Spring Security

Spring Security는 Spring 기반 어플리케이션의 보인을 담당하는 Spring 하위 프레임 워크.

인증(Authenticate)과 인가(Authorize)를 담당한다.

`인증(Authenticate)` : 보호된 리소스에 접근한 대상이 누구인지 확인한다. 즉, 작업을 수행해도 되는 주체인지 확인

`인가(Authorize)` : 해당 리소스에 대해 접근 가능한 권한을 가지고 있는지 확인한다. 즉, 어떤 것을 할 수 있는지 확인

스프링 시큐리티에서는 주로 servlet filter들의 filter chain으로 구성된 위임 모델을 사용한다.

보안과 관련해서 체계적으로 많은 옵션을 제공해주기 때문에 개발자 입장에서는 일일이 보안 관련 로직을 작성하지 않아도 되어 편리해진다.

인증 -> 인가

Spring Security는 기본적으로 인증을 성공하게 되면 인가 절차를 진행하게 되며, 인가 과정에서 해당 리소스에 대한 접근 권한이 있는지 확인하게 된다.

Spring Security에서는 이러한 인증과 인가를 위해 Principal을 아이디, Credential을 비밀번호로 사용하는 `Credential 기반의 인증 방식`을 사용한다.

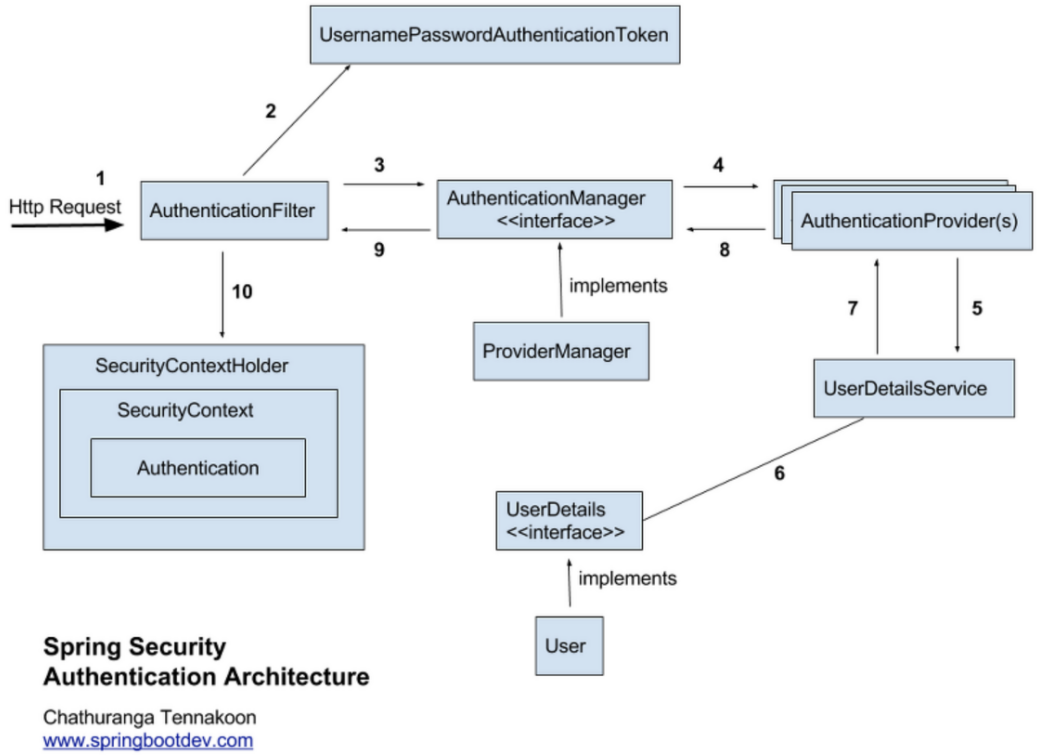
Credential 기반 인증 방식

`접근 주체(Principal)` : 보호받는 리소스에 접근하는 대상

`비밀번호(Credential)` : 리소스에 접근하는 대상의 비밀번호

인증

Architecture



동작 과정

1. ID, PW 기반의 인증 요청이 오면 `UsernamePasswordAuthenticationFilter`에서 가로챈다.
2. UsernamePasswordAuthenticationFilter는 request에서 username과 password를 가져와 이를 기반으로 `UsernamePasswordAuthenticationToken` (Authentication 인터페이스의 구현체)를 생성해 반환한다.
3. `AuthenticationManager` 는 Authentication 객체를 받아 인증하고 인증되었다면 인증(isAuthenticated = true)된 Authentication 객체를 돌려주는 authenticate() 메서드를 구현하도록 하는 인터페이스.

   이를 구현한게 `ProviderManager` 인데, Spring Security가 생성, 등록하고 관리하는 빈이기 때문에 직접 구현할 필요는 없다.

   인증을 담당하는 클래스이지만 실제로는 멤버변수로 가지고 있는 `AuthenticationProvider(s)` 에게 인증을 위임한다.
4. `AuthenticationProvider` 는 interface이므로 이를 구현한 클래스가 필요하다.

   authenticate() 메서드와 supports() 메서드를 구현하게 된다.

   `authenticate()` : 인증 과정을 진행

   `supports()` : 현재 AuthenticationProvider가 파라미터로 받은 Authentication 객체를 인증처리할 수 있는지 확인
5. authenticate() 메서드의 과정에서 `UserDetailsService` 인터페이스의 구현체를 통해 DB에 저장된 사용자 정보를 불러오게 되며 [6] `UserDetail` 인터페이스를 구현한 구현체로 반환받는다.

   반환받은 정보를 가지고 인증이 완료된 UsernamePasswordAuthenticationToken(Authentication) 객체를 생성해 반환하며 AuthenticationFilter가 받는다.
6. 이렇게 구현된 인증절차를 통해 인증이 완료되면 Authentication을 `SecurityContextHolder` 객체 안의 `SecurityContext` 에 저장하게 되고

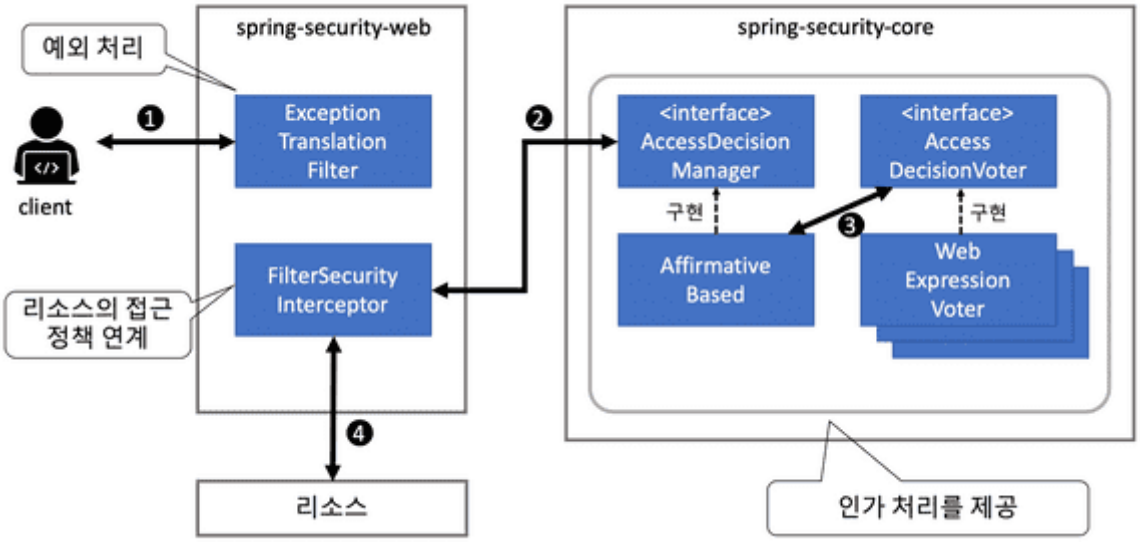
```
Authentication authentication =
SecutiryContextHolder.getContext().getAuthentication();
```

위와 같은 방식으로 저장된 인증 객체를 전역적으로 사용할 수 있게 된다.

인가 (Authorize)

- 애플리케이션에서 사용자가 접근할 수 있는 리소스를 제어하기 위한 기능.
- 각 리소스에 대한 접근 정책을 미리 정의, 접근 시 정책 확인해서 허용 여부 결정.
- 웹 리소스, 자바 메서드, 도메인 객체에 대한 접근 정책 정의 가능.

Architecture



동작 과정

1. 클라이언트가 임의의 리소스에 접근하면 `FilterSecurityInterceptor` 클래스가 가로챈다.
2. `FilterSecurityInterceptor` 클래스는 `AccessDecisionManager` 인터페이스의 메서드를 호출하고 리소스에 대한 접근 가능 여부를 확인하게 된다.
3. 기본적으로 AccessDecisionManager는 `AffirmativeBased` 클래스에 implements 하여 구현한다. AffirmativeBased 클래스는 `AccessDecisionVoter` 인터페이스의 메서드를 호출하고 접근 가능 여부에 대한 투표 결과를 받는다.
4. 위와 같은 과정을 통해 AccessDecisionManager가 허용할 때에만 FilterSecurityInterceptor에서 다음 단계로 나아가 리소스에 접근 할 수 있게 된다.

참고

- https://changrea.io/spring/spring-security-authorization/

참고

- https://wildeveloperetrain.tistory.com/50
- https://wildeveloperetrain.tistory.com/51