Spring Security를 활용한 보안인증 응용

팀명 : 코드러너
팀장 : 박재용
팀원 : 김민혜, 노윤건
기간 : 2024.02.05. ~ 2024.02.07

목 차

Ⅰ. 배경 및 개요

1. 배경

- 프로젝트를 진행하며 제작한 시스템의 회원관리 기능과 관련해 수업시간에 배운 Spring Security를 도입하게 되었다. Spring Security**(ver-6.2.1)** 활용을 통해 인증 및 인가와 로그인/로그아웃에 관련된 전후처리를 간편하게 구현할 수 있었다. 그리고 이 외에 어떤 기능들이 있는지 추가적으로 살펴보기 위해 Spring Security의 전체적인 개념과 핵심 기능에 대해 조사한 내용을 아래에 소개하고자 한다.

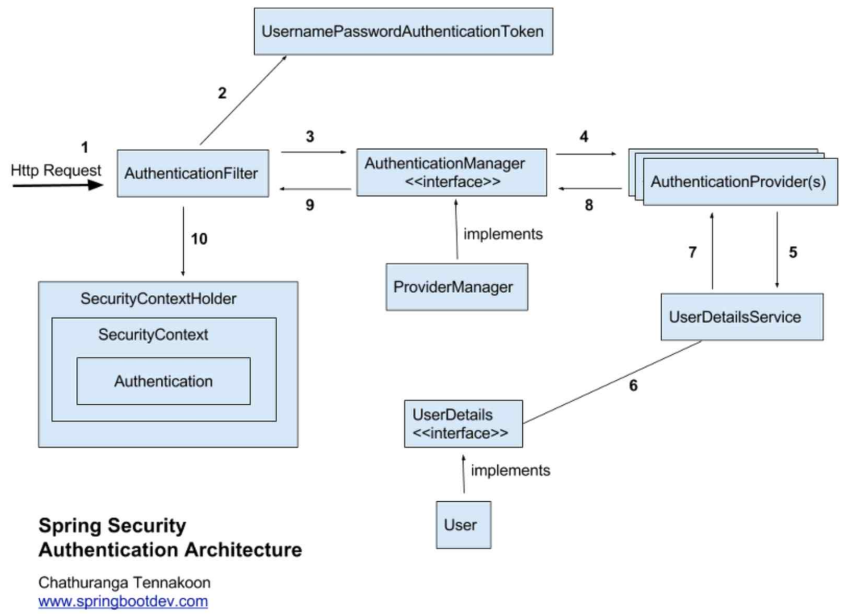
2. 개요

- Spring Security는 Spring 기반의 애플리케이션의 보안(인증과 권한, 인가 등)을 담당하는 스프링 하위 프레임워크이다. Spring Security는 '인증'과 '권한'에 대한 부분을 Filter 흐름에 따라 처리하고 있다. Filter는 Dispatcher Servlet으로 가기 전에 적용되므로 가장 먼저 URL 요청을 받는다. Spring Security는 보안과 관련해서 체계적으로 많은 옵션을 제공해주기 때문에 개발자 입장에서는 일일히 보안관련 로직을 작성하지 않아도 된다는 장점이 있다.

Ⅱ. Spring Security란

1. 동작원리

2-1. 인증관련 architecture



1) 사용자가 애플리케이션에 로그인정보와 함께 응답을 요청한다.
2) AuthenticationFilter가 요청을 가로채고 빈 UsernamePasswordAuthenticationToken객체를 생성하고, AuthenticationManager에게 전달한다.
3) AuthenticationManager가 AuthenticationProvider들에게 인증을 요구한다.
4) AuthenticationProvider들이 UserDetailsService에게 사용자 정보를 넘긴다.

5) UserDetailsService는 구현체를 통해 loadUserByUsername() 메서드를 호출하여 DB에 있는 사용자의 정보를 가져와 AuthenticationProvider들에 넘긴다.
6) AuthenticationProvider들은 사용자정보를 비교하고 일치하는 사용자 정보를 넘겨받은 AuthenticationProvider는 권한 등의 사용자 정보를 담은 Authentication객체를 반환한다.
7) AuthenticationManager는 전달받은 Authentication객체를 AuthenticationFilter로 넘긴다.
8) AuthenticationFilter는 전달받은 Authentication객체를 LoginSuccessHandler에게 전달하고 SecurityContextHolder에 저장한다.

2. 주요 기능

1) 인증 : 사용자가 시스템에 접근하면 제공된 정보를 검증하고 인증 여부를 결정한다.
2) 인가 : 인증된 사용자의 권한을 설정한다.
3) 세션 관리 : 사용자의 세션을 관리하고 로그인/로그아웃 등의 세션 관련 작업을 진행한다.
4) CSRF방어 : 보안을 강화하기 위한 CSRF 공격 방어 기능을 제공한다.
5) 보안헤더 추가 : 보안 관련 HTTP 헤더를 자동으로 추가해 애플리케이션의 보안을 강화한다.

Ⅲ. Spring Security 활용

- Spring Security 5.7 버전까지 지원되던 WebSecurityConfigurerAdapter가 삭제됨에 따라 Spring Security 6의 SecurityFilterChain interface를 활용하여 프로젝트의 보안설정을 아래와 같이 구현하였다.

```java
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final DdidaUserDetailsService ddidaUserDetailsService;
    private final LoginSuccessHandler loginSuccessHandler;

    @Bean
    protected SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.csrf((csrf) -> csrf.disable())
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers("/", "/join/**", "/newAdmin", "/login/**", "/logout
                        .requestMatchers("/static/**", "/css/**", "/js/**", "/img/**", "/fon
                        .requestMatchers("/admin/login", "/access/denied").permitAll() // 별
                        .requestMatchers("/sports/{rsrcNo}").permitAll()
                        .requestMatchers("/admin/**").hasRole("ADMIN")
                        .anyRequest().authenticated())

                .exceptionHandling(error -> error
                        .accessDeniedPage("/access/denied"))

                .formLogin(login -> login
                        .loginPage("/login")
                        .loginProcessingUrl("/login")
                        .failureUrl("/login?error=true")
                        .usernameParameter("username")
                        .passwordParameter("password")
                        .successHandler(loginSuccessHandler))

                .logout(logout -> logout
                        .logoutUrl("/logout")
                        .logoutSuccessUrl("/login")
                        .invalidateHttpSession(true)
                        .deleteCookies("JSESSIONID")); // 로그아웃 후  쿠키삭제

                return http.build();
    }
```

별도의 상속 없이 각각의 기능을 추가하며 커스터마이징이 가능했고
- 인증요청에 대한 인증 및 인가 처리
- 예외 핸들링
- 로그인 폼 관련 처리
- 로그아웃 처리

각 처리에 대한 람다와 메서드 체이닝으로 코드의 의미가 명료하고 직관적으로 표현되었다.

Ⅳ. 문제점 및 개선방안

1. 문제점

1) 본 시스템을 자주 이용하는 이용자일 경우 자동로그인 기능이 없으면 매번 로그인하는 게 번거로울 수 있다.

2) 하나의 사용자 계정을 여러 디바이스나 브라우저에서 로그인하여 이용하는 것은 사용자의 계정이 다른 사람에게 악용되거나 사용자의 혼란을 야기하는 등 문제를 발생시킬 수 있다.

(개선방향) 기존 웹사이트들의 인증관련 기능으로 자주 사용하던 '자동 로그인 / 중복 로그인 방지 / 강제 로그아웃' 등의 기능을 본 어플리케이션에 도입하여 이용자 편의성을 향상시키기 위해, Spring Security6에 해당 문제점을 개선해줄 수 있는 기능이 포함되었는지 조사해보기로 했다.

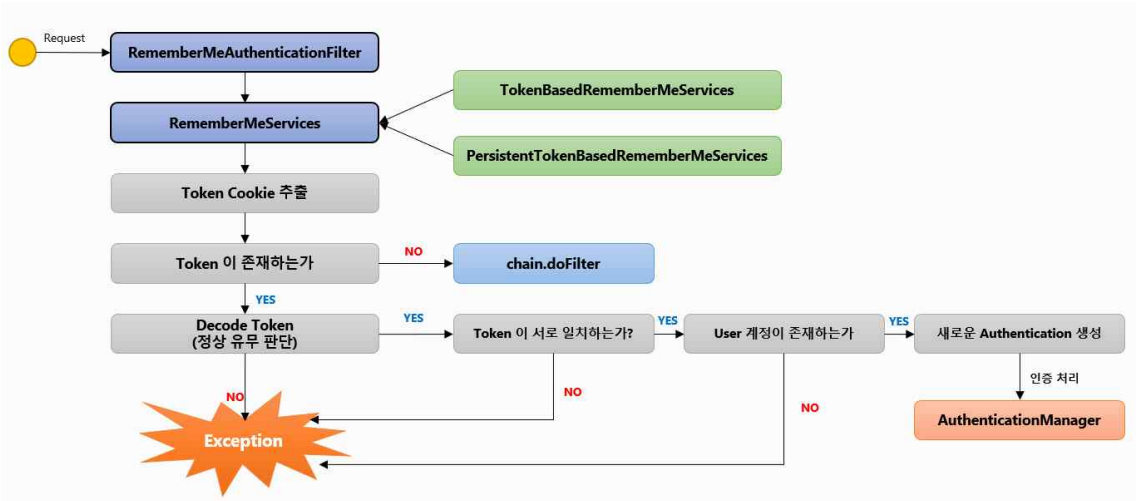
2. 개선방안

1) Remember Me 기능

A. 개요

- 사용자 세션이 만료되고 웹 브라우저가 종료된 후에도 애플리케이션이 사용자의 정보를 기억하는 기능이다.
- 사용자가 로그인을 할 때 Remember me기능을 활성화 시킬 경우 서버에서는 Remember Me 쿠키를 생성하게 된다. 그 후 로그인을 할 때는 애플리케이션에 저장되어 있는 Remember Me쿠키를 갖고 http header에 쿠키를 담아서 request를 보내게 되며 server는 http header를 확인한 후 토큰 기반의 인증을 통해 유효성 검사를 하고 로그인을 승인한다.

B. 작동원리



a) RememberMeAuthenticationFilter 실행.

- 인증객체(Authentication)가 Security Context에 없어야 필터가 실행된다.
- 또는 사용자 request header에 remember-me 쿠키 토큰이 존재해야 한다.
- 인증객체(Authentication)가 있다는 것은 로그인이 정상적으로 되었고, 회원 정보도 정상적으로 세션에서 찾을 수 있다라는 이야기이다. 따라서 이 필터가 실행될 필요가 없다.

b) TokenBasedRememberMeServices

- 서버 메모리에 있는 쿠키와 사용자가 보내온 remember-me 쿠키를 비교(기본적으로 14일간 존재)

c) PersistentTokenBasedRememberMeServices

- DB에 저장되어 있는 쿠키와 사용자가 보내온 remember-me 쿠키를 비교

d) 보안성, 이용자 편의성을 이유로 보통 PersistentTokenBasedRememberMeServices이 비교적 더 많이 사용되며, 둘 중 선택된 객체는 토큰에 해당하는 사용자 정보를 확인하고 결과를 Authentication객체에 저장한다. AuthenticationManager를 호출한다.

e) 호출된 AuthenticationManager는 Authentication 객체를 인수로 받아 인증절차를 수행한다.

※ 위 과정 중 RememberMeConfigurer는 Remember Me기능을 구성하는데 사용된다.

C. 반영

```
.rememberMe(remember -> remember // 자동 로그인
        .key("ddida")
        .tokenValiditySeconds(1000)
        .userDetailsService(ddidaUserDetailsService)
        .rememberMeParameter("remember-me"))
```

- **rememberMe**

  HttpSecurity에 있는 메서드이며 remember me인증을 구성할 수 있고 RememberMe Configurer를 리턴한다.

- **key**

  remember me 인증을 위해 생성된 토큰을 식별할 키를 설정한다. 기본값은 랜덤 생성되는 키이다.

- **tokenValiditySeconds**

  토큰이 얼마동안 유효할지 결정한다.

- **userDetailService**

  remember me 토큰이 유효할 때 UserDetails를 조회하는 데 사용되는 UserDetailsService를 커스텀 할 수 있다.

- **alwaysRemember**

  remember me parameter를 설정하지 않더라도(자동로그인 off) 한번 로그인 하고나면 애플리케이션이 쿠키를 생성해 세션이 만료되어도 로그인 상태를 유지하도록 한다. 본 프로젝트에서는 사용하지 않았다.

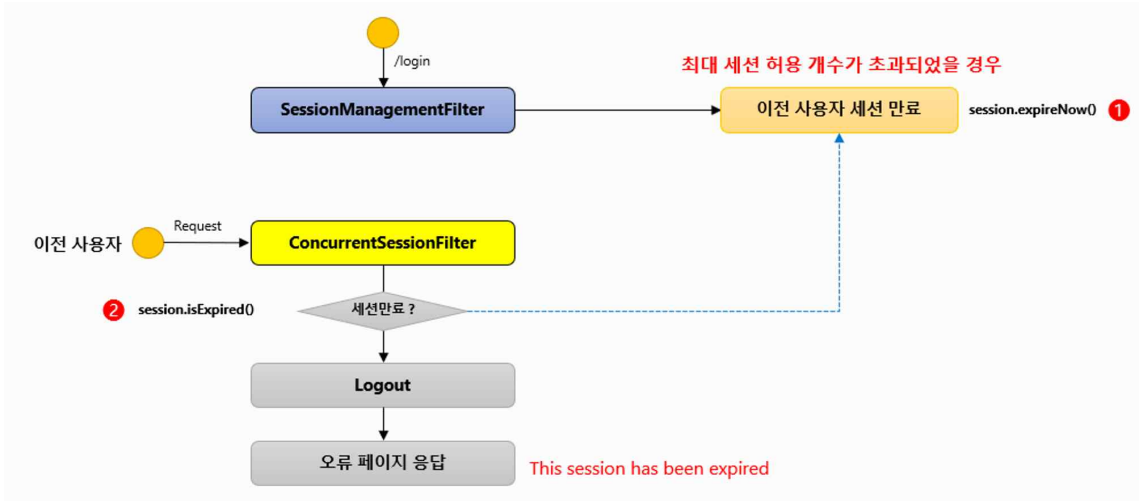
2) Session Management 기능

A. 개요

- 한 개의 계정에 대해 생성되는 세션의 개수를 설정하여 제한을 두고 세션이 최대 개수가 되었을 때 처리하는 방법을 설정하는 등 세션 관리에 대한 여러 설정을 위해 사용하는 기능이다.

Authentication Persistence and Session Management – spring 공식 문서

B. 작동원리



a) Login 시도

- 생성된 계정이 서버로부터 인증을 받게 되면 서버에 하나의 세션이 생성되며, 다른 브라우저나 디바이스에서 같은 계정으로 로그인을 할 경우 이전 서버에 생성된 세션을 공유하지 않고 새로운 세션을 생성하게 된다.

b) SessionManagementFilter

- 최대 세션 허용 개수를 확인하여 개수가 초과되었을 경우 정해진 로직을 수행한다. (현재 사용자 인증 실패/이전 사용자 세션 만료)

c) ConcurrentSessionFilter

- 사용자의 요청에 대하여 세션 만료 여부를 체크하여 만료인 경우 즉시 처리를 수행한다.

C. 반영

```
.sessionManagement(session -> session
        .maximumSessions(1) // 동일 계정 최대 세션갯수제한
        .maxSessionsPreventsLogin(false)
        .expiredUrl("/")); // 세션만료 이동url
```

- **sessionManagement**

  Spring Security에서 세션 관리를 구성하는 데 사용되는 메서드이다.

- **maximumSessions()**

  세션의 동시 허용 가능 최대수를 설정한다.

- **maxSessionsPreventsLogin()**

  설정한 최대 허용 세션 수가 되었을 때 추가적인 인증 요청(세션 생성)이 있을 경우 처리하는 방식을 정한다.

  true : 현재 사용자 인증 실패

  false : 기존 사용자 세션 만료

  해당 api를 이용하여 동시 로그인을 차단 하는 기능을 구현한다.

- **expiredUrl()**

  세션이 만료된 경우 이동 할 페이지를 설정한다.

사이트

"스프링 공식사이트",
https://docs.spring.io/spring-security/reference/servlet/authentication/architecture.html
https://docs.spring.io/spring-security/reference/servlet/authentication/session-management.html
https://docs.spring.io/spring-security/reference/servlet/authentication/rememberme.html

블로그

글쓴이 : 슬기로운 개발생활
제목 :『Spring Security의 구조(Architecture) 및 처리 과정 알아보기』
인용날짜 : 2024-02-05
https://dev-coco.tistory.com/174

글쓴이 : woonie.log
제목 :『Spring Security 동작 원리』
인용날짜 : 2024-02-05
https://velog.io/@kyungwoon/Spring-Security-%EB%8F%99%EC%9E%91-%EC%9B%90%EB%A6%AC

글쓴이 : Progrow
제목 :『[Spring Security] 인증 흐름 및 절차』
인용날짜 : 2024-02-05
https://somuchthings.tistory.com/197

글쓴이 : programmer life guidance 101
제목 :『Spring Security 01. Remember-Me 인증이란?』
인용날짜 : 2024-02-06
https://coder-in-war.tistory.com/entry/Spring-Security-01-Remember-Me-%EC%9D%B8%EC%A6%9D%EC%9D%B4%EB%9E%80

글쓴이 : 식빵
제목 :『[Spring Security] Remember Me 인증』
인용날짜 : 2024-02-06
https://velog.io/@dailylifecoding/spring-security-remember-me

글쓴이 : SeongWon Oh
제목 :『[Spring Security] RememberMe Filter 개념 & 사용법』
인용날짜 : 2024-02-06
https://velog.io/@seongwon97/Spring-Security-Remember-Me

글쓴이 : COLIN'S BLOG
제목 :『[[스프링] 스프링 시큐리티와 Remember Me 기능』
인용날짜 : 2024-02-06
https://colinch4.github.io/2023-12-21/09-50-42-007755-%EC%8A%A4%ED%94%84%EB%A7%81-%EC%8B%9C%ED%81%90%EB%A6%AC%ED%8B%B0%EC%99%80-remember-me-%EA%B8%B0%EB%8A%A5/

글쓴이 : SeungTaek
제목 :『스프링 시큐리티 - Remember Me 기능』
인용날짜 : 2024-02-06
https://velog.io/@gmtmoney2357/%EC%8A%A4%ED%94%84%EB%A7%81-%EC%8B%9C%ED%81%90%EB%A6%AC%ED%8B%B0-Remember-Me-%EA%B8%B0%EB%8A%A5

글쓴이 : 알리아 Alea
제목 :『[Spring Security] Remember Me / RememberMeAuthenticationFilter』
인용날짜 : 2024-02-06
https://velog.io/@bluedmoel/Spring-Security-%EC%8A%A4%ED%94%84%EB%A7%81-%EC%8B%9C%ED%81%90%EB%A6%AC%ED%8B%B0-2-Remember-Me