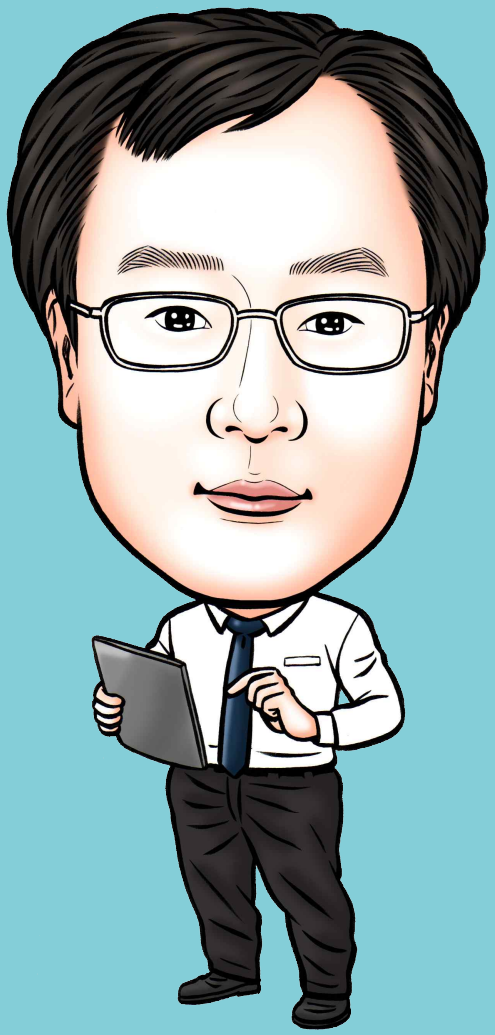
PART 1. 스프링의 스프링 부트



송 쌤

STEP 04

스프링 부트의 시큐리티

시큐리티의 소개

시큐리티의 이해

Spring Security는 Java 애플리케이션의 보안 기능을 제공하는 프레임워크로 다양한 통합 방식을 통해 애플리케이션의 보안을 강화할 수 있다.

다양한 보안 요구사항을 쉽게 구현할 수 있도록 다양한 기능과 유연한 확장성을 제공한다.

Spring Security는 애플리케이션의 인증이나 권한 부여를 관리하는 데 사용된다.

Spring Security는 Spring 애플리케이션에 보안 기능을 통합하는 표준 방법이다.

인증(authentication)과 권한 부여(authorization)를 처리한다.

- 인증(Authenticaton)
  사용자의 신원을 확인한다.
  사용자가 로그인 폼을 통해 제공한 사용자명과 비밀번호를 검증한다.
- 권한 부여(Authorization)
  인증된 사용자가 특정 리소스나 기능에 접근할 수 있는 권한을 확인한다.
- 보안 필터(Security Filter)
  HTTP 요청을 가로채어 보안 검사를 수행한다.

Spring Security 통합 방식과 인터셉터 방식은 Spring 기반 애플리케이션에서 보안을 적용하는 방법이지만, 구현 방법과 적용되는 레이어에서 차이가 있다.

인터셉터는 Spring MVC에서 HTTP 요청을 가로채어 처리하는 방법이다.

보안 외에도 요청 전후에 다양한 로직을 실행하기 위해 사용된다.

인터셉터는 주로 HandlerInterceptor 인터페이스를 구현하여 사용한다.

인터셉터는 보안을 처리할 수 있지만, Spring Security처럼 강력한 보안 기능을 제공하지는 않는다.

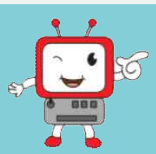
주로 간단한 인증이나 권한 검사를 위해 사용되며 세부적인 보안 정책을 적용하기 어렵다.

인터셉터 방식은 간단한 보안 요구 사항이나 특정 로직 실행을 위해 사용될 수 있다.

Spring Security는 보다 정교하고 강력한 보안 기능을 제공하며 엔터프라이즈 수준의 애플리케이션에 적합하다.

둘 중 어느 것을 사용할지는 애플리케이션의 보안 요구 사항과 복잡도에 따라 달라진다.

Spring Security 통합 방식	인터셉터 방식
인증 및 권한 부여, 세부적인 보안 정책 설정	HTTP 요청 전/후에 로직 실행(보안 외에도 사용)
상대적으로 복잡, 강력한 보안 기능 제공	비교적 간단, 제한적인 보안 기능 제공
고급 인증 및 권한 부여, 세션 관리, CSRF 보호 등	단순한 인증/권한 검증, 로그 및 감사 기능
높은 유연성, 다양한 보안 요구 사항 처리 가능	제한된 유연성, 간단한 보안 요구 사항 처리 가능
보안 기능 확장 및 커스터마이징 용이	보안 기능 확장 어려움, 제한적인 커스터마이징 가능



4.스프링 부트의 시큐리티

시큐리티의 구성

라이브러리 추가

코드 build.gradle

```
dependencies {

 //Spring Security의 핵심 기능과 함께 기본적인 보안 설정을 제공한다.
 implementation 'org.springframework.boot:spring-boot-starter-security'

 //Spring Security와 통합되어 타임리프 템플릿에서 보안 관련 정보를 쉽게 표시하고 제어한다.
 implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity6'

 //Spring Boot 기본 내장 톰캣을 제외하고 외부 톰캣에서 실행되도록 제어한다.
 providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
}
```

SecurityConfig 클래스

SecurityConfig 클래스는 Spring Security의 보안 설정을 정의한다.
애플리케이션의 보안 정책, 인증, 인가 절차를 설정한다.

SecurityFilterChain 인터페이스

Spring Security의 보안 필터 체인을 정의한다.
HTTP 요청에 대한 접근 제어를 설정합니다.
폼 로그인, 로그아웃, 예외 처리 등의 설정을 포함한다.

PasswordEncoder 인터페이스

비밀번호 인코딩 및 검증을 위한 설정이다.
다양한 인코딩 방식을 지원할 수 있다.

AuthenticationManagerBuilder 클래스

CustomUserDetailsService 클래스를 이용하여 사용자 인증 정보를 설정한다.
비밀번호 인코더를 설정하여 비밀번호를 검증한다.

CustomUserDetailsService 클래스

사용자 세부 정보를 로드하는 서비스다.
Spring Security는 이 클래스를 통해 사용자 인증 정보를 조회한다.

EmpRepository 인터페이스

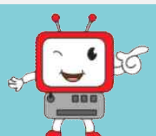
데이터베이스에서 사용자 정보를 조회하기 위한 리포지토리다.
findByEname 메서드를 통해 사용자 이름으로 사용자 정보를 검색한다.

loadUserByUsername 메서드

주어진 사용자 이름으로 사용자 정보를 로드한다.
사용자 이름이 admin일 경우에 하드코딩된 관리자 계정을 반환한다.
데이터베이스에서 사용자 정보를 조회하여 UserDetails 객체를 생성한다.

UserDetails 인터페이스

Spring Security에서 사용하는 사용자 정보 객체다.
사용자 이름, 비밀번호, 권한 정보를 포함한다.



4.스프링 부트의 시큐리티

시큐리티의 API

시큐리티의 어노테이션

애플리케이션의 보안 설정

Spring Security를 구성하고 활성화하며 Spring Security의 설정을 사용자 정의할 수 있는 방법을 제공하며 보안 구성 클래스를 정의할 수 있게 한다.
Spring Boot와 통합되어 더 쉽게 보안 구성을 할 수 있다.
WebSecurityConfigurerAdapter 클래스를 확장하여 사용자 정의 보안 규칙을 설정할 수 있다.
특정 HTTP 요청에 대한 보안 규칙을 정의하고 인증 및 권한 부여를 설정할 수 있다.
사용자 인증에 필요한 다양한 설정을 정의할 수 있으며 인메모리 인증, 데이터베이스 기반 인증 등을 설정할 수 있다.
@EnableWebSecurity 어노테이션은 클래스, 인터페이스에 부착할 수 있으며 런타임 동안 유지되고 리플랙션을 통해 런타임 시점에 접근할 수 있다.

- @EnableWebSecurity 어노테이션 → 스프링 부트 3 이상부터 생략가능

  Spring Security를 사용하여 애플리케이션의 보안 설정을 구성한다.

  - debug 엘리먼트

    Spring Security의 디버그 지원을 제어하며 기본값은 false다.

시큐리티의 자바

접근 거부

Spring Security에서 접근이 거부된 경우의 처리를 담당한다.
사용자가 인증된 상태에서 특정 리소스에 대한 접근 권한이 없을 때 발생하는 AccessDeniedException 클래스를 처리하기 위해 사용된다.
접근이 거부된 상황에서 특정 동작을 정의하는 데 사용된다.
기본 제공 구현체인 AccessDeniedHandlerImpl 클래스를 사용하거나 사용자 정의 구현체를 만들어 다양한 요구사항을 충족할 수 있다.
보안 정책을 강화하고 사용자에게 적절한 응답을 제공하여 보안 이벤트에 대한 명확한 처리를 가능하게 한다.
사용자가 인증된 상태에서 접근 권한이 없는 페이지나 리소스에 접근하려고 할 때 발생하는 AccessDeniedException 클래스를 처리한다.
접근이 거부된 상황에서 사용자에게 적절한 피드백인 에러 페이지, JSON 응답 등을 제공한다.
접근 거부 이벤트를 로깅하거나 감사 기록으로 남겨 보안 분석에 활용할 수 있다.
Spring Security에서 접근 거부 상황을 처리하기 위한 중요한 인터페이스다.
단일 메서드 handle 메서드를 통해 접근이 거부된 요청에 대해 커스텀 처리를 구현할 수 있다.
사용자에게 적절한 피드백을 제공하고 보안 이벤트를 효과적으로 관리할 수 있다.

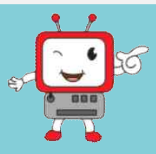
- AccessDeniedHandler 인터페이스

  접근 거부 처리를 위한 핸들러다.

  - handle(HttpServletRequest request, HttpServletResponse response, AccessDeniedException accessDeniedException) 추상 메서드

    접근이 거부된 상황에서 호출되며 사용자가 접근 권한이 없는 리소스에 접근하려고 할 때 필요한 처리를 수행한다.

    - request 파라미터 : 접근이 거부된 요청을 나타내는 객체다.
    - response 파라미터 : 응답을 나타내는 객체다.
    - accessDeniedException 파라미터 : 발생한 접근 거부 예외다.



4.스프링 부트의 시큐리티

HTTP 요청 처리

보안 필터들이 어떤 순서로 적용될지와 어떤 요청에 대해 필터가 적용될지를 결정하는 데 중요한 역할을 한다.
HTTP 요청을 처리하기 위해 적용될 보안 필터들의 순서와 구성을 정의한다.
특정 요청에 대해 필터 체인이 적용될지 여부를 결정한다.
다양한 보안 요구사항을 충족하기 위해 필터 체인을 유연하게 설정할 수 있다.
애플리케이션의 보안 필터들이 어떻게 적용될지를 정의한다.
필터 체인이 적용될 HTTP 요청 경로를 지정한다.
애플리케이션의 특정 보안 정책을 필터 체인을 통해 일관되게 적용할 수 있다.

● SecurityFilterChain 인터페이스

Spring Security에서 HTTP 요청을 처리하는 필터 체인의 구성을 정의한다.

■ matches(HttpServletRequest request) 추상 메서드

HTTP 요청이 필터 체인에 매칭되는지 여부를 결정한다.

- request 파라미터 : 검사할 HTTP 요청 객체다.

■ getFilters( ) 추상 메서드

필터 체인에 포함된 필터들의 리스트를 반환한다.
각 필터는 HTTP 요청을 처리하기 위한 특정 보안 기능을 수행한다.

HTTP의 보안 설정

다양한 보안 설정을 체인 형태로 정의할 수 있도록 지원하며 주로 WebSecurityConfigurerAdapter 클래스를 확장하여 사용한다.
HTTP 요청에 대한 인증 및 권한 부여 설정을 정의한다.
메서드 체인을 사용하여 설정을 단계별로 구성할 수 있다.
다양한 보안 요구사항을 충족하기 위한 유연한 설정이 가능하다.
HTTP 요청에 대한 보안 정책을 설정하고 적용한다.
특정 URL 패턴에 대한 접근 제어 규칙을 정의한다.
로그인 페이지, 성공 및 실패 URL 등을 설정하여 로그인 및 로그아웃 동작을 정의한다.
접근 거부 및 기타 보안 예외 상황에 대한 처리를 정의한다.

● HttpSecurity 클래스

Spring Security에서 HTTP 요청에 대한 보안 설정을 구성한다.

■ authorizeHttpRequests( ) 메서드

HTTP 요청에 대한 접근 제어 규칙을 설정하기 위한 설정 시작점을 반환한다.

■ requestMatchers(RequestMatcher... requestMatchers) 메서드

특정 요청 매처를 사용하여 요청을 필터링한다.

- requestMatchers 파라미터 : 요청을 매칭할 조건을 정의하는 RequestMatcher 객체들이다.

■ hasRole(String... role) 메서드

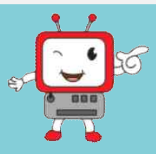
특정 역할을 가진 사용자만 접근할 수 있도록 설정한다.

- role 파라미터 : 요청을 허용할 역할 이름이다.

■ hasAnyRole(String... roles) 메서드

특정 역할 중 하나라도 가지고 있는 사용자만 접근할 수 있도록 설정한다.

- roles 파라미터 : 요청을 허용할 역할들의 배열이다.



4.스프링 부트의 시큐리티

■ anyRequest( ) 메서드

모든 요청에 대해 설정을 적용한다.

■ authenticated( ) 메서드

인증된 사용자만 접근할 수 있도록 설정한다.

■ formLogin( ) 메서드

폼 기반 로그인을 구성하기 위한 설정 시작점을 반환한다.

■ loginPage(String loginPage) 메서드

사용자 정의 로그인 페이지를 설정한다.

▪ loginPage 파라미터 : 사용자 정의 로그인 페이지의 URL이다.

■ permitAll( ) 메서드

인증 여부와 상관없이 모든 사용자에게 접근을 허용한다.

■ defaultSuccessUrl(String defaultSuccessUrl) 메서드

로그인 성공 시 리디렉션할 기본 URL을 설정한다.

▪ defaultSuccessUrl 파라미터 : 로그인 성공 후 리디렉션할 기본 URL이다.

■ failureUrl(String failureUrl) 메서드

로그인 실패 시 리디렉션할 URL을 설정한다.

▪ failureUrl 파라미터 : 로그인 실패 시 리디렉션할 URL이다.

■ logout( ) 메서드

로그아웃 설정을 위한 설정 시작점을 반환한다.

■ exceptionHandling( ) 메서드

예외 처리 설정을 위한 설정 시작점을 반환한다.

■ accessDeniedHandler(AccessDeniedHandler accessDeniedHandler) 메서드

접근 거부 상황을 처리하기 위한 핸들러를 설정한다.

▪ accessDeniedHandler 파라미터 : 접근 거부 처리 핸들러다.

■ build( ) 메서드

설정된 보안 필터 체인을 빌드하여 반환한다.

비밀번호 인코딩 검증

Spring Security에서 비밀번호를 인코딩하고 인코딩된 비밀번호와 원본 비밀번호를 비교하는 데 사용된다.
비밀번호의 보안성을 강화하기 위해 사용되며 다양한 인코딩 알고리즘을 지원한다.
비밀번호를 인코딩하여 저장한다.
인코딩된 비밀번호는 원본 비밀번호와 다르게 보이며 보안성을 높인다.
원본 비밀번호와 인코딩된 비밀번호를 비교하여 일치 여부를 검증한다.
다양한 인코딩 알고리즘을 사용할 수 있어 필요에 따라 선택할 수 있다.
데이터베이스에 비밀번호를 안전하게 저장하기 위해 인코딩된 형태로 변환한다.
로그인 시 입력한 비밀번호와 저장된 인코딩된 비밀번호를 비교하여 인증을 수행한다.

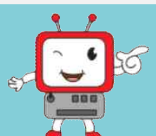
● PasswordEncoder 인터페이스

Spring Security에서 비밀번호를 인코딩하고 인코딩된 비밀번호와 원본 비밀번호를 비교한다.

■ encode(CharSequence rawPassword) 추상 메서드

원본 비밀번호를 인코딩하여 저장 가능한 형태로 변환한다.
인코딩된 비밀번호는 데이터베이스 등에 안전하게 저장할 수 있다.

▪ rawPassword 파라미터 : 인코딩할 원본 비밀번호다.



4.스프링 부트의 시큐리티

비밀번호 인코딩 관리

여러 종류의 인코딩 알고리즘을 지원하고 필요에 따라 적절한 인코더를 선택할 수 있는 DelegatingPasswordEncoder 클래스를 생성하여 인코딩된 비밀번호의 유연한 관리와 검증을 가능하게 한다.
인스턴스화되지 않으며 스테틱 메서드만을 제공하는 유틸리티 클래스다.
여러 종류의 비밀번호 인코더를 생성하고 관리할 수 있다.
DelegatingPasswordEncoder를 생성하여 다양한 인코딩 알고리즘을 동적으로 지원한다.
다양한 인코딩 알고리즘을 지원하는 비밀번호 인코더를 생성한다.

● PasswordEncoderFactories 클래스

다양한 PasswordEncoder 구현체를 생성하고 관리한다.

■ createDelegatingPasswordEncoder( ) 스테틱 메서드

DelegatingPasswordEncoder 객체를 생성하여 반환하며 생성된 DelegatingPasswordEncoder 클래스는 여러 종류의 인코딩 알고리즘을 지원하며 기본적으로 bcrypt 알고리즘을 사용한다.
반환된 인코더는 다양한 인코딩 방식이 적용된 비밀번호를 처리할 수 있도록 설정된다.

인증 관리자

다양한 인증 메커니즘을 설정하고, 이를 통해 사용자 인증을 처리할 수 있다.
여러 종류의 인증 소스를 설정할 수 있으며 인메모리, JDBC, LDAP 등 다양한 소스에서 사용자 정보를 가져와 인증을 처리할 수 있다.
메서드 체인을 통해 설정을 유연하게 구성할 수 있다.
사용자 정의 인증 프로바이더를 추가하여 고유한 인증 로직을 구현할 수 있다.
인증 관리자(AuthenticationManager)를 설정하고 구성하여, 다양한 인증 방법을 지원한다.
다양한 인증 프로바이더를 등록하여 사용자가 다양한 소스로 인증할 수 있도록 한다.

● AuthenticationManagerBuilder 클래스

Spring Security에서 인증 관리자(AuthenticationManager)를 구성한다.

사용자 인증 및 권한 부여

주어진 사용자 이름에 대한 사용자 세부 정보를 가져오는 메서드를 정의한다.
사용자 이름(username)을 기반으로 사용자 세부 정보를 로드한다.
인증 프로세스와 분리된 인터페이스로, 다양한 방식으로 구현할 수 있다.
사용자 저장소가 데이터베이스, LDAP, 인메모리 등 다양한 형태일 수 있도록 유연하게 확장할 수 있다.
로그인 시 입력된 사용자 이름을 기반으로 사용자 정보를 로드하여 인증을 처리한다.
로드된 사용자 정보에 포함된 권한 정보를 기반으로 권한 부여를 처리한다.
UserDetails 객체를 반환하여 Spring Security가 사용자 인증 및 권한 부여를 수행할 수 있도록 한다.

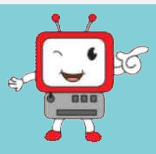
● UserDetailsService 인터페이스

Spring Security에서 사용자 인증 및 권한 부여를 처리한다.

■ loadUserByUsername(String username) 추상 메서드

사용자 이름을 입력 받아 해당 사용자의 세부 정보를 반환한다.
주어진 사용자 이름에 해당하는 사용자 세부 정보를 포함한 UserDetails 객체를 반환한다.
반환된 UserDetails 객체는 Spring Security가 인증 및 권한 부여를 수행하는 데 사용된다.

- username 파라미터 : 로드할 사용자의 사용자 이름이다.



4.스프링 부트의 시큐리티

사용자 인증 및 권한 부여 처리

인증된 사용자에 대한 핵심 정보를 제공한다.
인증된 사용자에 대한 기본 정보인 사용자 이름, 비밀번호, 권한 등을 제공한다.
사용자 정보를 담기 위한 표준 인터페이스로, 애플리케이션의 필요에 맞게 확장할 수 있다.
사용자 계정의 활성화 여부, 계정 만료 여부, 자격 증명 만료 여부, 계정 잠금 여부와 같은 보안 관련 속성을 포함한다.
인증된 사용자의 세부 정보를 저장하고 관리한다.
사용자가 가진 권한 정보를 제공하여 권한 부여 프로세스를 지원한다.
계정의 활성화 상태, 만료 여부, 자격 증명 만료 여부, 잠금 여부를 확인하여 추가적인 보안 검사를 수행한다.

- UserDetails 인터페이스

  Spring Security에서 사용자 인증 및 권한 부여를 처리한다.

사용자 이름 예외 확인

주어진 사용자 이름에 해당하는 사용자가 존재하지 않는 경우를 명확하게 나타낸다.
명시적으로 처리되지 않아도 되며 필요에 따라 예외 핸들러에서 처리할 수 있다.

- UsernameNotFoundException 클래스

  사용자 이름으로 사용자를 찾을 수 없을 때 발생한다.

사용자의 인증 및 권한 부여 정보 관리

Spring Security에서 사용자의 기본 구현체로 사용되며 사용자 이름, 비밀번호, 권한 등 사용자 정보를 저장한다.
사용자 인스턴스를 생성할 때 빌더 패턴을 사용하여 유연하고 가독성 높은 방식으로 설정할 수 있다.
사용자의 인증 정보와 권한 정보를 관리하고 Spring Security의 인증 및 권한 부여 프로세스에서 사용된다.
UserDetailsService 인터페이스와 함께 사용하여 사용자 정보를 로드하고 AuthenticationManager 클래스에서 인증을 처리하는 데 사용됩니다.

- User 클래스

  사용자의 인증 및 권한 부여 정보를 저장하고 관리한다.

  - withUsername(String username) 메서드

    지정된 사용자 이름을 가진 사용자 빌더 객체를 생성한다.
    사용자 인스턴스를 빌드하는 첫 번째 단계다.

    - username 파라미터 : 사용자 이름을 나타내는 문자열이다.

  - password(String password) 메서드

    지정된 비밀번호를 사용자 빌더에 설정한다.
    {noop} 접두사는 비밀번호가 평문임을 나타낸다.
    {noop} 접두사를 사용하여 비밀번호가 인코딩되지 않았음을 나타낼 수 있다.

    - password 파라미터 : 사용자의 비밀번호를 나타내는 문자열이다.

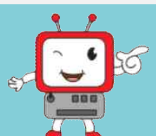
  - roles(String... roles) 메서드

    지정된 역할(권한)을 사용자 빌더에 설정한다.
    각 역할은 권한 부여 프로세스에서 사용된다.

    - roles 파라미터 : 사용자가 가지고 있는 역할인 권한들을 나타내는 문자열 배열다.

  - build( ) 메서드

    설정된 정보로 User 객체를 생성하고 반환한다.
    빌더 패턴의 최종 단계로 앞서 설정한 사용자 이름, 비밀번호, 역할 등의 정보를 기반으로 사용자 객체를 만든다.



4.스프링 부트의 시큐리티

컬렉션 조작

Java에서 컬렉션을 조작하기 위한 여러 유틸리티 메서드를 제공하는 클래스다.
인스턴스화할 수 없으며 모든 메서드가 스테틱 메서드로 제공된다.
컬렉션을 정렬, 검색, 변환, 동기화, 변경 불가능하게 만드는 등의 작업을 수행할 수 있는 메서드를 포함한다.
컬렉션 프레임워크와 함께 사용되어, 일관되고 표준화된 방식으로 컬렉션을 조작할 수 있다.
컬렉션 객체를 다양한 방식으로 조작하는 작업을 단순화한다.
싱글톤 리스트, 변경 불가능한 컬렉션 등의 특수한 형태의 컬렉션 객체를 생성할 수 있다.
컬렉션을 정렬하고 검색하는 작업을 지원하며 멀티스레드 환경에서 컬렉션을 안전하게 사용할 수 있도록 동기화된 컬렉션을 생성한다.

● Collections 클래스

다양한 컬렉션 작업을 지원하며 컬렉션을 조작한다.

■ singletonList(T o) 스테틱 메서드

주어진 요소를 포함하는 크기 1의 불변 리스트를 생성한다.
반환된 리스트는 요소를 추가, 제거, 변경할 수 없는 불변 리스트다.

- o 파라미터 : 리스트에 포함할 단일 요소다.

권한의 문자열 표현

사용자에게 부여된 권한을 문자열로 표현하며 권한을 나타내는 문자열 하나를 저장하는 단순한 구현체다.
생성 시 권한 문자열을 설정한 후 변경할 수 없는 불변 객체다.
인증된 사용자의 권한을 저장하고 관리하는 데 사용된다.
Spring Security의 권한 검증 메커니즘에서 사용자의 권한을 확인하는 데 사용된다.
역할 기반 접근 제어(RBAC)에서 사용자가 특정 역할을 가지고 있는지 확인할 때 사용된다.

● SimpleGrantedAuthority 클래스

Spring Security에서 사용자의 권한을 문자열로 표현하는 간단하고 기본적인 구현체다.

STUDY