Spring.boot.security

InformAtelier

Features

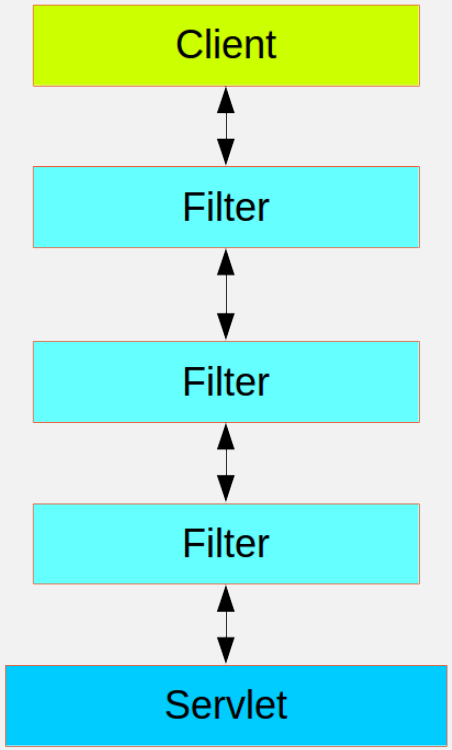
- Comprehensive and extensible support for both Authentication and Authorization
- Protection against attacks like session fixation, clickjacking, cross site request forgery, etc
- Servlet API integration
- Optional integration with Spring Web MVC
- Social integration

InformAtelier

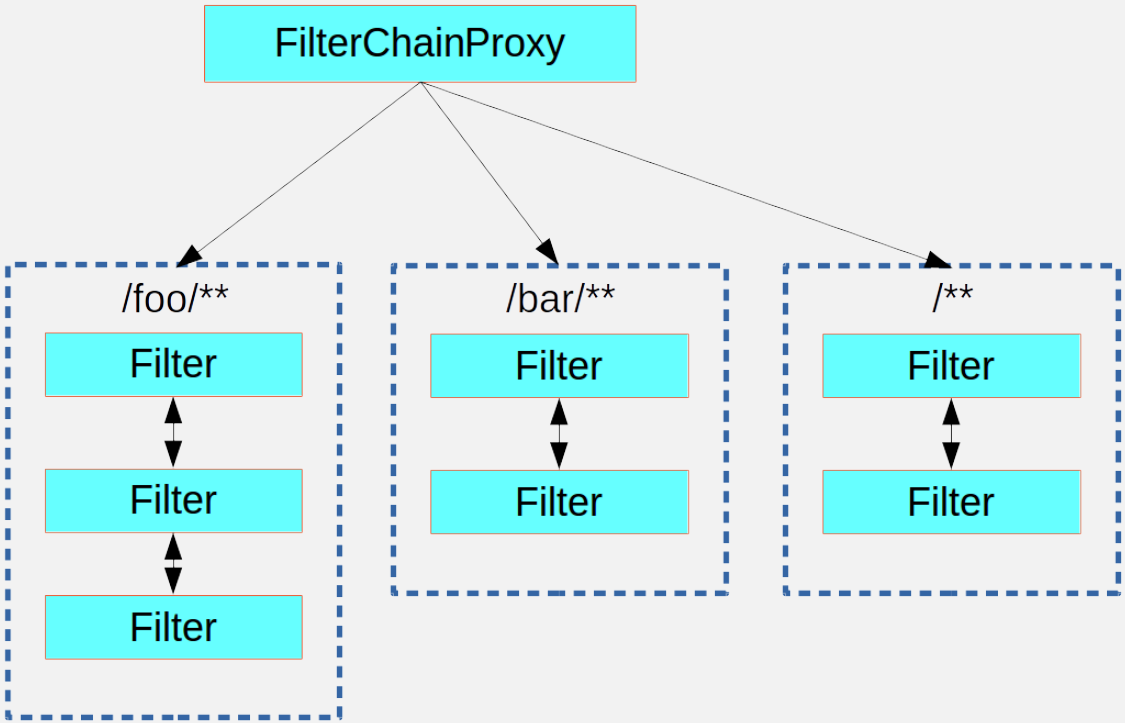
Dependency

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Security Servlet Filter



Filter Chain Proxy



Authentication Strategy

```
public interface AuthenticationManager {

  Authentication authenticate(Authentication authentication)
    throws AuthenticationException;

}
```

ProviderManeger

- The most commonly used implementation of AuthenticationManager is ProviderManager, which delegates to a chain of AuthenticationProvider

Authentication Principal

```
@RequestMapping(value = "/username", method = RequestMethod.GET)
@ResponseBody
public String currentUserName(Authentication authentication) {
  UserDetails userDetails = (UserDetails) authentication.getPrincipal();
  System.out.println("User has authorities: " userDetails.getAuthorities());
  return authentication.getName();
}
```

Java Security Principal

```
import java.security.Principal;


    @RequestMapping(value = "/username", method = RequestMethod.GET)
    @ResponseBody
    public String currentUserNameSimple(HttpServletRequest request) {
        Principal principal = request.getUserPrincipal();
        return principal.getName();
    }
```

Authentication Bonus

- Cookie, Session
- Remember Me
- LDAP

InformAtelier

MVC Integration

- Login/logout out of the box.
- Default closed protection to relax.

InformAtelier

Authorization Strategy

- AccessDecisionManager delegate to a chain of AccessDecisionVoter.
- An AccessDecisionVoter considers an Authentication (representing a principal) and a secure Object (web resource or method) which as been decorated with ConfigAttributes (string for principal role)

Authorization Bonus

- Declarative routes and annotations.
- WebSockets.

InformAtelier

Social Authorizations

- Oauth2 client, server
- OpenID

InformAtelier

UserDetails Strategy

```
public interface UserDetails extends Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();
    String getPassword();
    String getUsername();
    boolean isAccountNonExpired();
    boolean isAccountNonLocked();
    boolean isCredentialsNonExpired();
    boolean isEnabled();
}
```

Custom UserDetailsService

```
@Service
public class MyUserService implements UserDetailsService {…}


public interface UserDetailsService {
UserDetails loadUserByUsername(String username) throws
                                        UsernameNotFoundException;
}
```

Enable Security Config

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true)
public class WebSecurity extends WebSecurityConfigurerAdapter {
   @Override
   protected void configure(HttpSecurity http) throws Exception {...}
   @Override
   protected void configure(AuthenticationManagerBuilder auth)
                                                        throws Exception {...}
}
```

Config Routes

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
        .antMatchers("/","/index").permitAll()
        .antMatchers("/dashboard").authenticated()
        .antMatchers("/profile").hasAnyRole(Roles.ADMIN.name())
```

Config Login Logout

```
            .and()
.formLogin().loginPage("/user/login")
            .permitAll()
            .defaultSuccessUrl("/deck/my")
            .and()
.logout().logoutSuccessUrl("/user/login")
            .authenticated();
```

Config Credentials

```
    @Override
    protected void configure(AuthenticationManagerBuilder auth)
                                                        throws Exception {
        userDetailService.setPasswordEncoder(passwordEncoder());
        auth.userDetailsService(userDetailService);
}
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
```

Method Securty

```
@Service
public class MyService {

  @Secured("ROLE_USER")
  public String secure() {
    return "Hello Security";
  }

}
```

OWASP Top 10

1) Injection
2) Broken Authentication
3) Sensitive Data Exposure
4) XML Extenal Entities (XEE)
5) Broken Access Control
6) Security Misconfiguration
7) Cross Site Scripting
8) Insecure Deserialization
9) Known Deps Vulnerability
10) Insufficient Monitoring

Config Protections

```
http.csrf().disable();

http.headers().disable();

http.headers().frameOptions().disable();

http.headers().xssProtection().enable();

http.headers().contentTypeOptions().disable();

http.headers().cache().enable();

Http.headers().httpStrictTransportSecurity()
```

Config Content Policy

```
http.headers()
        .contentSecurityPolicy(
"script-src 'self' https://trustedscripts.example.com;
object-src https://trustedplugins.example.com;
report-uri /csp-report-endpoint/");
```

InformAtelier

Sessions

- Concurrency
- Timeout
- Fixation
- Delete on logout

InformAtelier

Session Concurrency

```
@Bean
//
public HttpSessionEventPublisher httpSessionEventPublisher() {
   return new HttpSessionEventPublisher();
}
@Override
protected void configure(HttpSecurity http) throws Exception {
   http.sessionManagement().maximumSessions(2)
}
```

Session Timeout

```
http.sessionManagement()
  .invalidSessionUrl("/invalidSession.html");
```

InformAtelier

Session Fixation

```
http.sessionManagement()
  .sessionFixation().migrateSession()
```

none - Don’t do anything. The original session will be retained.

newSession - Create a new "clean" session, without copying the existing session data (Spring Security-related attributes will still be copied).

migrateSession - Create a new session and copy all existing session attributes to the new session. This is the default in Servlet 3.0 or older containers.

changeSessionId - Do not create a new session. Instead, use the session fixation protection provided by the Servlet container

Spring Session Project

- Clustered Sessions
- WebSocket Keep Alive
- WebFlux Reactive Application Containers

InformAtelier

Guide

- https://spring.io/blog/2010/08/02/spring-security-in-google-app-engine/

Riferimenti

- https://spring.io/guides/topicals/spring-security-architecture/
- https://spring.io/projects/spring-security
- https://spring.io/projects/spring-security-oauth
- https://oauth.net/
- https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf