

Spring Security

SpringBoot 기준으로 서술

[Java](#), [Spring](#)

Overview

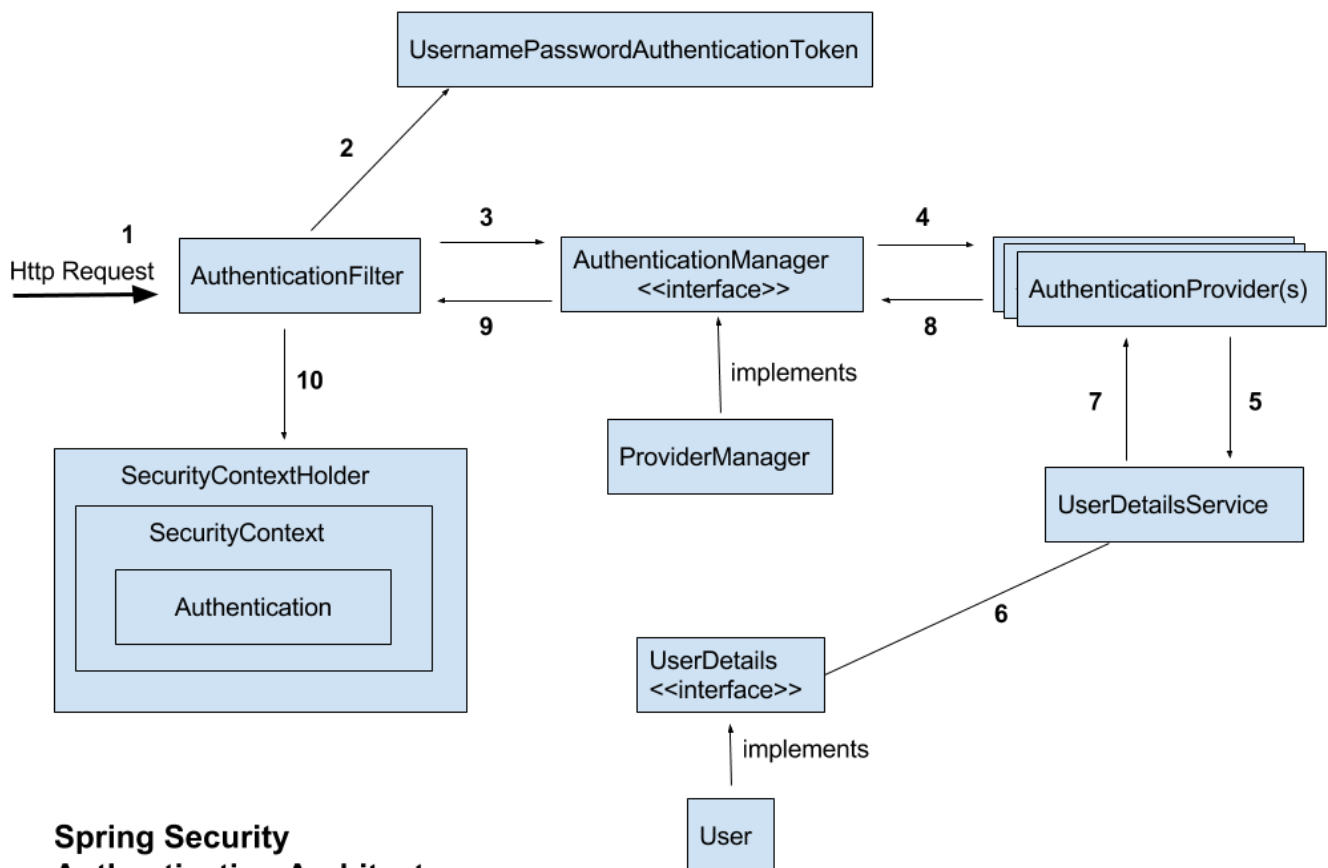
- Authentication : 인증. “유효한 사용자인가?”
- Authorization : 허가(권한 부여). 유효한 사용자가 이 작업을 할 수 있는가?

Documents

- [Spring Security Architecture](#)
- [Spring Security : Authentication Architecture](#)
- [\[Docs\] Spring Security - Servlet Applications](#)
- [Samples](#)

Authentication

[Docs](#)



Spring Security Authentication Architecture

Chathuranga Tennakoon
www.springbootdev.com

- 모두 Interface

classDiagram class AuthenticationManager { authenticate(Authentication authentication) Authentication } class AuthenticationProvider { authenticate(Authentication authentication) Authentication supports(Class~?~ authentication) boolean } class UserDetailsService { loadUserByUsername(String username) UserDetails } class UserDetails { getAuthorities() Collection~? extends GrantedAuthority~ getPassword() String isAccountNonExpired() boolean isAccountNonLocked() boolean isCredentialsNonExpired() boolean isEnabled() boolean } AuthenticationManager --> AuthenticationProvider : authRequest AuthenticationProvider <..> UserDetailsService : 서비스 주입 후 authenticate()에서 사용자 정보를 꺼내어 사용 UserDetailsService -- UserDetails

AuthenticationManager

- AuthenctaionManager
 - 기본 구현체 : [ProviderManager](#)
 - [AuthenticationManagerBuilder](#) : [WebSecurityConfigurerAdapter](#) 의 `configure(AuthenticationManagerBuilder auth)` Method로 “auth” 인자를 사용할 수 있는데, 이 인자의 “userDetailsService(T userDetailsService)”로 “UserDetailsService” 주입이 가능함.

```

AuthenticationManager - org.springframework.security.authentication
  AuthenticationManagerDelegator - org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration
  AuthenticationManagerDelegator - org.springframework.security.config.method.GlobalMethodSecurityBeanDefinitionParser
  AuthenticationManagerDelegator - org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter
  NoOpAuthenticationManager - org.springframework.security.oauth2.client.filter.OAuth2ClientAuthenticationProcessingFilter
  NoOpAuthenticationManager - org.springframework.security.access.intercept.AbstractSecurityInterceptor
  OAuth2AuthenticationManager - org.springframework.security.oauth2.provider.authentication
  ProviderManager - org.springframework.security.authentication

```

AuthenticationProvider

- AuthenticationProvider

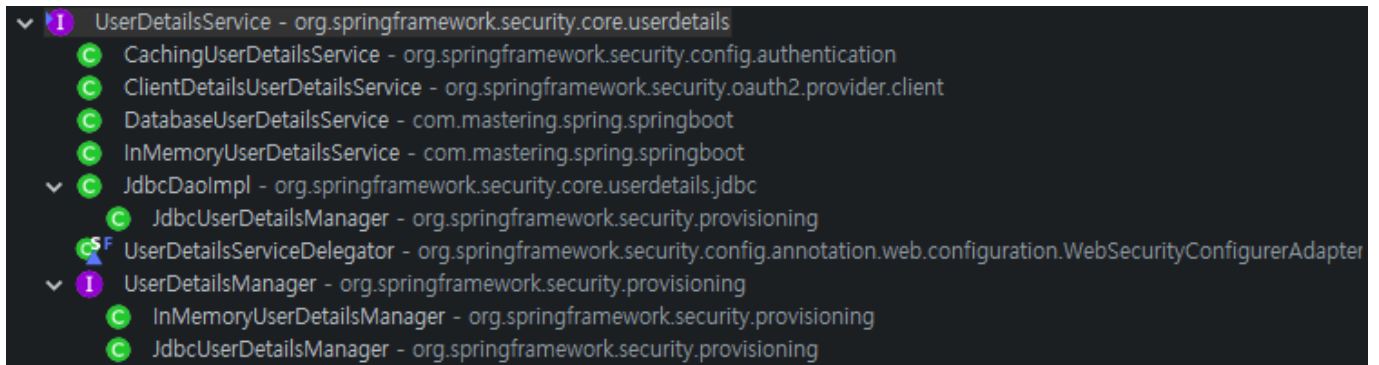
```

AuthenticationProvider - org.springframework.security.authentication
  AbstractJaasAuthenticationProvider - org.springframework.security.authentication.jaas
    DefaultJaasAuthenticationProvider - org.springframework.security.authentication.jaas
    JaasAuthenticationProvider - org.springframework.security.authentication.jaas
  AbstractUserDetailsAuthenticationProvider - org.springframework.security.authentication.dao
    DaoAuthenticationProvider - org.springframework.security.authentication.dao
    AnonymousAuthenticationProvider - org.springframework.security.authentication
    NullAuthenticationProvider - org.springframework.security.config.authentication.AuthenticationManagerBeanDefinitionParser
    OidcAuthenticationRequestChecker - org.springframework.security.config.annotation.web.configurers.oauth2.client.OAuth2LoginConfigurer
    PreAuthenticatedAuthenticationProvider - org.springframework.security.web.authentication.preauth
    RememberMeAuthenticationProvider - org.springframework.security.authentication
    RemoteAuthenticationProvider - org.springframework.security.authentication.rcp
    RunAsImplAuthenticationProvider - org.springframework.security.access.intercept
    TestingAuthenticationProvider - org.springframework.security.authentication

```

UserDetailsService

“UserDetailsService에는 UserDetails loadUserByUsername(String username)”만 있는데, 여기서 UserDetails는 이름, 비밀번호, 인증/인가등의 기본적인 사용자 정보를 담는 Class.



또 다른 사용자 정보를 담는 **Bean**을 만들기보다 이를 상속해서 사용하는게 일을 덜 할 수 있으리라 판단된다.

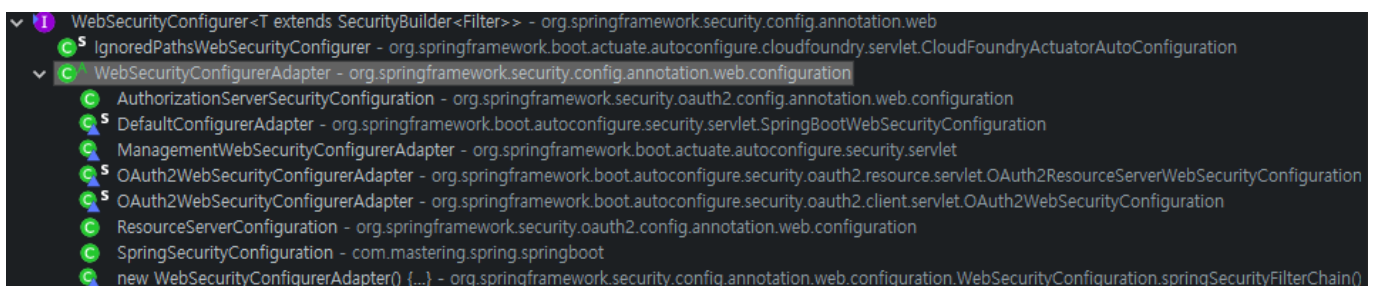
마음에 드는 구현체가 없다면 이 Interface를 상속받아서 재정의.

자식 Interface인 **UserDetailsManager**는 비밀번호 변경, 사용자 생성/삭제/갱신, 사용자 존재 여부등의 기능이 추가로 있음.

WebSecurityConfigurerAdapter

전역적인(globally) authentication이 필요할 때 사용.

WebSecurityConfigurerAdapter를 상속해서 사용한다.



- WebSecurityConfigurerAdapter, AbstractConfiguredSecurityBuilder, AbstractSecurityBuilder : abstract class
- WebSecurity : final class

```
classDiagram
    class SecurityConfigurer~O, B extends SecurityBuilder~O~ {
        init(B builder)
        configure(B builder)
    }
    class WebSecurityConfigurer~T extends SecurityBuilder~Filter~
    class SecurityBuilder~O~ {
        build() O
    }
    class AbstractSecurityBuilder~O~
    class AbstractConfiguredSecurityBuilder~O, B extends SecurityBuilder~O~
    class WebSecurityConfigurer <|-- WebSecurityConfigurer : ~Filter, T extends SecurityBuilder~Filter~
    class WebSecurityConfigurerAdapter : ~WebSecurity~ SecurityBuilder <|-- AbstractSecurityBuilder
    class AbstractSecurityBuilder <|-- AbstractConfiguredSecurityBuilder
    class AbstractConfiguredSecurityBuilder <|-- WebSecurity : ~Filter, WebSecurity~ Aware <|-- ApplicationContextAware
    class SecurityBuilder <|-- WebSecurity : ~Filter~ ApplicationContextAware <|-- WebSecurity
```

Authorization

[Docs](#)

Protection Against Exploits

[Docs](#)

From:

<http://ledyx.xyz/> - **Ledyx WIKI**

Permanent link:

<http://ledyx.xyz/doku.php?id=back-end:spring:security&rev=1619708564>

Last update: **2021/04/29 15:02**

