

Webservices beveiligen

Veerle Ongenae



Webservice beveiligen

- Authenticatie: gebruiker herkennen (bv. login – wachtwoord)
 - PasswordEncoder: wachtwoord veilig bewaren
- Autorisatie: rechten instellen
- CSRF: Cross Site Request Forgery
- HTTPS gebruiken



Authenticatie

- Gebruikers configureren

Configuratiebestand

Standaardbeveiliging uitgeschakeld en vervangen

@Configuration

@EnableWebSecurity

```
public class SecurityConfig {
```

Methode die bean levert → gebruikt bij Dependency Injection

@Bean

Encrypteren wachtwoorden

```
public UserDetailsService userDetailsService(BCryptPasswordEncoder bCryptPasswordEncoder) {  
    InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();
```

```
    manager.createUser(User.withUsername("user")  
        .password(bCryptPasswordEncoder.encode("userPass"))  
        .roles("USER")  
        .build());
```

Gebruikers aanmaken en aanpassen
in het geheugen → testen

Interface om op basis van loginnaam gebruikersgegevens op
te halen

```
    manager.createUser(User.withUsername("admin")  
        .password(bCryptPasswordEncoder.encode("adminPass"))  
        .roles("USER", "ADMIN")  
        .build());  
    return manager; }  
...  
}
```



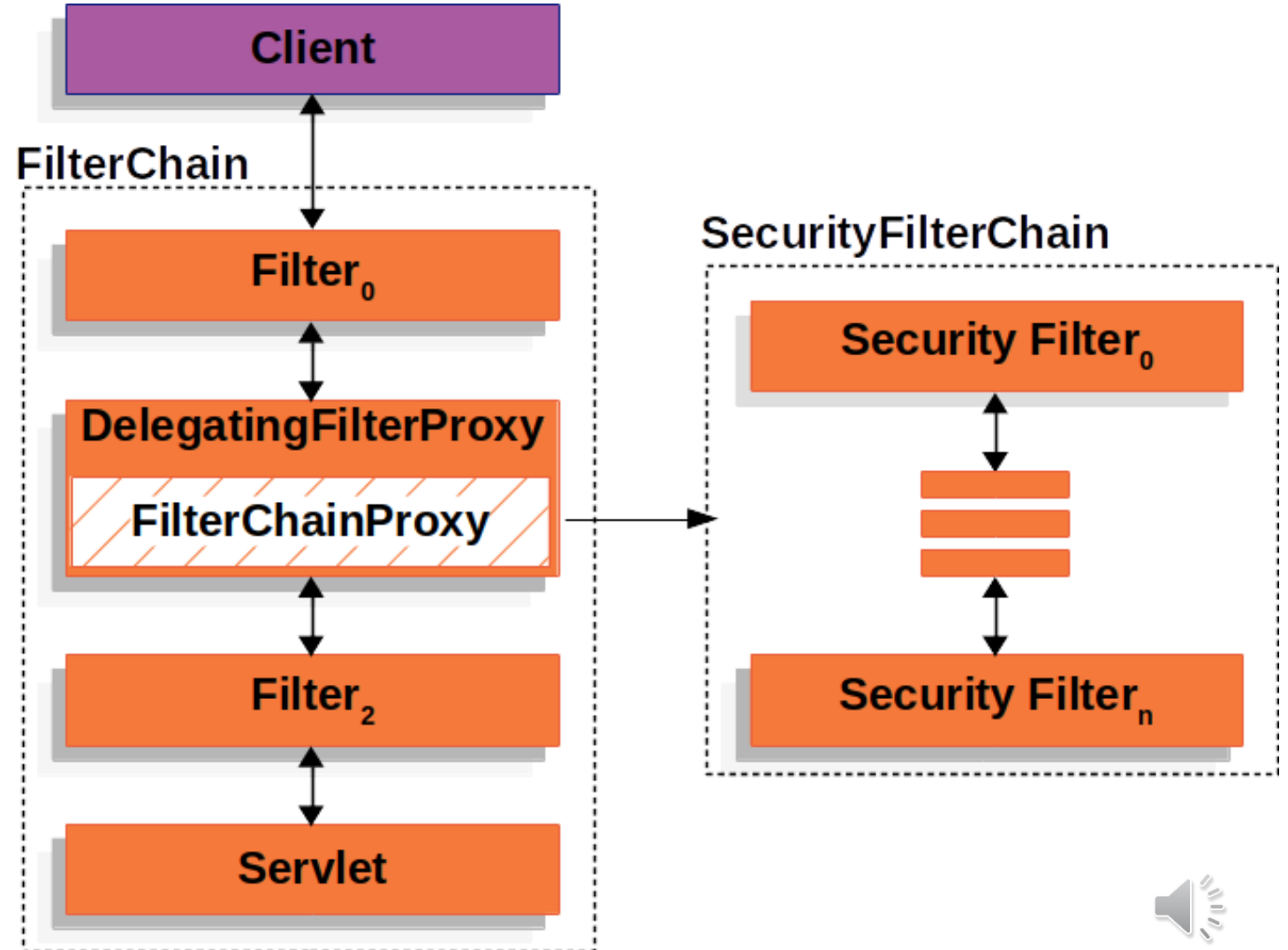
Webservice beveiligen

- Authenticatie: gebruiker herkennen (bv. login – wachtwoord)
 - PasswordEncoder: wachtwoord veilig bewaren
- Autorisatie: rechten instellen
- CSRF: Cross Site Request Forgery
- HTTPS gebruiken



Autorisatie

- Rechten bepalen
- Toegang beperken



bron: <https://docs.spring.io/spring-security/reference/servlet/architecture.html>

Autorisatie

- Beveiliging configureren

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

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


Bean voor reeks beveiligingsfilters

Configuratie beveiliging HTTP-berichten

Beperken toegang

Basis HTTP-authenticatie via headers

SecurityFilterChain maken



Autorisatie – paden

- Beveiliging configureren – op basis van pad

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

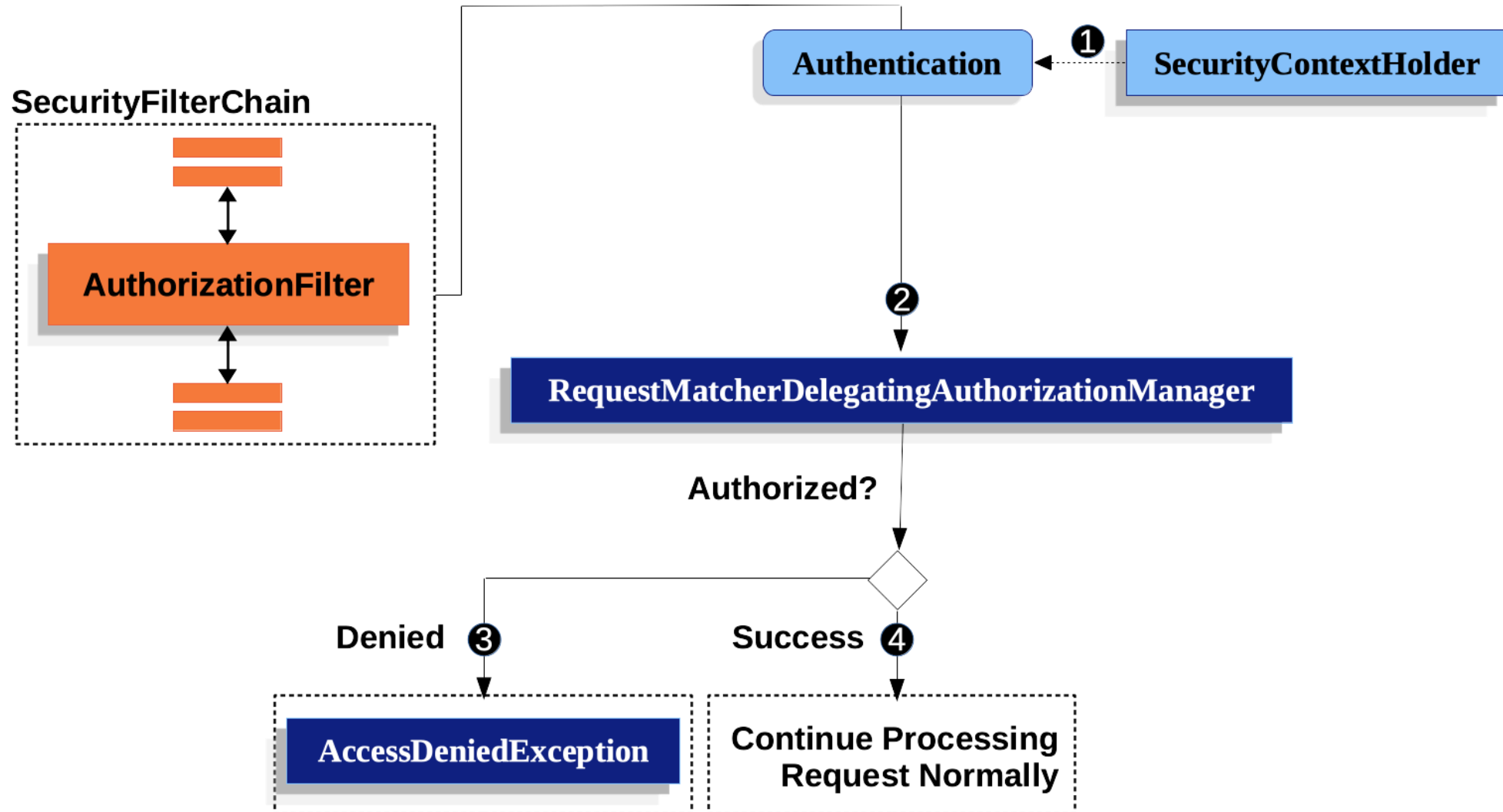
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((authz) -> authz
                .requestMatchers("/api/admin/**").hasRole("ADMIN")
                .requestMatchers("/api/user/**").hasRole("USER")
                .anyRequest().authenticated()
            );
        return http.build();
    }
    ...
}
```

Pad naar REST-service

Gewenste rol: wie heeft toegang?



Autorisatie - principe



Autorisatie – methodes webservice

- Beveiliging configureren – methodes webservice

```
@Configuration
@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true, jsr250Enabled = true)
public class SecurityConfig {

    ...

}
```

Beveiliging op methodes

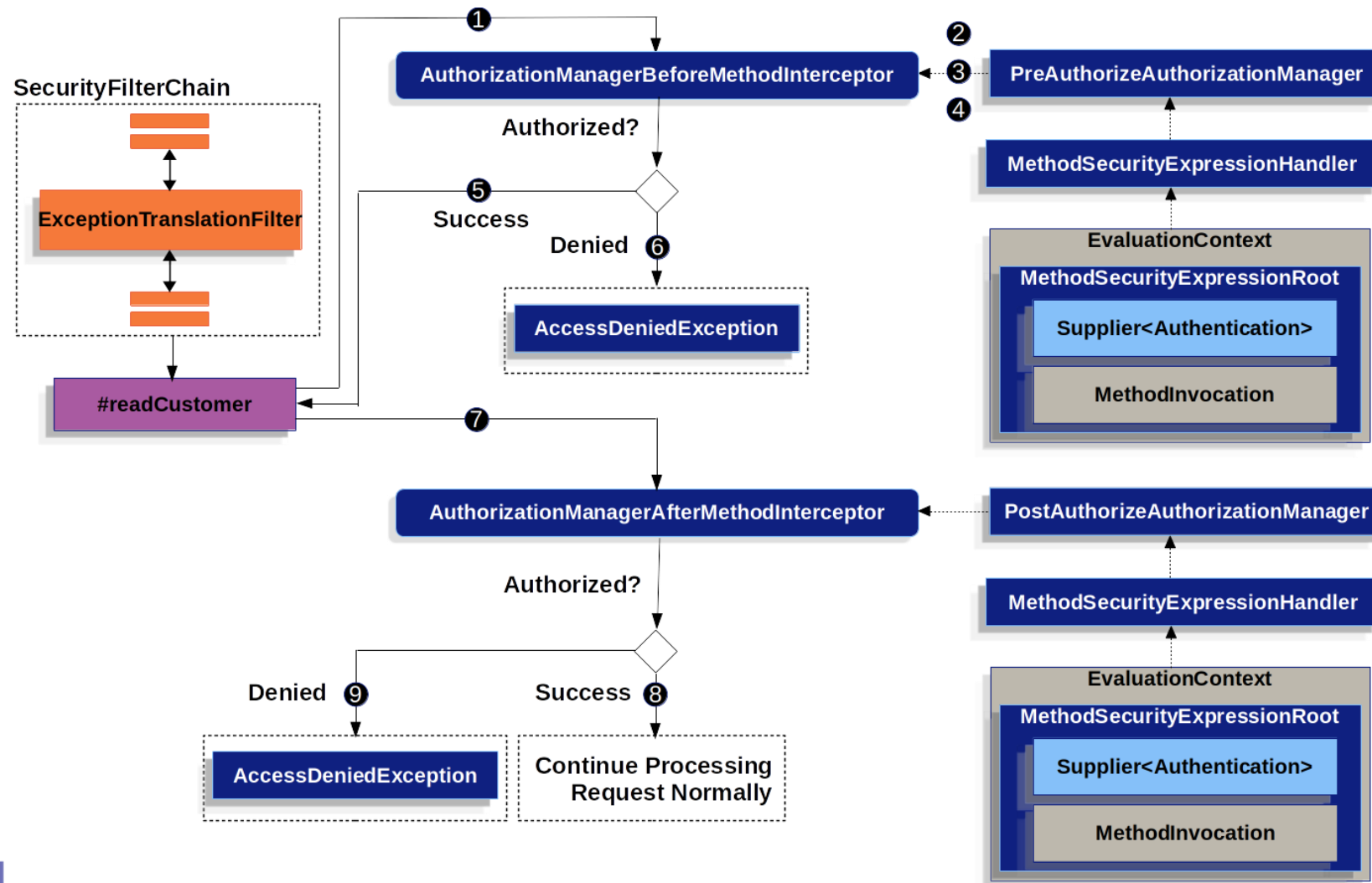
Voor- of na uitvoeren methode

Beveiliging op basis van rollen

Beveiliging op basis van
specifieke Java-annotaties



Autorisatie op methodes: principe



Autorisatie – op methodes op basis van rollen

- Beveiliging configureren – methodes webservice - rollen

```
@Secured({ "ROLE_VIEWER", "ROLE_EDITOR" })  
public void deleteUser(...) {  
    ...  
}
```



Autorisatie – op methodes op basis van rollen (JSR 250)

- Beveiliging configureren – methodes webservice - rollen

```
@RolesAllowed("ROLE_VIEWER")
public String getUsername2() {
    ...
}

@RolesAllowed({ "ROLE_VIEWER", "ROLE_EDITOR" })
public boolean isValidUsername2(String username) {
    ...
}
```



Autorisatie – voor of na uitvoeren methode

- Beveiliging configureren – conditie voor of na uitvoeren methode

```
@PreAuthorize("hasRole('ROLE_VIEWER')")
public String getUsernameInUpperCase() {
    ...
}

@PreAuthorize("#username == authentication.principal.username")
public String getMyRoles(String username) {
    ...
}
```

Voor uitvoeren methode

Uitdrukking in Spring Expression Language



Autorisatie op requests versus methodes

	Op request/bericht	Op methode
Type autorisatie	Grofmazig	Fijnmazig
Plaats configuratie	In configuratieklasse	Op een specifieke methode
Type configuratie	Met lambda-uitdrukkingen	Met annotaties
Definitie autorisatie	In code	Spring Expression Language

bron: <https://docs.spring.io/spring-security/reference/servlet/authorization/method-security.html>



Webservice beveiligen

- Authenticatie: gebruiker herkennen (bv. login – wachtwoord)
 - PasswordEncoder: wachtwoord veilig bewaren
- Autorisatie: rechten instellen
- CSRF: Cross Site Request Forgery
- HTTPS gebruiken



Wat is CSRF?

Website bank

```
<form method="post"
  action="/transfer">
<input type="text"
  name="amount"/>
<input type="text"
  name="routingNumber"/>
<input type="text"
  name="account"/>
<input type="submit"
  value="Transfer"/>
</form>
```

```
POST /transfer HTTP/1.1
Host: bank.example.com
Cookie: JSESSIONID=randomid
Content-Type: application/x-www-form-urlencoded
```

```
amount=100.00&routingNumber=1234&account=9876
```

Hackers site

```
<form method="post"
  action="https://bank.example.com/transfer">
<input type="hidden"
  name="amount"
  value="100.00"/>
<input type="hidden"
  name="routingNumber"
  value="evilsRoutingNumber"/>
<input type="hidden"
  name="account"
  value="evilsAccountNumber"/>
<input type="submit"
  value="Win Money!"/>
</form>
```



Beveiliging tegen CSRF

- Hoe onderscheid maken tussen beide HTTP-aanvragen? (bank site vs hackers site)
- Synchronizer Token Pattern
 - Naast cookie ook een willekeurig token in HTTP-request

```
<form method="post"
  action="/transfer">
<input type="hidden"
  name="_csrf"
  value="4bfd1575-3ad1-4d21-96c7-4ef2d9f86721"/>
<input type="text"
  name="amount"/>
<input type="text"
  name="routingNumber"/>
<input type="hidden"
  name="account"/>
<input type="submit"
  value="Transfer"/>
</form>
```

```
POST /transfer HTTP/1.1
Host: bank.example.com
Cookie: JSESSIONID=randomid
Content-Type: application/x-www-form-urlencoded

amount=100.00&routingNumber=1234&account=9876&_csrf=4bfd1575-3ad1-4d21-96c7-4ef2d9f86721
```

Beveiliging tegen CSRF

- Hoe onderscheid maken tussen beide HTTP-aanvragen? (bank site vs hackers site)
- Synchronizer Token Pattern
- SameSite-attribuut op cookie

```
Set-Cookie: JSESSIONID=randomid; Domain=bank.example.com; Secure; HttpOnly; SameSite=Lax
```




CSRF in Spring Security


- Standaard aan

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            // ...
            .csrf((csrf) -> csrf
                .ignoringRequestMatchers("/api/*")
            );
        return http.build();
    }
    ...
}
```



Uitschakelen CSRF voor bepaalde webservices



Webservice beveiligen

- Authenticatie: gebruiker herkennen (bv. login – wachtwoord)
 - PasswordEncoder: wachtwoord veilig bewaren
- Autorisatie: rechten instellen
- CSRF: Cross Site Request Forgery
- HTTPS gebruiken

