[구현] JWT 토큰을 사용한 인증처리

발급된 JWT 토큰을 Gateway 에서 Authorization 하시오.

10 분 이내 완료

INTO THE LAB

Instruction

Gateway 에서의 JWT 토큰기반 인증

인증,인가에 대한 자세한 내용은 **MSASchool** 에서 확인할 수 있다.

http://msaschool.io/operation/design/design-seven/

- **Spring Security** 와 **Spring oauth2** 를 사용해 **Resource Owner, Client, Authorization Server, Resource Server** 간의 인증/인가를 실습한다.
- 여기서 **Resouce** 란 **Gateway** 를 경유하는 **Rest APIs** 들이며, **Gateway** 가 **Client** 이자 **Resource Server** 역할을 한다.
- **JWT** 기반 **Access_Token** 을 활용한다.

gateway 서비스에서 리소스서버 설정

- 게이트웨이 서비스의 **ResourceServerConfiguration.java** 파일을 열어본다.
- **spring-cloud-gateway** 는 **webflux** 로 기동되기 때문에 **@EnableWebFluxSecurity** 를 적용한다.
- **ServerHttpSecurity** 생성시, **oauth2ResourceServer()** 리소스 서버역할을 부여하고 **.jwt()** 를 선언해 **jwt** 형식의 **Authorization** 을 지정한다.
- 인증/인가를 위한 **Url** 은 **JWK(Json Web Key)**로 정의해 **application.yaml** 에 선언되어 있다.

```
spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          jwk-set-uri: http://localhost:8090/.well-known/jwks.json
```

- **8090** 포트의 서버는 인증**(oauth)** 서버이다. 인증서버에서 인증 **Url** 인 **jwks.json** 엔드포인트가 **GetMapping** 으로 선언되어 있다.

JwkSetEndpointConfiguration.java

- **Gateway** 서비스**(Client)**는 인증서버에 **uengine-client:uengine-secret** 정보로 등록되어 있다.

OAuth2AuthorizationServerConfig.java

서비스 구동

- 게이트웨이 서비스와 일반 마이크로 서비스인 **order** 서비스를 실행한다.

```
cd gateway
mvn spring-boot:run
cd order
mvn spring-boot:run
```

- **8081** 로 기동된 **order** 서비스를 바로 접근해 본다. **(접근됨)**
    - **http localhost:8081/orders**
- **8088** 로 기동된 **gateway** 서비스를 통하여 접근해 본다. **(401 Unauthorized)**
    - **http localhost:8088/orders**

oauth 서버에서 토큰 발급

- 토큰을 발급하려면, 사용자가 있어야 한다.
- **oauth** 서비스의 **AuthorizationServerApplication.java** 에서 초기 사용자 **username="1@uengine.org" / password = "1"** 로 사전 등록되어 있다.

OAuth 인증서버 구동

- 인증 서버를 실행한다.

```
cd oauth
mvn spring-boot:run
```

- 서버를 실행 후, 토큰을 요청하는 **API(/oauth/token)** 를 호출하여 토큰을 가져온다.
    - 이때 **Basic** 뒤의 **base64** 값은 인증서버에 등록된 **Gateway** 의 인코딩된 **CLIENT_ID:CLIENT_SECRET** 정보이다.
- `http --form POST localhost:8088/oauth/token "Authorization: Basic dWVuZ2luZS1jbGllbnQ6dWVuZ2luZS1zZWNyZXQ=" grant_type=password username=1@uengine.org password=1`
- 출력된 **access_token** 을 복사하여 **https://jwt.io/** 페이지에 접속 후 **decode** 해 본다.

Header, Payload, Signature 로 파싱된다.

- 가져온 데이터의 **"access_token":** 부분이 토큰 정보이다. 이 토큰을 이용하여 게이트웨이를 통하여 주문 서비스를 조회한다.

- `export access_token=[TOKEN 입력]`

```
echo $access_token
http localhost:8088/orders "Authorization: Bearer
 $access_token"
```

- **$access_token** 부분에 넘어온 토큰 정보를 입력한다.
- 토큰정보를 일부러 틀리게 하여 호출하여 본다.

- 토큰이 유효한지 체크하여 본다
  - **http --form POST localhost:8088/oauth/check_token token=$access_token**
  - 토큰정보를 일부러 틀리게 하여 호출하여 본다.

```
{
    "error": "invalid_token",
    "error_description": "Invalid access token"
}
```

Service Clear

- 다음 **Lab** 을 위해 기동된 모든 서비스 종료

```
fuser -k 8081/tcp
fuser -k 8088/tcp
fuser -k 8090/tcp
```

상세설명

CheckPoints

1. oauth 서버에서 토큰 발급하여 access_token 이라는 환경변수로 설정하고, 게이트웨이를 통하여 주문서비스를 조회한다.