JWT

Header

{

“alg” :”HS512”,

“kid” : “key”

}

UTF-8인코딩 후 base64인코딩

Payload:실질적으로 인증에 필요한 데이터를 저장

{

“sub” : “user”,

“iat”: 1231231231”,//발행시간

“exp”: 123123313//만료시간

}

Signature //토근에 대한 진위여부 판단을 해주는 부분

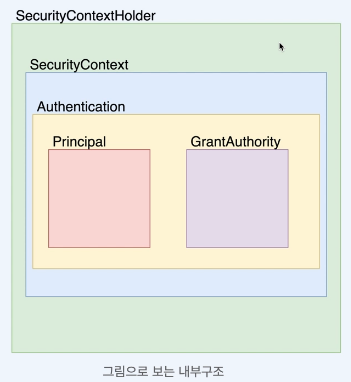
Header와 Payload합친뒤 비밀키로 hash생성하여 암호화

Mock user로 인증받아 시큐리티 테스트 가능

이때 user는 security user를 import 해야함

Refresh 토큰

테이블 , id,user\_id,token,expiryDate



SecurityContextHolder에서 getContext(static메소드)로 SecurityContext가져올 수 있음

SeucrityContext:접근 주체와 인증에대한 정보 담고있음(Authentication담고있ㅇㅁ)

Authentication

ThreadLocal(기본 자바라이브러리)

서버가 요청을 받으면 thread1개 생성되는데 그안에 securitycontext가 생김

->동시에 여러 요청 와도 독립적으로 관리 가능

Password encoder

->비밀번호를 안전하게 관리

1.회원가입한 password암호화해서 저장되어야함

2.로그인할 때 받은 비밀번호와 회원가입할때의 비밀번호를 비교할 수 있어야함

해시함수 사용

->암호화는 쉽지만 복호화가 거의 불가능한 방식 알고리즘

1.각각 같은 해시함수로 암호화해서 같은 암호일 때 패스워드 통과

springboot에서 interface로 구현되어있음

DelegatingPasswordEncoder로 5개중 하나 선택해서 사용할 수 있음

Bcryptpasswordencoder 주로 사용

Security Filter

요청 전, 응답 후 어떤작업을하도록 하는 것

Request->filter->controller

SecurityContextPersistenceFilter 보통 두번쨰로 실행되는 필터

정보주체와 인증에 대한 정보 담고있음

세션에 저장된 정보 가져오거나 없으면 만들어줌

BasicAuthentication Filter

안쓸꺼며 명시적으로 disable해줌

필터 적용시 따로 로그인 과정하지않아도 일회성으로 페이지 불러 올 수 있음

->세션 필요없이 요청떄마다 인증 이루어짐(stateless)

->매번 반복 노출되기 떄문에 보안에 취약해 https사용해야함

UsernamePasswordAuthentication Filter

로그인 버튼 누를시 해당필터 실행

Csrf Filter

csrfAttack방어하는 filter

csrf:위조된 페이지로 사기치는 것

csrf토큰(정상페이지가 소유)을 두고 악의적 공격을 방어함

RememberMeAuthenticationFilter

로그인 사실을 기억할 수 있도록 해주는 필터(웹사이트에 있는 로그인 유지하기)

AnonymousAuthentication Filter

인증안된 유저 요청하면 익명유저로 만들어 인증에 넣어ㅜ는 필터

인증안되도 null이아닌 기본 인증 넣어주는 개념

FilterSecurityInterceptor

넘어온 인증내용 기반으로 최종 인가 판단내림-> 보통 뒤쪽에 위치

Exception TranslationFilter

AuthenticationException :인증에 실패

AccessDeniedException:인가에 실패 FilterSecurityInterceptor에서 판단 후 여기서 예외처리