Spring Security

홍정기

Architecture

인증

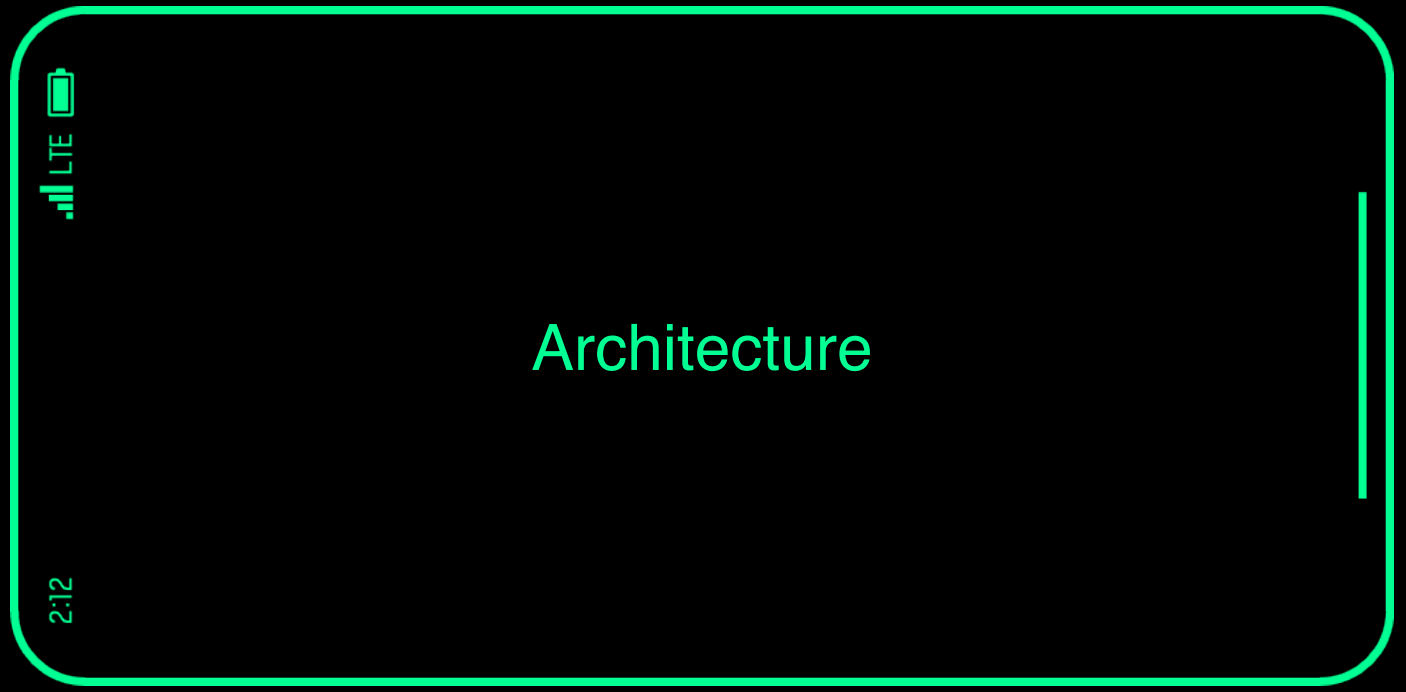
인가

정리

Demo

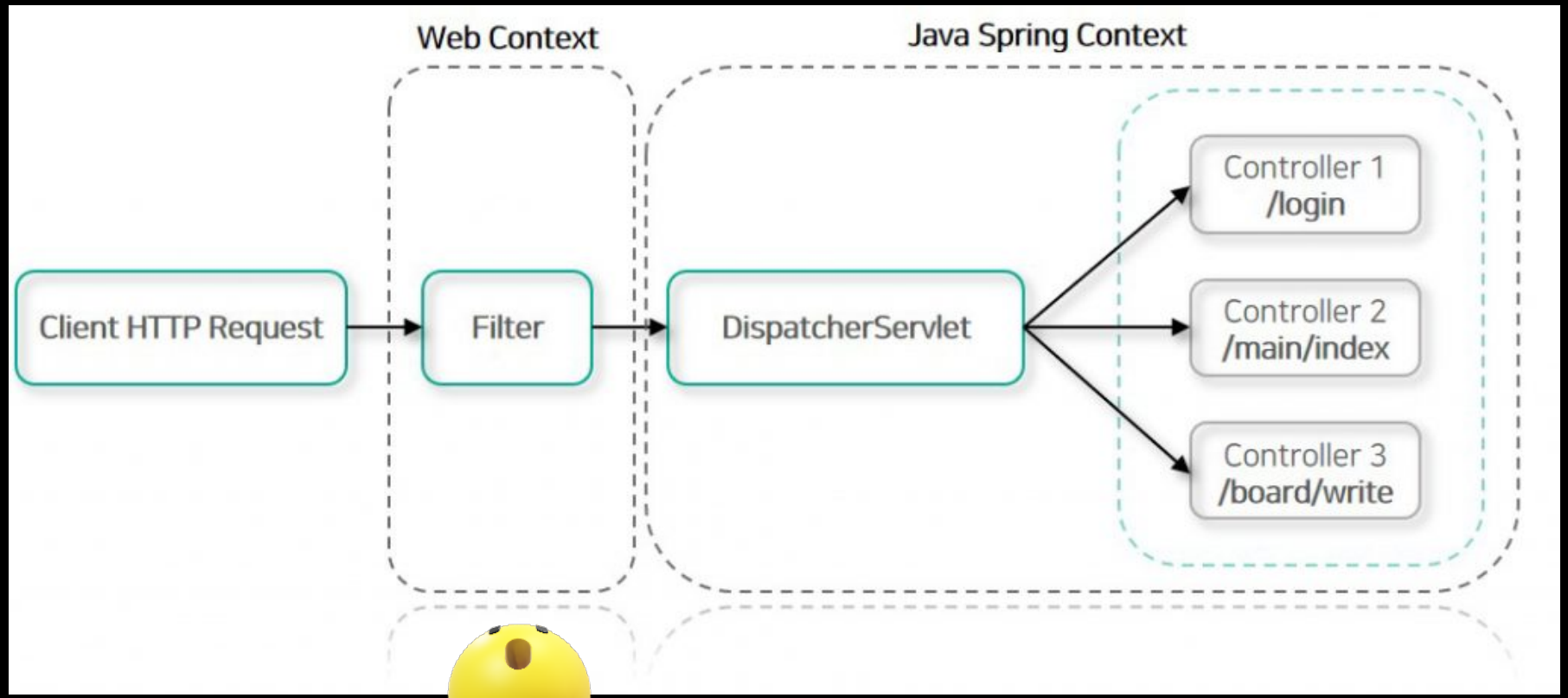
INDEX

OBJECTIVE

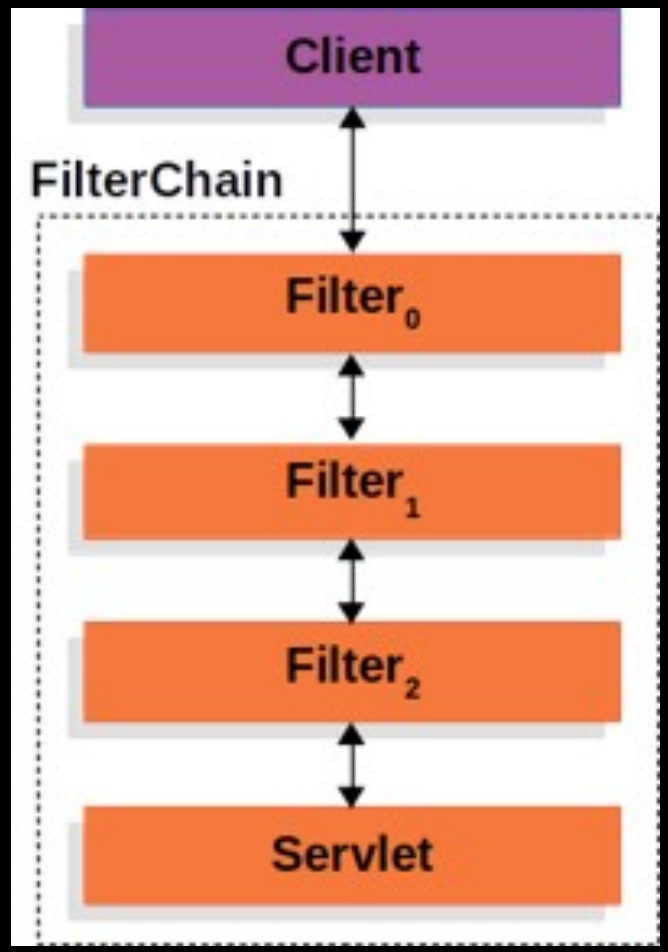


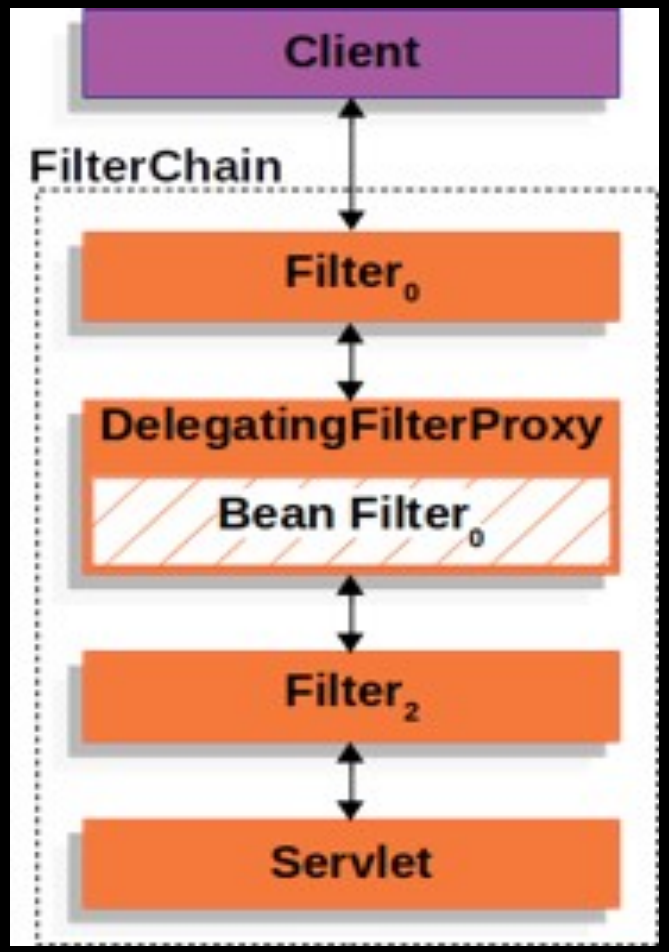
Architecture

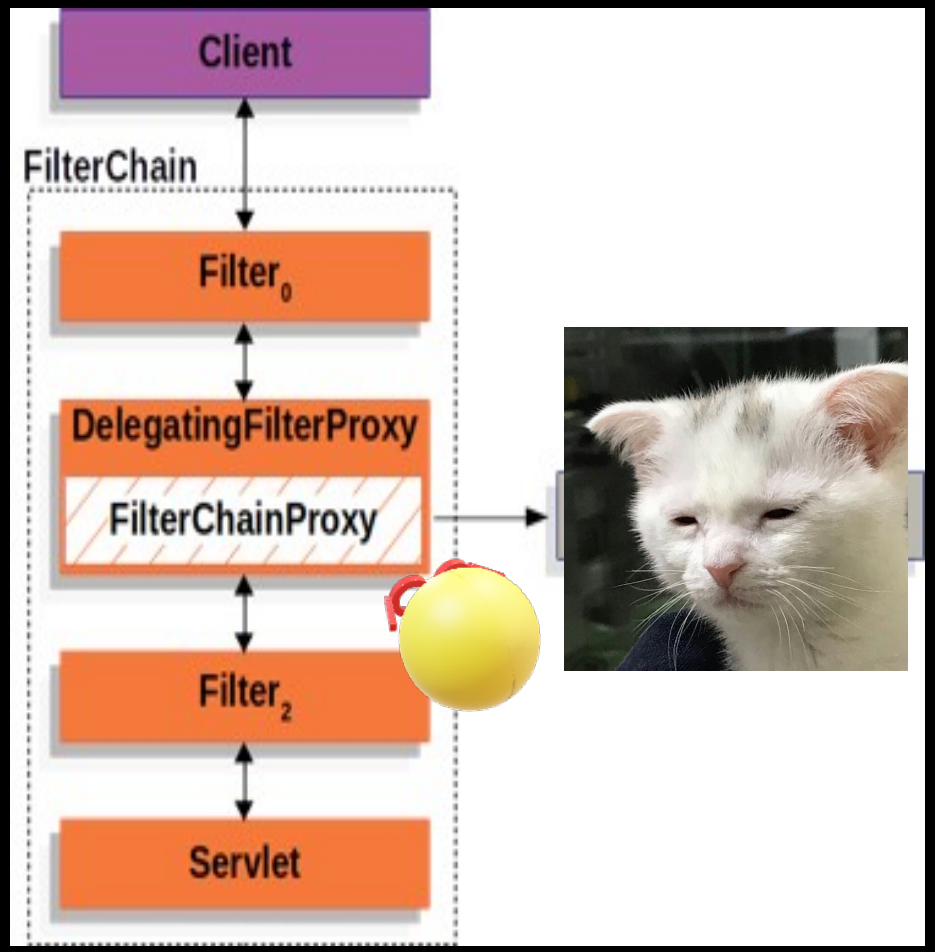
Architecture



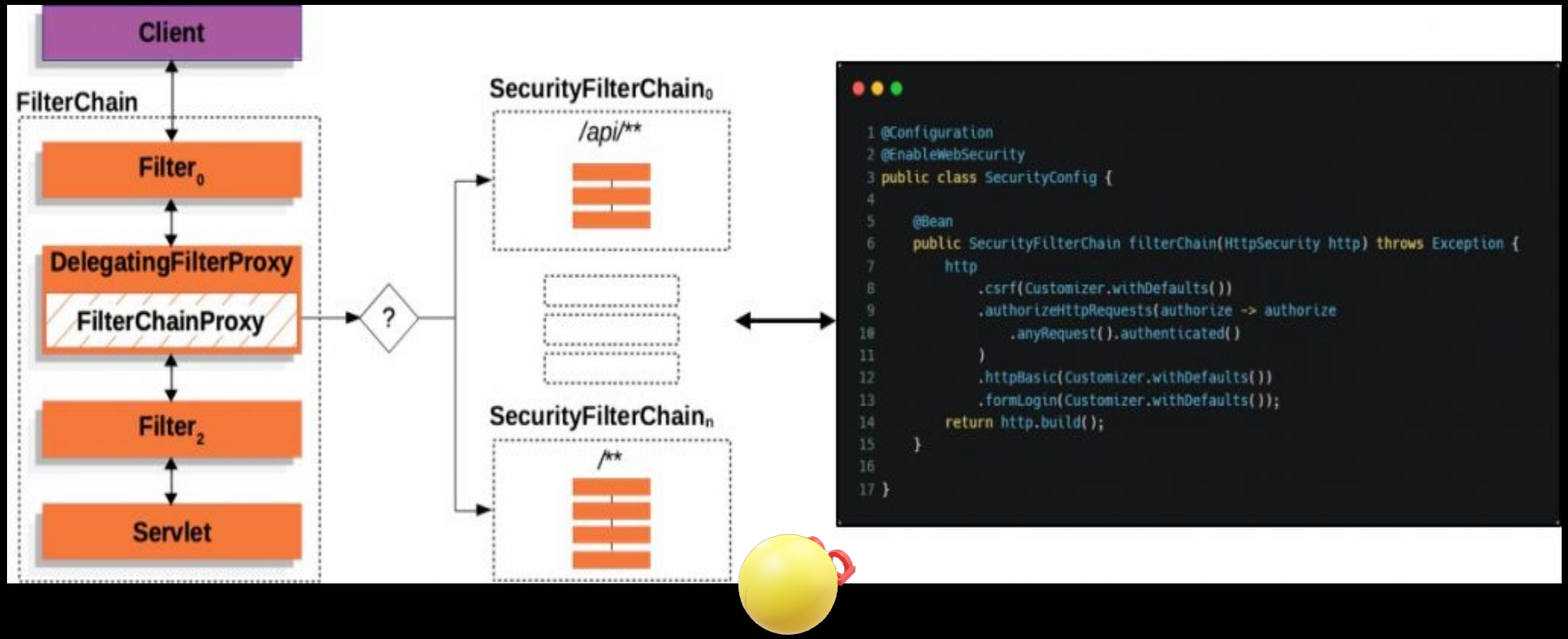
Architecture





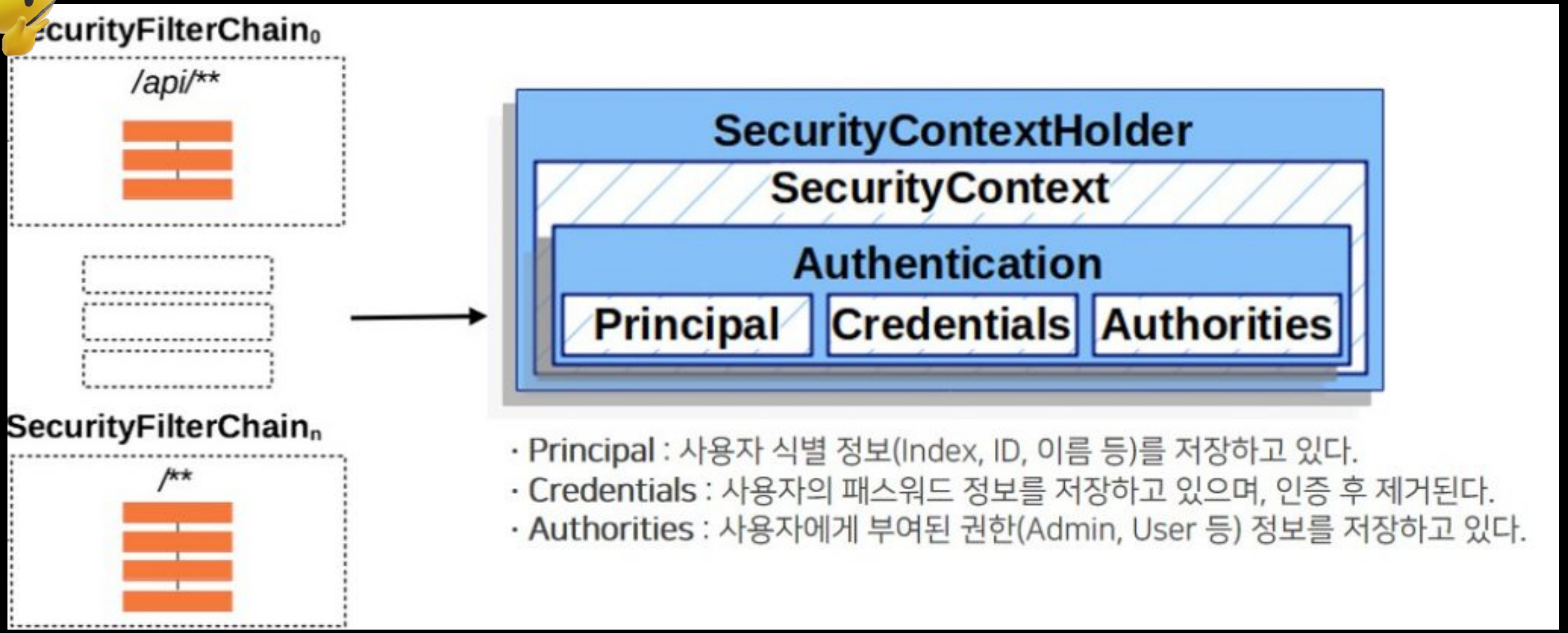


Architecture



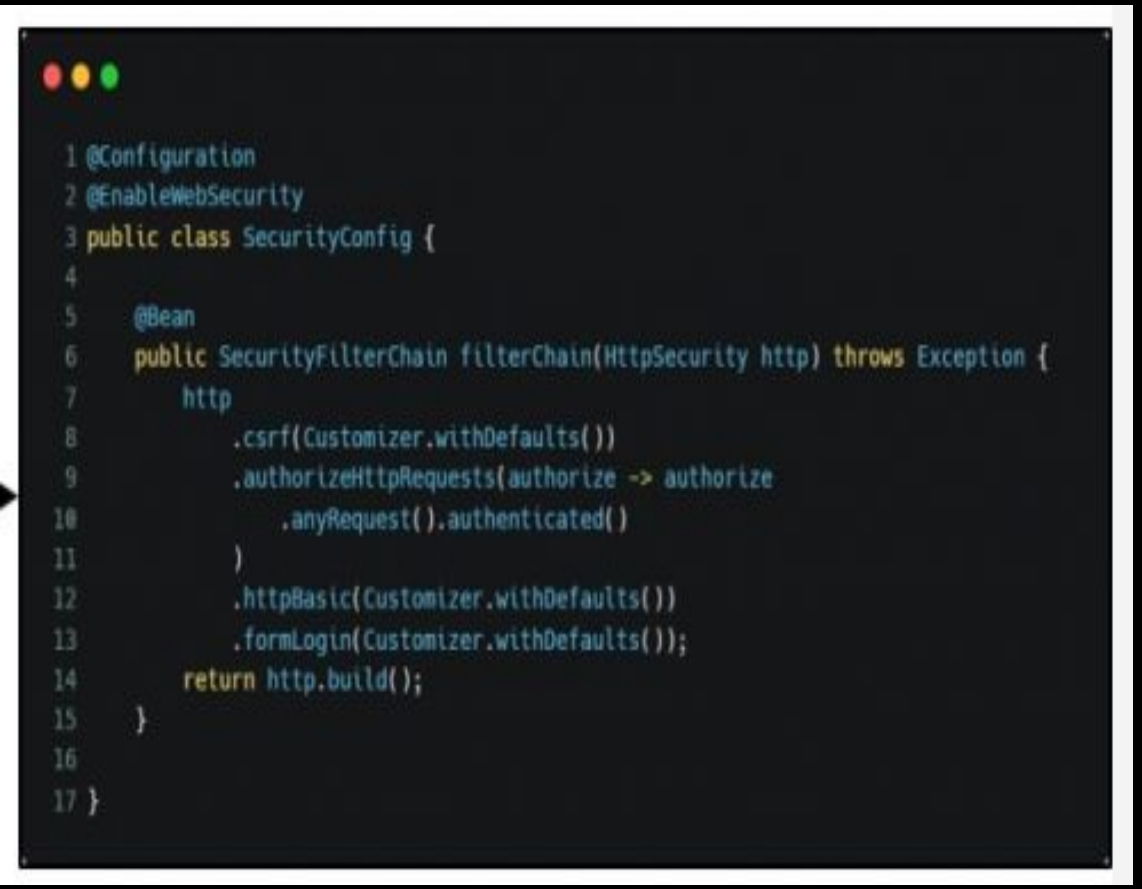

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf(Customizer.withDefaults())
            .authorizeHttpRequests(authorize -> authorize
                .anyRequest().authenticated()
            )
            .httpBasic(Customizer.withDefaults())
            .formLogin(Customizer.withDefaults());
        return http.build();
    }

}
```

Architecture



- Principal : 사용자 식별 정보(Index, ID, 이름 등)를 저장하고 있다.
- Credentials : 사용자의 패스워드 정보를 저장하고 있으며, 인증 후 제거된다.
- Authorities : 사용자에게 부여된 권한(Admin, User 등) 정보를 저장하고 있다.

Architecture

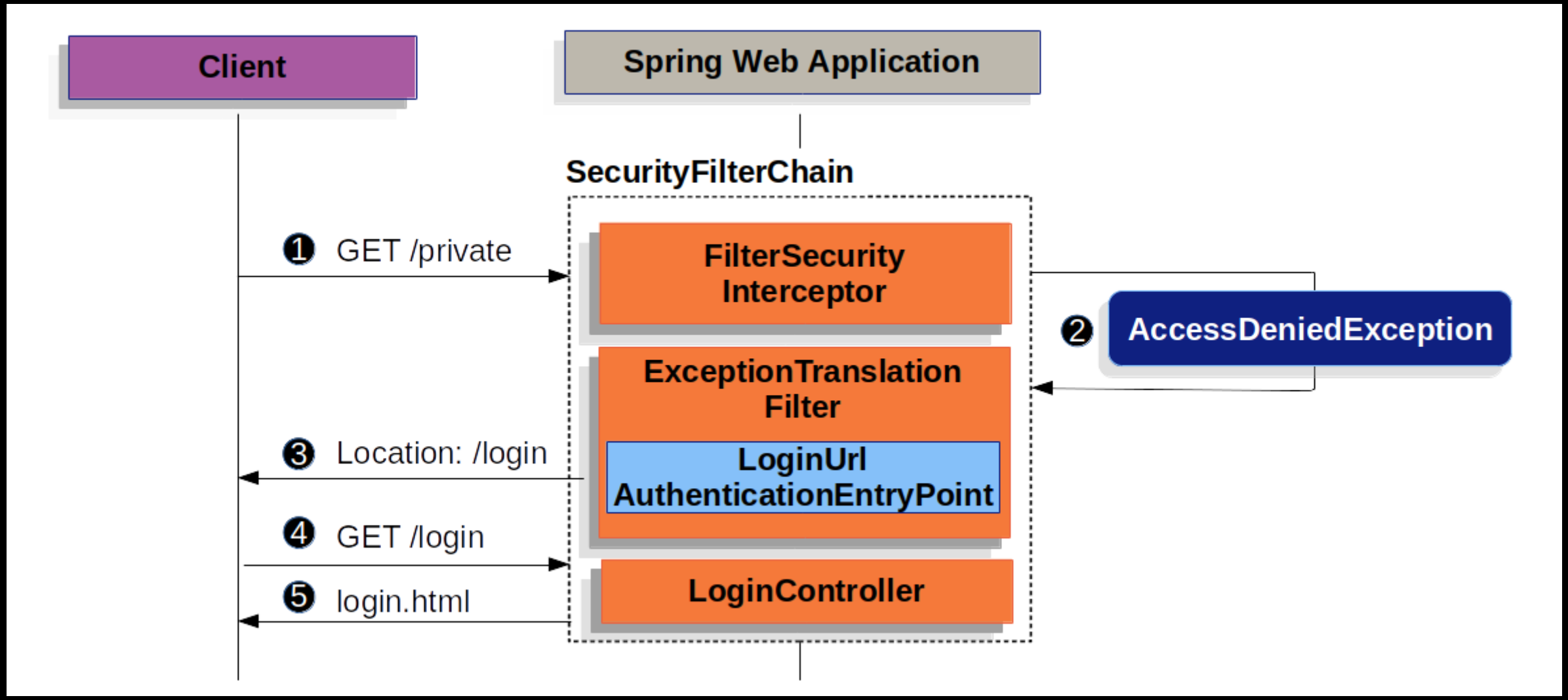


```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf(Customizer.withDefaults())
            .authorizeHttpRequests(authorize -> authorize
                .anyRequest().authenticated()
            )
            .httpBasic(Customizer.withDefaults())
            .formLogin(Customizer.withDefaults());
        return http.build();
    }

}
```

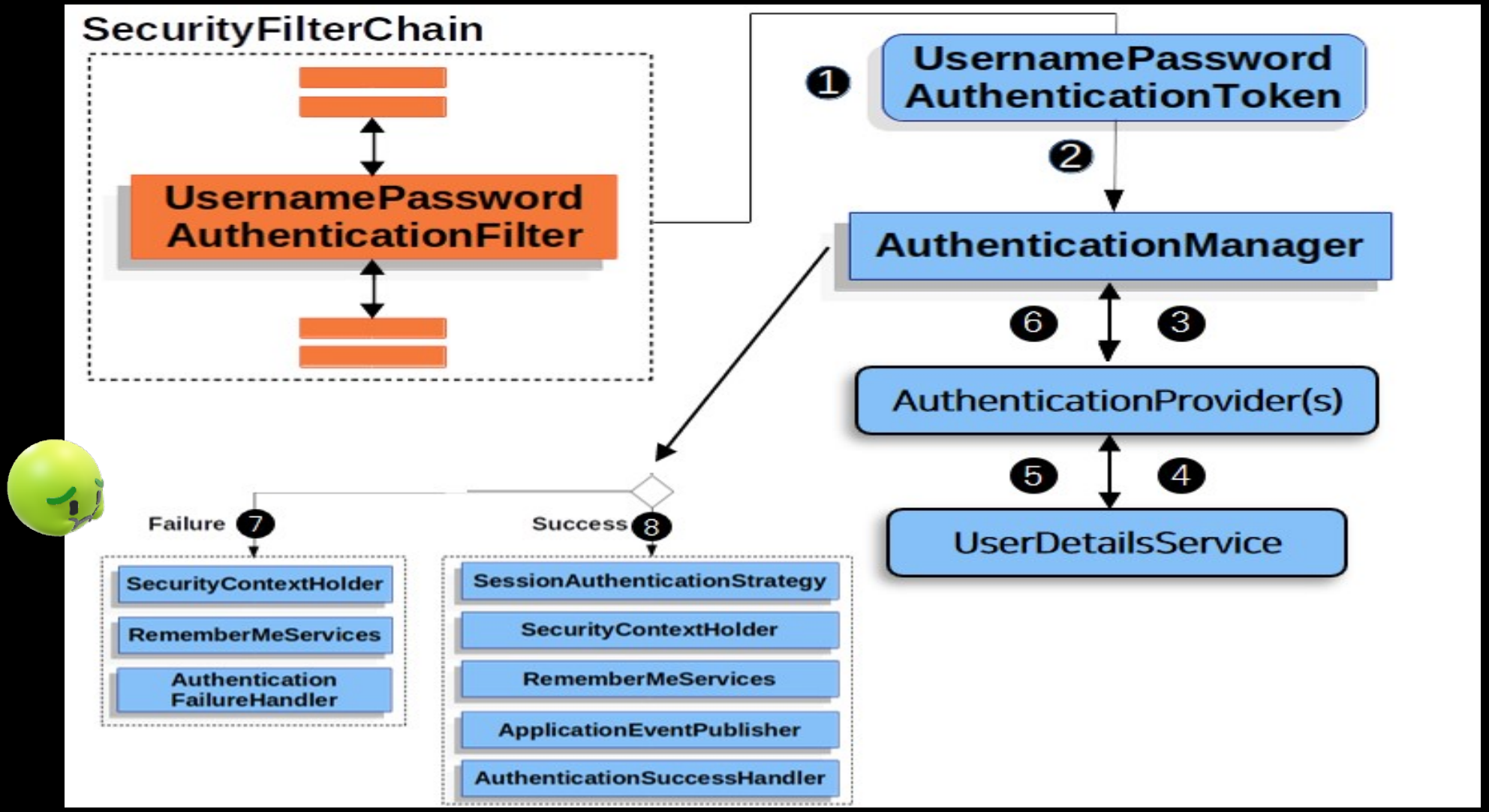
HttpSecurity 인증 API	설명
formLogin()	사용자 이름과 비밀번호 기반의 기본적인 로그인 기능을 설정한다.
logout()	사용자 로그아웃 기능을 설정한다.
csrf()	크로스사이트 요청 위조(CSRF) 공격을 방지하기 위한 기능을 설정한다.
httpBasic()	HTTP Basic 인증 방식을 사용한 기본적인 인증 기능을 설정한다.
sessionManagement()	사용자 세션과 관련된 설정을 수행한다.
rememberMe()	쿠키 정보를 통한 사용자 자동 로그인 기능을 설정한다.
exceptionHandling()	사용자 인증, 인가 실패 시 발생하는 예외 처리에 대해 설정한다.
addFilter()	사용자가 정의한 필터를 설정한다.

OBJECTIVE

인증


Client
Spring Web Application
SecurityFilterChain
1 GET /private
FilterSecurity Interceptor
2
AccessDeniedException
ExceptionTranslation Filter
LoginUrl AuthenticationEntryPoint
3 Location: /login
4 GET /login
5 login.html
LoginController

인증

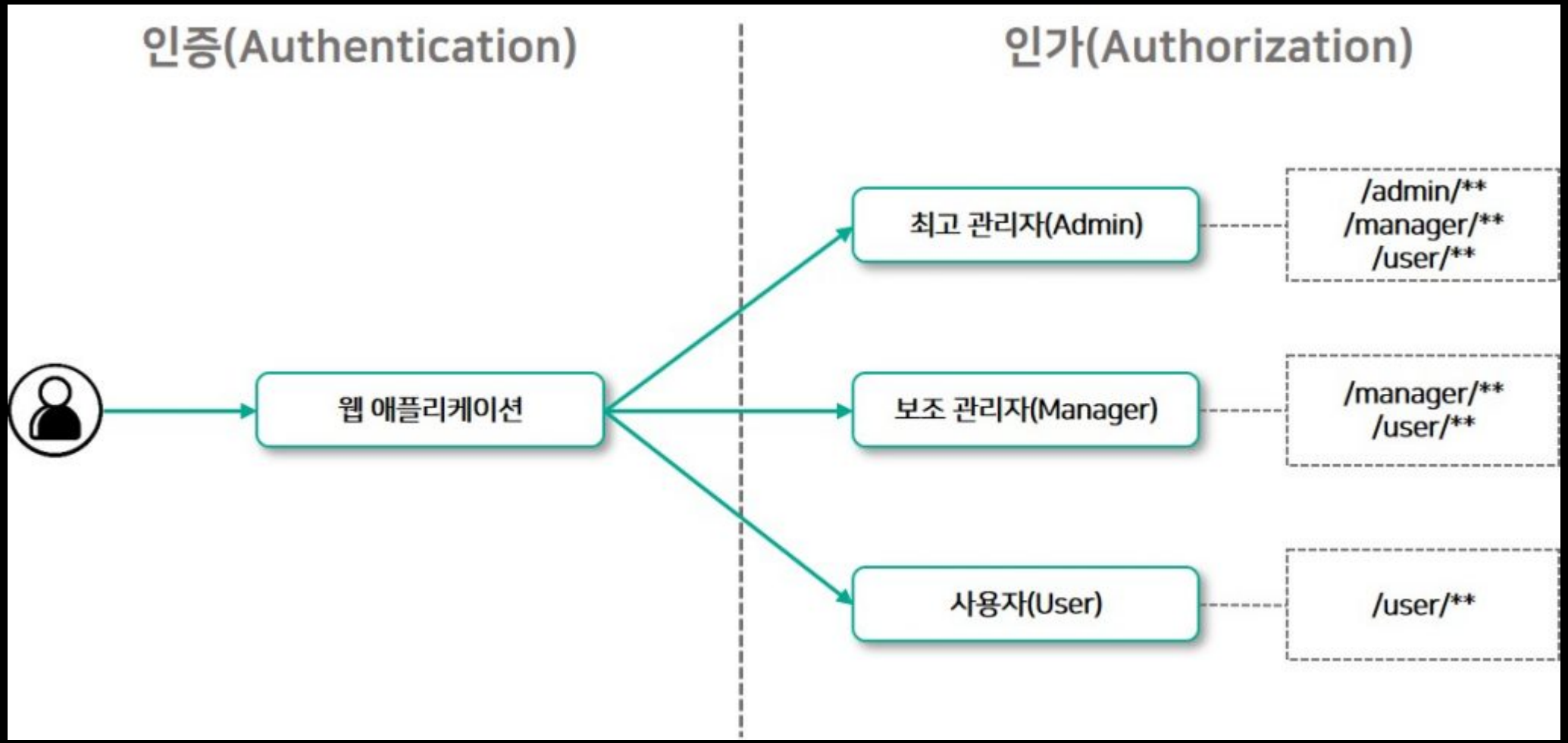

SecurityFilterChain
UsernamePassword AuthenticationFilter
1
UsernamePassword AuthenticationToken
2
AuthenticationManager
6
3
AuthenticationProvider(s)
5
4
UserDetailsService
Failure 7
Success 8
SecurityContextHolder
RememberMeServices
Authentication FailureHandler
SessionAuthenticationStrategy
SecurityContextHolder
RememberMeServices
ApplicationEventPublisher
AuthenticationSuccessHandler

인증

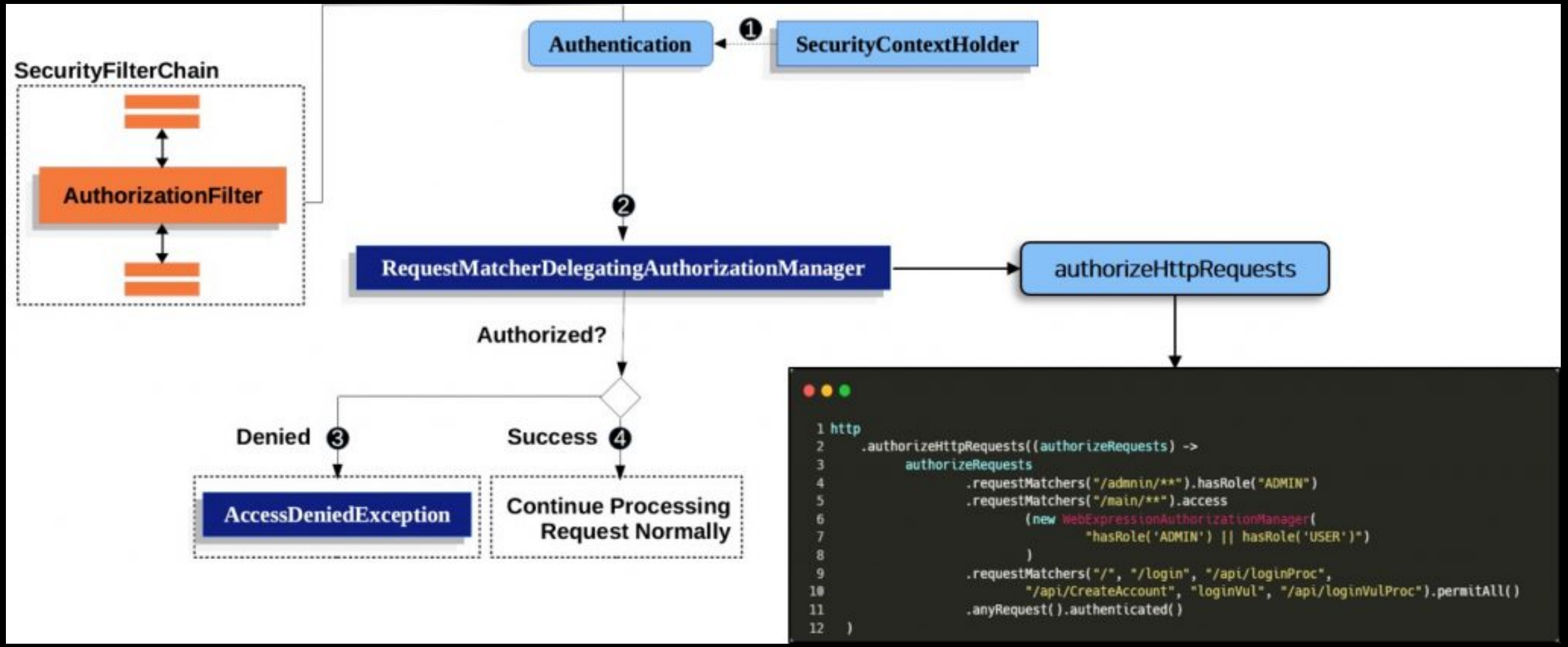
```java
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .requestCache(request ->
                request.requestCache(requestCache))
        .formLogin((formLogin) ->
                formLogin
                        .loginPage("/login")
                        .loginProcessingUrl("/api/loginProc")
                        .usernameParameter("userId")
                        .passwordParameter("userPw")
                        .defaultSuccessUrl("/main/index")
                        .failureHandler(customAuthenticationFailureHandler)
                        .permitAll()
        )

    return http.build();
}
```

OBJECTIVE

인가


인증(Authentication)
인가(Authorization)
웹 애플리케이션
최고 관리자(Admin)
보조 관리자(Manager)
사용자(User)
/admin/**
/manager/**
/user/**
/manager/**
/user/**
/user/**

인가




```
http
    .authorizeHttpRequests((authorizeRequests) ->
        authorizeRequests
                .requestMatchers("/admnin/**").hasRole("ADMIN")
                .requestMatchers("/main/**").access
                        (new WebExpressionAuthorizationManager(
                                "hasRole('ADMIN') || hasRole('USER')")
                        )
                .requestMatchers("/", "/login", "/api/loginProc",
                        "/api/CreateAccount", "loginVul", "/api/loginVulProc").permitAll()
                .anyRequest().authenticated()
)
```

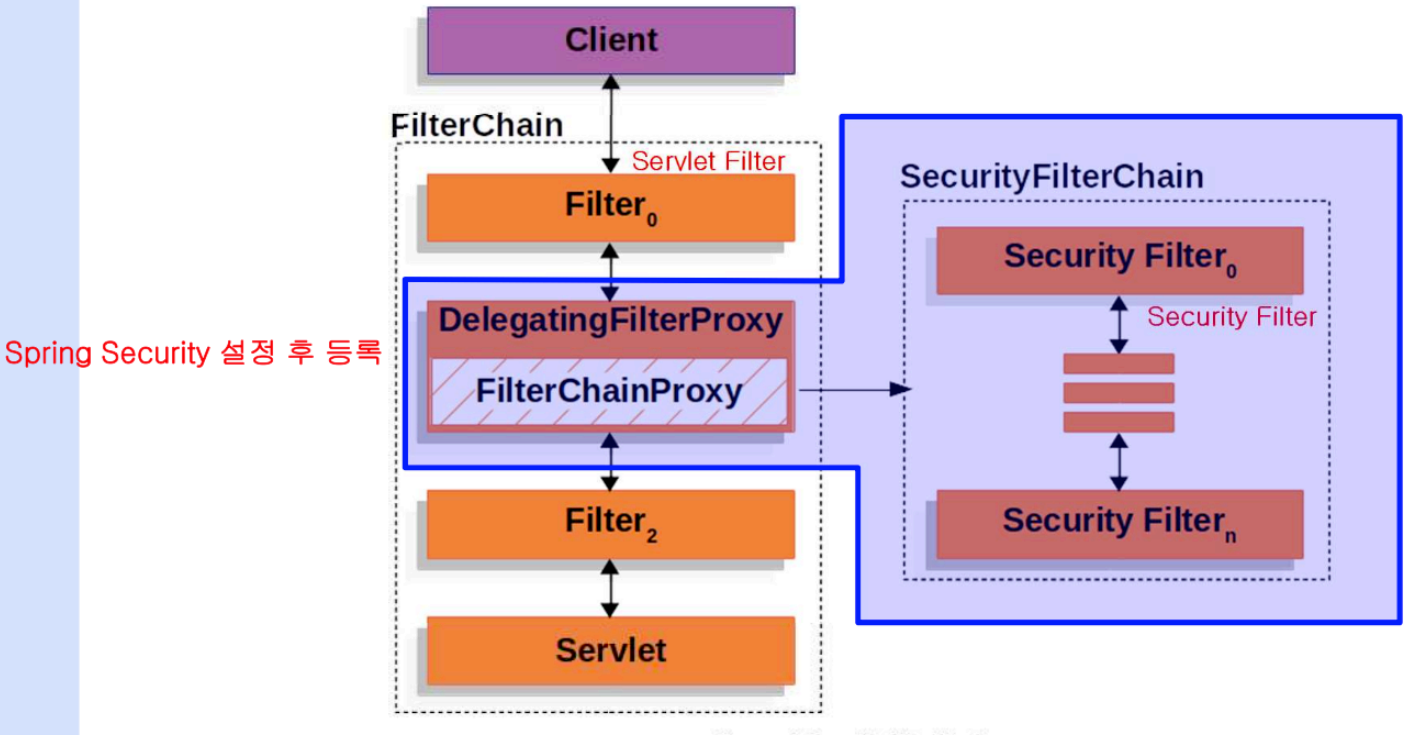
정리

정리

순서	필터 이름	설명
1	DisableEncodeUrlFilter	URL에 parameter를 제외하도록 설정(ex. session id)
2	WebAsyncManagerIntegrationFilter	비동기 작업에서 쓰레드들 간의 SecurityContext 관리
3	SecurityContextHolderFilter	HTTP 요청 사이에 SecurityContext 정보를 유지(session)
4	HeaderWriterFilter	응답 헤더에 보안 관련 헤더 추가
5	*CorsFilter* **기본 Filter**	CORS 설정 (다른 도메인(출처)에서 오는 요청을 허용)
6	*CsrfFilter*	CSRF 공격 방지를 위한 토큰 유효성 검증
7	*LogoutFilter*	로그아웃 요청 처리
8	UsernamePasswordAuthenticationFilter	Form 방식의 username과 password를 사용한 인증 요청 처리
9	DefaultLoginPageGeneratingFilter	기본 로그인 페이지 생성
10	DefaultLogoutPageGeneratingFilter	기본 로그아웃 페이지 생성
11	BasicAuthenticationFilter	http basic 기반 로그인 처리
12	RequestCacheAwareFilter	이전에 접근하려고 했던 URL로 리다이렉트
13	SecurityContextHolderAwareRequestFilter	ServletRequest에 SpringSecurity 기능을 포함하도록 랩핑
14	AnonymousAuthenticationFilter	보안 주제가 없는 경우 익명으로 인증 처리
15	ExceptionTranslationFilter	인증, 인가 예외를 처리 (예외 시작점)
16	AuthorizationFilter(FilterSecurityInterceptor)	접근권한에 따른 인가 수행 (authorizeHttpRequests())

정리

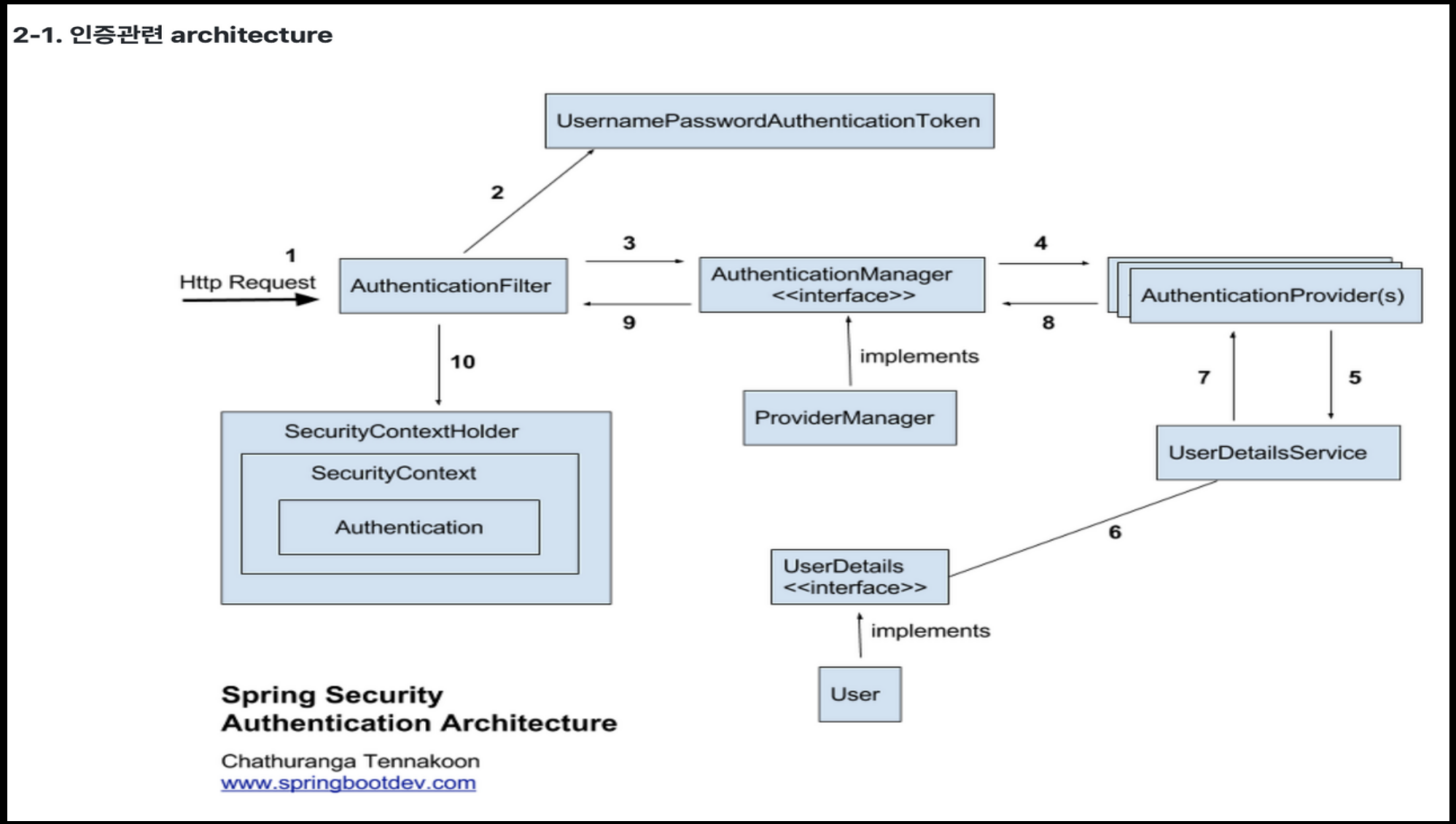
Spring Security 6.3.3




```
public class MeetingFilter implements Filter {

  public void doFilter(  ServletRequest request,
                         ServletResponse response,
                         FilterChain chain) {

    // 전처리
    chain.doFilter(request, response); // next filter
    // 후처리
}
```

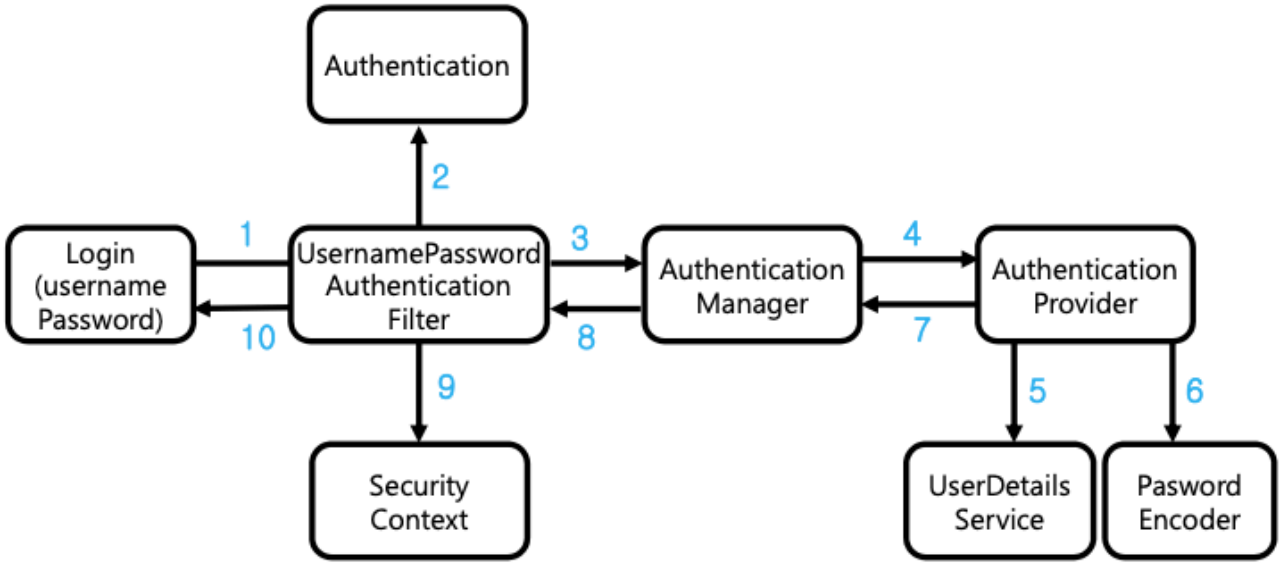
정리

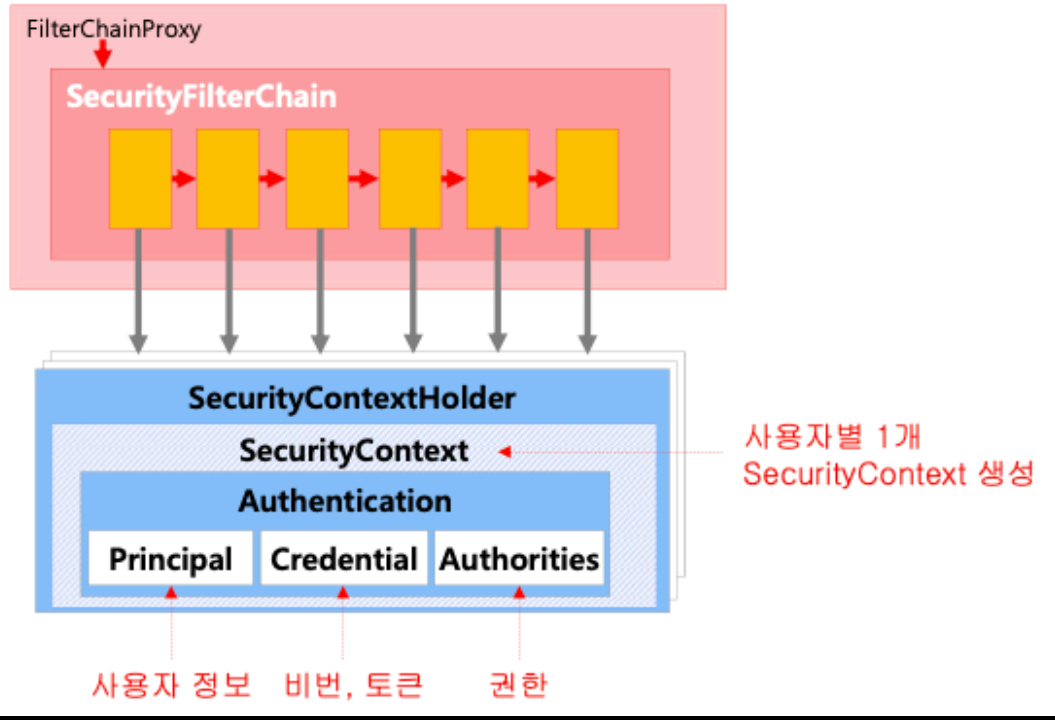
2-1. 인증관련 architecture



정리

Authentication Flow





Demo