer::disable)
        .securityMatcher("/**") // map current config to given resource path
        .sessionManagement(sessionManagementConfigurer
            -> sessionManagementConfigurer.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .formLogin(AbstractHttpConfigurer::disable)
        .httpBasic(AbstractHttpConfigurer::disable)
        .addFilterBefore(
            new JwtAuthenticationFilter(jwtTokenValidator),
            UsernamePasswordAuthenticationFilter.class)
        .authorizeHttpRequests(registry -> registry
            .requestMatchers("/api/user/v1/auth/**").permitAll() (로그인, 회원가입, 리프레쉬는 인증 절차 X)
            .anyRequest().authenticated()
        );

    return http.build();
}
```

- + CORS 설정
- + CSRF 설정
- + 보안 필터 적용 범위 지정
- + 세션 관리 정책 설정
- + 폼 로그인 비활성화
- + HTTP Basic 인증 비활성화
- + 커스텀 필터 등록
- + 요청별 인가(Authorization) 규칙 설정

Gateway 프로젝트 코드 작성 – WebSecurityConfig (계속)



```
@Bean
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration config = new CorsConfiguration();

    config.setAllowCredentials(true);
    // config.setAllowedOrigins(List.of("*"));
    config.setAllowedOriginPatterns(List.of("*"));
    config.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "PATCH", "OPTIONS"));
    config.setAllowedHeaders(List.of("*"));
    config.setExposedHeaders(List.of("*"));

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", config);

    return source;
}
```

CORS 설정
+ 메소드, 헤더 등 허용 설정