



ChatGPT 3.5

You
como implemento JWT con Spring

ChatGPT

La implementación de JSON Web Tokens (JWT) en una aplicación Spring es bastante común y puede lograrse de varias maneras. A continuación, te proporcionaré una guía básica de cómo puedes implementar JWT en una aplicación Spring usando la biblioteca Spring Security.

1. Agregar dependencias Maven/Gradle:

Asegúrate de tener las dependencias necesarias en tu archivo `pom.xml` (si estás utilizando Maven) o `build.gradle` (si estás utilizando Gradle):

Maven:

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

<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-api</artifactId>
  <version>0.11.2</version>
</dependency>
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-impl</artifactId>
  <version>0.11.2</version>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-jackson</artifactId>
  <version>0.11.2</version>
  <scope>runtime</scope>
</dependency>
```

Gradle:

```
gradle
implementation 'org.springframework.boot:spring-boot-starter-security'
implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
implementation 'io.jsonwebtoken:jjwt-impl:0.11.2'
implementation 'io.jsonwebtoken:jjwt-jackson:0.11.2'
```

2. Configurar Spring Security con JWT:

Crea una clase que extienda `WebSecurityConfigurerAdapter` y anótala con `@EnableWebSecurity`. Sobrescribe el método `configure` para configurar la seguridad y agregar el filtro de JWT:

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

    @Autowired
    private JwtTokenProvider jwtTokenProvider;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // Desactivar la protección CSRF (puedes activarla si es necesario)
        http.csrf().disable();

        // Configurar la autorización
        http.authorizeRequests()
            .antMatchers("/api/public/**").permitAll() // Rutas públicas
            .anyRequest().authenticated();

        // Configurar el filtro de JWT
        http.apply(new JwtTokenFilterConfigurer(jwtTokenProvider));
    }
}
```

3. Crear el proveedor de tokens JWT:

Crea una clase que implemente `JwtTokenProvider` y proporcione métodos para generar y validar tokens JWT:

```
java
public interface JwtTokenProvider {
    String generateToken(Authentication authentication);

    Authentication getAuthentication(String token);

    boolean validateToken(String token);
}
```

4. Implementar el filtro de JWT:

Crea una clase para el filtro que extraiga y valide el token JWT de las solicitudes:

```
java
```

```

public class JwtTokenFilter extends OncePerRequestFilter {

    private JwtTokenProvider jwtTokenProvider;

    public JwtTokenFilter(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Override
    protected void doFilterInternal(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain filterChain
    ) throws ServletException, IOException {
        String token = jwtTokenProvider.resolveToken(request);
        if (token != null && jwtTokenProvider.validateToken(token)) {
            Authentication auth = jwtTokenProvider.getAuthentication(token);
            if (auth != null) {
                SecurityContextHolder.getContext().setAuthentication(auth);
            }
        }
        filterChain.doFilter(request, response);
    }
}

```

5. Configurar el filtro en la aplicación:

Crea una clase de configuración para registrar el filtro:

```

java
public class JwtTokenFilterConfigurer extends SecurityConfigurerAdapter {

    private JwtTokenProvider jwtTokenProvider;

    public JwtTokenFilterConfigurer(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Override
    public void configure(HttpSecurity http) {
        JwtTokenFilter customFilter = new JwtTokenFilter(jwtTokenProvider);
        http.addFilterBefore(customFilter, UsernamePasswordAuthenticationFilter);
    }
}

```

Con estas configuraciones básicas, deberías tener una implementación funcional de JWT en tu aplicación Spring. Asegúrate de ajustar las rutas, configuraciones y detalles específicos según las necesidades de tu aplicación.

📄 📌 🗑️ ↻

Message ChatGPT...



ChatGPT can make mistakes. Consider checking important information.

