Java - Integração Segura do Keycloak com Spring Boot

Keycloak é uma solução open-source de IAM (Identity and Access Management) amplamente utilizada para autenticação, autorização e gerenciamento de identidades em aplicações modernas.

Neste guia, você aprenderá a integrar o Keycloak com o Spring Boot, passo a passo, com boas práticas, códigos comentados e relação com versões LTS do Java.

1. Requisitos

- **Java 17 (LTS)**
- **Spring Boot 3.x**
- **Keycloak 22 ou superior**
- **Maven ou Gradle**

Recomendação: Java 17 é a versão LTS mais estável e compatível com as versões recentes do Spring Boot.

2. Instalação e Execução do Keycloak

2.1 Subindo o Keycloak com Docker

docker run -p 8080:8080 quay.io/keycloak/keycloak:22.0.1 start-dev

2.2 Configuração Inicial

- Acesse: http://localhost:8080
- Crie um **Realm**: fastcode-realm
- Crie um **Client**:
  - Client ID: springboot-client
  - Client Type: confidential
  - Root URL: http://localhost:8081
- Crie um **usuário**: user1 com senha senha123

Boas Práticas: Não exponha client_secret no código-fonte. Use variáveis de ambiente ou ferramentas como Spring Vault.]

3. Criando o Projeto Spring Boot

3.1 Dependências no Maven

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

3.2 Configuração application.yml

```
spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: http://localhost:8080/realms/fastcode-realm
```

Explicação: issuer-uri informa ao Spring onde obter as chaves públicas para validação de JWTs emitidos pelo Keycloak.

4. Protegendo Endpoints

4.1 Criando o Controller

```
@RestController
public class HelloController {

  @GetMapping("/public")
  public String publico() {
    return "Acesso livre";
  }

  @GetMapping("/private")
  @PreAuthorize("hasRole('user')")
  public String privado() {
    return "Acesso restrito ao papel 'user'";
  }
}
```

Nota: Para que @PreAuthorize funcione, ative com @EnableMethodSecurity na configuração de segurança.

4.2 Configurando a Segurança

```
@Configuration
@EnableMethodSecurity
public class SecurityConfig {

  @Bean
  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
      .authorizeHttpRequests(auth -> auth
        .requestMatchers("/public").permitAll()
        .anyRequest().authenticated()
      )
      .oauth2ResourceServer(OAuth2ResourceServerConfigurer::jwt);
    return http.build();
  }
}
```

Boas Práticas: Centralize a configuração de segurança e mantenha clara a distinção entre acessos públicos e privados.

5. Testando com JWT Token

5.1 Obtendo o Token via cURL

```
curl -X POST \
  http://localhost:8080/realms/fastcode-realm/protocol/openid-connect/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d
"username=user1&password=senha123&grant_type=password&client_id=springboot-client&client_secret=SEU_SECRET"
```

5.2 Chamando Endpoint Protegido

curl -H "Authorization: Bearer SEU_TOKEN" http://localhost:8081/private

✅ **Dica:** Use ferramentas como Postman ou Insomnia para facilitar testes com autenticação.

6. Comparativo: Java LTS e Spring Boot

Java Version	Spring Boot Suporte	Comentário
Java 8	Spring Boot 2.x	Legado, sem suporte a novas APIs
Java 11	Spring Boot 2.x/3.x	Estável, mas com APIs limitadas
Java 17	Spring Boot 3.x	Recomendado para projetos atuais
Java 21	Spring Boot 3.3+	Alta performance, uso crescente

Recomendação: Prefira Java 17 ou Java 21 para novos projetos. Ambos são LTS e possuem suporte prolongado.

Dica EducaCiência FastCode

Ao longo deste tutorial, você aprendeu:

- *Como executar o Keycloak localmente com Docker*
- *Configurar Realms, Clients e Usuários*
- *Proteger endpoints REST com Spring Boot*
- *Validar tokens JWT emitidos pelo Keycloak*
- *Aplicar boas práticas de segurança*
- *Escolher a versão correta do Java LTS para seu projeto*

A integração do Spring Boot com o Keycloak traz um modelo robusto e escalável de controle de acesso, alinhado às melhores práticas de arquitetura moderna. Essa abordagem favorece a segurança centralizada, facilita integrações com outros sistemas e reduz a complexidade do código de autenticação.

Mantenha sempre seus ambientes e dependências atualizados, invista em testes de integração com autenticação e monitore os logs de auditoria do Keycloak.

Referências Recomendadas

- *https://www.keycloak.org/documentation*
- *Spring Security - Documentação Oficial*
- *RFC 6749 - https://oauth.net/2/*
- *https://auth0.com/resources/ebooks/jwt-handbook-pt?utm_source=google&utm_campaign=amer_latam-pt_bra_all_ciam-all_dg-ao_auth0_search_google_text_kw_pt-generic-authentication_utm2&utm_medium=cpc&utm_id=aNK4z000000UCviGAG&gad_source=1&gad_campaignid=12973797247&gclid=Cj0KCQjwlrvBBhDnARIsAHEQgORWBBh4UcgZmouRwnEsoqNEv-nXp-W42tltrZxSb-Gh8iOXJADfIncaAuBOEALw_wcB*
- *Roadmap Java LTS - Oracle*

EducaCiência FastCode para a comunidade