MSA 11주차

∷ 스터디원	소현
	Done
Ø URL	https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/7
를 비고	MSA Section10
■ 제출 마감일	@November 22, 2023

Micrcoservice 간 통신

Communication Types

- Synchronous HTTP communication : 동기적인 통신
- Asynchronous communication over AMQP : AMQP를 사용한 비동기적인 통신
- Rest Template

Rest Template 사용

User Service

• Bean 등록

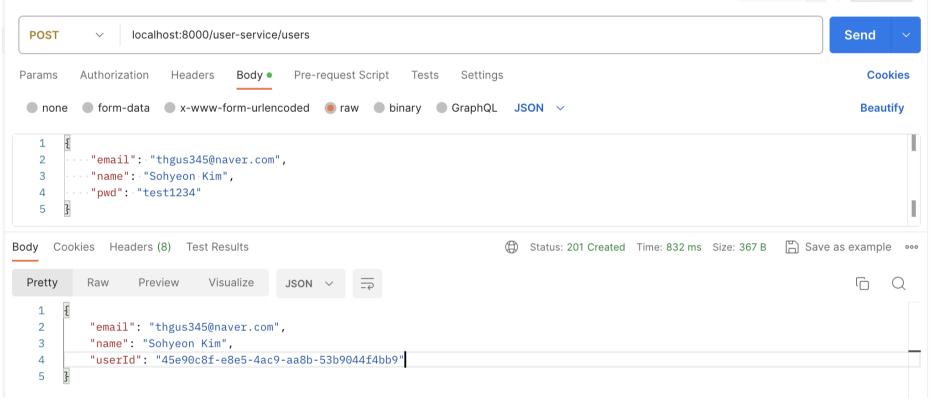
```
@Configuration
public class RestTemplateConfig {
     @Bean
     public RestTemplate restTemplate() {
        return new RestTemplate();
     }
}
```

• Controller 수정

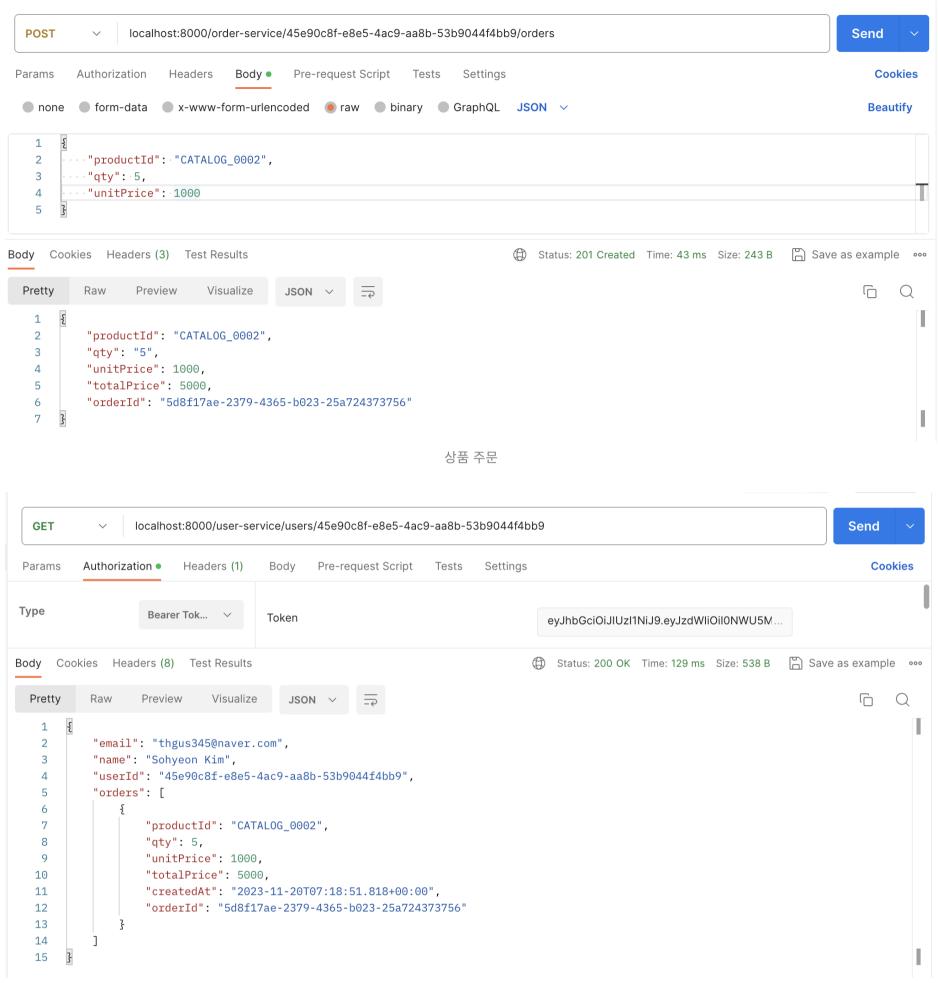
Native File Repo

```
order_service:
  url: http://localhost:8000/order-service/%s/orders
```

Test



사용자 추가



사용자 조회 : 주문 정보 조회 성공

LoadBalanced

```
@Configuration
public class RestTemplateConfig {
    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

```
url: http://ORDER-SERVICE/order-service/%s/orders # http://{mircoservice-name}/{url}
...
```

FeignClient 사용

- Feign Client → HTTP Client
 - 。 Rest call 을 추상화 한 Spring Cloud Netflix 라이브러리
- 사용방법
 - 。 호출하려는 HTTP Endpoint 에 대한 Interface 를 생성
 - o @FeignClient 선언
- Load balanced 지원

User service

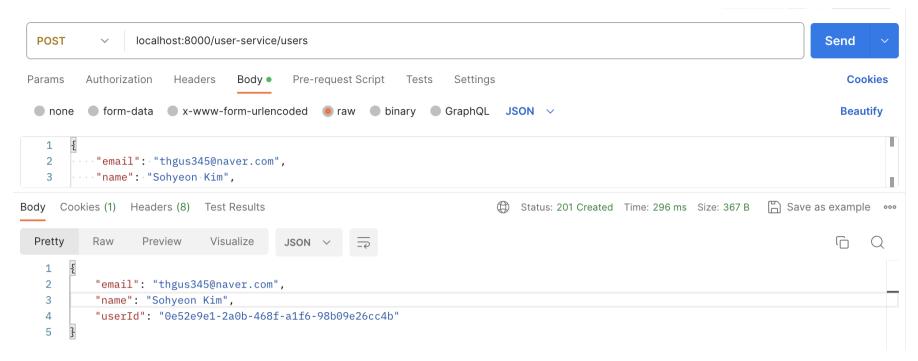
• dependencies 추가

• Feign Client 생성

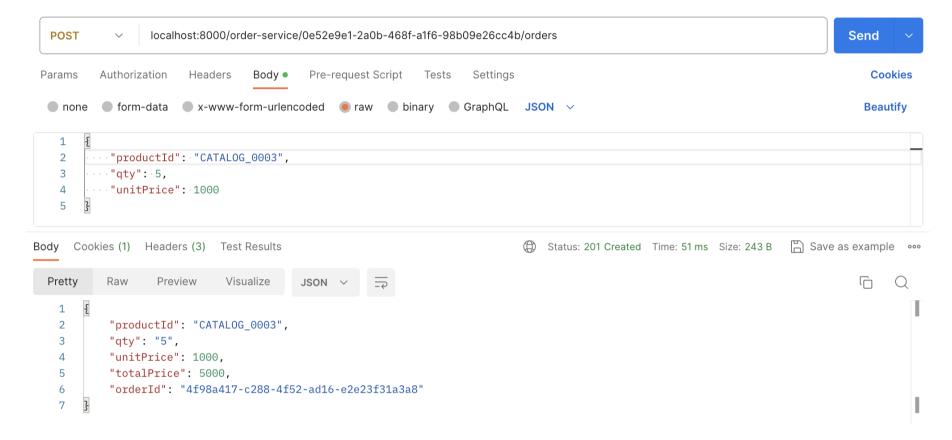
```
@FeignClient(name = "order-service")
public interface OrderServiceClient {
    @GetMapping("/order-service/{userId}/orders")
    List<ResponseOrder> getOrders(@PathVariable String userId);
}
```

Test

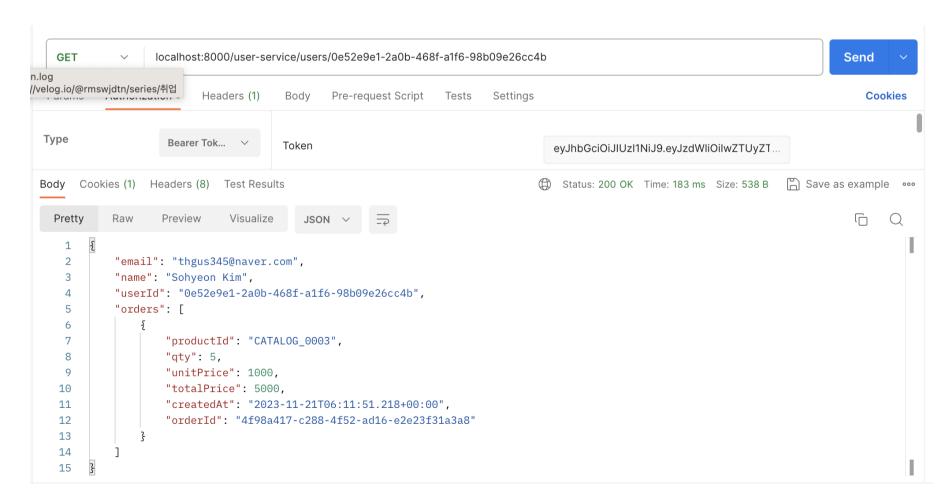
• 사용자 추가



• 상품 주문



• 사용자 조회 (주문 목록 포함)



FeignClient 예외 처리

User service

• 설정 파일 수정

```
logging:
level:
com.example.userservice.client: DEBUG
```

• 빈 등록

```
@Configuration
public class LoggerConfig {

    @Bean
    public Logger.Level feignLoggerLevel() {
       return Logger.Level.FULL;
    }
}
```

• try - catch

```
. . .
@Override
public UserDto getUserByUserId(String userId) {
    /* Using a feign client */
    List<ResponseOrder> orders = getOrdersByFeignClient(userId);
    . . .
}
private List<ResponseOrder> getOrdersByFeignClient(String userId) {
    List<ResponseOrder> orders = null;
    /* Feign exception handling */
    try {
        orders = orderServiceClient.getOrders(userId);
    } catch(FeignException ex) {
        log.error(ex.getMessage());
    }
    return orders;
}
. . .
```

ErrorDecoder를 이용한 예외 처리

User service

• 빈 등록

```
@Configuration
public class DecoderConfig {

    @Bean
    public FeignErrorDecoder getFeignErrorDecoder() {
        return new FeignErrorDecoder();
    }
}
```

• FeignErrorDecoder 추가

```
public class FeignErrorDecoder implements ErrorDecoder {
    @Override
    public Exception decode(String methodKey, Response response) {
        switch (response.status()) {
            case 400 -> {
            case 404 -> {
                if (methodKey.contains("getOrders")) {
                    return new ResponseStatusException(
                        HttpStatus.valueOf(response.status()),
                        "User's orders is empty.");
                }
            }
            default -> {
                return new Exception(response.reason());
            }
        return null;
    }
}
```

데이터 동기화 문제

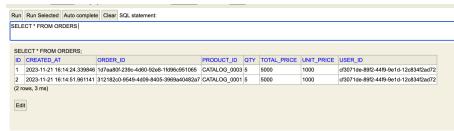
Multiple Orders Service

- Orders Service 2개 기동
 - 。 Users 의 요청 분산 처리
 - 。 Orders 데이터도 분산 저장 → 데이터 동기화 문제
 - 하나의 Database 사용
 - Database 간의 동기화 (Messaging Queuing server 활용)
 - Kafka Connector + DB

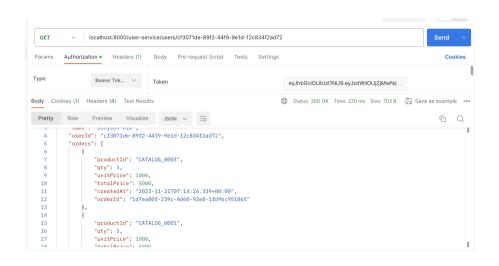
데이터 동기화 문제 예시

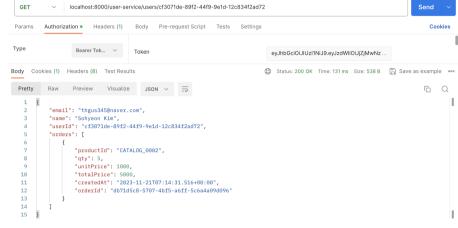
- Orders service 2개 기동
- 상품 3개 주문 → 2개의 DB에 주문 데이터가 나눠 저장됨





• 사용자 조회 (주문 내역 포함) → 사용자의 주문 데이터가 번갈아서 조회됨





8

Pull Request

https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/7