Fine grained RBAC w/ Resource Server

Keycloak, JWT 기반 OAuth2 책임의 분산 (3/3) - 마이크로서비스와 OAuth2 구성요소인 Authorization Server/ Client/ Resource Server 를 활용해 Single Sign-On 구현 모형을 실습한다. OAuth2 에서 Client(Gateway)와 Resource Server(마이크로서비스)가 분리된 환경으로 Authorization Server 로는 Keycloak 을 활용한다.

Instruction

Keycloak, JWT 기반 OAuth2 - Resource Server

OAuth2 Resource Server 설정

- 주문 마이크로서비스를 Resource Server 로 설정한다.
- Gateway 로부터 JWT Token 을 전달받아 클레임에 포함된 User Role 기반의 Fine grained 한 ACL 을 적용한다.
- Platform 에서 작업이 원활히지 않을 경우, Local 에서 수행한다.

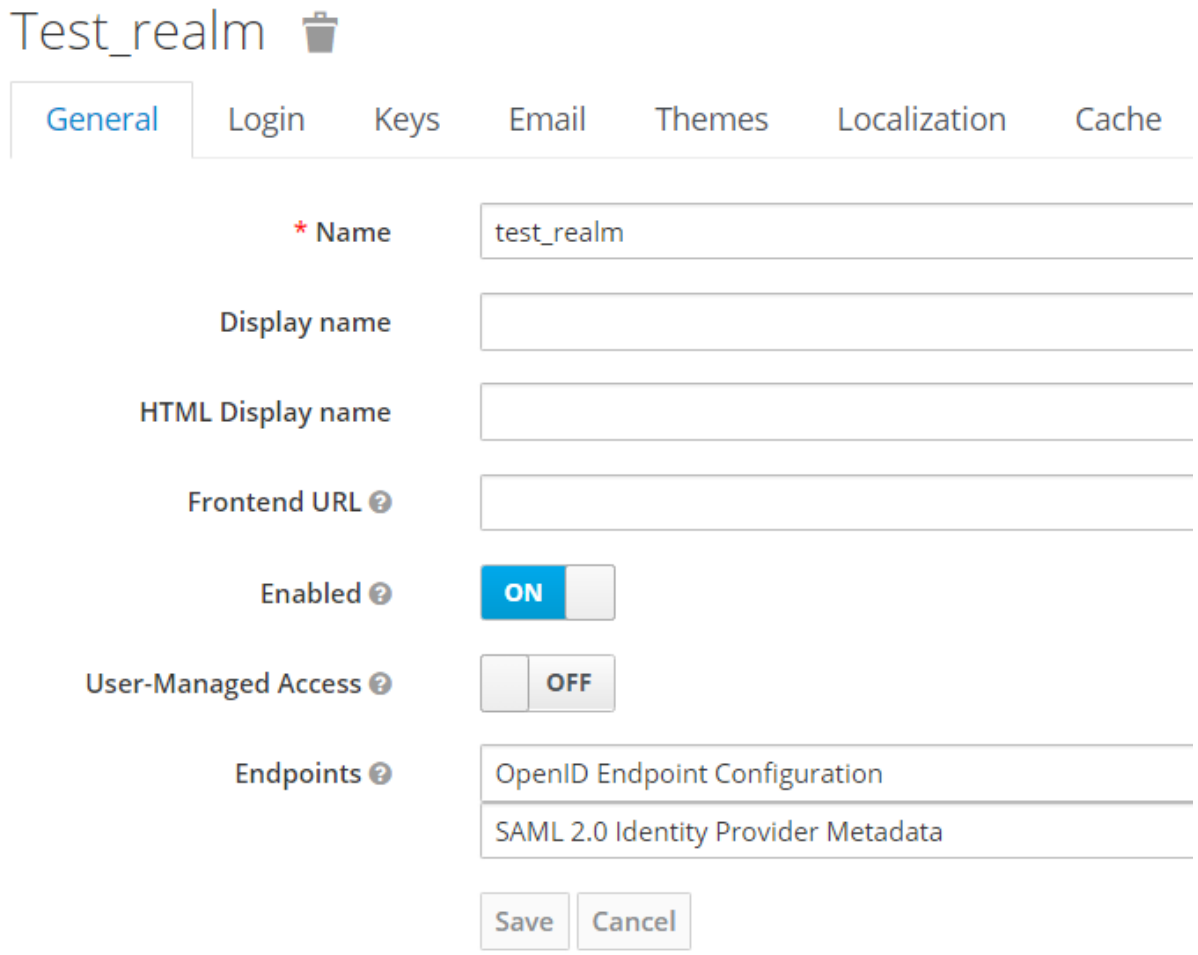
Local 머신에 IDE(IntelliJ, VSCode)와 JDK 11 이상이 설치되어 있어야 한다.

OAuth2 Resource Server 설정

- Order 마이크로서비스의 pom.xml 을 통해 Resource Server 설정에 필요한 라이브러리(oauth2-resource-server)를 확인한다.
- application.yml 에 oauth2-resource-server 설정을 주입한다.
- application.yml 을 열어 주석 부분을 해제한다.

```
#  security:
#    oauth2:
#      resourceserver:
#        jwt:
#          jwk-set-uri:
http://localhost:8080/realms/my_realm/protocol/openid-connect/certs
```

12 행과 같이 keycloak Endpoint 정보는 Realm 에서 OIDC 링크를 통해 확인 가능하다.



Resource Server Security Configuration

- **security 패키지의 ResourceSecurityConfig.java 파일을 열어 기본 설정을 확인한다.**
- **메소드 레벨의 Spring security 를 적용하기 위해 @EnableGlobalMethodSecurity 을 설정한다.**
- **Gateway 로부터 전달받은 JWT 토큰으로부터 Claim 을 추출하여 Spring security 에 Injection 한다.**

메소드 레벨 **Fine grained Role** 적용

- **Order 서비스의 Controller.java 를 열어 Role 설정을 확인한다.**
- **javax.annotation.security.RolesAllowed 를 활용해 메소드 레벨 ACL 을 적용한다.**
- **Spring Security 의 @Secured 로도 제어 가능하다.**

Keycloak > JWT Claim > Spring Security 로 전달되었다.

Order 서비스 메소드 레벨 **ACL** 테스트

- 브라우저로 http://localhost:8088 에 접속한다.
- 접속 후, 인증한 User 의 Role 에 따른 응답을 확인해 본다.

```
http://localhost:8088/orders
http://localhost:8088/orders/placeAnOrder
http://localhost:8088/orders/orderManage
```

Service Clear

- 다음 Lab 을 위해 기동된 모든 서비스 종료

```
fuser -k 8080/tcp
fuser -k 8081/tcp
fuser -k 8088/tcp
```

CheckPoints

1. 모든 요구사항을 만족하는가