

SpringBoot 기반의 MSA 과정

inky4832@daum.net

1장. 환경 설정



목 차

JDK설치

STS 설치

Kafka 설치

PostgreSQL 설치

데이터베이스 생성

1. JDK 설치

1) JDK설치

<http://java.oracle.com>

[Overview](#) [Technical Details](#)

Top Downloads

- [Java SE](#)
- [Java EE and GlassFish](#)
- [Java Card](#)
- [JDeveloper and ADF](#)
- [Enterprise Pack for Eclipse](#)
- [NetBeans IDE](#)

Newest Downloads


- [Java SE 16](#)
- [Java SE 15.0.2](#)
- [Java SE 11.0.10 \(LTS\)](#)
- [Java SE 8u281](#)
- [Java SE Embedded](#)
- [Java Card 3.1](#)


Java SE 8

Java SE 8u281 is the latest release for the Java SE 8 Platform.

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)

Oracle JDK

 [JDK Download](#)

 [Server JRE Download](#)

1. JDK 설치

Solaris x64 (SVR4 package)	134.68 MB	jdk-8u281-solaris-x64.tar.Z
Solaris x64	92.66 MB	jdk-8u281-solaris-x64.tar.gz
Windows x86	154.69 MB	jdk-8u281-windows-i586.exe
Windows x64	166.97 MB	jdk-8u281-windows-x64.exe

download 받아서 기본설치 또는 임의의 디렉토리에 설치한다.
이후에 JAVA_HOME과 PATH 환경변수를 다음과 같이 설정한다.

시스템 변수 편집

변수 이름(N):

변수 값(V):

디렉터리 찾아보기(D)...

파일 찾아보기(F)...

시스템 변수(S)

변수	값
NUMBER_OF_PRO... 12	
OS	Windows_NT
Path	%JAVA_HOME%\bin;C:\W...

2. STS 설치

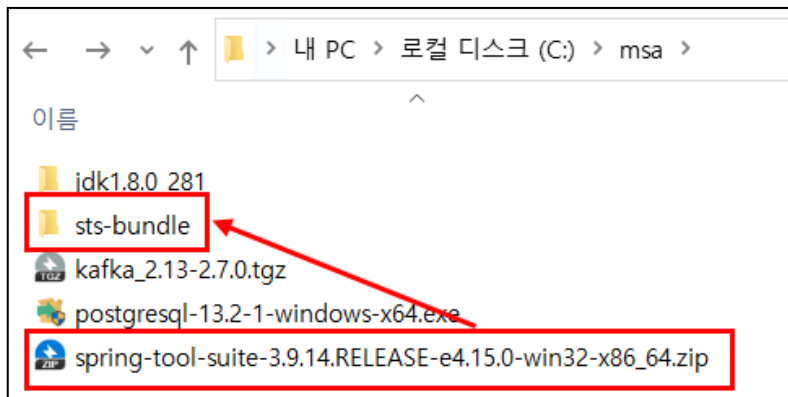
2) STS 설치 (Eclipse 4.15, STS 3.9.14)

<https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3>

full distribution on Eclipse 4.15

- https://download.springsource.com/release/STS/3.9.14.RELEASE/dist/e4.15/spring-tool-suite-3.9.14.RELEASE-e4.15.0-win32-x86_64.zip
- https://download.springsource.com/release/STS/3.9.14.RELEASE/dist/e4.15/spring-tool-suite-3.9.14.RELEASE-e4.15.0-macosx-cocoa-x86_64.dmg
- https://download.springsource.com/release/STS/3.9.14.RELEASE/dist/e4.15/spring-tool-suite-3.9.14.RELEASE-e4.15.0-linux-gtk-x86_64.tar.gz

download 받아서 c:\wmsa 디렉토리에 저장하고 압축을 푼다.



STS.exe 파일 실행후 workspace 설정 및 Text file Encoding 값을 UTF-8로 변경한다.

3. Kafka 설치

3) Kafka 설치

<http://kafka.apache.org/downloads>

DOWNLOAD

2.7.0 is the latest release. The current stable version is 2.7.0.

You can verify your download by following these [procedures](#) and using these [KEYS](#).

2.7.0

- Released Dec 21, 2020
- [Release Notes](#)
- Source download: [kafka-2.7.0-src.tgz](#) ([asc](#), [sha512](#))
- Binary downloads:
 - Scala 2.12 - [kafka_2.12-2.7.0.tgz](#) ([asc](#), [sha512](#))
 - Scala 2.13 - [kafka_2.13-2.7.0.tgz](#) ([asc](#), [sha512](#))

download 받아서 c:\wmsa 디렉토리에 저장하고 압축을 푼다.

4. PostgreSQL 설치

4) Postgresql 설치

<https://www.postgresql.org/download/windows/>

Windows installers

Interactive installer by EDB

Download the installer certified by EDB for all supported PostgreSQL versions.

This installer includes the PostgreSQL server, pgAdmin; a graphical tool for managing and developing your data PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial,

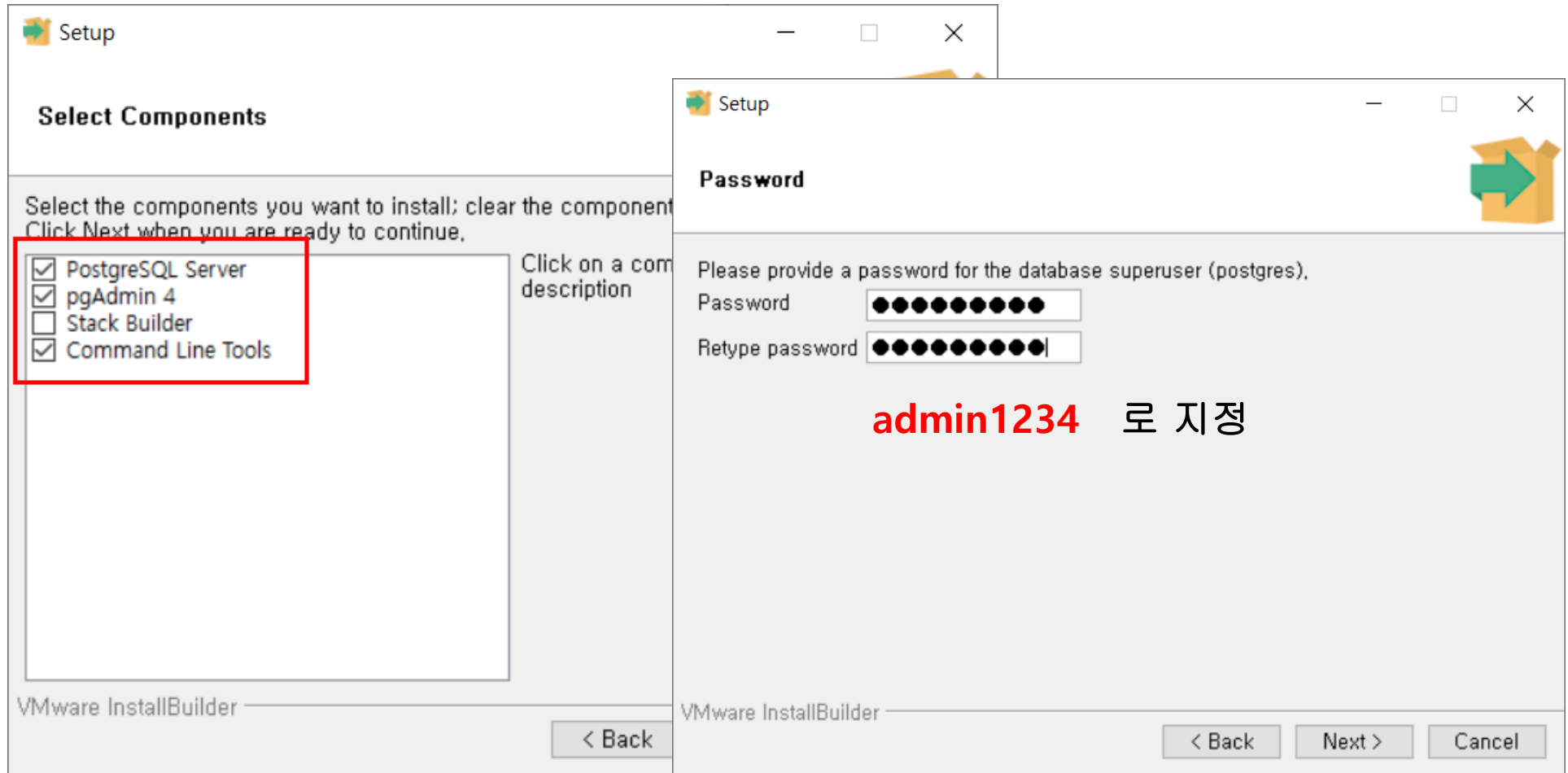
This installer can run in graphical or silent install modes.

The installer is designed to be a straightforward, fast way to get up and running with PostgreSQL on Windows.




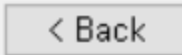
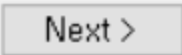
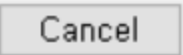
Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
13.2	N/A	N/A	Download	Download	N/A
12.6	N/A	N/A	Download	Download	N/A
11.11	N/A	N/A	Download	Download	N/A

4. PostgreSQL 설치

download 받아서 c:\Wmsa 디렉토리에 저장하고 기본설정방법으로 설치한다.

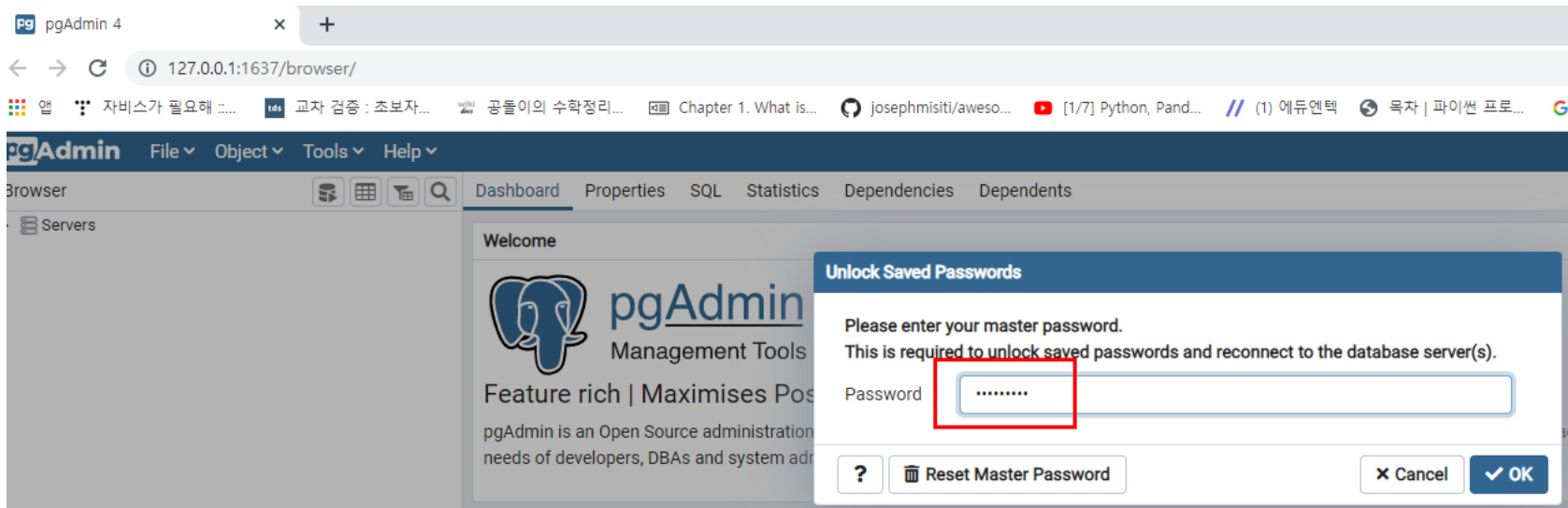
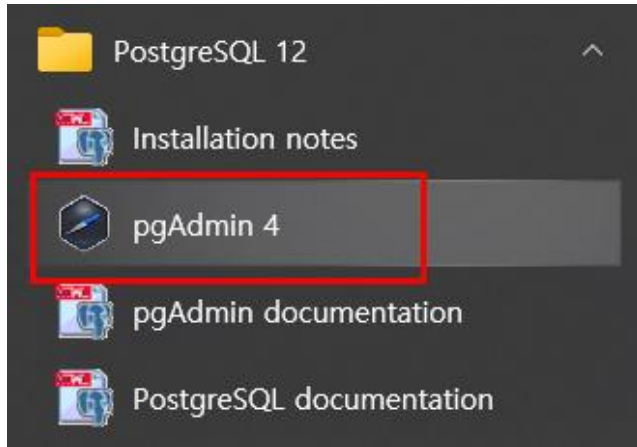


4. PostgreSQL 설치

 Setup Port	 Setup Advanced Options 
<p>Please select the port number the server will use.</p> <p>Port <input type="text" value="5432"/></p>	<p>Select the locale to be used by the new database cluster.</p> <p>Locale <input type="text" value="Korean, Korea"/></p>
VMware InstallBuilder	VMware InstallBuilder   

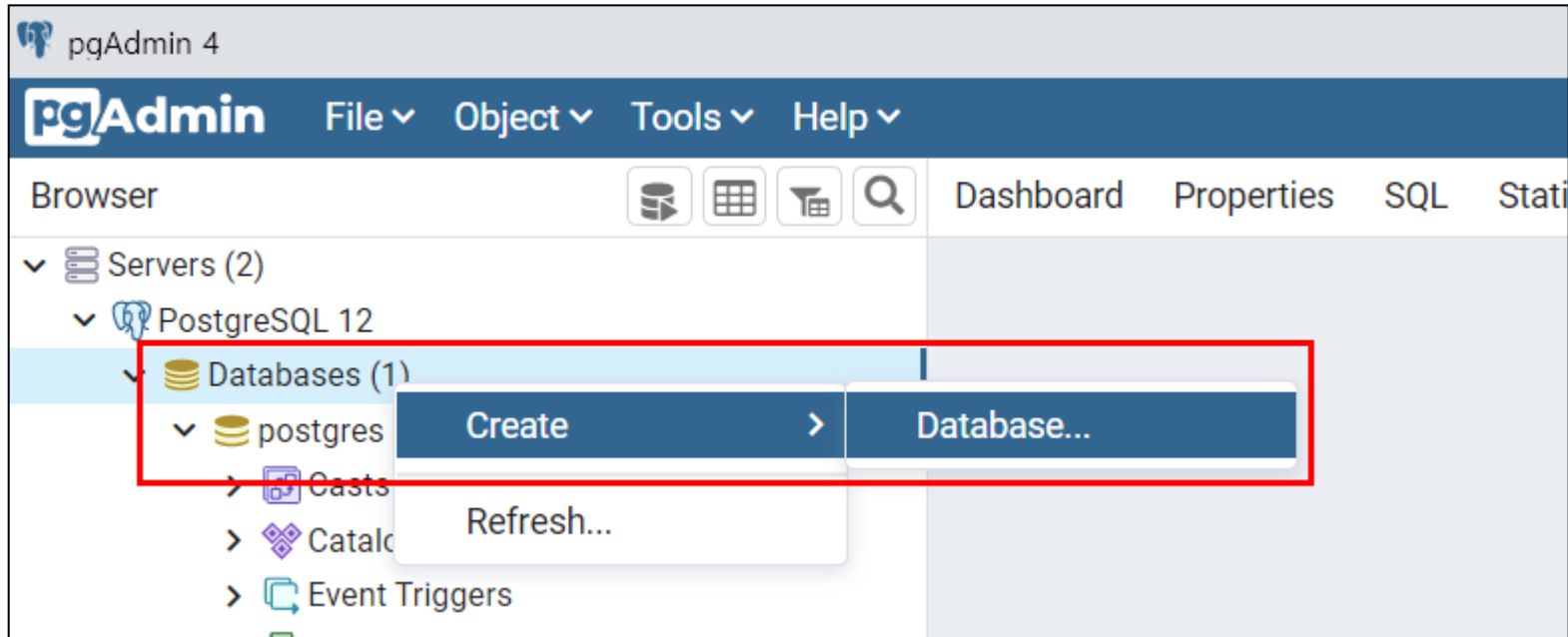
5. 데이터베이스 생성

데이터베이스를 생성하기 위하여 pgAdmin 4 를 선택한다.



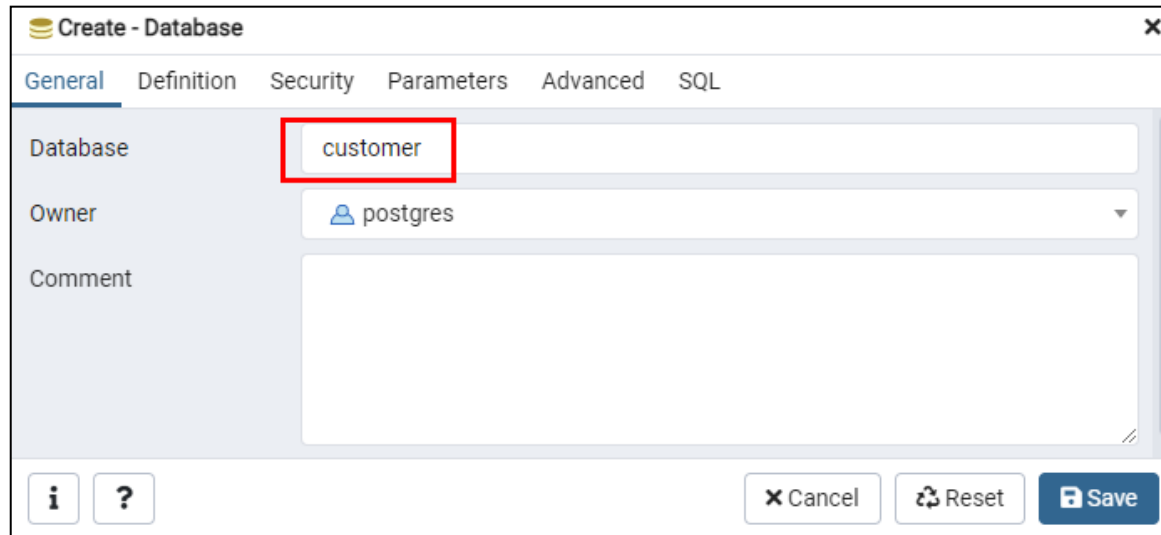
5. 데이터베이스 생성

PostgreSQL 12의 Databases를 클릭한 후 마우스 우 클릭을 하고 create database를 선택한다.



5. 데이터베이스 생성

Database 란에 customer를 입력한 후 Save를 클릭한다.



Create - Database

General Definition Security Parameters Advanced SQL

Database: customer

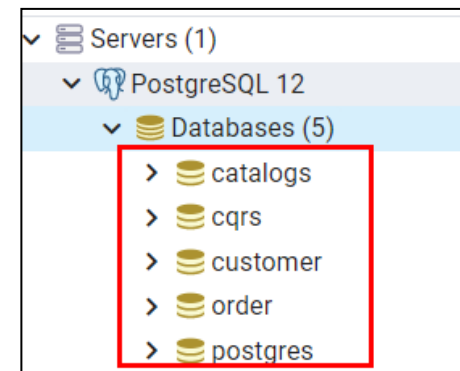
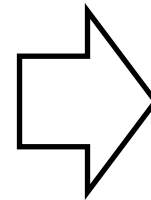
Owner: postgres

Comment:

Cancel Reset Save

같은 방식으로 아래의 db를 생성한다

order
catalogs
cqrs



2장. E-commerce 실습



실습 시나리오 및 구성

E-commerce 마이크로서비스 구성도

DB 명세서 (논리/물리 ERD)

Application Architecture

1. E-commerce 실습 시나리오 및 구성

E-commerce 실습 시나리오

E-commerce는 기존 운영되던 Monolithic Application을 Microservices Application으로 변환하여 구축하고 있다. 이에 고객(Customer), 주문(Order), 상품(Catalogs) 및 고객상세조회(CustomerInquiry)의 Microservices 및 화면 Application으로 분리하여 개발을 진행하고 있다.

Microservice 구성

Microservice	설명
Customer	고객에 대한 정보를 관리하는 Microservice
Order	주문 등록 및 내역을 관리하는 Microservice
Catalogs	상품 등록 및 목록 제공하는 Microservice
Customer Inquiry	고객 상세 조회 Microservice로서 CQRS 패턴 사용

3. 마이크로서비스 별 API 목록

마이크로서비스 별 API 목록		
마이크로서비스명	컴포넌트명	API명(오퍼레이션명)
Customer	고객	고객등록(), 고객삭제(), 고객수정() 고객기본조회() 고객상세조회() 고객전체조회() 고객존재여부조회()
Order	주문	주문등록() 주문조회()
Catalogs	상품	상품등록(), 상품삭제(), 상품수정() 상품조회() 상품전체조회()
Customer Inquiry	고객조회	고객상세조회()

3장. 기본 실습

(Hello World!)



Hello World! 출력

Swagger 설정

1. Hello World! 출력

실습 상세

실습 개요

MSA 구현 환경으로 많이 사용되는 Spring Boot 기반의 개발 환경에 익숙해지기 위한 간단한 실습이다. 이를 위해 프로젝트를 생성한 후, API를 개발하고 테스트하는 실습을 진행한다.

1. 프로젝트 생성

<http://start.spring.io> 접속하여 다음 정보로 프로젝트를 생성한 후 다운 받는다.

- Project: Maven Project
- Language: Java
- Spring Boot: 2.3.10
- Project Metadata/Group: com.msa
- Project Metadata/Artifact: Hello_World
- Project Metadata/Packing: jar
- Project Metadata/Java: 8
- Dependency: Spring Web

1. Hello World! 출력



Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.5.0 (SNAPSHOT) ☐ 2.5.0 (RC1) ☐ 2.4.6 (SNAPSHOT) ☐ 2.4.5
☐ 2.3.11 (SNAPSHOT) ☒ 2.3.10

Project Metadata

Group	com.msa
Artifact	Hello_World
Name	Hello_World
Description	Demo project for Spring Boot
Package name	com.msa.Hello_World

Packaging ☒ Jar ☐ War

Java ☐ 16 ☐ 11 ☒ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

GENERATE CTRL + G

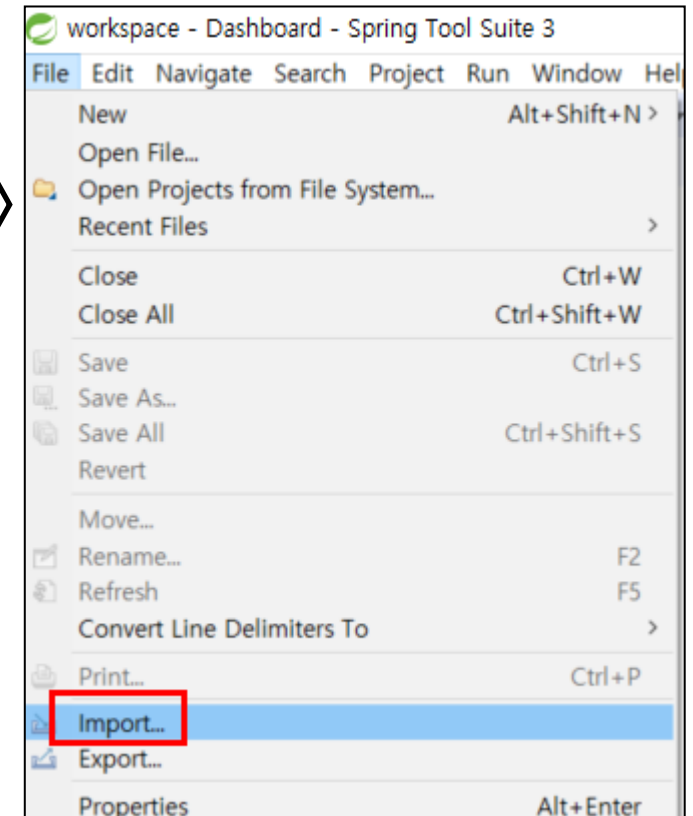
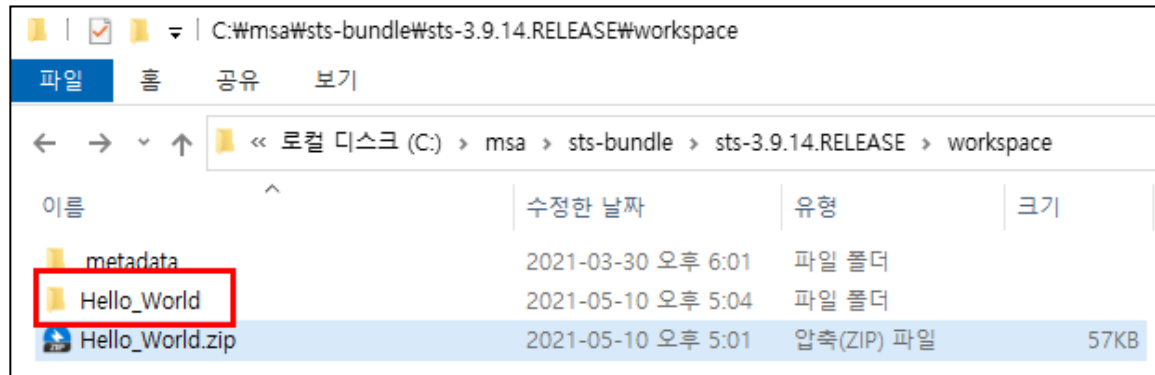
EXPLORE CTRL + SPACE

SHARE...

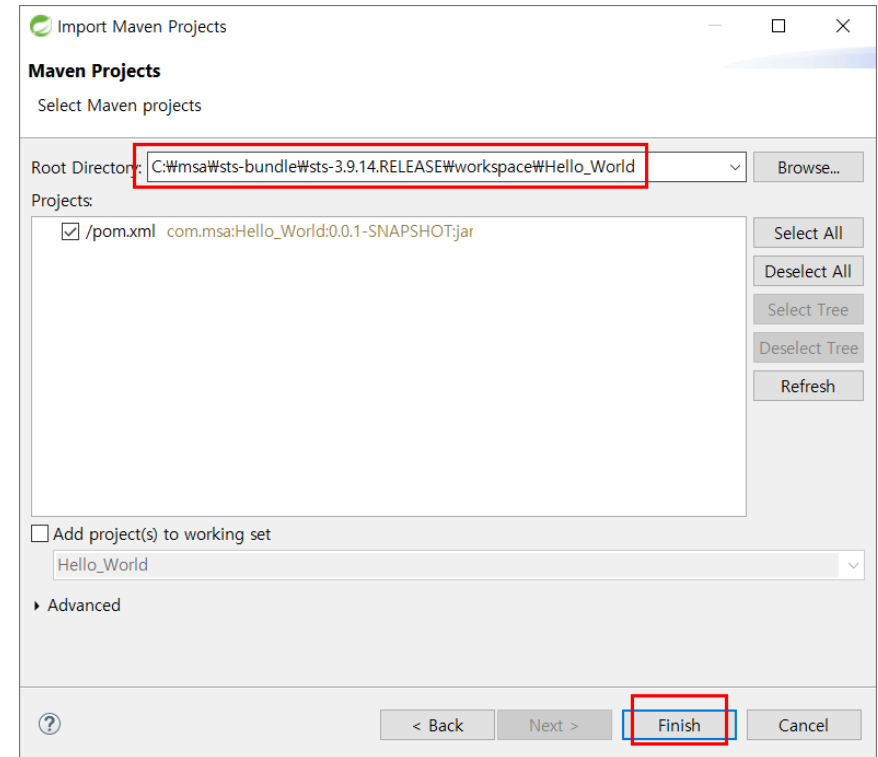
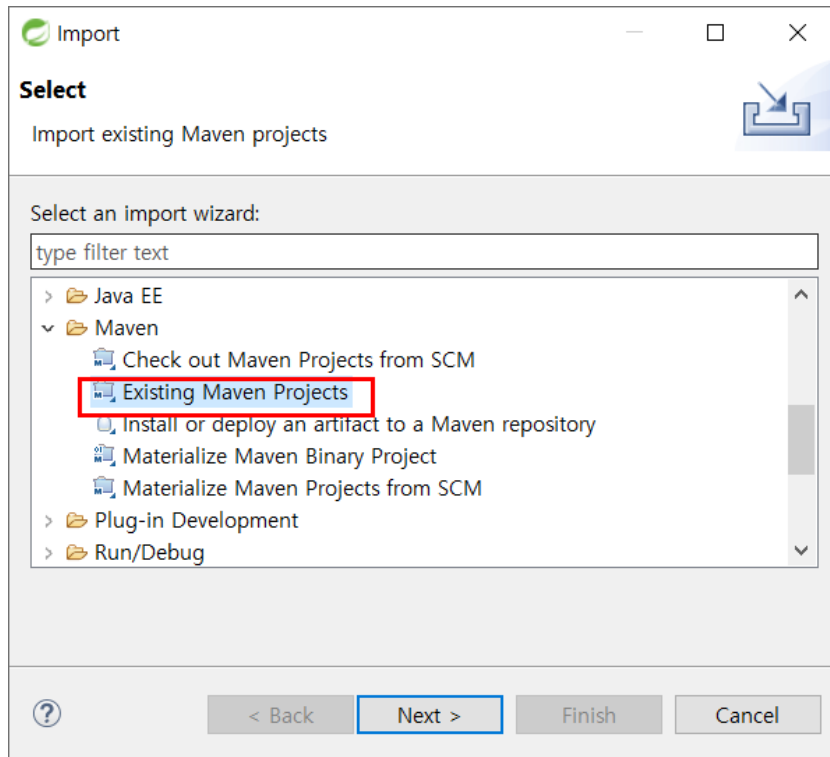
1. Hello World! 출력

2. 프로젝트 import

생성한 프로젝트를 다운로드 한 후 workspace 에 복사 및 압축을 풀고 import 한다.



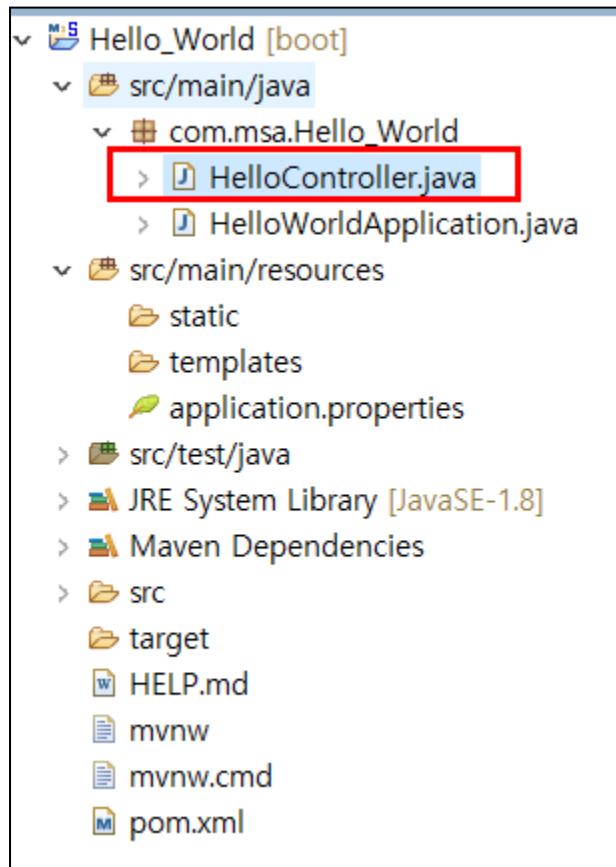
1. Hello World! 출력



1. Hello World! 출력

3. API 개발 및 실행

HelloController.java 파일을 생성한다.



```
1 package com.msa.Hello_World;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4
5
6
7 @RestController
8 public class HelloController {
9
10     @RequestMapping(value="/hello", method=RequestMethod.GET)
11     public String hello() {
12         return "Hello World";
13     }
14 }
```

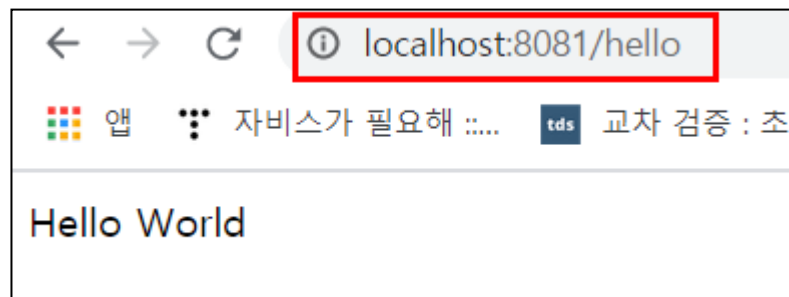
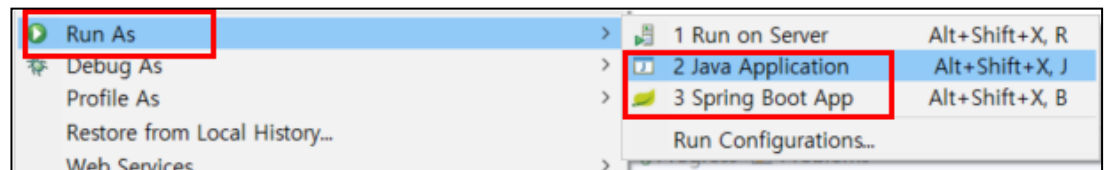
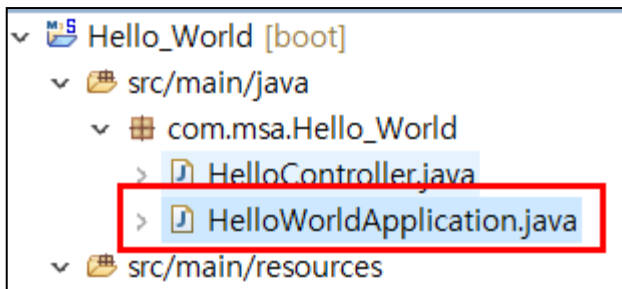
1. Hello World! 출력

4. Server 포트 변경

application.properties 파일에서 server 포트를 8081 값으로 지정한다.

```
application.properties
1 server.port=8081
2
```

5. 실행후 요청



2. Swagger 설정

1) Swagger 개요

Java API에 대한 Java Doc이 있는 것처럼 Rest API의 경우도 문서화가 필요한데, 현재까지 가장 많이 사용하고 있는 것이 SpringFox 라는 오픈소스 프로젝트에서 만든 Swagger이다.

2) Swagger 기능

개발자가 만든 REST API 서비스를 설계, 빌드, 문서화할 수 있도록 하는 오픈소스 프로젝트로서 대표적인 기능은 다음과 같다.

- API 디자인
- API 빌드
- API 문서화
- API 테스트 (*)
- API 표준화

API를 통해 Parameter, 응답정보, 예제등과 같은 스펙(Spec) 전달이 용이하고, 실제 사용되는 Parameter로 테스트까지도 가능하다.

2. Swagger 설정

3) Swagger 의존성 검색

http://mvnrepository.com 에서 springfox 검색어 지정



2. Swagger 설정 (2.9.2 버전으로 통일)

가. Swagger2

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
```

나. Swagger-UI

Maven Gradle SBT Ivy Grape Leiningen

```
<!-- https://mvnrepository.com/artifact/io.springfox
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

2. Swagger 설정

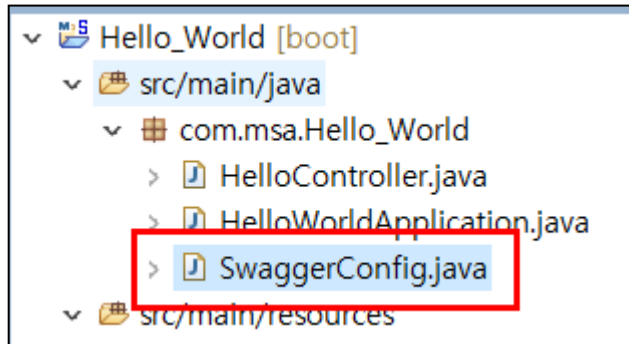
4) Swagger 의존성 설정

pom.xml에 다음과 같이 설정한다.

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

2. Swagger 설정

5) SwaggerConfig.java 파일 추가



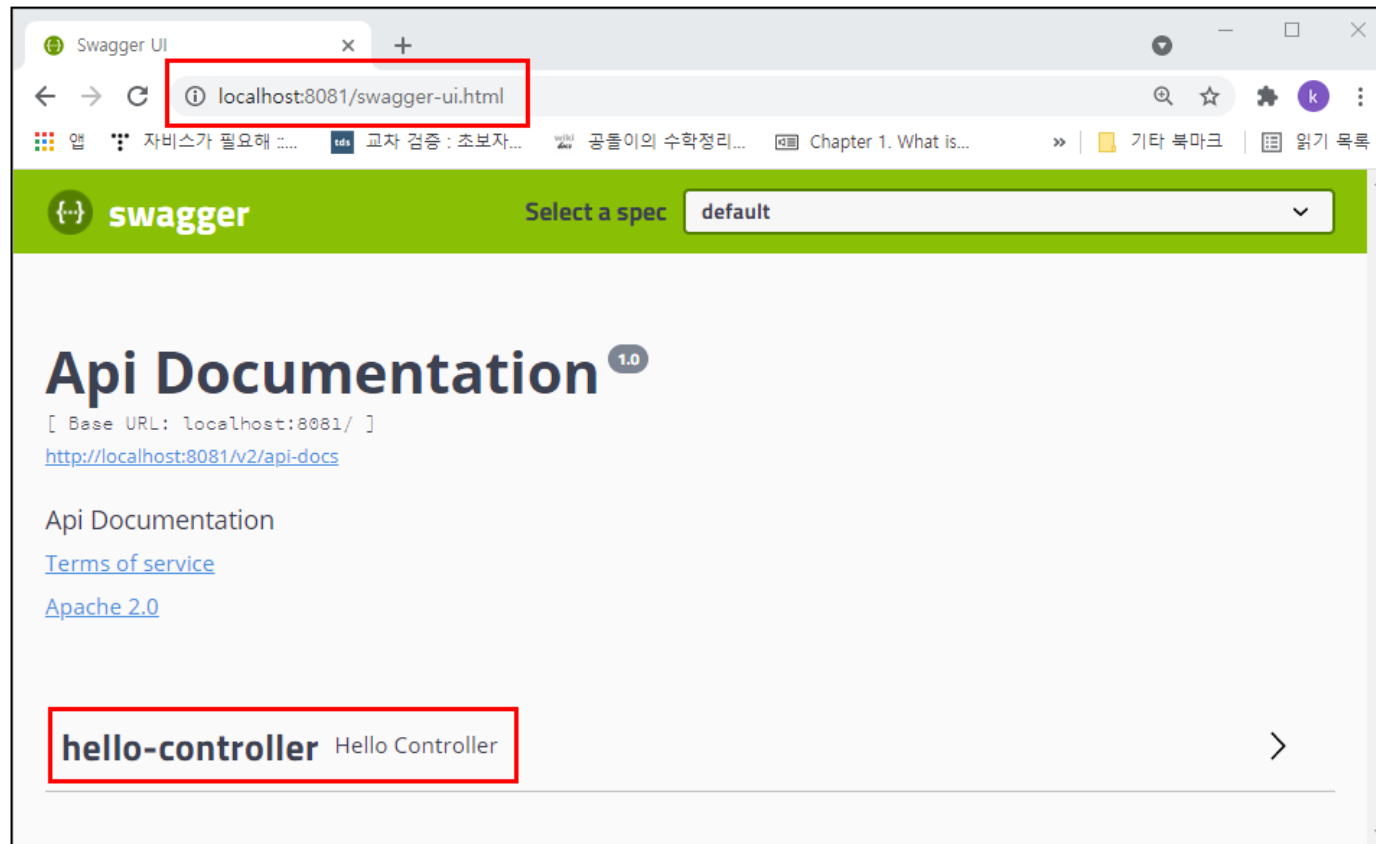
```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.msa.Hello_World"))
            .paths(PathSelectors.any())
            .build();
    }
}
```

2. Swagger 설정

7) Swagger UI 실행 및 테스트 진행

<http://localhost:8081/swagger-ui.html>



2. Swagger 설정

hello-controller Hello Controller ▼

GET /hello hello

Parameters Try it out



Parameters Cancel

No parameters

Execute



Code	Details
200	<div>Response body</div> <div>Hello World</div>

4장. MSA 구현 실습-1

(Customer 서비스 개발)

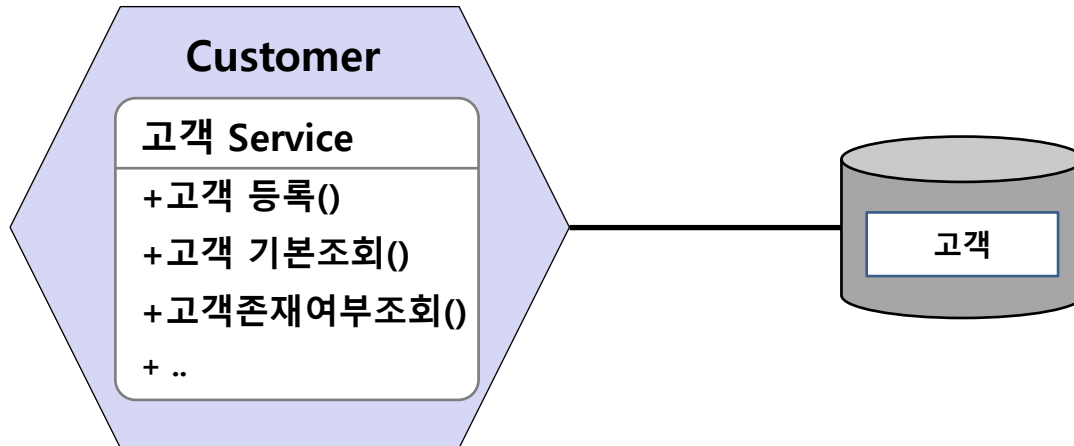


Customer 마이크로서비스 구축
ecommerce_customer 프로젝트

1. Customer 마이크로서비스 구축

1) 개요

Customer Microservice의 고객 등록과 고객 기본조회 API를 개발하여 완성하고, 개발한 API 테스트를 진행한다.



1. Customer 마이크로서비스 구축

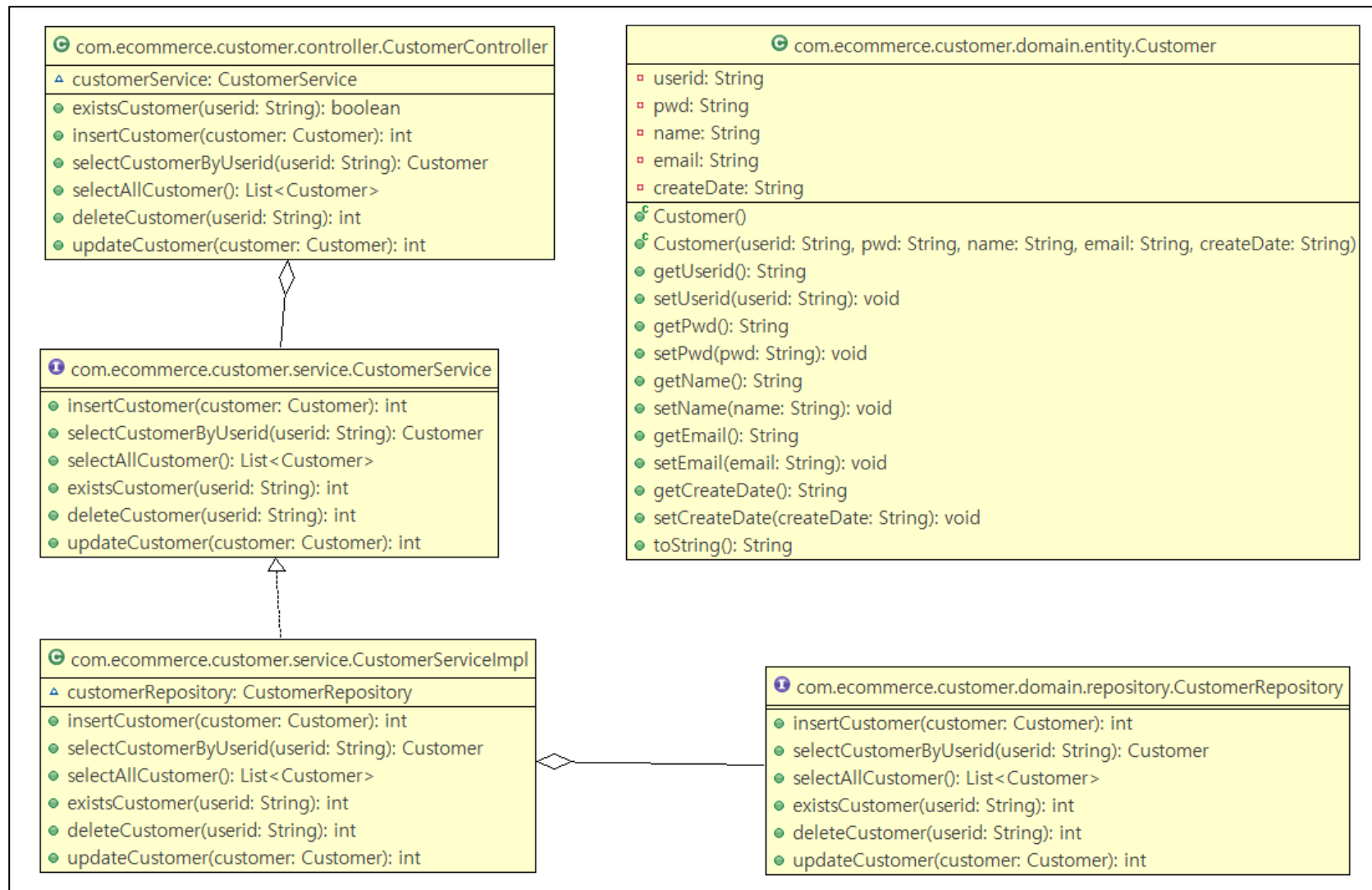
2) 요구사항

아래의 요구사항에 만족하도록 API를 개발한다.

API 명	URL	HTTP method	annotation
고객존재여부	/rest/customers/{userid}/exists	GET	@PathVariable
고객 등록	/rest/customer	POST	@RequestBody
고객 조회	/rest/customers/{userid}	GET	@PathVariable
고객전체조회	/rest/customers	GET	
고객 삭제	/rest/customers/{userid}	DELETE	@PathVariable
고객 수정	/rest/customer	PUT	@RequestBod

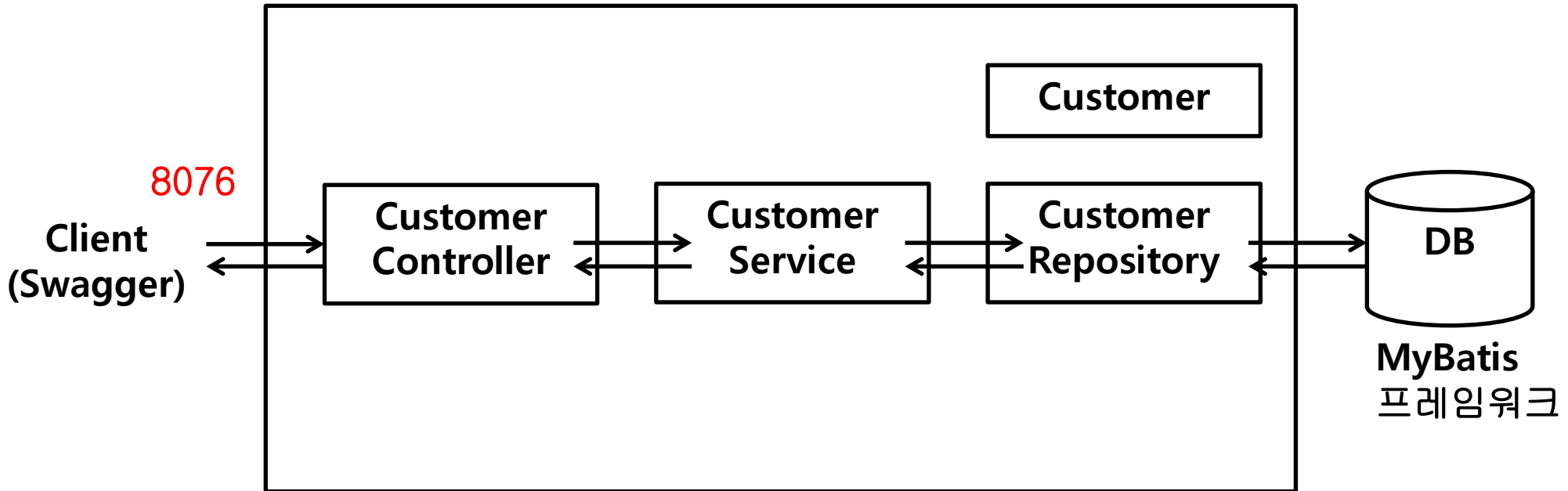
1. Customer 마이크로서비스 구축

3) 클래스 다이어그램



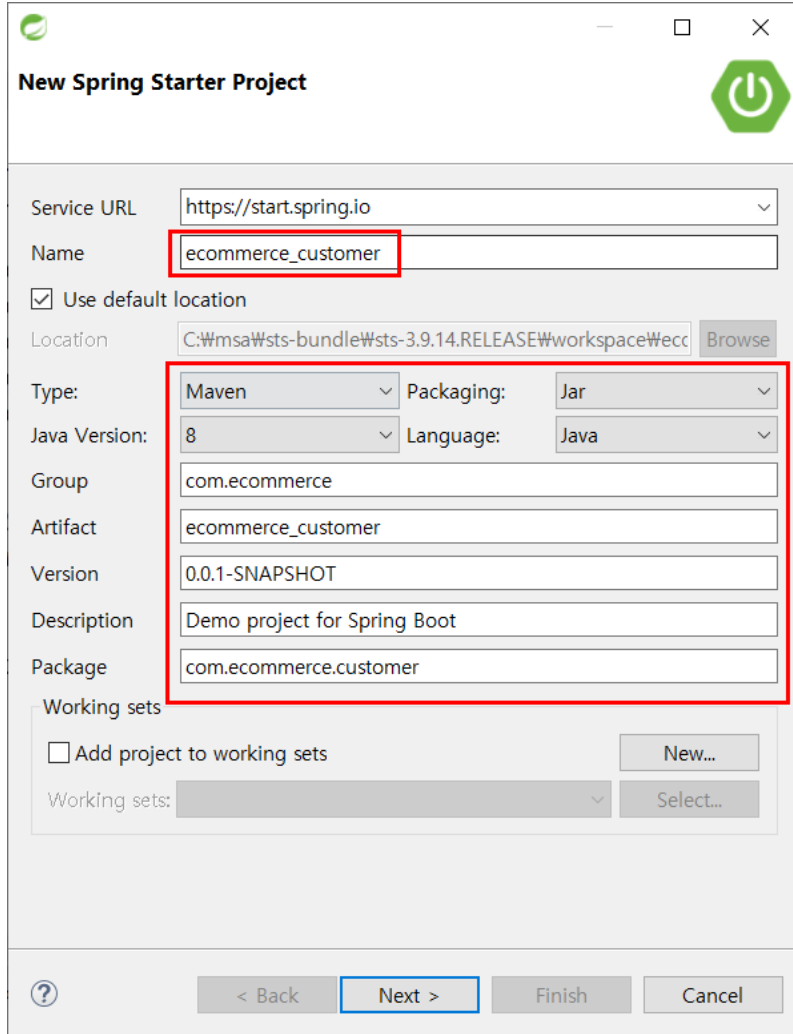
1. Customer 마이크로서비스 구축

4) 아키텍처



2. ecommerce_customer 프로젝트

1) ecommerce_customer 프로젝트 생성



New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

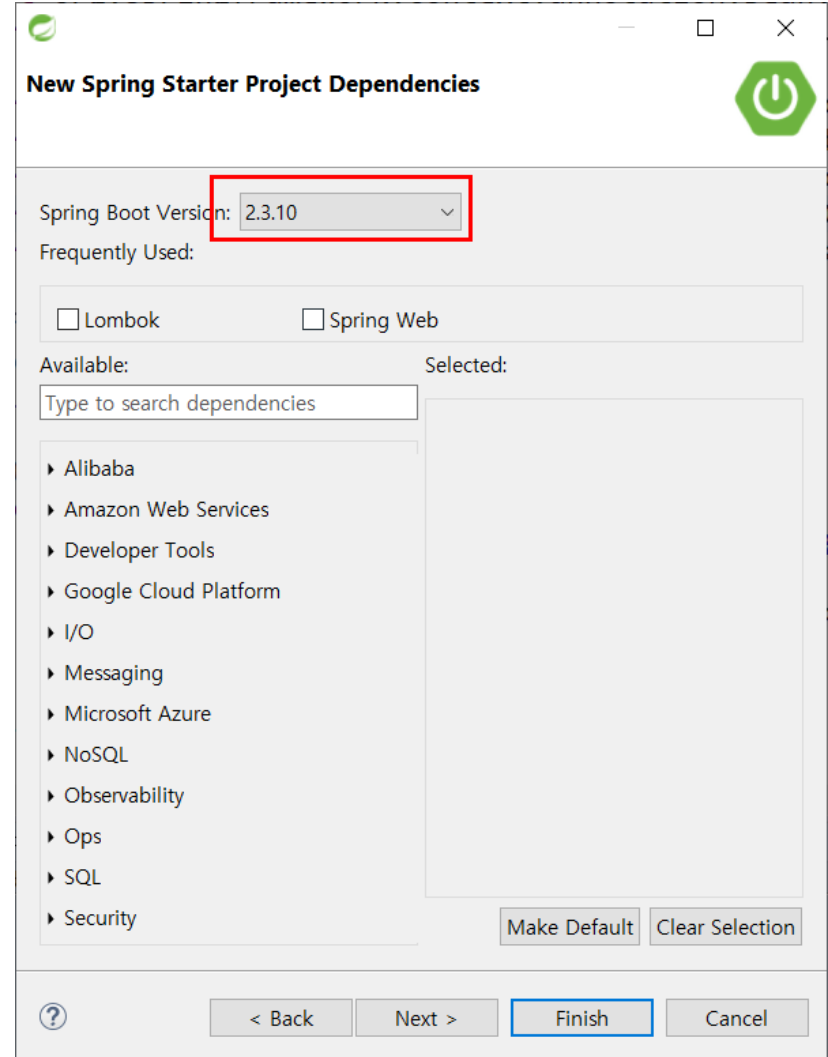
Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☐ Lombok ☐ Spring Web

Available:

Selected:

- ▶ Alibaba
- ▶ Amazon Web Services
- ▶ Developer Tools
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ SQL
- ▶ Security

2. ecommerce_customer 프로젝트

2) pom.xml 파일에 dependency 설정

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>2.1.4</version>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>

<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>2.4.1</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

2. ecommerce_customer 프로젝트

3) com.ecommerce.customer.domain.entity.Customer 빈 생성

```
@Alias("Customer")
public class Customer implements Serializable{

    private String userid;
    private String pwd;
    private String name;
    private String email;
    private String createDate;
```

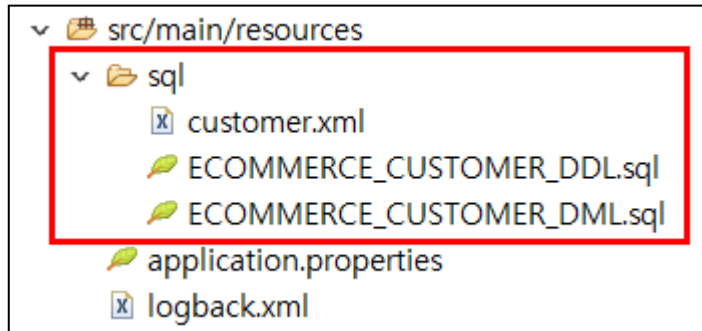
4) com.ecommerce.customer.domain.repository.CustomerRepository 생성

```
@Mapper
public interface CustomerRepository {

    public int insertCustomer(Customer customer) throws Exception;
    public Customer selectCustomerByUserId(String userid) throws Exception;
    public List<Customer> selectAllCustomer() throws Exception;
    public int existsCustomer(String userid) throws Exception;
    public int deleteCustomer(String userid) throws Exception;
    public int updateCustomer(Customer customer) throws Exception;
}
```

2. ecommerce_customer 프로젝트

5) 제공된 sql 파일과 MyBatis mapper 파일 복사



```
<mapper namespace="com.ecommerce.customer.domain.repository.CustomerRepository">
  <insert id="insertCustomer" parameterType="com.ecommerce.customer.domain.entity.Customer">
    insert into MSA_CUSTOMER ( userid, pwd, name, email)
    values (#{userid}, #{pwd}, #{name}, #{email})
  </insert>

  <select id="selectCustomerByUserid" resultType="Customer"
    parameterType="string">
    select userid, pwd, name, email, create_date as createDate
    from MSA_CUSTOMER
    where userid = #{userid}
  </select>
```

2. ecommerce_customer 프로젝트

6) com.ecommerce.customer.service.CustomerService 인터페이스 작성

```
public interface CustomerService {  
  
    public int insertCustomer(Customer customer) throws Exception;  
    public Customer selectCustomerByUserid(String userid) throws Exception;  
    public List<Customer> selectAllCustomer() throws Exception;  
    public int existsCustomer(String userid) throws Exception;  
    public int deleteCustomer(String userid) throws Exception;  
    public int updateCustomer(Customer customer) throws Exception;  
}
```

2. ecommerce_customer 프로젝트

7) com.ecommerce.customer.service.CustomerServiceImpl 빈 생성

```
@Service("customerService")
public class CustomerServiceImpl implements CustomerService {

    @Autowired
    CustomerRepository customerRepository;

    @Override
    public int insertCustomer(Customer customer) throws Exception {
        return customerRepository.insertCustomer(customer);
    }

    @Override
    public Customer selectCustomerByUserid(String userid) throws Exception {
        return customerRepository.selectCustomerByUserid(userid);
    }
}
```


2. ecommerce_customer 프로젝트

8) com.ecommerce.customer.controller.CustomerController 빈 생성

```
@RestController
public class CustomerController {

    @Autowired
    CustomerService customerService;

    @ApiOperation(value = "고객존재여부", httpMethod = "GET", notes = "고객존재여부")
    @RequestMapping(value = "/rest/customers/{userid}/exists", method = RequestMethod.GET)
    public boolean existsCustomer(@PathVariable("userid") String userid) throws Exception{
        boolean result = false;
        if(customerService.existsCustomer(userid) > 0) {
            result = true;
        }
        return result;
    }

    @ApiOperation(value = "고객 등록", httpMethod = "POST", notes = "고객 등록")
    @PostMapping(value = "/rest/customer")
    public int insertCustomer(@RequestBody Customer customer) throws Exception{
        customer.setUserId(UUID.randomUUID().toString());
        return customerService.insertCustomer(customer);
    }
}
```

2. ecommerce_customer 프로젝트

9) com.ecommerce.customer.config.SwaggerConfig 빈 생성

```
package com.ecommerce.customer.config;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors
                .basePackage("com.ecommerce.customer.controller"))
            .paths(PathSelectors.any())
            .build();
    }
}
```

2. ecommerce_customer 프로젝트

10) application.properties 파일에 속성값 설정

```
server.port=8076
server.servlet.context-path=/ecommerce/customer

#spring
spring.application.name=ecommerce-customer

# PostgreSQL
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.url=jdbc:log4jdbc:postgresql://${POSTGRES}:5432/customer
spring.datasource.username=postgres
spring.datasource.password=admin1234
spring.datasource.schema=classpath:sql/ECOMMERCE_CUSTOMER_DDL.sql
spring.datasource.data=classpath:sql/ECOMMERCE_CUSTOMER_DML.sql
spring.datasource.initialization-mode=always

# MyBatis
mybatis.type-aliases-package=com.ecommerce.customer.domain.entity
mybatis.mapper-locations=classpath:sql/*.xml

# Env
POSTGRES=localhost

# actuator 엔드포인트 노출
# https://docs.spring.io/spring-boot/docs/2.3.10.RELEASE/reference/html/proc
management.endpoints.web.exposure.include=info,health,beans,metrics
```

2. ecommerce_customer 프로젝트

11) 실행 및 Swagger UI 요청

<http://localhost:8076/ecommerce/customer/swagger-ui.html>

customer-controller Customer Controller	
POST	/rest/customer 고객 등록
PUT	/rest/customer 고객 수정
GET	/rest/customers 고객 전체 조회
GET	/rest/customers/{userid} 고객 조회
DELETE	/rest/customers/{userid} 고객 삭제
GET	/rest/customers/{userid}/exists 고객존재여부

5장. MSA 구현 실습-2 (Catalogs 서비스 개발)

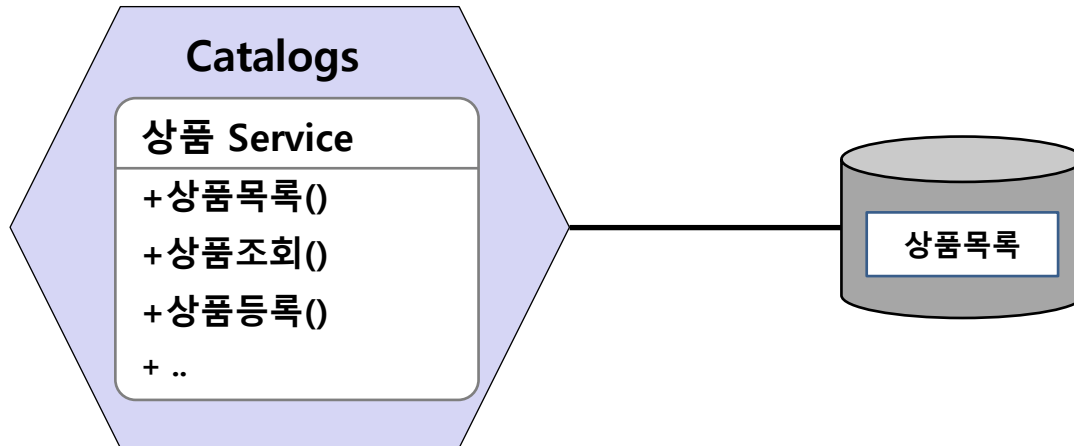


Catalogs 마이크로서비스 구축
ecommerce_catalogs 프로젝트

1. Catalogs 마이크로서비스 구축

1) 개요

Catalogs Microservice의 상품목록과 상품등록 API를 개발하여 완성하고, 개발한 API 테스트를 진행한다.



1. Catalogs 마이크로서비스 구축

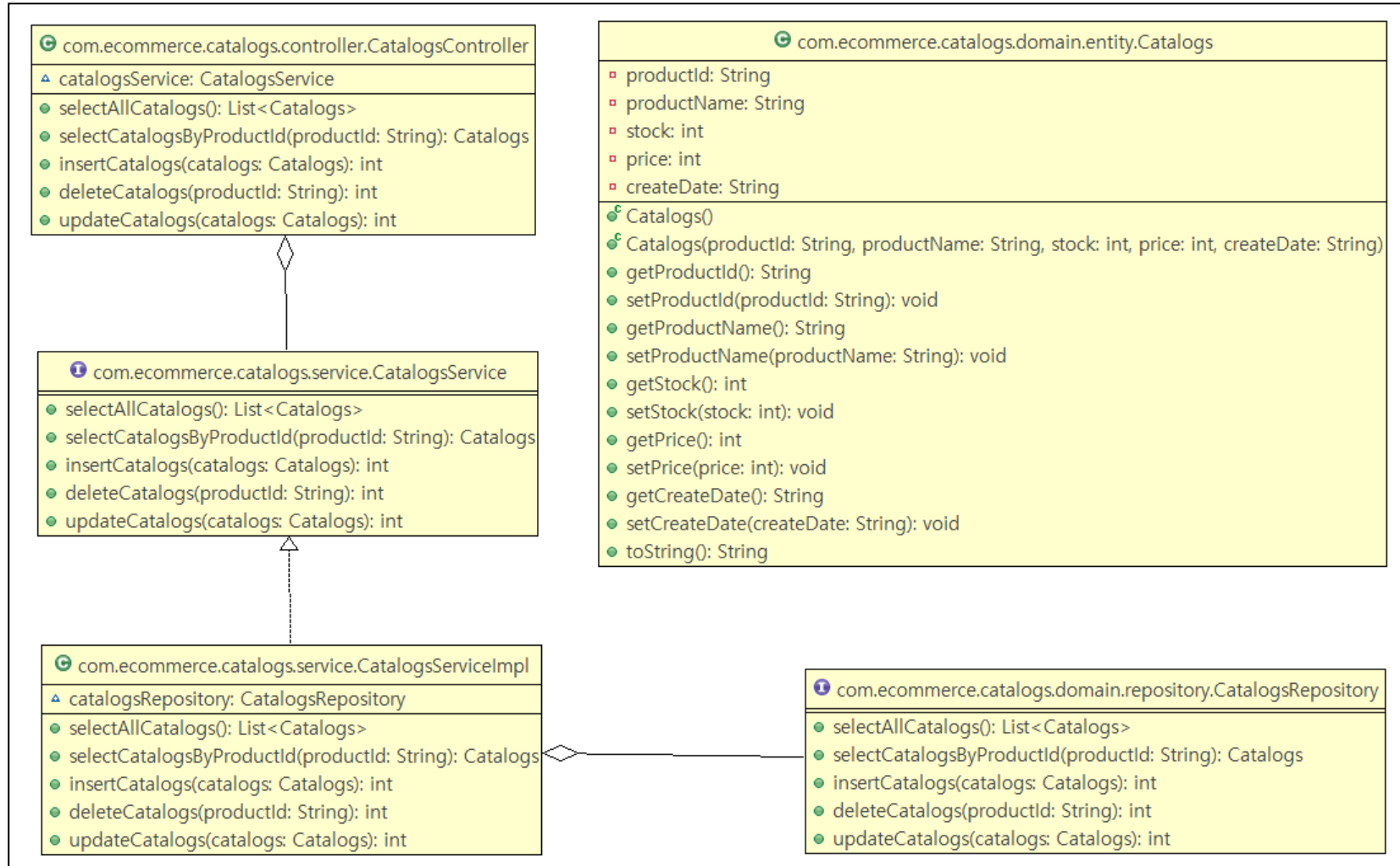
2) 요구사항

아래의 요구사항에 만족하도록 API를 개발한다.

API 명	URL	HTTP method	annotation
상품전체조회	/rest/catalogs	GET	
상품조회	/rest/catalogs/{productId}	GET	@PathVariable
상품등록	/rest/catalog	POST	@RequestBody
상품 삭제	/rest/catalog/{productId}	DELETE	@PathVariable
상품 수정	/rest/catalog	PUT	@RequestBody

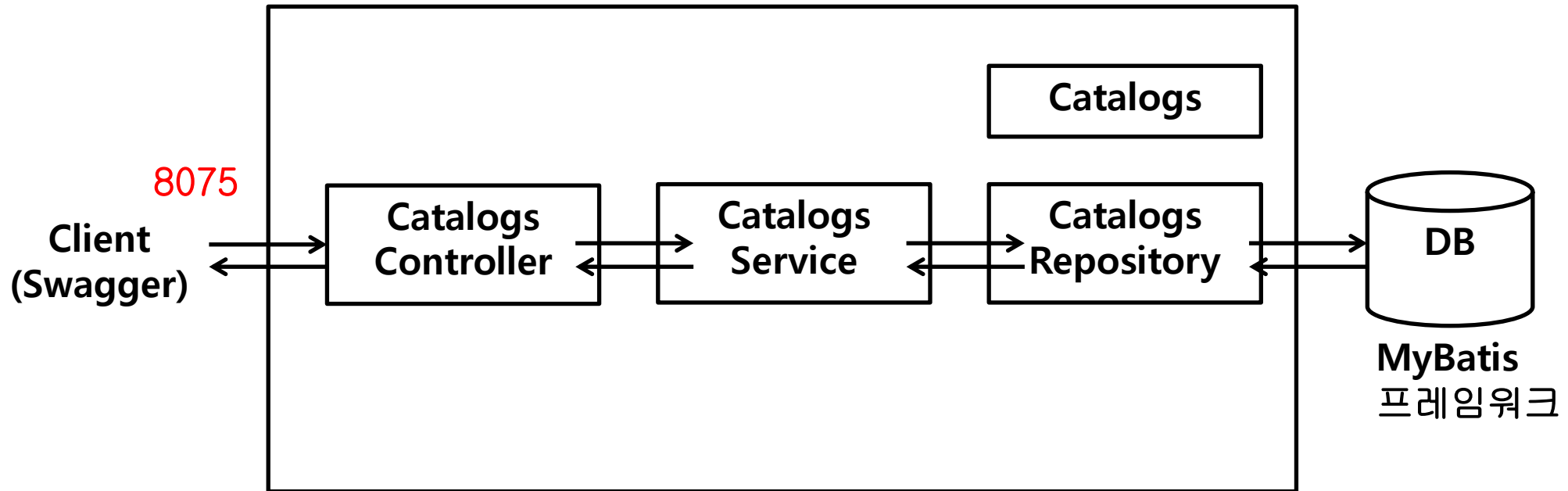
1. Catalogs 마이크로서비스 구축

3) 클래스 다이어그램



1. Catalogs 마이크로서비스 구축

4) 아키텍처



2. ecommerce_catalogs 프로젝트

1) ecommerce_catalogs 프로젝트 생성

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☐ Lombok ☐ Spring Web

Available:

Selected:

- ▶ Alibaba
- ▶ Amazon Web Services
- ▶ Developer Tools
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ SQL
- ▶ Security

2. ecommerce_catalogs 프로젝트

2) pom.xml 파일에 dependency 설정

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>2.1.4</version>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>

<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>2.4.1</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

2. ecommerce_catalogs 프로젝트

3) com.ecommerce.catalogs.domain.entity.Catalogs 빈 생성

```
@Alias("Catalogs")
public class Catalogs {

    private String productId;
    private String productName;
    private int stock;
    private int price;
    private String createDate;
```

4) com.ecommerce.catalogs.domain.repository.CatalogsRepository 생성

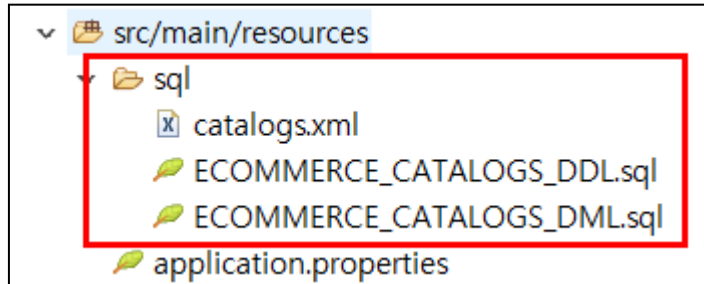
```
@Mapper
public interface CatalogsRepository {

    public List<Catalogs> selectAllCatalogs() throws Exception;
    public Catalogs selectCatalogsByProductId(String productId) throws Exception;
    public int insertCatalogs(Catalogs catalogs) throws Exception;
    public int deleteCatalogs(String productId) throws Exception;
    public int updateCatalogs(Catalogs catalogs) throws Exception;

}
```

2. ecommerce_catalogs 프로젝트

5) 제공된 sql 파일과 MyBatis mapper 파일 복사



```
<mapper namespace="com.ecommerce.catalogs.domain.repository.CatalogsRepository">

    <select id="selectAllCatalogs" resultType="Catalogs">
        select product_id, product_name, stock, price, create_date
        from MSA_CATALOGS
    </select>

    <select id="selectCatalogsByProductId" resultType="Catalogs" parameterType="string">
        select product_id, product_name, stock, price, create_date
        from MSA_CATALOGS
        where product_id = #{product_id}
    </select>

</mapper>
```

2. ecommerce_catalogs 프로젝트

6) com.ecommerce.catalogs.service.CatalogsService 인터페이스 작성

```
public interface CatalogsService {  
  
    public List<Catalogs> selectAllCatalogs() throws Exception;  
    public Catalogs selectCatalogsByProductId(String productId) throws Exception;  
    public int insertCatalogs(Catalogs catalogs) throws Exception;  
    public int deleteCatalogs(String productId) throws Exception;  
    public int updateCatalogs(Catalogs catalogs) throws Exception;  
}
```

7) com.ecommerce.catalogs.service.CatalogsServiceImpl 빈 생성

```
@Service("catalogsService")  
public class CatalogsServiceImpl implements CatalogsService {  
  
    @Autowired  
    CatalogsRepository catalogsRepository;  
  
    @Override  
    public List<Catalogs> selectAllCatalogs() throws Exception {  
        return catalogsRepository.selectAllCatalogs();  
    }  
  
    @Override  
    public Catalogs selectCatalogsByProductId(String productId) throws Exception {  
        return catalogsRepository.selectCatalogsByProductId(productId);  
    }  
}
```

2. ecommerce_catalogs 프로젝트

8) com.ecommerce.catalogs.controller.CatalogsController 빈 생성

```
@RestController
public class CatalogsController {

    @Autowired
    CatalogsService catalogsService;

    @ApiOperation(value = "상품 전체 조회", httpMethod = "GET", notes = "상품 전체 조회")
    @GetMapping(value="/rest/catalogs")
    public List<Catalogs> selectAllCatalogs() throws Exception{
        return catalogsService.selectAllCatalogs();
    }

    @ApiOperation(value = "상품 조회", httpMethod = "GET", notes = "상품 조회")
    @GetMapping(value="/rest/catalogs/{productId}")
    public Catalogs selectCatalogsByProductId(@PathVariable("productId") String productId)
        return catalogsService.selectCatalogsByProductId(productId);
    }
}
```

2. ecommerce_catalogs 프로젝트

9) com.ecommerce.catalogs.config.SwaggerConfig 빈 생성

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors
                .basePackage("com.ecommerce.catalogs.controller"))
            .paths(PathSelectors.any())
            .build();
    }
}
```


2. ecommerce_catalogs 프로젝트

10) application.properties 파일에 속성값 설정

```
server.port=8075
server.servlet.context-path=/ecommerce/catalogs

#spring
spring.application.name=ecommerce-catalogs

# PostgreSQL
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.url=jdbc:log4jdbc:postgresql://${POSTGRES}:5432/catalogs
spring.datasource.username=postgres
spring.datasource.password=admin1234
spring.datasource.schema=classpath:sql/ECOMMERCE_CATALOGS_DDL.sql
spring.datasource.data=classpath:sql/ECOMMERCE_CATALOGS_DML.sql
spring.datasource.initialization-mode=always

# MyBatis
mybatis.type-aliases-package=com.ecommerce.catalogs.domain.entity
mybatis.mapper-locations=classpath:sql/*.xml
mybatis.configuration.map-underscore-to-camel-case=true

# Env
POSTGRES=localhost

# actuator \uC5D4\uB4DC\uD3EC\uC778\uD2B8 \uB178\uC9C0
# https://docs.spring.io/spring-boot/docs/2.3.10.RELEASE/reference/html/production-environment.html
management.endpoints.web.exposure.include=info,health,beans,metrics
```

2. ecommerce_catalogs 프로젝트

11) 실행 및 Swagger UI 요청

<http://localhost:8075/ecommerce/catalogs/swagger-ui.html>

catalogs-controller Catalogs Controller	
POST	/rest/catalog 상품 등록
PUT	/rest/catalog 상품 수정
DELETE	/rest/catalog/{productId} 상품 삭제
GET	/rest/catalogs 상품 전체 조회
GET	/rest/catalogs/{productId} 상품 조회

6장. MSA 구현 실습-3 (Order 서비스 개발)

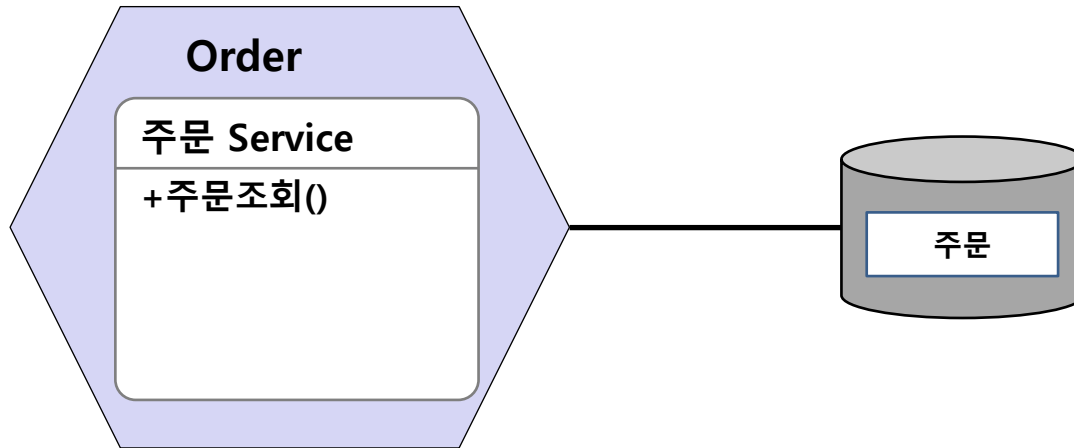


Order 마이크로서비스 구축
ecommerce_order 프로젝트

1. Order 마이크로서비스 구축

1) 개요

Order Microservice의 주문조회 API를 개발하여 완성하고, 개발한 API 테스트를 진행한다.



1. Order 마이크로서비스 구축

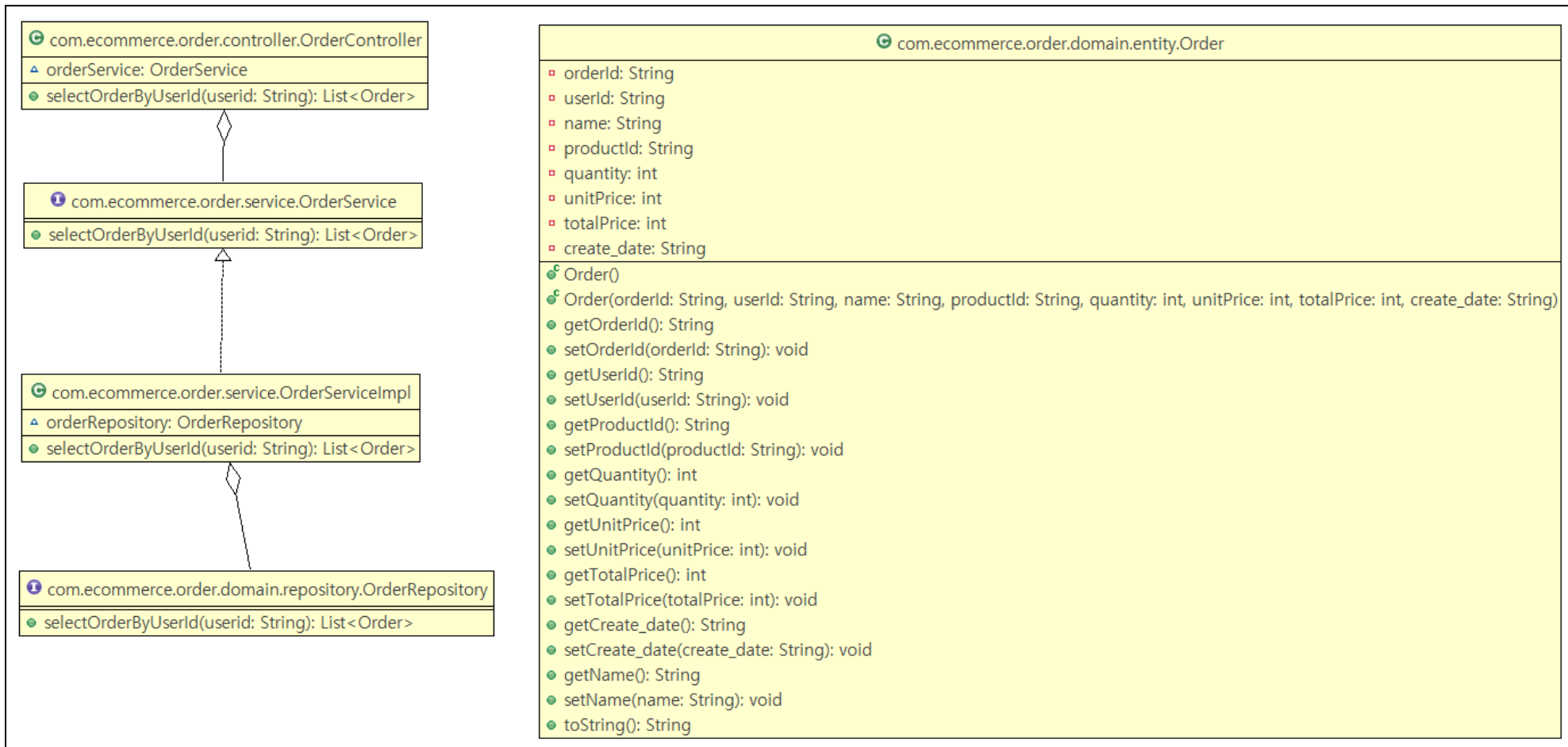
2) 요구사항

아래의 요구사항에 만족하도록 API를 개발한다.

API 명	URL	HTTP method	Annotation
주문조회	/rest/orders/{userid}	GET	@PathVariable

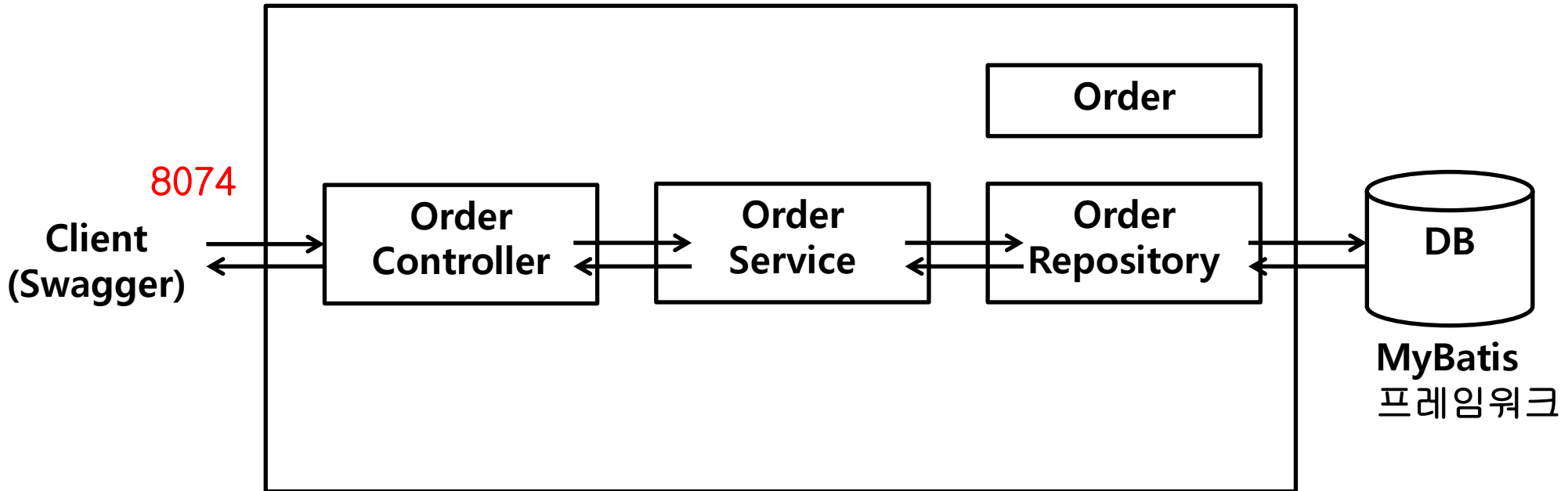
1. Order 마이크로서비스 구축

3) 클래스 다이어그램



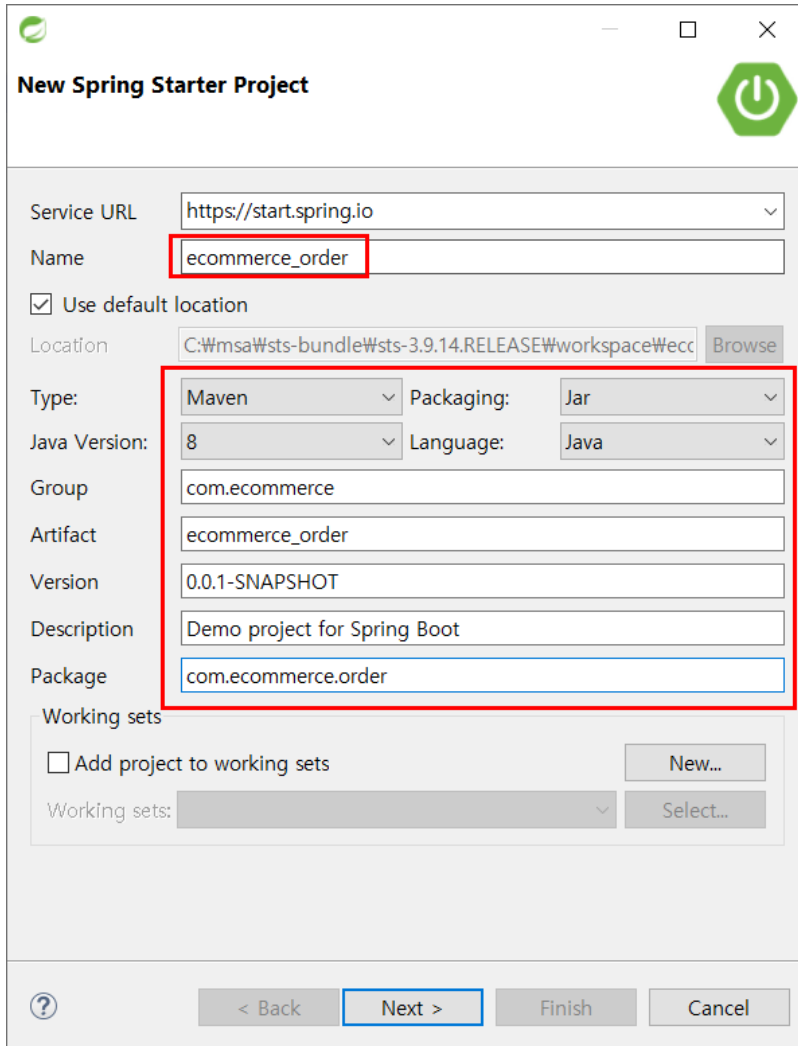
1. Order 마이크로서비스 구축

4) 아키텍처



2. ecommerce_order 프로젝트

1) ecommerce_order 프로젝트 생성



New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

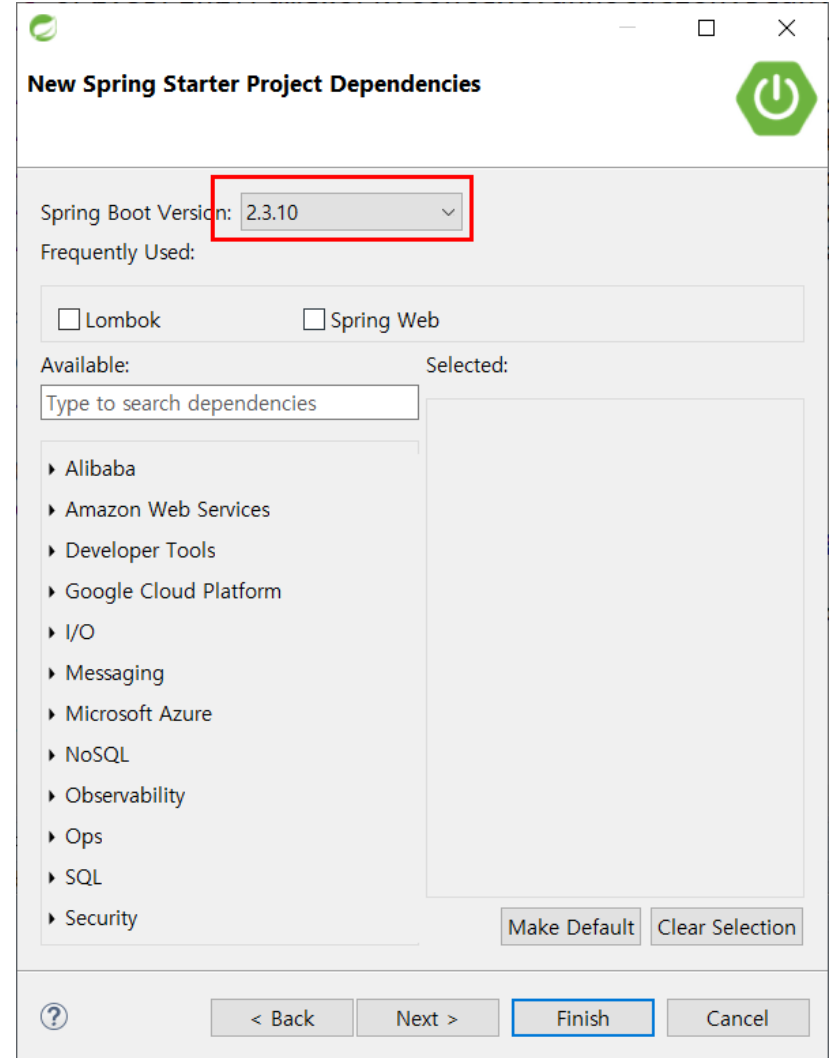
Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☐ Lombok ☐ Spring Web

Available:

Selected:

- ▶ Alibaba
- ▶ Amazon Web Services
- ▶ Developer Tools
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ SQL
- ▶ Security

2. ecommerce_order 프로젝트

2) pom.xml 파일에 dependency 설정

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>2.1.4</version>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>

<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>2.4.1</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

2. ecommerce_order 프로젝트

3) com.ecommerce.order.domain.entity.**Order** 빈 생성

```
@Alias("Order")
public class Order implements Serializable{

    private String orderId; //pk
    private String userId;
    private String name;
    private String productId;
    private int quantity;
    private int unitPrice;
    private int totalPrice;
    private String create_date;
```

4) com.ecommerce.order.domain.repository.**OrderRepository** 생성

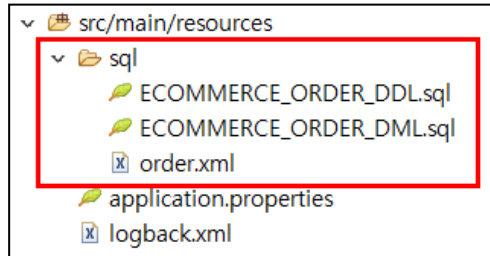
```
@Mapper
public interface OrderRepository {

    public List<Order> selectOrderByUserId(String userid) throws Exception;

}
```

2. ecommerce_order 프로젝트

5) 제공된 sql 파일과 MyBatis mapper 파일 복사



```
<mapper namespace="com.ecommerce.order.domain.repository.OrderRepository">

    <select id="selectOrderByUserId" resultType="Order" parameterType="string">
        select orderId, userId, productId, quantity, unitPrice, totalPrice, create_date
        from MSA_ORDER
        where userId = #{userId}
    </select>
</mapper>
```

2. ecommerce_order 프로젝트

6) com.ecommerce.order.service.OrderService 인터페이스 작성

```
public interface OrderService {  
    public List<Order> selectOrderByUserId(String userid) throws Exception;  
}
```

7) com.ecommerce.order.service.OrderServiceImpl 빈 생성

```
@Service("orderService")  
public class OrderServiceImpl implements OrderService {  
    @Autowired  
    OrderRepository orderRepository;  
  
    @Override  
    public List<Order> selectOrderByUserId(String userid) throws Exception {  
        return orderRepository.selectOrderByUserId(userid);  
    }  
}
```

2. ecommerce_order 프로젝트

8) com.ecommerce.order.controller.OrderController 빈 생성

```
@RestController
public class OrderController {

    @Autowired
    OrderService orderService;

    @ApiOperation(value = "주문 조회", httpMethod = "GET", notes = "주문 조회")
    @GetMapping(value="/rest/orders/{userid}")
    public List<Order> selectOrderByUserId(@PathVariable("userid") String userid) throws Exception{
        return orderService.selectOrderByUserId(userid);
    }
}
```

2. ecommerce_order 프로젝트

9) com.ecommerce.order.config.SwaggerConfig 빈 생성

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors
                .basePackage("com.ecommerce.order.controller"))
            .paths(PathSelectors.any())
            .build();
    }
}
```

2. ecommerce_order 프로젝트

10) application.properties 파일에 속성값 설정

```
server.port=8074
server.servlet.context-path=/ecommerce/order

#spring
spring.application.name=ecommerce-order

# PostgreSQL
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.url=jdbc:log4jdbc:postgresql://${POSTGRES}:5432/order
spring.datasource.username=postgres
spring.datasource.password=admin1234
spring.datasource.schema=classpath:sql/ECOMMERCE_ORDER_DDL.sql
spring.datasource.data=classpath:sql/ECOMMERCE_ORDER_DML.sql
spring.datasource.initialization-mode=always

# MyBatis
mybatis.type-aliases-package=com.ecommerce.order.domain.entity
mybatis.mapper-locations=classpath:sql/*.xml

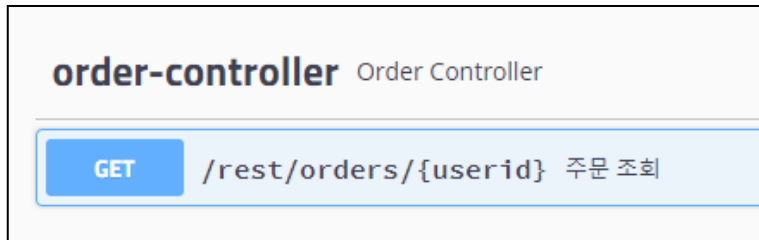
# Env
POSTGRES=localhost

# actuator \uC5D4\uB4DC\uD3EC\uC778\uD2B8 \uB178\uC9C
# https://docs.spring.io/spring-boot/docs/2.3.10.RELEASE/reference/html/pro
management.endpoints.web.exposure.include=info,health,beans,metrics
```

2. ecommerce_order 프로젝트

11) 실행 및 Swagger UI 요청

<http://localhost:8074/ecommerce/order/swagger-ui.html>





감사합니다.
