

# SpringSecurity architecture

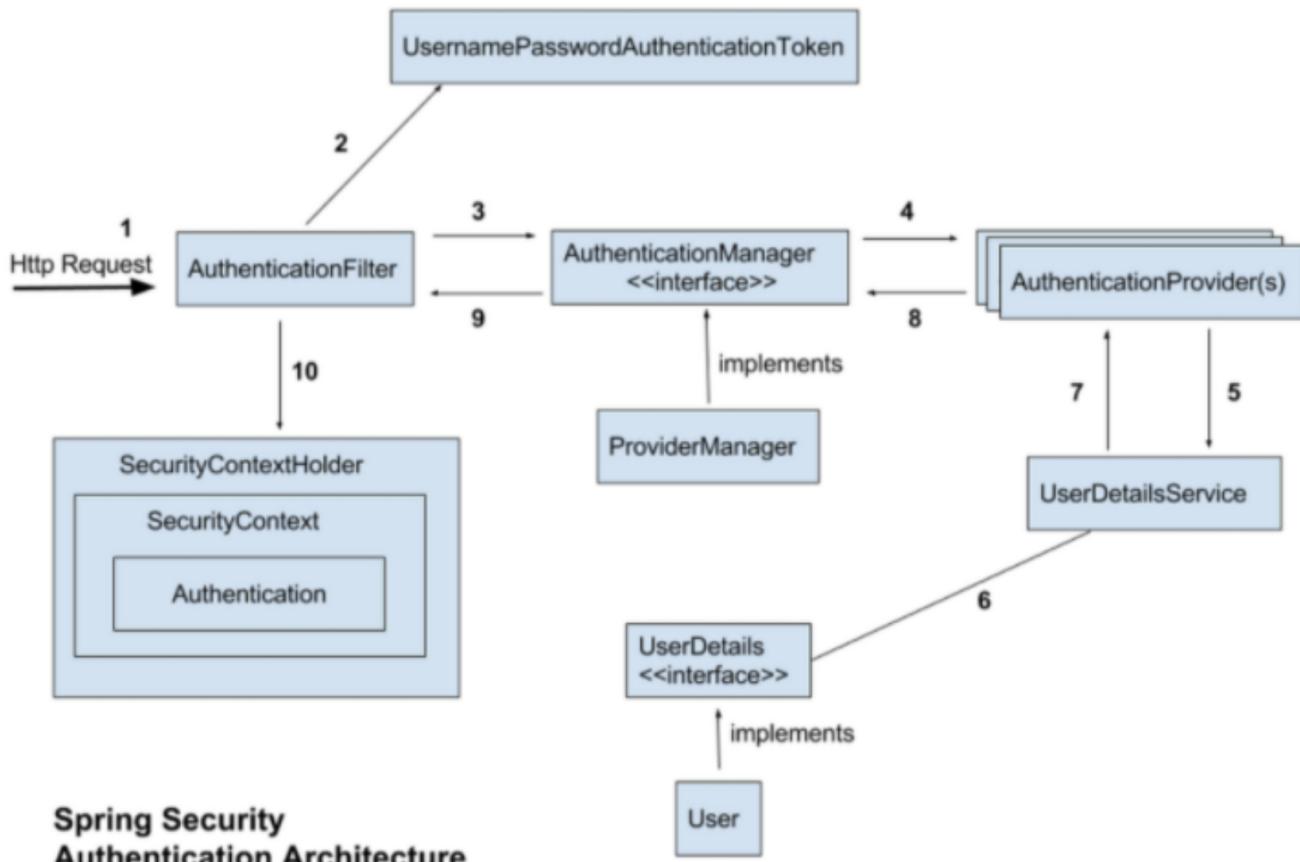
# 수업차시	2
# 페이지수	22

## 01. SpringSecurity architecture

### 01-01. Security architecture

#### 01-01-01. Security architecture

1. 사용자의 로그인 요청
2. AuthenticationFilter에서 사용자의 아이디와 비밀번호를 기반으로 UsernamePasswordAuthenticationToken을 발행한다.
3. 생성된 AuthenticationManager에 전달
4. AuthenticationManager 등록된 AuthenticationProvider에 사용자 정보조회 요청
5. UserDetailsService에서 사용자의 정보를 조회하고 UserDetails에 정보를 저장하여 반환함  
조회시 입력된 username을 기반으로 조회함
6. AuthenticationProvider에 조회된 사용자의 username, password를 비교
7. 일치하는 경우 AuthenticationManager에 객체를 반환함
8. 해당 인증 정보를 SecurityContextHolder에 저장함



## 01-01-02. Security 개념

### • Authorization & Authentication

Spring security는 인증 절차를 거친 후에 인가 절차를 진행하게 되며 인가 과정에서 해당 리소스에 대한 접근 권한이 있는지를 확인하게 된다.

인가를 위해서 Principal을 아이디로, Credential을 비밀번호로 사용하는 Credential 기반의 인증 방식을 사용하게 된다.

- Authorization (인증) : 해당 사용자가 본인이 맞는지를 확인하는 절차
- Authentication (인가) : 인증된 사용자가 요청한 자원에 접근 가능한지를 결정하는 절차
- 접근 주체 (Principal) : 보호받는 Resource에 접근하는 대상
- 비밀번호 (Credential) : Resource에 접근하는 대상의 비밀번호

### • UsernamePasswordAuthenticationToken

해당 클래스는 Spring Security에서 제공하는 Authentication 인터페이스의 특정 구현이다.

사용자 이름과 암호를 자격 증명으로 포함하는 인증 요청을 나타낸다.

클래스는 사용자의 이름 및 암호 기반 인증을 수행할 때 사용자의 자격 증명을 캡슐화하는데 사용되며 일반적으로 인증 프로세스 중에 생성되며 인증을 위해 인증 관리자에게 전달된다.

해당 토큰은 인증을 위해 관리자에게 전달되며 일반적으로 사용자 데이터베이스 또는 기타 인증 메커니즘에 대해 자격 증명을 유효성 검사를 진행한다.

인증에 성공하면 인증된 Authentication 객체가 반환되며 이 객체는 SecurityContextHolder에서 설정하여 보안 컨텍스트 내에서 사용자의 인증을 설정할 수 있다.

이외로 다음과 같은 클래스를 활용할 수 있다.

1. RememberMeAuthenticationToken
2. JwtAuthenticationToken
3. OAuth2AuthenticationToken

### • AuthenticationManager

해당 인터페이스는 Authentication 객체 인증을 담당하는 Spring Security의 핵심 구성 요소로 사용자 자격 증명의 유효성을 검사하고 보안 컨텍스트 내에서 사용자의 id를 설정하는데 사용된다.

AuthenticationManager 인터페이스의 기본 메소드는 authenticate(Authentication authentication)이다. 이 메서드는 Authentication 객체를 입력으로 사용하고 인증 프로세스를 수행하고 인증에 성공하면 Authentication 객체를 반환하고 인증에 실패하면 예외를 throw한다.

### • ProviderManager

Spring Security의 AuthenticationManager 인터페이스 구현으로 인증 요청을 AuthenticationProvider 구현 체인에 위임하는 핵심 구성 요소이다.

ProviderManager는 복합 인증 공급자 역할을 진행하게 되며 즉, AuthenticationProvider 인스턴스 목록을 보유하고 주어진 인증 요청에 대하여 적절한 인증 객체를 찾기 위해 반복한다.

ProviderManager의 Authenticate 메서드가 호출되면 해당 목록의 각 AuthenticationProvider에 인증 요청을 순차적으로 위임 하며 각 공급자는 인증 요청에 따라 특정 인증 논리를 수행한다. 공급자가 요청을 성공적으로 인증하면 인증된 Authentication 객체를 반환하여 공급자가 요청을 인증할 수 없는 경우 null을 반환하게 된다. AuthenticationException을 발생시킨다.

- UserDetailsService

인증 프로세스 중에 사용자별 데이터를 로드하는 Spring Security의 핵심 구성 요소로 백엔드 데이터 소스에서 사용자 아이디, 암호 및 권한과 같은 사용자 세부 정보를 검색하는데 사용하게 된다.

인터페이스의 기본 메서드는 사용자 이름을 입력으로 사용하고 개체를 반환하는 UserDetailsService이다. 개체는 사용자의 세부 정보를 나타내며 인증 및 권한 부여 목적으로 사용된다. loadUserByUsername(String username)을 상속받아서 사용한다.

해당 예제에서 CustomUserDetailsService 클래스 인터페이스를 구현한다.

userDetailsService 메서드 loadUserByUsername 일반적으로 데이터베이스 LDAP 서버 또는 기타 사용자 리포지토리와 같은 데이터 소스에서 사용자 세부 정보를 검색한다.

userDetails 사용자가 발견되면 사용자의 세부 정보를 나타내는 개체를 만들고 반환한다.

인터페이스 UserDetails는 사용자 이름, 암호(인코딩됨) 및 GrantedAuthority 사용자의 권한이나 역할을 나타내는 객체 모음을 검색하기 위한 메서드를 제공한다.

- UserDetails

인터페이스는 핵심 사용자 정보를 나타내는 객체로 사용자 이름, 암호, 권한 및 사용자의 관련 속성을 포함하여 사용자에 대한 세부 정보를 캡슐화 하여 관리한다.

UserDetails 개체는 일반적으로 인증 중인 사용자를 나타내기 위해 인증 프로세스 중에 사용되며 UserDetails 인터페이스에서 제공하는 몇 가지 중요한 특성 및 메서드이다.

- 메서드

1. getUsername() : 사용자의 이름을 반환
2. getPassword() : 사용자의 비밀번호를 반환한다.
3. getAuthorities() : 사용자와 관련된 권한 또는 역할 나타내는 GrantedAuthority 객체 모음이다.
4. isEnabled() : 사용자 계정의 활성화 또는 비활성화 여부를 나타낸다.
5. isAccountNonExpired() : 사용자의 계정이 만료되었는지 여부를 나타낸다.
6. isAccountNonLocked() : 사용자 계정이 잠겨 있는지 여부를 나타낸다.
7. isCredentialsNonExpired() : 사용자의 자격 증명(암호) 만료 여부를 나타낸다.

- SecurityContextHolder

애플리케이션 내에서 현재 사용자의 보안 컨텍스트에 대한 액세스를 제공하는 중앙 구성 요소이다.

인증된 사용자 세부 정보 및 부여된 권한과 같은 보안 관련 정보를 저장하고 액세스하는 컨테이너 역할을 하게 된다.

Security 프레임워크의 일부이며 보안 컨텍스트와 상호 작용하는 정적 메서드를 제공한다.

- SecurityContextHolder의 주요 역할

1. 보안 컨텍스트 : SecurityContextHolder는 현재 사용자의 보안 관련 정보를 보유하는 SecurityContext 객체를 유지한다.
  2. 보안 컨텍스트 모드 : securityContextHolder는 보안 컨텍스트를 관리하기 위해 다양한 모드를 지원한다.
    - MODE\_THREADLOCAL : 이 모드에서 보안 컨텍스트는 'ThreadLocal'을 사용하여 현재 스레드에 바인딩 된다. 각 스레드에는 자체 보안 컨텍스트가 있으며 다른 스레드에서 코드를 실행할 때 컨텍스트가 자동으로 전파된다.
    - MODE\_INHERITABLETHREADLOCAL : MODE\_THREADLOCAL과 유사하지만 보안 컨텍스트는 원래 스레드에서 생성된 자식 스레드에 의해 상속된다.
  3. 보안 컨텍스트 설정 : 정적 SecurityContextHolder.setContext(SecurityContext context) 메서드를 사용하여 보안 컨텍스트를 설정하는 것이 가능하다.
- 일반적으로 이는 수동 또는 AuthenticationManager 또는 UserDetailsService와 같은 Spring Security 구성 요소에 의해 성공적인 인증 후에 수행된다.

- 4. 보안 컨텍스트 획득 : 보안 컨텍스트를 얻기 위해 SecurityContextHolder.getContext() 메서드를 사용하여 액세스하는 것이 가능하다.  
SecurityContextHolder.getContext() 메서드를 사용하여 액세스하는 것이 가능하다.  
SecurityContext에서 권한과 함께 인증된 사용자를 나타내는 Authentication 객체를 얻는 것이 가능하다.
- 5. 보안 컨텍스트 지우기 : 보안 컨텍스트를 정리하면 정적 SecurityContextHolder.clearContext() 메서드를 호출할 되며 이것은 일반적인 로그아웃 중 혹은 보안 컨텍스트가 더 이상 필요하지 않을 때 수행된다.