05. Spring Security

## Introductie tot Spring Security

Veiligheid is een van de belangrijkste aspecten van webapplicaties en Spring Security biedt een uitgebreide oplossing voor het beveiligen van webapplicaties. Spring Security is een krachtige en flexibele module die authenticatie en autorisatie biedt voor webapplicaties die zijn gebouwd met het Spring Framework.

In deze tutorial gaan we leren hoe we Spring Security kunnen gebruiken om onze webapplicaties te beveiligen. We gaan de basisbegrippen van Spring Security behandelen en hoe we deze kunnen configureren in onze webapplicatie.

## Voordelen van Spring Security

Met Spring Security kun je verschillende beveiligingsmechanismen toevoegen aan je webapplicatie, zoals authenticatie, autorisatie, HTTPS-ondersteuning, bescherming tegen Cross-Site Request Forgery (CSRF), Single Sign-On (SSO) en meer. Het biedt ook integratie met andere populaire frameworks en bibliotheken, zoals Spring MVC, Spring Boot, OAuth2 en OpenID Connect.

## Basisbegrippen van Spring Security

Voordat we beginnen met het configureren van Spring Security, moeten we enkele basisbegrippen begrijpen. Hieronder vind je enkele belangrijke begrippen die we gaan behandelen in deze tutorial:

* **Authentication (Authenticatie)**: Het proces van het verifiëren van de identiteit van de gebruiker.
* **Authorization (Autorisatie)**: Het proces van het verlenen van toegang tot specifieke delen van de applicatie op basis van de rol of rechten van de gebruiker.
* **Principal**: De entiteit (gebruiker) die probeert toegang te krijgen tot de applicatie.
* **Role**: Een verzameling autorisaties die aan een gebruiker zijn toegewezen.
* **Granted Authority**: Een specifieke autorisatie die aan een gebruiker is toegewezen.
* **Authentication Provider**: Een interface die wordt gebruikt om de authenticatie van een gebruiker uit te voeren.

## Configuratie van Spring Security

Nu we de basisbegrippen van Spring Security hebben behandeld, kunnen we doorgaan met de configuratie. Spring Security wordt geconfigureerd door een aantal beveiligingsfilters toe te voegen aan de Servlet-filterketen. Deze filters worden geconfigureerd in een Java-configuratieklasse of XML-configuratiebestand.

In deze tutorial gaan we de configuratie van Spring Security behandelen met behulp van een Java-configuratieklasse. Hieronder vind je de stappen die we gaan volgen:

1. Voeg de Spring Security-afhankelijkheid toe aan je project.
2. Configureer de beveiligingsfilters en -regels in een Java-configuratieklasse.
3. Schakel Spring Security in voor je webapplicatie.
4. Implementeer gebruikersauthenticatie en autorisatie.
5. Beveilig de endpoints van je applicatie.

Stap 1: Voeg de Spring Security-afhankelijkheid toe aan je project.

Om Spring Security te kunnen gebruiken, moeten we de juiste afhankelijkheden toevoegen aan ons project. Dit kan worden gedaan door de volgende afhankelijkheid toe te voegen aan het pom.xml-bestand:

<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-web</artifactId>  
 <version>${spring-security.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-config</artifactId>  
 <version>${spring-security.version}</version>  
</dependency>  
Copy

De ${spring-security.version} variabele moet worden vervangen door de versie van Spring Security die je wilt gebruiken.

Stap 2: Configureer de beveiligingsfilters en -regels in een Java-configuratieklasse.

Nu we de Spring Security-afhankelijkheid hebben toegevoegd aan ons project, kunnen we beginnen met het configureren van de beveiligingsfilters en -regels in een Java-configuratieklasse. Dit kan gedaan worden door een klasse te maken die de interface WebSecurityConfigurerAdapter implementeert.

@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
   
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers("/", "/home").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll();  
 }  
   
 @Autowired  
 public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
 auth  
 .inMemoryAuthentication()  
 .withUser("user").password("password").roles("USER");  
 }  
}  
Copy

In dit voorbeeld configureren we de beveiligingsfilters en -regels met behulp van de configure(HttpSecurity http)-methode. We geven aan dat alle verzoeken die naar de server worden gestuurd, geautoriseerd moeten worden, behalve de verzoeken naar de startpagina en de homepagina. De inlogpagina wordt toegestaan en uitloggen is ook toegestaan.

Stap 3: Schakel Spring Security in voor je webapplicatie.

Om Spring Security in te schakelen voor je webapplicatie, moeten we een klasse maken die de interface WebApplicationInitializer implementeert.

public class SecurityWebApplicationInitializer extends AbstractSecurityWebApplicationInitializer {  
   
}  
Copy

Door deze klasse aan ons project toe te voegen, wordt Spring Security ingeschakeld voor onze webapplicatie.

Stap 4: Implementeer gebruikersauthenticatie en autorisatie.

Om gebruikersauthenticatie en autorisatie te implementeren, moeten we de configureGlobal(AuthenticationManagerBuilder auth)-methode implementeren in onze SecurityConfig-klasse. In deze methode definiëren we de gebruikers die toegang hebben tot onze webapplicatie en hun bijbehorende rollen. In dit voorbeeld hebben we één gebruiker met de rol "USER".

@Autowired  
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
  
}  
Copy

Om gebruikers en rollen toe te voegen aan de beveiligingsconfiguratie, moeten we een gebruikersservice implementeren die gebruikersinformatie en -rollen ophaalt uit een database of andere gegevensbron. Deze service wordt vervolgens geconfigureerd in de beveiligingsconfiguratieklasse, samen met de rollen en rechten van de gebruikers.

1. Beveilig de endpoints van je applicatie.

Nadat de beveiligingsconfiguratie is geconfigureerd en de gebruikers- en rolgegevens zijn opgehaald, moeten we de endpoints van de applicatie beveiligen met de juiste autorisatieregels. Dit kan worden gedaan met behulp van annotaties op de controller- of serviceklassen van de applicatie.

## Conclusie

Met deze stappen kun je Spring Security configureren en gebruiken om je webapplicaties te beveiligen tegen ongeautoriseerde toegang en aanvallen.