Spring Web Security Getting Start

Spring MVC

Getting Start
Spring Web Security

- About Security
- Getting Start
- Spring Security Filters
- Security Filter Chain Configurations

About Security

- Enterprise Application တွေဟာ လက်ရှိ Request ကို ဘယ်သူတွေက ဆောင်ရွက်နေတာလဲ၊ ပြီးတော့ လုပ်ပိုင်ခွင့်ရောရှိရဲ့လား ဆိုတာ ဆုံးဖြတ်နိုင်ဖို့လိုအပ်ပါတယ်

- လက်ရှိအသုံးပြုနေသူဟာ ဘယ်သူလဲ ဆိုတာကို စစ်ဆေးပေးတာကတော့ Authentication ဖြစ်ပြီး၊ Web Resource တွေကို ဘယ်လိုသူတွေမှ အသုံးပြုခွင့်ရှိသလဲဆိုတာကို သတ်မှတ်ပေးနိုင်တာကတော့ Authorization ပဲဖြစ်ပါတယ်

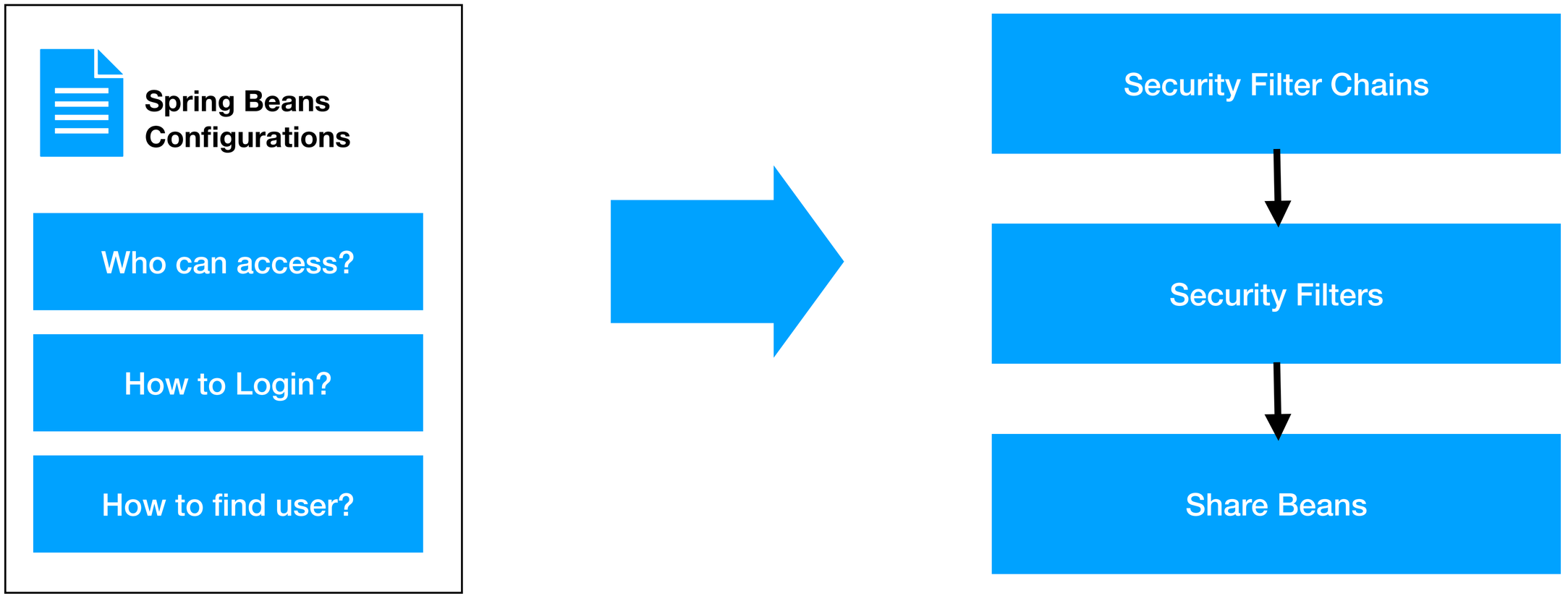
- Spring Security မှာ Authentication, Authorization နှင့် အခြား Security Attack တွေကို တားဆီးပေးဖို့ Protection Mechanism တွေကို Declaration Style ဖြင့် ရေးသားနိုင်အောင် ပြင်ဆင်ပေးထားပါတယ်

Using Spring Security

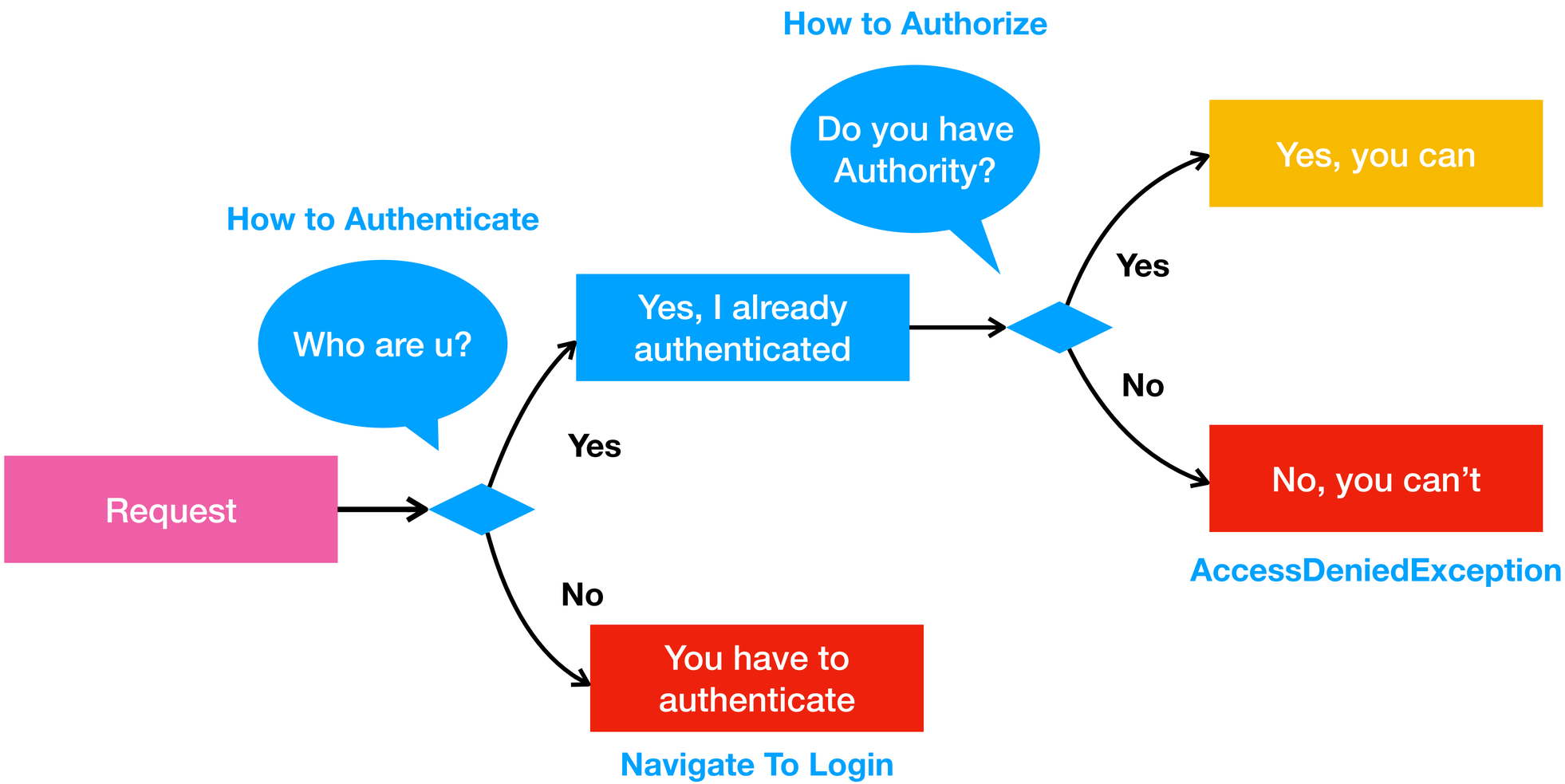
- Spring Web Project တွေမှာ Spring Security ကို အသုံးပြုလိုတယ်ဆိုရင် အခြေခံအားဖြင့် Configuration တွေကို နှစ်နေရာမှာ ရေးသားပေးရမှာ ဖြစ်ပါတယ်

- ပထမတစ်ခုကတော့ Spring Bean Configuration တွေမှာ ရေးသားပေးရမှာ ဖြစ်ပြီး နောက်တစ်ခုက Spring Configuration ကြောင့် ရရှိလာတဲ့ Security Filter တွေကို Servlet Filter တွေကနေ အသုံးပြုနိုင်အောင် Configuration ကို ရေးသားပေးရမှာ ပဲ ဖြစ်ပါတယ်

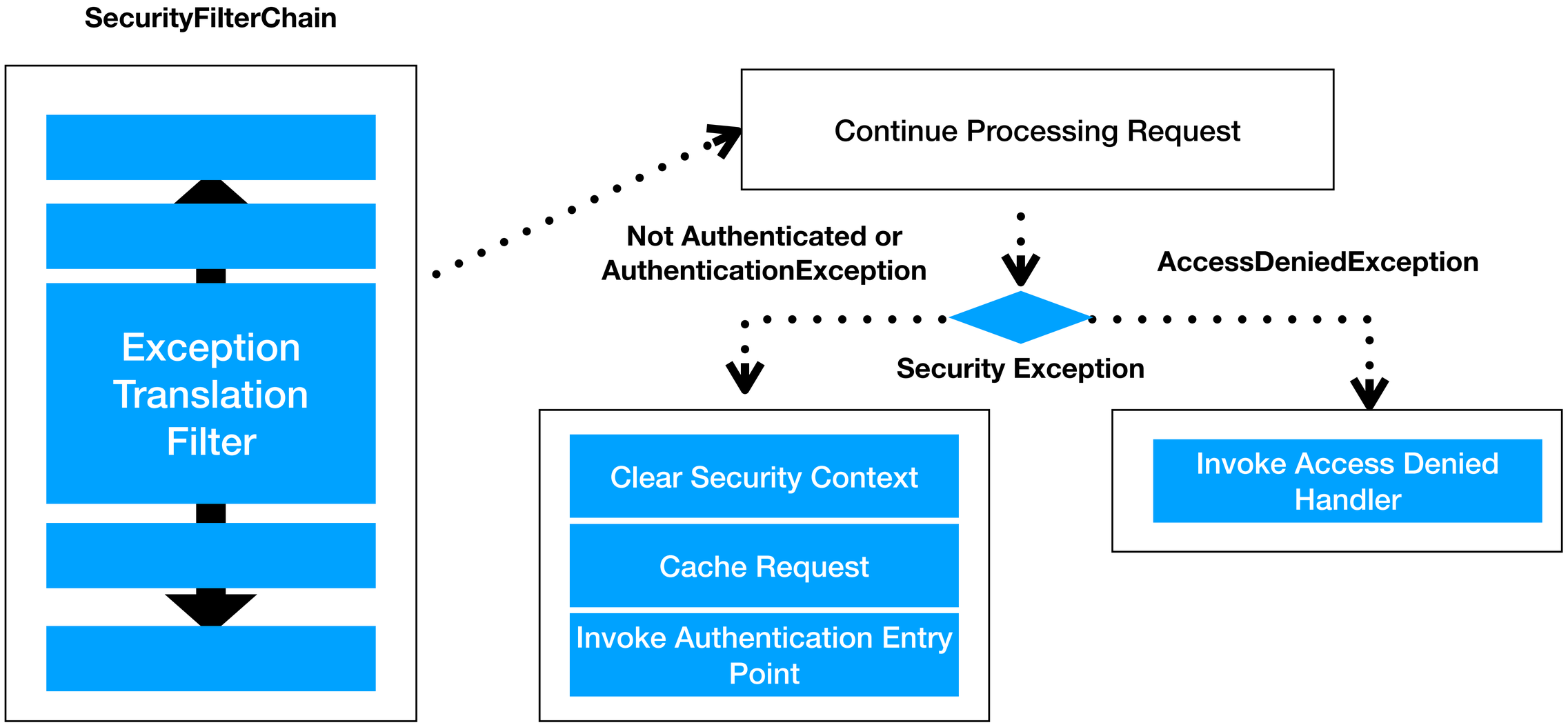
Spring Security Configuration



Security of Web Request



Handling Security Exceptions



Getting Start

- Spring Security ကို အသုံးပြုနိုင်ဖို့အတွက် Maven Dependency အနေဖြင့် spring-security-web နှင့် spring-security-config တို့ကို ထည့်သွင်းထားဖို့လိုအပ်ပါတယ်

- Spring Security Configuration အနေဖြင့် Web Resources တွေကို ဘယ်လို Role တွေက အသုံးပြုနိုင်သလဲဆိုတာနဲ့ ဘယ်လို User တွေရှိတာလဲ ဆိုတာကို သတ်မှတ်ထားဖို့လိုအပ်ပါတယ်

- နောက်ဆုံး Request တွေက DispatcherServlet ဆီကို မရောက်ခင် Security ကို စစ်ဆေးနိုင်ဖို့ အတွက် Spring Security က ရေးပေးထားတဲ့ DelegatingFilterProxy ကို Registration လုပ်ထားပေးရမှာ ဖြစ်ပါတယ်

Getting Start - Maven Dependency

```xml
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>6.1.0</version>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>6.1.0</version>
</dependency>
```

Getting Start - Security Config

```
<security:http pattern="/" security="none" />

<security:http>

    <security:intercept-url pattern="/admin/**" access="hasAuthority('Admin')"/>
    <security:intercept-url pattern="/member/**" access="hasAnyAuthority('Admin', 'Member')"/>

    <security:form-login />
    <security:logout/>

</security:http>

<security:user-service>
    <security:user name="Admin" authorities="Admin" password="{noop}Admin"/>
    <security:user name="Member" authorities="Member" password="{noop}Member"/>
</security:user-service>
```

Getting Start - Filter Mapping

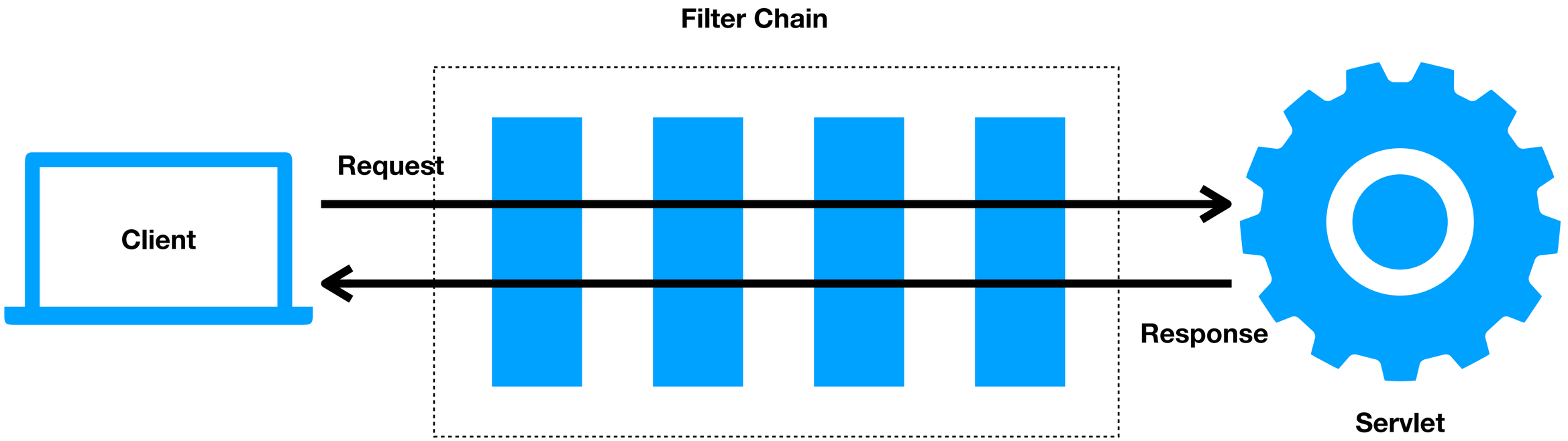
```
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>

<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

- Spring Security ကို ဆောင်ရွက်နိုင်ဖို့အတွက် DeligatingFilterProxy ကို Servlet Filter အနေဖြင့် Registration လုပ်ပေးထားဖို့လိုအပ်ပါတယ်

Spring Security Mechanism

- Spring Security Framework ဟာ Declarative Style ဖြင့် ရေးသားနိုင်အောင် ပြင်ဆင်ထားတဲ့ Framework တစ်ခု ဖြစ်ပါတယ်
- Web Security နှင့်ပတ်သက်တဲ့ Configuration တွေကို ရေးသားခြင်းအားဖြင့် နောက်ကွယ်မှာ Security Filters များ နှင့် Share Bean တွေကို တည်ဆောက်ပေးနိုင်ပါတယ်
- Security Mechanism ကို အသုံးပြုနိုင်ဖို့အတွက် Servlet Filter ထဲကနေ Spring Managed Security Filter များကို အသုံးပြုဖို့လိုအပ်ပါတယ်
- ထို့ကြောင့် Spring Security ကို အသုံးပြုနိုင်ဖို့ Spring ကနေရေးပေးထားတဲ့ DelegatingFilterProxy Class ကို Servlet Filter အနေဖြင့် Registration လုပ်ပေးရမှာ ဖြစ်ပါတယ်

Servlet Filters



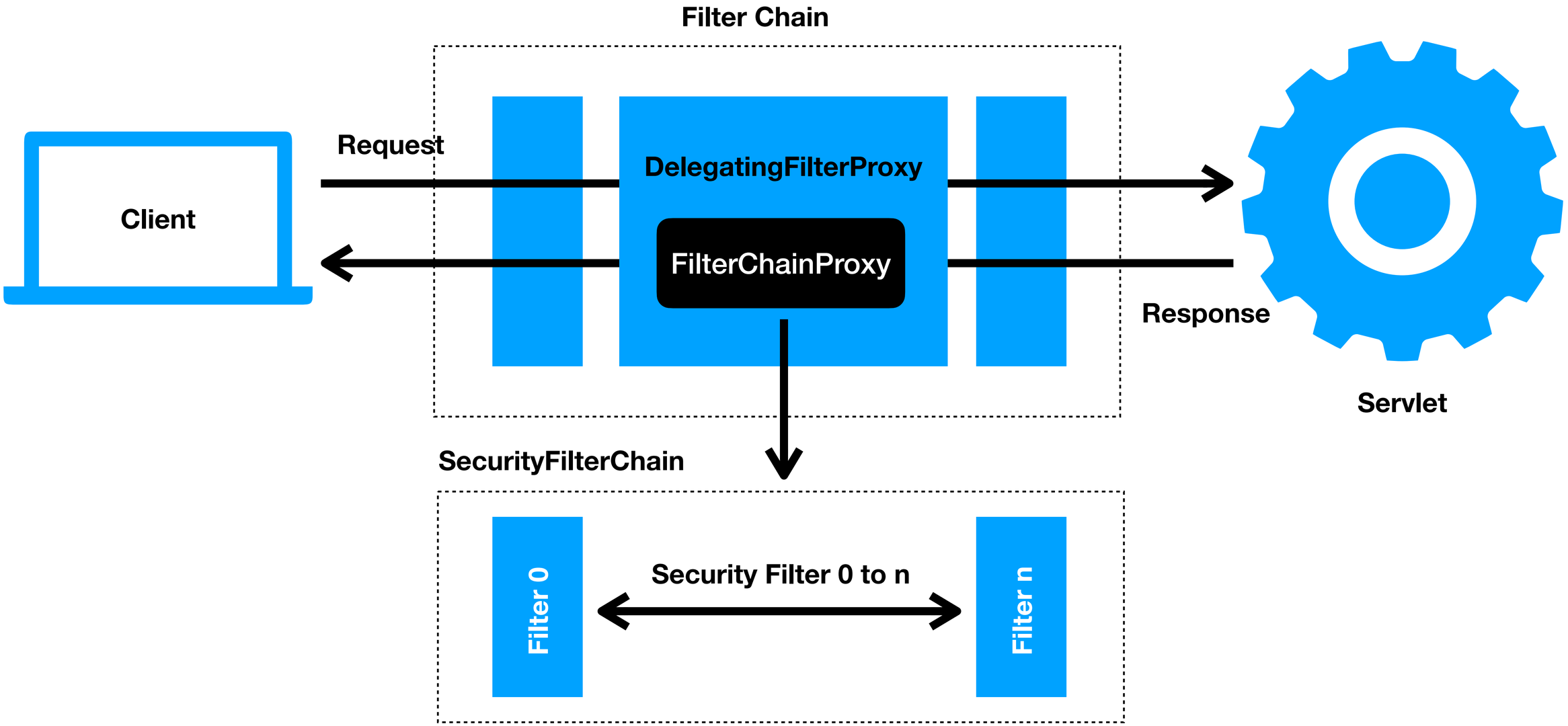
Servlet Filters

- Servlet Application တွေမှာ Web Filter တွေကို URL တွေနဲ့ Map လုပ်ပြီး သတ်မှတ် ရေးသား ထားနိုင်ပါတယ်

- Server ဆီကို Request တစ်ခုရောက်လာခဲ့ရင် URL အပေါ်မှာ Map လုပ်ထားတဲ့ Filter တွေကို Chain အဖြစ်တည်ဆောက်ပေးပြီး၊ Chain အတွင်းမှာပါရှိတဲ့ Filter တစ်ခုချင်းစီကို ဖြတ်ပြီး Target Resource ကို ရောက်ရှိစေတာ ဖြစ်ပါတယ်

- တဖန် Target Resource (Servlet) ကနေ ပြန်လိုက်တဲ့ Response ဟာလဲ Filter တွေကနေပဲ ဖြတ်ပြီး Client ဆီကို ရောက်ရှိစေတာ ဖြစ်ပါတယ်

Servlet Filters



DelegatingFilterProxy

- Spring Web Security ကို အသုံးပြုနိုင်ဖို့မ DelegatingFilterProxy Class ကို Servlet Filter အနေဖြင့် ပြင်ဆင်ပေးထားပါတယ်
- DelegatingFilterProxy ကို Servlet Filter အနေနှင့် Registration လုပ်ပြီးမှသာ Security ကို အသုံးပြုနိုင်မှာ ဖြစ်ပါတယ်
- DelegatingFilterProxy ဟာ Servlet Lifecycle အတွင်းမှာ အလုပ်လုပ်ပြီး၊ Spring ApplicationContext ကိုတော့ တိုက်ရိုက် Access မလုပ်နိုင်ပါဘူး
- DelegatingFilterProxy အတွင်းမှာပါတဲ့ FilterChainProxy ကနေတဆင့် Spring Manage Bean တွေကို Access လုပ်ပြီး Spring Security Mechanism ကို ဆောင်ရွက်စေတာပဲ ဖြစ်ပါတယ်

FilterChainProxy

- Spring MVC အတွက် Security ကို ဆောင်ရွက်ဖို့အတွက် Support တွေဟာ FilterChainProxy အတွင်းမှာ ပါဝင်ကြပါတယ်

- FilterChainProxy ဟာ Spring Security ကနေ ရေးသားပေးထားတဲ့ Special Filter တစ်မျိုးဖြစ်ပြီး၊ Spring Context အတွင်းမှာရှိတဲ့ SecurityFilterChain Beans တွေကို Access လုပ်ပေးနိုင်ပါတယ်

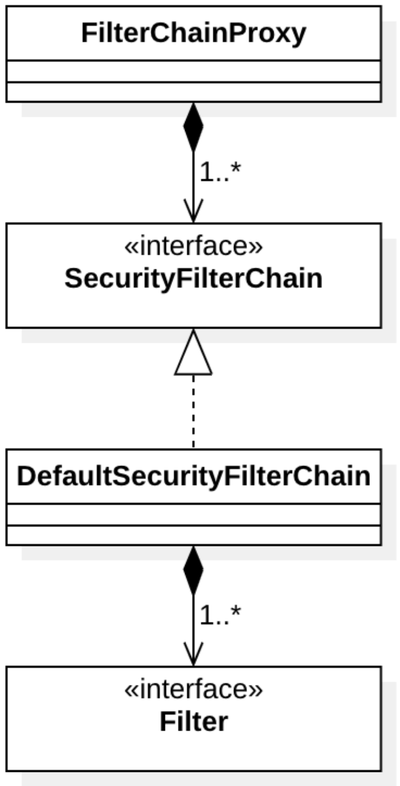
- ကျွန်တော်တို့ ရေးသားထားတဲ့ Security Configuration တွေအပေါ်မူတည်ပြီး တစ်ခုကို တစ်ခုထက်မကသော Security Filter များဖြင့် ဖွဲ့စည်းတည်ဆောက်ပေးနိုင်ပါတယ် SecurityFilterChain Bean တွေကို တည်ဆောက်ပေးနိုင်ပါတယ်

SecurityFilterChain

- SecurityFilterChain အတွင်းမှာ Current Request အတွက် အသုံးပြုဖို့လိုတဲ့ Spring Security Filter တွေပါဝင်ကြပါတယ်

- Application အတွင်းမှာ ရေးသားထားတဲ့ Security Configuration အပေါ်မူတည်ပြီး Spring Security Filter တွေကို Spring Bean အဖြစ်တည်ဆောက်ပေးပြီး SecurityFilterChain အတွင်းမှာ Chain အနေဖြင့် ထည့်သွင်းပေးလိုက်မှာ ဖြစ်ပါတယ်

Spring Web Security



- Spring Security ဟာ web security အတွက် FilterChainProxy ဖြင့် ဆောင်ရွက်ပေးပြီး သက်ဆိုင်ရာ Request URL Pattern အလိုက် Match ဖြစ်တဲ့ SecurityFilterChain ကို အသုံးပြုပေးနိုင်ပါတယ်

- Spring Security ဟာ Security Configuration ထဲမှာရှိတဲ့ SecurityFilterChain တွေကို စုစည်းပြီး FilterChainProxy Bean ကို တည်ဆောက်ပေးနိုင်ပါတယ်

- Web Application တစ်ခုမှာ တစ်ခုထက်မကသော SecurityFilterChain Object ကို သတ်မှတ် အသုံးပြုနိုင်ပါတယ်

Security Filters

- ForceEagerSessionCreationFilter
- ChannelProcessingFilter
- WebAsyncManagerIntegrationFilter
- SecurityContextPersistenceFilter
- HeaderWriterFilter
- CorsFilter
- CsrfFilter
- LogoutFilter
- OAuth2AuthorizationRequestRedirectFilter
- Saml2WebSsoAuthenticationRequestFilter
- X509AuthenticationFilter
- AbstractPreAuthenticatedProcessingFilter
- CasAuthenticationFilter
- OAuth2LoginAuthenticationFilter
- Saml2WebSsoAuthenticationFilter
- UsernamePasswordAuthenticationFilter

Security Filters

- DefaultLoginPageGeneratingFilter
- DefaultLogoutPageGeneratingFilter
- ConcurrentSessionFilter
- DigestAuthenticationFilter
- BearerTokenAuthenticationFilter
- BasicAuthenticationFilter
- RequestCacheAwareFilter
- SecurityContextHolderAwareRequestFilter
- JaasApiIntegrationFilter
- RememberMeAuthenticationFilter
- AnonymousAuthenticationFilter
- OAuth2AuthorizationCodeGrantFilter
- SessionManagementFilter
- ExceptionTranslationFilter
- FilterSecurityInterceptor / AuthorizationFilter
- SwitchUserFilter

HttpSecurity Configuration

- SecurityFilterChain Bean တစ်ခုချင်းစီအတွက် Configure လုပ်ပေးနိုင်တဲ့ Interface ဖြစ်ပါတယ်
- XML Configuration ရဲ့ <sec:http> Tag ဖြင့်လဲ သတ်မှတ် အသုံးပြုနိုင်ပါတယ်
- SecurityFilterChain မှာ သတ်မှတ်ထားတဲ့ URL Pattern နဲ့ ကိုက်ညီတဲ့ Request တွေမှာ ဆိုရင် အသုံးပြုပေးမှာ ဖြစ်တယ်
- တကယ်လို SecurityFilterChain မှာ URL Pattern သတ်မှတ်ထားခြင်းမရှိဘူးဆိုပါက Any Request အဖြစ် အသုံးပြုသွားမှာ ဖြစ်ပါတယ်
- HttpSecurity ဖြင့် SecurityFilterChain အတွင်းမှာပါဝင်မည့် Spring Security Filter တွေကို သတ်မှတ်ရေးသားသွားရမှာ ဖြစ်ပါတယ်

SecurityFilterChain - XML

```xml
<security:http pattern="/resources/**" security="none" />
<security:http pattern="/" security="none" />

<security:http>
    <security:intercept-url pattern="/admin/**" access="hasRole('ADMIN')" />
    <security:intercept-url pattern="/member/**" access="hasAnyRole('ADMIN', 'MEMBER')" />

    <security:http-basic/>
</security:http>
```

SecurityFilterChain - Java

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

    @Bean
    WebSecurityCustomizer webSecurityCustomizer() {
        return web -> web.ignoring().requestMatchers("/resources/**", "/");
    }

    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(registry -> registry
                .requestMatchers("/admin/**").hasRole("ADMIN")
                .requestMatchers("/member/**").hasRole("MEMBER"));

        http.formLogin(Customizer.withDefaults());

        http.logout(logout -> logout.logoutSuccessUrl("/"));

        return http.build();
    }
}
```

HttpSecurity Configurations

Configuration	Description
Authorized Request	Current HttpServletRequest အတွက် Authorization Configurations
Login Configuration	Form Login, HTTP Basic Login, Remember Me Login, OAuth2 Login, SAML Login Configurations
Logout Configuration	Logout လုပ်မည့် Mechanism နှင့်ပတ်သက်သော Configurations
Custom Filters	Custom Filter များကို ရေးသားအသုံးပြုလိုတဲ့အခါ ရေးသားရန်လိုအပ်ပါတယ်
Others Configurations	အခြားသော Security Filter တွေကို ဖြည့်စွက်ဖို့အတွက် အသုံးပြုနိုင်ပါတယ်