


MSA 6주차

스터디원	소현
Status	Done
URL	https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/3
비고	MSA Section5
제출 마감일	@October 18, 2023 1:00 PM

Catalogs and Orders Microservice

 Index

- Users Microservice 기능 추가
- Catalogs Microservice 프로젝트 생성
- Orders Microservice 프로젝트 생성

Users Microservice - 사용자 조회(1)

 Features

☒ 신규 회원 등록

☐ 회원 로그인

☐ 상세 정보 확인

☐ 회원 정보 수정/삭제

☐ 상품 주문

☐ 주문 내역 확인

Name	URI	HTTP Method	Develop
신규 회원 등록	/users	POST	<input checked="" type="checkbox"/>
전체 회원 조회	/users	GET	
회원 정보, 주문 내역 조회	/users/{user_id}	GET	
작동 상태 확인	/health_check	GET	<input checked="" type="checkbox"/>
환영 메시지	/welcome	GET	<input checked="" type="checkbox"/>

API Gateway Service 변경

- User Service route 추가 (포트번호가 아닌 네이밍으로 편리하게 접속하기 위해서)

```
...
spring:
  application:
    name: gateway-service
  cloud:
    gateway:
      ...
      routes:
        - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/user-service/**
...

```

서버 실행

- Eureka Server 실행 → Gateway Service, User Service 실행

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - 172.30.43.241:gateway-service:8000
USER-SERVICE	n/a (1)	(1)	UP (1) - user-service:8dd4a06293f3f381b54fe3eea09c3549

Eureka Server 대시보드

Users Microservice와 Spring Cloud Gateway 연동

User Service 변경

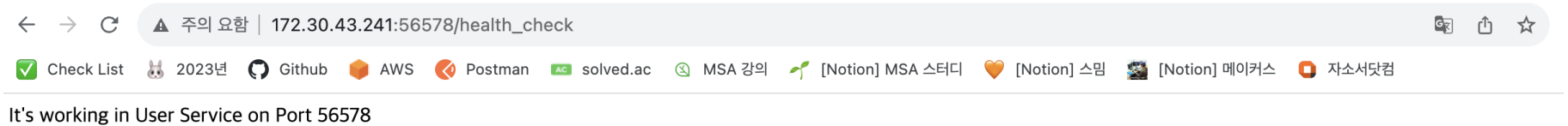
- `/health_check` 변경

```
@GetMapping("/health_check")
public String status(HttpServletRequest request) {
    return String.format("It's working in User Service on Port %s", request.getServerPort());
}
```

- WebSecurity 파일 변경

```
.requestMatchers(new AntPathRequestMatcher("/health_check/**")).permitAll()
```

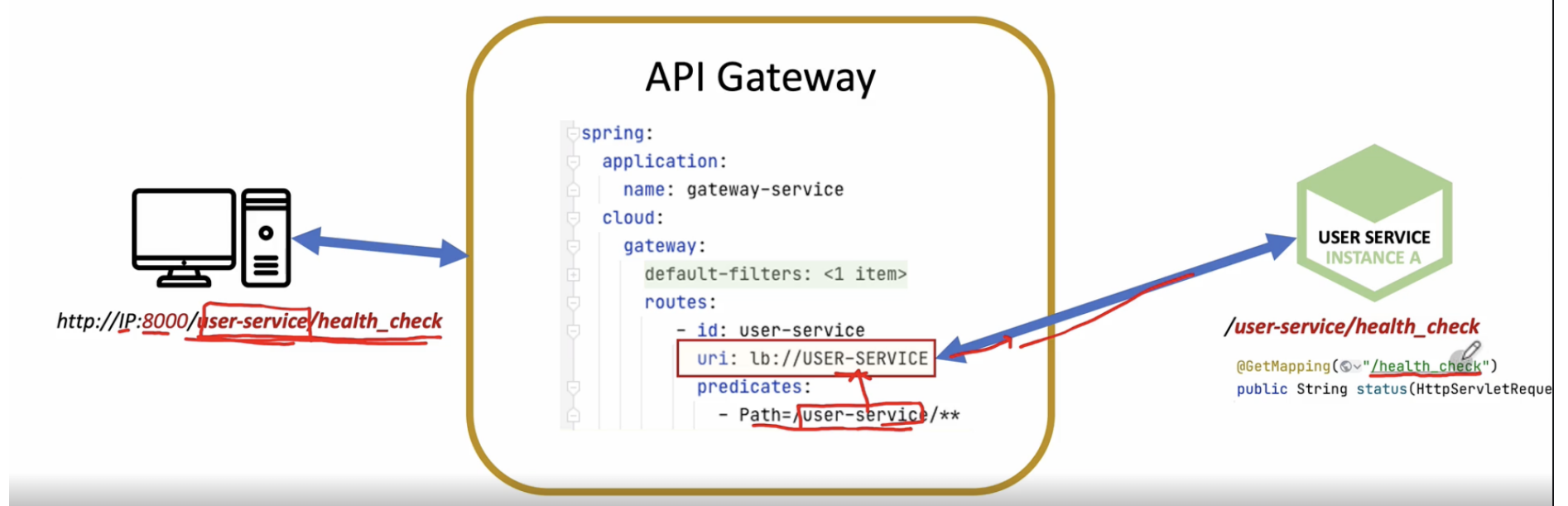
- 실행



Eureka Server → USER-SERVICE 서버 접속

Gateway 연동

- Users Service의 URI와 API Gateway의 URI가 다름

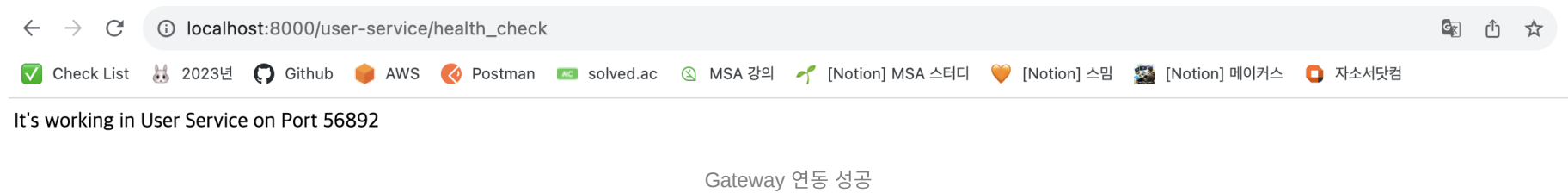


- UserService 변경

```
@GetMapping("/user-service/health_check")
public String status(HttpServletRequest request) {
    return String.format("It's working in User Service on Port %s", request.getServerPort());
}
```

```
.requestMatchers(new AntPathRequestMatcher("/user-service/health_check/**")).permitAll()
```

- 실행 `localhost:8080/user-service/health_check`



Users Microservice - 사용자 조회(2)

Response Data 추가

- ResponseUser

```
@Data
@JsonInclude(JsonInclude.Include.NON_NULL) // Null이 아닌 데이터만 전달
public class ResponseUser {
    private String email;
    private String name;
    private String userId;

    private List<ResponseOrder> orders; // 주문 내역
}
```

- ResponseOrder

```
@Data
public class ResponseOrder {
    private String productId; // 상품 id
    private Integer qty; // 주문 수량
    private Integer unitPrice; // 단가
    private Integer totalPrice; // 전체 가격
    private Date createdAt; // 주문 날짜

    private String orderId; // 주문 id
}
```

UserService 변경

- 메소드 추가

```
...
public interface UserService {
    ...
    UserDto getUserByUserId(String userId); // 회원 상세 조회
    Iterable<UserEntity> getUserByAll(); // 전체 회원 조회
}
```

```
...
public class UserServiceImpl implements UserService {

    ...

    @Override
    public UserDto getUserByUserId(String userId) {
        UserEntity userEntity = userRepository.findById(userId)
            .orElseThrow(() -> new UsernameNotFoundException("User Not Found"));

        UserDto userDto = new ModelMapper().map(userEntity, UserDto.class);

        List<ResponseOrder> orders = new ArrayList<>();
        userDto.setOrders(orders);
    }
}
```

```

        return userDto;
    }

    @Override
    public Iterable<UserEntity> getUserByAll() {
        return userRepository.findAll();
    }
}

```

UserController 변경

- RequestMapping 추가

```

...
@RequestMapping("/user-service/")
public class UserController {

    ...

    @GetMapping("/users")
    public ResponseEntity<List<ResponseUser>> getUsers() {
        Iterable<UserEntity> userList = userService.getUserByAll();

        List<ResponseUser> result = new ArrayList<>();
        userList.forEach(o -> result.add(new ObjectMapper().map(o, ResponseUser.class)));

        return ResponseEntity.ok(result);
    }

    @GetMapping("/users/{userId}")
    public ResponseEntity<ResponseUser> getUser(@PathVariable String userId) {
        UserDto userDto = userService.getUserById(userId);
        ResponseUser returnValue = new ObjectMapper().map(userDto, ResponseUser.class);

        return ResponseEntity.ok(returnValue);
    }
}

```

Users Microservice - 사용자 조회(3)

회원 전체 조회

GET localhost:8000/user-service/users

Send

Params Authorization Headers Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 21 ms Size: 464 B Save as example

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "email": "thgus345@gmail.com",
4      "name": "Sohyeon Kim",
5      "userId": "13ccf2f4-f172-46ea-ba58-8b68fe99bb70"
6    },
7    {
8      "email": "thgus345@naver.com",
9      "name": "Sohyeon Kim",
10     "userId": "3e4841e7-bd96-4163-88d5-4e45fb3f79cc"
11   }
12 ]

```

회원 추가 후 조회 요청 결과

회원 상세 조회

회원 상세 조회 (없는 id 조회)

Catalogs Microservice - 개요



Features

- ☒ 신규 회원 등록
 - ☐ 회원 로그인
 - ☒ 상세 정보 확인
 - ☐ 회원 정보 수정/삭제
 - ☐ 상품 주문
 - ☒ 주문 내역 확인

Name	Microservice	URI	Http Method
상품 목록 조회	Catalogs	/catalog-service/catalogs	GET
회원 별 상품 주문	Orders	/order-service/{user_id}/orders	POST
회원 별 주문 내역 조회	Orders	/order-service/{user_id}/orders	GET

프로젝트 생성

- **dependencies**
 - Spring Boot DevTools
 - Lombok
 - Spring Web
 - Spring Data JPA
 - Eureka Discovery Client
 - H2 Database
 - [org.modelmapper](#)
- **application yamll 파일**

```
server:
  port: 0

spring:
  application:
```

```

    name: catalog-service
  h2:
    console:
      enabled: true
      settings:
        web-allow-others: true
      path: /h2-console
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:mem:testdb
    username: sa
  jpa:
    hibernate:
      ddl-auto: create-drop
    show-sql: true
    generate-ddl: true

  eureka:
    client:
      service-url:
        defaultZone: http://localhost:8761/eureka

  logging:
    level:
      com.example.catalogservice: DEBUG

```

Catalogs Microservice - 기능 구현

data sql 추가

```

insert into catalog(product_id, product_name, stock, unit_price) values ('CATALOG_0001', 'Berlin', 100, 1500)
insert into catalog(product_id, product_name, stock, unit_price) values ('CATALOG_0002', 'Tokyo', 100, 900);
insert into catalog(product_id, product_name, stock, unit_price) values ('CATALOG_0003', 'Stockholm', 100, 1200);

```

Run
Run Selected
Auto complete
Clear

SQL statement:

SELECT * FROM CATALOG

SELECT * FROM CATALOG;

STOCK	UNIT_PRICE	CREATED_AT	ID	PRODUCT_ID	PRODUCT_NAME
100	1500	2023-10-13 17:59:53.158757	1	CATALOG_0001	Berlin
100	900	2023-10-13 17:59:53.159816	2	CATALOG_0002	Tokyo
100	1200	2023-10-13 17:59:53.159991	3	CATALOG_0003	Stockholm

(3 rows, 2 ms)

Edit

Entity, Repository 추가



Serializable 목적

: 가지고 있는 객체를 다른 네트워크로 전송하거나 데이터베이스에 보관하기 위해서 사용

```

@Data
@Entity
@Table(name = "catalog")
public class CatalogEntity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, length = 120, unique = true)
    private String productId;
    @Column(nullable = false)
    private String productName;
    @Column(nullable = false)

```

```

    private Integer stock;
    @Column(nullable = false)
    private Integer unitPrice;

    @Column(nullable = false, updatable = false, insertable = false)
    @ColumnDefault(value = "CURRENT_TIMESTAMP")
    private Date createdAt;
}

```

```

public interface CatalogRepository extends JpaRepository<CatalogEntity, Long> {
    Optional<CatalogEntity> findByProductId(String productId);
}

```

DTO, VO 추가

```

@Data
public class CatalogDto implements Serializable {
    private String productId;
    private Integer qty;
    private Integer unitPrice;
    private Integer totalPrice;

    private String orderId;
    private String userId;
}

```

```

@Data
@JsonInclude(JsonInclude.Include.NON_NULL)
public class ResponseCatalog {
    private String productId;
    private String productName;
    private Integer stock;
    private Integer unitPrice;
    private Date createdAt;
}

```

Service 추가

```

public interface CatalogService {
    Iterable<CatalogEntity> getAllCatalogs();
}

```

```

@Service
@Slf4j
@RequiredArgsConstructor
public class CatalogServiceImpl implements CatalogService {
    private final CatalogRepository repository;

    @Override
    public Iterable<CatalogEntity> getAllCatalogs() {
        return repository.findAll();
    }
}

```

Controller 추가

```

@RestController
@RequiredArgsConstructor
@RequestMapping("/catalog-service")
public class CatalogController {
    private final CatalogService catalogService;
}

```

```

@GetMapping("/health_check")
public String status(HttpServletRequest request) {
    return String.format("It's working in Catalog Service on Port %s", request.getServerPort());
}

@GetMapping("/catalogs")
public ResponseEntity<List<ResponseCatalog>> getCatalogs() {
    Iterable<CatalogEntity> catalogList = catalogService.getAllCatalogs();

    List<ResponseCatalog> result = new ArrayList<>();
    catalogList.forEach(o -> result.add(new ObjectMapper().map(o, ResponseCatalog.class)));

    return ResponseEntity.ok(result);
}
}

```

Gateway 라우트 추가

```

...
spring:
  application:
    name: gateway-service
  cloud:
    gateway:
      ...
      routes:
        ...
        - id: catalog-service
          uri: lb://CATALOG-SERVICE
          predicates:
            - Path=/catalog-service/**
...

```

서버 실행 및 API 테스트

The screenshot shows a web browser interface for testing an API. The address bar shows a GET request to `localhost:8000/catalog-service/catalogs`. The response is a JSON array of three catalog items:

```

[
  {
    "productId": "CATALOG_0001",
    "productName": "Berlin",
    "stock": 100,
    "unitPrice": 1500,
    "createdAt": "2023-10-13T08:59:53.158+00:00"
  },
  {
    "productId": "CATALOG_0002",
    "productName": "Tokyo",
    "stock": 100,
    "unitPrice": 900,
    "createdAt": "2023-10-13T08:59:53.159+00:00"
  },
  {
    "productId": "CATALOG_0003",
    "productName": "Stockholm",
    "stock": 100,
    ...
  }
]


```

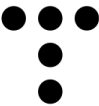
상품 전체 조회

Trouble Shooting

- SQL 실행 실패

[에러해결] data.sql Error creating bean with name 'dataSourceScriptDatabaseInitializer' defined in class path resource

에러 : data.sql로 초기 data insert 할 때 에러 발생함. data.sql Error creating bean with name 'dataSourceScriptDatabaseInitializer' defined in class path resource 원인 : 스프링 버전 2.4.x에서는 그냥 되는데 스프링 버전 2.5.x에서는 스프링 2.5에서 SQL Script DataSource Initialization의 기능이 변경되어 data.sql 스크립트는 hibernate가 초기화되기 전에 실행되며 hibernate에 의해 생성된 스키마에
 <https://charactermail.tistory.com/353>



Orders Microservice - 개요



Features

☒ 상품 목록 조회

☐ 회원 별 상품 주문

☐ 회원 별 주문 내역 조회

Name	Microservice	URI	Http Method
상품 목록 조회	Catalogs	/catalog-service/catalogs	GET
회원 별 상품 주문	Orders	/order-service/{user_id}/orders	POST
회원 별 상품 내역 조회	Orders	/order-service/{user_id}/orders	GET

프로젝트 생성

• Dependencies

- Spring Boot DevTools
- Lombok
- Spring Web
- Spring Data JPA
- H2 Database
- Eureka Discovery Client
- org.modelmapper

• application yml 파일

```
server:
  port: 0

spring:
  application:
    name: order-service
  h2:
    console:
      enabled: true
      settings:
        web-allow-others: true
      path: /h2-console
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:mem:testdb
  jpa:
    hibernate:
      ddl-auto: update
      show-sql: true

eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka

logging:
  level:
    com.example.catalogservice: DEBUG
```

- API Gateway Service - Order 라우트 추가

Orders Microservice - 기능 구현(1)

Entity, Repository 추가

```
@Data
@Entity
@Table(name = "orders")
public class OrderEntity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, length = 120, unique = true)
    private String productId;
    @Column(nullable = false)
    private Integer qty;
    @Column(nullable = false)
    private Integer unitPrice;
    @Column(nullable = false)
    private Integer totalPrice;

    @Column(nullable = false)
    private String userId;
    @Column(nullable = false, unique = true)
    private String orderId;

    @Column(nullable = false, updatable = false, insertable = false)
    @ColumnDefault(value = "CURRENT_TIMESTAMP")
    private Date createdAt;
}
```

```
public interface OrderRepository extends JpaRepository<OrderEntity, Long> {
    Optional<OrderEntity> findByOrderId(String orderId);
    Iterable<OrderEntity> findByUserId(String userId);
}
```

DTO, VO 추가

```
@Data
public class OrderDto implements Serializable {
    private String productId;
    private Integer qty;
    private Integer unitPrice;
    private Integer totalPrice;

    private String orderId;
    private String userId;
}
```

```
@Data
@JsonInclude(JsonInclude.Include.NON_NULL)
public class ResponseOrder {
    private String productId;
    private String qty;
    private Integer unitPrice;
    private Integer totalPrice;
    private Date createdAt;

    private String orderId;
}
```

```

@Data
public class RequestOrder {
    private String productId;
    private Integer qty;
    private Integer unitPrice;
}

```

Service 추가

```

public interface OrderService {
    OrderDto createOrder(OrderDto orderDetails); // 상품 주문
    OrderDto getOrderById(String orderId); // 주문 상세 조회
    Iterable<OrderEntity> getOrdersByUserId(String userId); // 회원 별 주문 조회
}

```

```

@Service
@RequiredArgsConstructor
public class OrderServiceImpl implements OrderService {
    private final OrderRepository orderRepository;

    @Override
    public OrderDto createOrder(OrderDto orderDetails) {
        orderDetails.setOrderId(UUID.randomUUID().toString());
        orderDetails.setTotalPrice(orderDetails.getQty() * orderDetails.getUnitPrice());

        ModelMapper mapper = new ModelMapper();
        mapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRICT);
        OrderEntity orderEntity = mapper.map(orderDetails, OrderEntity.class);

        orderRepository.save(orderEntity);

        return mapper.map(orderEntity, OrderDto.class);
    }

    @Override
    public OrderDto getOrderById(String orderId) {
        OrderEntity orderEntity = orderRepository.findById(orderId)
            .orElseThrow(() -> new EntityNotFoundException("INVALID ORDER"));
        return new ModelMapper().map(orderEntity, OrderDto.class);
    }

    @Override
    public Iterable<OrderEntity> getOrdersByUserId(String userId) {
        return orderRepository.findByUserId(userId);
    }
}

```

Orders Microservice - 기능 구현(2)

Controller 추가

```

@RestController
@RequiredArgsConstructor
@RequestMapping("/order-service")
public class OrderController {
    private final OrderService orderService;

    @GetMapping("/health_check")
    public String status(HttpServletRequest request) {
        return String.format("It's working in Order Service on Port %s", request.getServerPort());
    }

    @PostMapping("/{userId}/orders")

```

```

public ResponseEntity<ResponseOrder> createOrder(@PathVariable String userId, @RequestBody RequestOrder c
    ModelMapper mapper = new ModelMapper();
    mapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRICT);

    OrderDto orderDto = mapper.map(orderDetails, OrderDto.class);
    orderDto.setUserId(userId);
    OrderDto createdOrder = orderService.createOrder(orderDto);

    ResponseOrder responseOrder = mapper.map(createdOrder, ResponseOrder.class);

    return ResponseEntity.status(HttpStatus.CREATED).body(responseOrder);
}

@GetMapping("/{userId}/orders")
public ResponseEntity<List<ResponseOrder>> getOrders(@PathVariable String userId) {
    Iterable<OrderEntity> orderList = orderService.getOrdersByUserId(userId);

    List<ResponseOrder> result = new ArrayList<>();
    orderList.forEach(o -> result.add(new ModelMapper().map(o, ResponseOrder.class)));

    return ResponseEntity.ok(result);
}
}

```

서버 실행 및 API 테스트



실행 순서

: Eureka Server → Api Gateway Service → User Service, Catalog Service, Order Service

- 사용자 추가

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8000/user-service/users
- Body Type:** JSON
- Request Body:**

```

{
  "email": "thgus345@naver.com",
  "name": "Sohyeon Kim",
  "pwd": "test1234"
}

```
- Status:** 201 Created
- Time:** 1068 ms
- Size:** 367 B
- Response Body:**

```

{
  "email": "thgus345@naver.com",
  "name": "Sohyeon Kim",
  "userId": "e4b464d3-fa10-42c6-9f2a-dfb2397c2c86"
}

```

- 사용자 조회

GET localhost:8000/user-service/users/e4b464d3-fa10-42c6-9f2a-dfb2397c2c86 Send

Params Authorization Headers Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results Status: 200 OK Time: 178 ms Size: 374 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "email": "thgus345@naver.com",
3   "name": "Sohyeon Kim",
4   "userId": "e4b464d3-fa10-42c6-9f2a-dfb2397c2c86",
5   "orders": []
6 }
```

- 상품 조회

GET localhost:8000/catalog-service/catalogs Send

Params Authorization Headers Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (3) Test Results Status: 200 OK Time: 455 ms Size: 493 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "productId": "CATALOG_0001",
4     "productName": "Berlin",
5     "stock": 100,
6     "unitPrice": 1500,
7     "createdAt": "2023-10-13T15:47:05.802+00:00"
8   },
9   {
10    "productId": "CATALOG_0002",
11    "productName": "Tokyo",
12    "stock": 100,
13    "unitPrice": 900,
14    "createdAt": "2023-10-13T15:47:05.804+00:00"
15  },
16  {
17    "productId": "CATALOG_0003",
18    "productName": "Stockholm",
19    "stock": 100,
```

- 상품 주문

POST localhost:8000/order-service/e4b464d3-fa10-42c6-9f2a-dfb2397c2c86/orders Send

Params Authorization Headers Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "productId": "CATALOG_0002",
3   "qty": 5,
4   "unitPrice": 1000
5 }
```

Body Cookies Headers (3) Test Results Status: 201 Created Time: 76 ms Size: 243 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "productId": "CATALOG_0002",
3   "qty": "5",
4   "unitPrice": 1000,
5   "totalPrice": 5000,
6   "orderId": "f77b0949-1d94-4583-b78a-9211aa1abf48"
7 }
```

- 회원 별 주문 조회

GETlocalhost:8000/order-service/e4b464d3-fa10-42c6-9f2a-dfb2397c2c86/ordersSend

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettingsCookies

BodyCookiesHeaders (3)Test Results

Body

RawPreviewVisualizeJSON

```
1  [
2    {
3      "productId": "CATALOG_0001",
4      "qty": "10",
5      "unitPrice": 1500,
6      "totalPrice": 15000,
7      "createdAt": "2023-10-13T15:51:29.109+00:00",
8      "orderId": "62bd2775-00b9-4a87-ac67-b5f45d834555"
9    },
10   {
11     "productId": "CATALOG_0002",
12     "qty": "5",
13     "unitPrice": 1000,
14     "totalPrice": 5000,
15     "createdAt": "2023-10-13T15:53:08.572+00:00",
16     "orderId": "f77b0949-1d94-4583-b78a-9211aa1abf48"
17   }
18 ]
```

Status: 200 OKTime: 81 msSize: 453 BSave as example

Github Code (PR)

<https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/3>