**GraphQL로 백엔드 데이터 통합**

* 주문,상품,배송 서비스를 모두 기동한다.
* 터미널 3개를 열어서 각각의 프로젝트로 이동한 후, run을 실행한다.
* 주문서비스 기동(8081)

**cd reqres\_orders**

**mvn spring-boot:run**

* 상품서비스 기동(8085)

**cd reqres\_products**

**mvn spring-boot:run**

* 배송서비스 기동(8082)

**cd reqres\_delivery**

**mvn spring-boot:run**

* 1개의 주문을 생성한다.

**http localhost:8081/orders productId=1 quantity=1 customerId=**[**1@uengine.org**](mailto:1@uengine.org)

* GraphQL 기동(8089)

**cd apollo\_graphql**

**npm install**

**npm start**

* GraphQL Playground

작성한 GraphQL Type, Resolver 명세확인, 데이터 요청 및 테스트가 가능한 워크벤치

* Labs > 포트열기 > 8089로 WebUI에 접속

**서비스 조회**

* 전체 주문서비스

**query getOrders {**

**orders {**

**productId**

**productName**

**quantity**

**price**

**}**

**}**

* 단일 주문서비스( id=1 주문서비스 )

**query getOrderById {**

**order(orderId: 1) {**

**productId**

**productName**

**quantity**

**price**

**}**

**}**

* 복합 서비스 조회  
  order 서비스의 연결된 product, delivery 정보조회

**query Query{**

**orders {**

**quantity**

**customerId**

**state**

**product {**

**price**

**name**

**}**

**delivery {**

**deliveryAddress**

**}**

**}**

**}**

**GraphQL 파일 참고**

1. src/graphql/resolvers.js

* 데이터를 가져오는 구체적인 과정을 구현
* 서비스의 액션들을 함수로 지정, 요청에 따라 데이터를 반환(Query), Mutation(입력, 수정, 삭제) 하는 Query 또는 구현체 작성

**예시)**

**const resolvers = {**

**//typeDefs의 객체 유형 정보(Order, Query, Product) 호출 선언**

**Query: {**

**//...**

**}**

**Order: {**

**deliveries: (root, args, {dataSources}) => {}**

**// 함수명: (parent, args, context, info) => {}**

**// \* parent : 루트에 대한 resolver의 반환 값.**

**// \* args : 함수 호출시 args 또는 {parameter}으로 인자값.**

**// \* context :**

**특정 작업을 위해 실행되는 모든 resolver에 전달되는 개체,**

**데이터베이스 연결과 같은 컨텍스트를 공유.**

**{dataSources}: xxx-rest-api.js와 연결된 데이터 호출.**

**// \* info : 필드명, 루트에서 필드까지의 경로 등 작업의 실행 상태.**

**}**

**}**

1. src/graphql/typeDefs.js
   * GraphQL 명세서에서 사용될 데이터, 요청의 타입 (gql로 생성됨)

* Type Definitions
* 객체 타입과 필드명 선언

**type Delivery {**

**id: Long!**

**orderId: Long**

**productId: Long**

**customerId: String**

**deliveryAddress: String**

**deliveryState: String**

**orders: [Order]**

**order(orderId: Long): Order**

**}**

**type Order {**

**id: Long!**

**productId: Long**

**customerId: String**

**state: String**

**deliveries: [Delivery]**

**delivery(deliveryId: Long): Delivery**

**}**

**// []: 배열**

**// !: 필수값**

1. src/restApiServer/xxx-rest-api.js
   * apollo-datasource-rest의 해당 서비스의 호출 함수및 호출 경로 설정.

**import {RESTDataSource} from 'apollo-datasource-rest';**

**// apollo-datasource-rest 모듈**

**class orderRestApi extends RESTDataSource {**

**constructor() {**

**super();**

**this.baseURL = 'http://order:8080';**

**// 해당 서비스의 호출 주소 정보.**

**}**

**// 함수명()**

**async getOrders() {**

**const data = await this.get('/orders', {})**

**// baseURL 이후 url 호출 정보.**

**var value = this.stringToJson(data);**

**// 호출정보 String to Json 으로 변경.**

**return value**

**// 호출 정보 리턴.**

**}**

**async getOrder(id) {**

**// ...**

**}**

**stringToJson(str){**

**if(typeof str == 'string'){**

**str = JSON.parse(str);**

**}**

**return str;**

**}**

**}**

1. src/index.js
   * 선언부 호출 매핑및 선언.

**import {ApolloServer} from 'apollo-server';**

**import resolvers from './graphql/resolvers.js';**

**import typeDefs from './graphql/typeDefs.js';**

**import orderRestApi from './restApiServer/order-rest-api.js'**

**import deliveryRestApi from './restApiServer/delivery-rest-api.js'**

**const server = new ApolloServer({**

**typeDefs,**

**resolvers,**

**dataSources: () => ({**

**orderRestApi: new orderRestApi(),**

**deliveryRestApi: new deliveryRestApi()**

**}),**

**// dataSources 선언 하여 xxxRestApi 호출정보.**

**});**

**server.listen({**

**port: 8089,**

**}).then(({url}) => {**

**console.log(`🚀 Server ready at ${url}`);**

**});**

Checkpoints

1. 모든 요구사항을 만족하는가