✅ Spring Security와 JWT 간단 정리

1

인증 Authentication 사용자의 신원을 입증하는 과정으로,

로그인을 할 때 확인 과정이 인증에 해당됩니다.

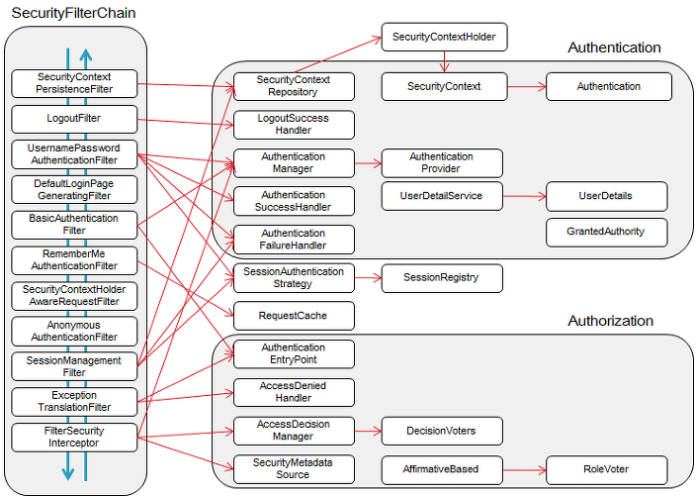
인가 Authorization 사이트의 특정 부분에 접근할 수 있는지 권한을 확인하는 작업으로,

관리자 페이지에 접속이 가능한 사용자인지 확인하는 과정이 인가에 해당됩니다.

2

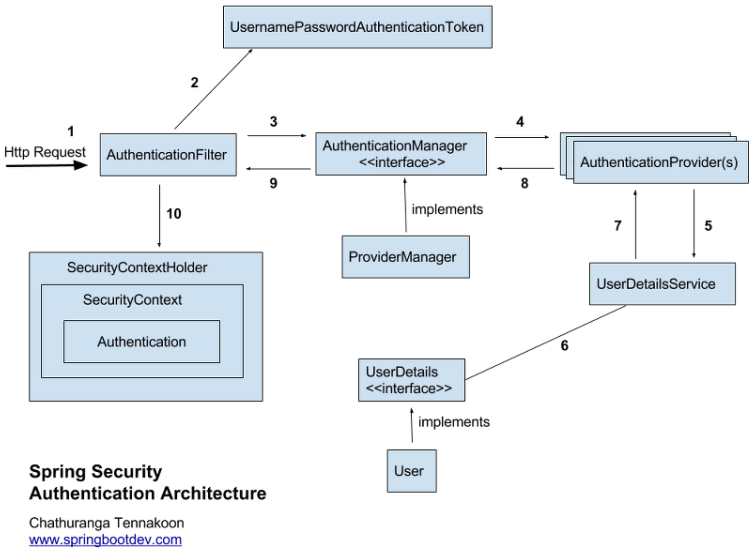
Spring Security 스프링 기반의 애플리케이션의 보안(인증과 권한, 인가 등)을 담당하는 스프링 하위 프레임워크입니다. Spring Security는 필터를 기반으로 동작하며 각 필터에서 인증, 인가 관련 작업을 처리합니다.

▼ Spring Security FilterChain



이렇게 필터 종류는 정말.. 많은데 필요할 때 찾아보면서 custom해서 쓰면 된다고 합니다!

▼ Spring Security의 Authentication 처리 과정

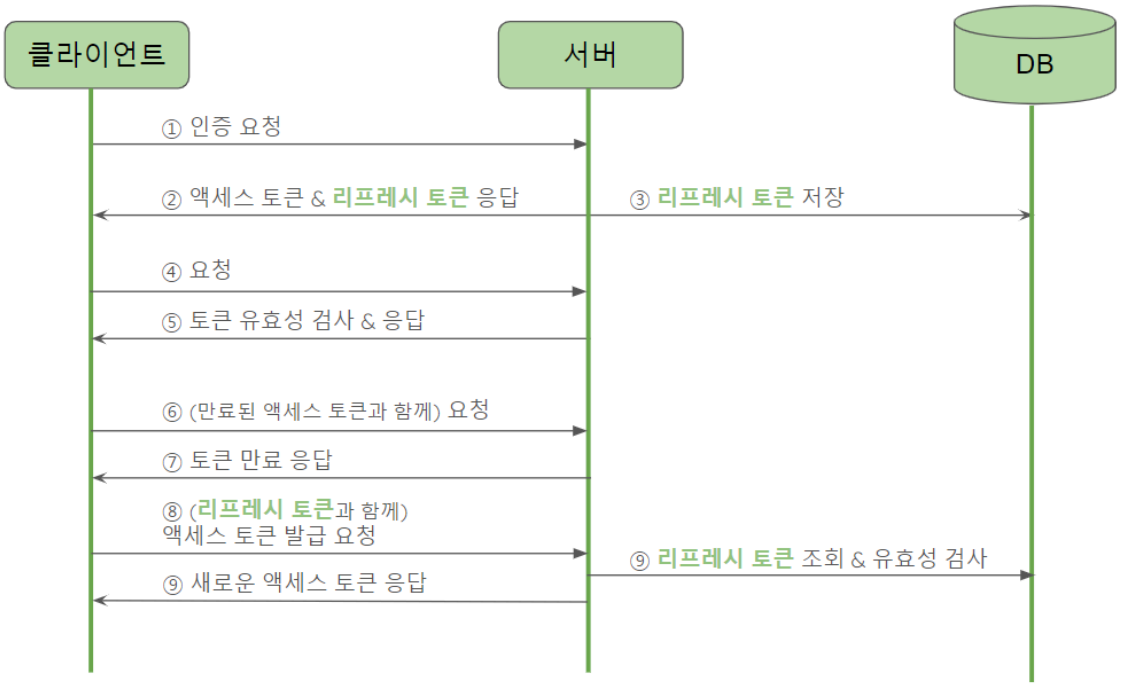


1. 사용자가 id,pw 입력 → `HTTPServletRequest`에 정보 전달
2. `AuthenticationFilter`가 id,pw 유효성 검사 진행 후,
   id,pw,인증 여부를 가지는 객체(UsernamePasswordAuthenticationToken) 생성
3. 객체를 `AuthenticationManager`에게 보내어 인증 요청 전달
   여기서
   `ProviderManager`가 인증 처리를 수행하는데, ProviderManager는 여러 개의 AuthenticationProvider를 관리하고, 각 AuthenticationProvider들에게 인증을 위임함
4. `AuthenticationProvider`는 id,pw를 확인하는 실제 인증 로직 가지고 인증을 수행
5. UserDetailsService에 id 전달
6. `UserDetailsService`(인터페이스)는 id로 DB 조회
7. 조회를 통해 찾은 사용자 정보를 객체(UserDetails)로 만들어 AuthenticationProvider에게 전달

   `AuthenticationProvider`가 UserDetails의 정보를 비교해 실제 인증 처리
8. ~10. `Authentication` 객체 생성해 `SecurityContextHolder`에 저장
   SecurityContextHolder는 현재 사용자의 인증 정보를 유지하면서 해당 정보를 통해 사용자의 접근 권한 확인
   a. 인증 성공하면 AuthenticationSuccessHandler
   b. 인증 실패하면 AuthenticationFailureHanlder

3

`JWT` `(Json Web Token)` '헤더.내용(payload).서명(signature)' 으로 구성되며 우리가 흔히 사용하는 웹 토큰입니다.



액세스/리프레시 토큰을 통해서는 위의 그림과 같이 (2)에서 발급 받은 토큰으로 (4)에서 클라이언트가 서버에게 API 요청을 할 때 토큰과 같이 요청을 하는 형식입니다. 이때 HTTP 요청 헤더의 Authorization 키 값에 `Bearear JWT값` 형태를 넣어 보내야 발급받은 JWT로 인증 과정이 이루어집니다.

위의 과정들이 이어지면서 토큰을 통해 사용자 확인이 서버에서 이루어지기 때문에 토큰이 만료될 때까지 해당 토큰으로 인증 처리가 가능하므로 **"액세스 토큰의 만료 시간을 짧게 하고, 리프레시 토큰을 통해 액세스 토큰을 재발급 받는 형식"**으로 구현이 많이 이루어집니다.

> 💡 지난번 프로젝트에서 그림과 같은 흐름으로 구현을 한 경험이 있는데, 탈취를 막기 위한 데이터 처리나 더 자세한 코드 구현과 관련해서 공부를 다시 해봐야 할 것 같다는 생각이 듭니다. Security는 찾아볼수록 공부할 내용이 계속 나와서 어려운 것 같지만.. 이번 프로젝트에 다시 적용해보면서 튼튼하게 공부해보고자 합니다!!