**Keycloak, JWT기반 OAuth2 - Resource Server**

**OAuth2 Resource Server 설정**

* 주문 마이크로서비스를 Resource Server로 설정한다.
* Gateway로부터 JWT Token을 전달받아 클레임에 포함된 User Role기반의 Fine grained한 ACL을 적용한다.
* Platform에서 작업이 원활히지 않을 경우, Local에서 수행한다.

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

**OAuth2 Resource Server 설정**

* Order 마이크로서비스의 pom.xml을 통해 Resource Server 설정에 필요한 라이브러리(oauth2-resource-server)를 확인한다.
* application.yml에 oauth2-resource-server 설정을 주입한다.
* application.yml을 열어 주석 부분을 해제한다.

**# security:**

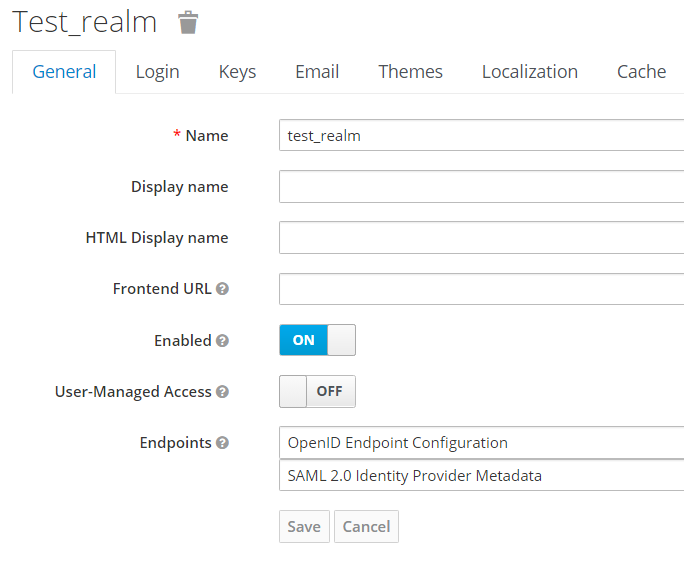
**# oauth2:**

**# resourceserver:**

**# jwt:**

**# jwk-set-uri: http://localhost:8080/realms/my\_realm/protocol/openid-connect/certs**

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



**Resource Server Security Configuration**

* security 패키지의 ResourceSecurityConfig.java 파일을 열어 기본 설정을 확인한다.
* 메소드 레벨의 Spring security를 적용하기 위해 @EnableGlobalMethodSecurity을 설정한다.
* Gateway로부터 전달받은 JWT 토큰으로부터 Claim을 추출하여 Spring security에 Injection한다.

**메소드 레벨 Fine grained Role 적용**

* Order 서비스의 Controller.java를 열어 Role 설정을 확인한다.
* javax.annotation.security.RolesAllowed 를 활용해 메소드 레벨 ACL을 적용한다.
* Spring Security의 @Secured 로도 제어 가능하다.

Keycloak > JWT Claim > Spring Security로 전달되었다.

**Order 서비스 메소드 레벨 ACL 테스트**

* 브라우저로 http://localhost:8088에 접속한다.
* 접속 후, 인증한 User의 Role에 따른 응답을 확인해 본다.

**http://localhost:8088/orders**

**http://localhost:8088/orders/placeAnOrder**

**http://localhost:8088/orders/orderManage**

**Service Clear**

* 다음 Lab을 위해 기동된 모든 서비스 종료

**fuser -k 8080/tcp**

**fuser -k 8081/tcp**

**fuser -k 8088/tcp**

Checkpoints

1. 모든 요구사항을 만족하는가

Hints