목차

0. Lombok**/JPA 연관관계:** Lombok 설치.pdf/JPA 연관관계실습.pdf/JPA 연관관계실습.mdj

1. Spring Legacy + JDBC/SPRINGJDBC + JSP

2. Spring Legacy + MyBatis + JSP

3. Spring Legacy + JPA + JSP

4. Spring boot + JDBC/SPRINGJDBC + JSP

5. Spring boot + MyBatis + JSP

6. Spring boot + JPA + Thymeleaf(Spring boot default)

7. Spring boot + MyBatis + Thymeleaf(Spring boot default)

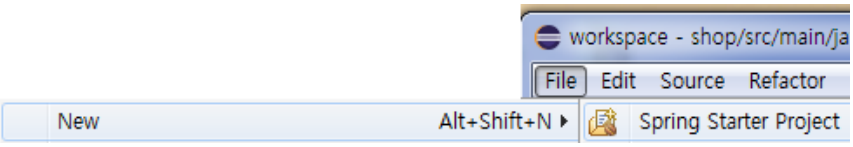
8. Spring boot + JPA + JSP

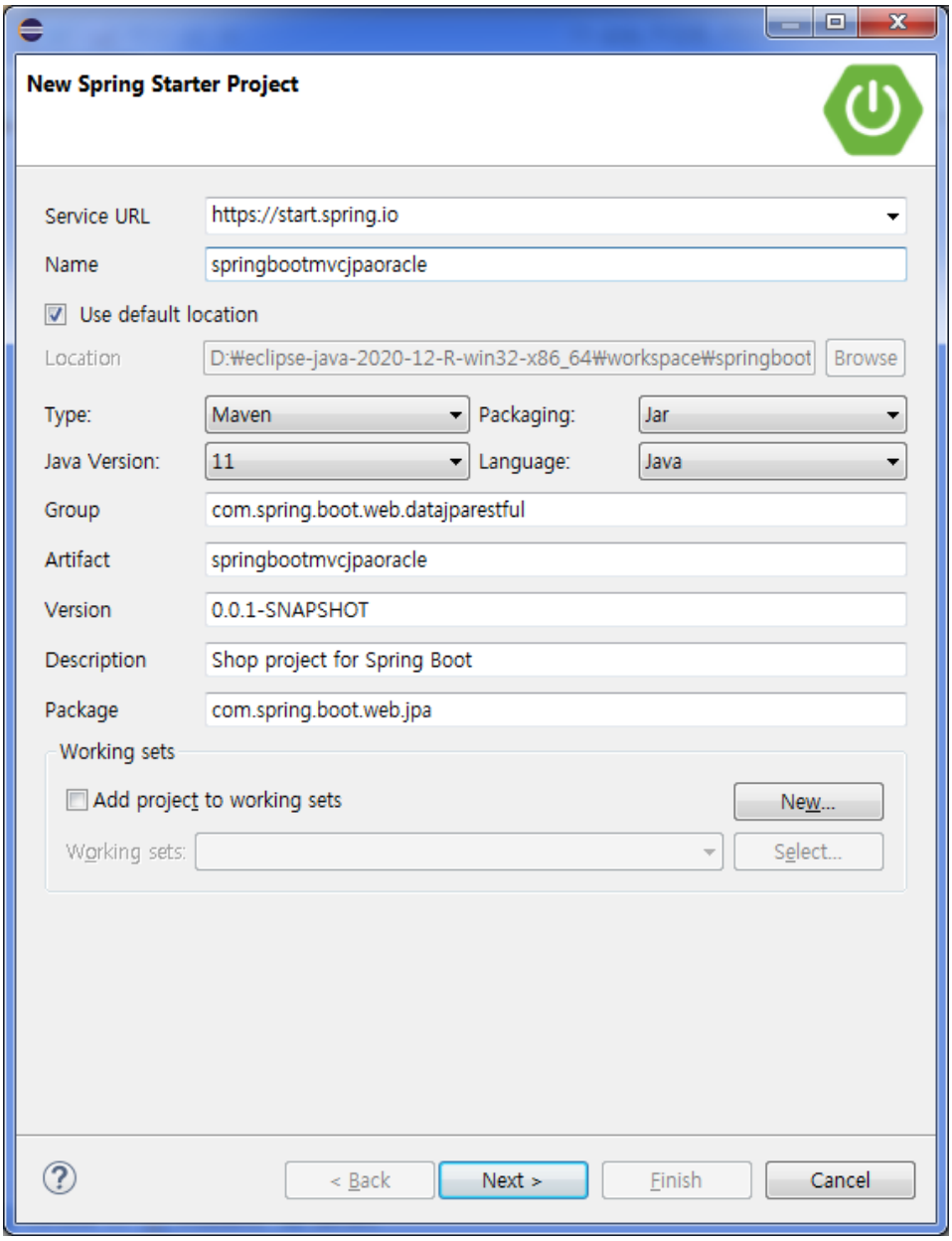
8. Spring Boot + JPA + JSP

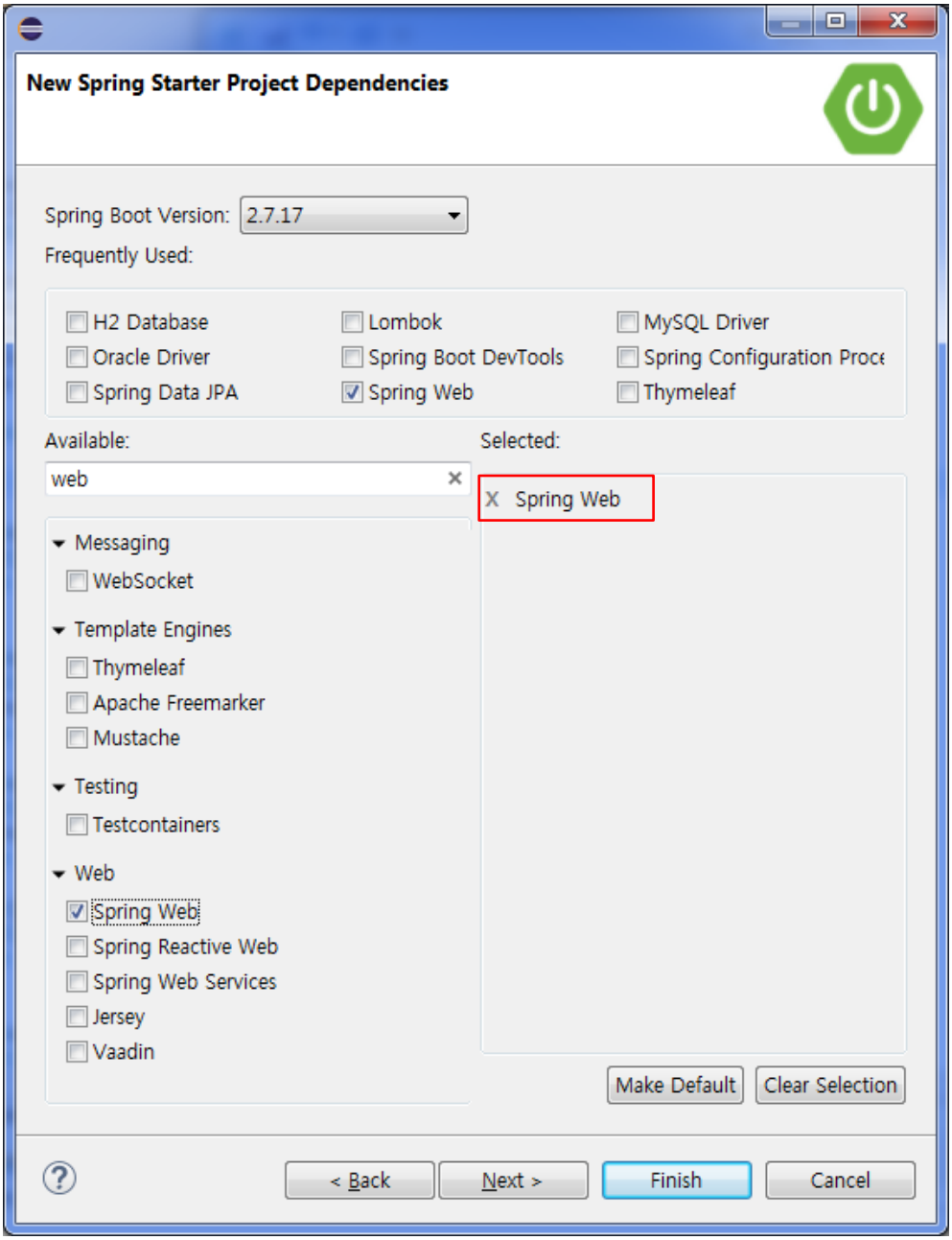
8.1. JPA 연관관계 설정: JPA 연관관계실습.pdf 참조

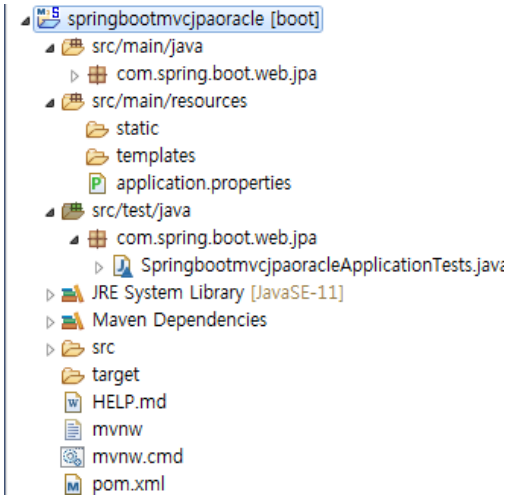
8.2. 스프링 부트 웹 프로젝트 만들기(스프링부트_웹실습 2.pdf 참조)

가. 프로젝트 생성





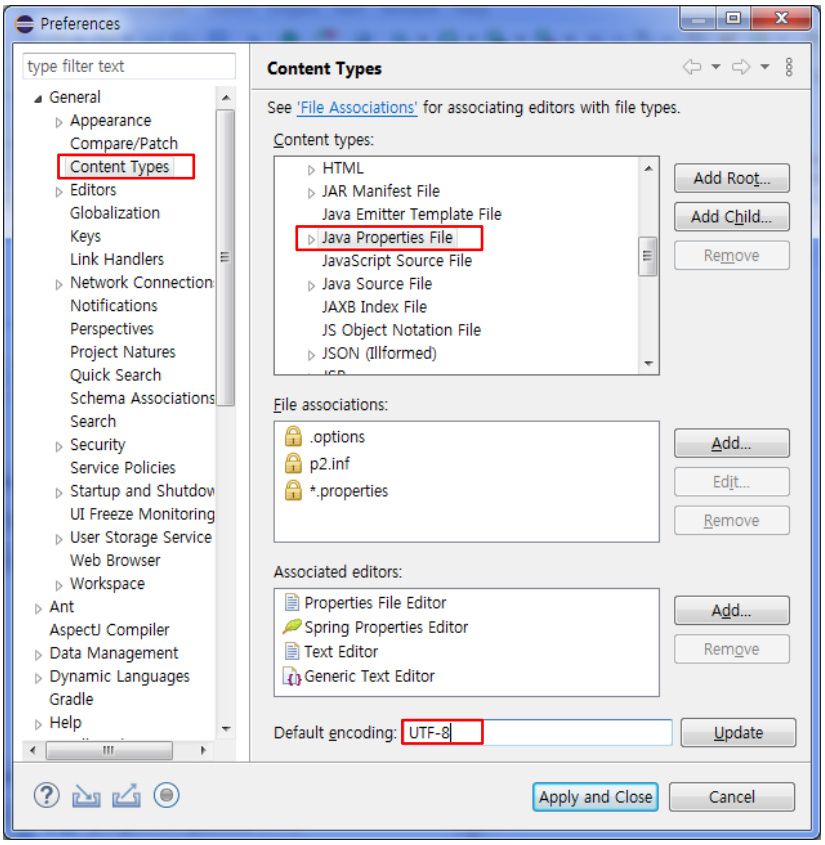

New Spring Starter Project Dependencies
Spring Boot Version: 2.7.17
Frequently Used:
H2 Database
Lombok
MySQL Driver
Oracle Driver
Spring Boot DevTools
Spring Configuration Proce
Spring Data JPA
Spring Web
Thymeleaf
Available:
Selected:
web
X Spring Web
Messaging
WebSocket
Template Engines
Thymeleaf
Apache Freemarker
Mustache
Testing
Testcontainers
Web
Spring Web
Spring Reactive Web
Spring Web Services
Jersey
Vaadin
Make Default
Clear Selection
< Back
Next >
Finish
Cancel


springbootmvcjpaoracle [boot]
src/main/java
com.spring.boot.web.jpa
src/main/resources
static
templates
application.properties
src/test/java
com.spring.boot.web.jpa
SpringbootmvcjpaoracleApplicationTests.java
JRE System Library [JavaSE-11]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

나. pom.xml 수정: jsp 처리 모듈 추가 -> **Update Maven**

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
</dependency>
```

다. 설정파일(application.properties) 한글 깨짐 방지 설정: window > preferences

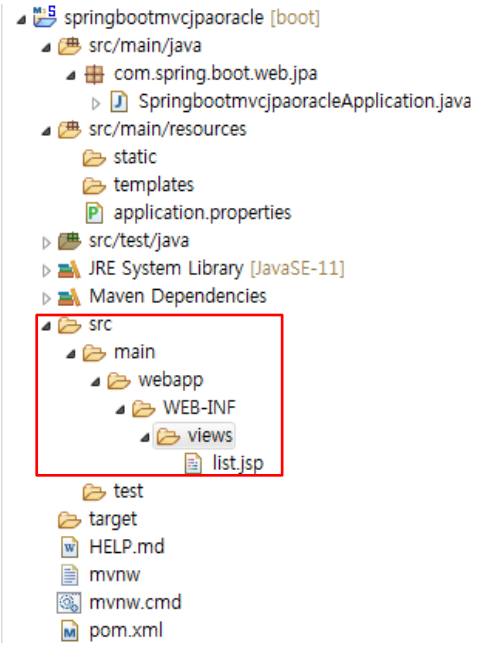


라. JSP 설정: src/main/resources/application.properties

```
# 서버포트 지정
server.port=8082

# 컨텍스트 패스 지정
server.servlet.context-path=/jsp

# 뷰 리졸버 설정
```

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.j 네
```

마. JSP 폴더와 테스트 jsp 파일 생성




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
		<title>User List</title>
	</head>
<body>
	<table border="1">
		<tr>
			<th>성명</th>
		</tr>
		<c:forEach var="item" items="${list}">
			<tr><!-- 첫번째 줄 시작 -->
				<td>${item}</td>
			</tr><!-- 첫번째 줄 끝 -->
		</c:forEach>
		<tr>
			<td>${sum}</td>
		</tr>
	</table>
</body>
</html>
```

바. 서블릿 설정 파일 생성:
src/main/java/com/spring/boot/web/jpa/ServletInitializer.java

사. 테스트 컨트롤러 생성: src/main/java/com/spring/boot/web/jpa /TestController.java

아. 테스트: http://localhost:8082/jsp/test?score1=10&score2=20

8.3. 스프링 부트 웹 + JPA(단일 테이블) 프로젝트 만들기

1) JPA , 오라클 드라이버 모듈 추가 및 오라클 버전 수정 -> **Maven Update**

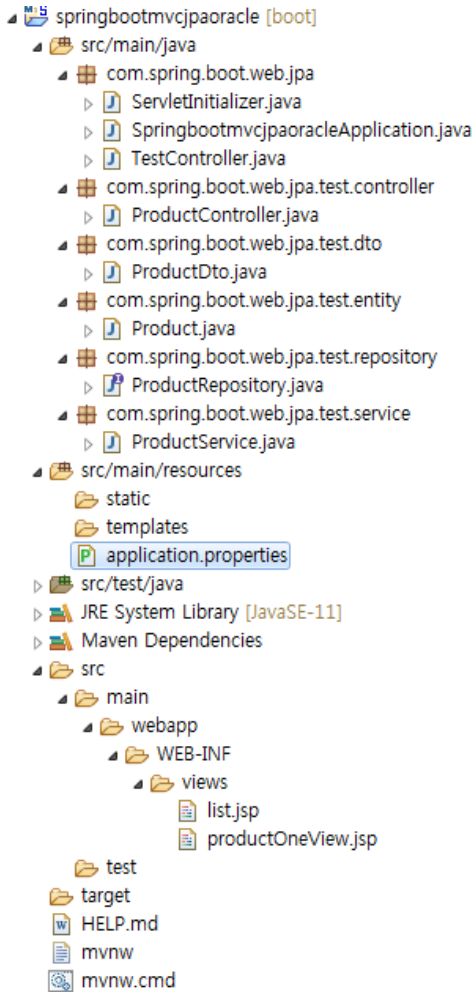
```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>com.oracle.database.jdbc</groupId>
        <artifactId>ojdbc6</artifactId>
        <version>11.2.0.4</version>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
```

```
</dependency>
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
</dependency>
```

2) 데이터소스 및 JPA 설정

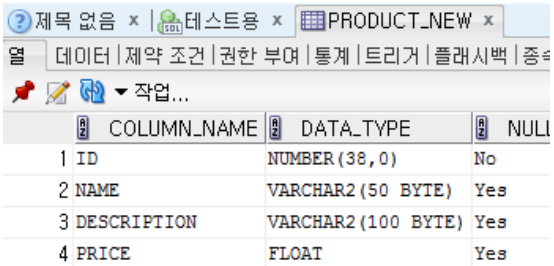
```
# 서버포트 지정
server.port=8082
# 컨텍스트 패스 지정
server.servlet.context-path=/jsp
# 뷰 리졸버 설정
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
# 데이터 소스 설정
spring.datasource.username=oracle_test
spring.datasource.password=woseven
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
# JPA SQL 보기
spring.jpa.properties.hibernate.show_sql=true
# SQL 포맷 보기 좋게
spring.jpa.properties.hibernate.format_sql=true
# SQL문의 파라메터 값 출력
logging.level.org.hibernate.type.descriptor.sql=trace
# 자동 SQL문 생성 방언
spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect
# 테이블 업데이트만
spring.jpa.hibernate.ddl-auto=update
```

3) 테스트 관련 클래스 만들기



4) 소스코드

가. 테이블

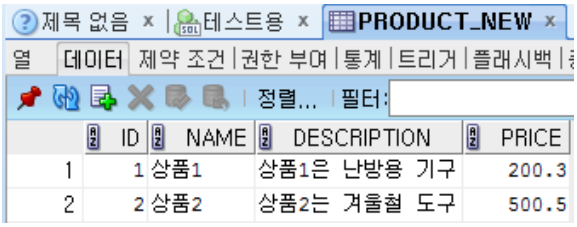


제목 없음 | 테스트용 | PRODUCT_NEW

열 | 데이터 | 제약 조건 | 권한 부여 | 통계 | 트리거 | 플래시백

작업...

	COLUMN_NAME	DATA_TYPE	NULL
1	ID	NUMBER(38,0)	No
2	NAME	VARCHAR2(50 BYTE)	Yes
3	DESCRIPTION	VARCHAR2(100 BYTE)	Yes
4	PRICE	FLOAT	Yes



제목 없음 | 테스트용 | PRODUCT_NEW

열 | 데이터 | 제약 조건 | 권한 부여 | 통계 | 트리거 | 플래시백

정렬... | 필터:

	ID	NAME	DESCRIPTION	PRICE
1	1	상품1	상품1은 난방용 기구	200.3
2	2	상품2	상품2는 겨울철 도구	500.5

나. 엔터티: Product.java

```
package com.spring.boot.web.jpa.test.entity;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;
```

```
@Entity
@Table(name = "product_new")
@Getter
@Setter
@ToString
public class Product {
	@Id
	private int id;
	private String name;
	private String description;
	private float price;
}
```

다. DTO(Data Transfer Object): ProductDto.java

```
package com.spring.boot.web.jpa.test.dto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class ProductDto {
	private int id;
	private String name;
	private String description;
	private float price;
}
```

라. DAO(Data Access Object == Repository): ProductRepository.java

```
package com.spring.boot.web.jpa.test.repository;

import org.springframework.data.repository.CrudRepository;

import com.spring.boot.web.jpa.test.entity.Product;

import java.lang.String;
import java.util.List;

public interface ProductRepository
	extends CrudRepository<Product, Integer> {
}
```

마. 서비스: ProductService.java

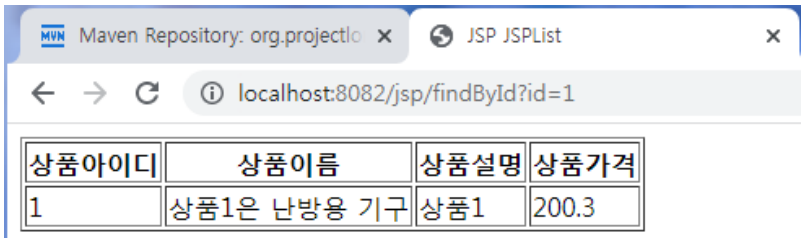
```java
package com.spring.boot.web.jpa.test.service;

import javax.persistence.EntityNotFoundException;
import javax.transaction.Transactional;
import org.springframework.stereotype.Service;

import com.spring.boot.web.jpa.test.dto.ProductDto;
import com.spring.boot.web.jpa.test.entity.Product;
import com.spring.boot.web.jpa.test.repository.ProductRepository;

import lombok.RequiredArgsConstructor;

@Service
@Transactional
@RequiredArgsConstructor
public class ProductService {

	private final ProductRepository productRepository;

	public ProductDto findById(int id) {
		Product product = productRepository.findById(id).orElseThrow(EntityNotFoundException::new);

		ProductDto productDto = new ProductDto(product.getId(), product.getDescription(),
						product.getName(), product.getPrice());
		return productDto;
	}
}
```

바. 컨트롤러: ProductController.java

```java
package com.spring.boot.web.jpa.test.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.spring.boot.web.jpa.test.dto.ProductDto;
import com.spring.boot.web.jpa.test.service.ProductService;

import lombok.RequiredArgsConstructor;

@Controller
@RequiredArgsConstructor
public class ProductController {

	private final ProductService productService;

	@GetMapping(value = "/findById")
	public String findById(@RequestParam("id") String id,
		ModelMap modelMap)
```

```
    {
        ProductDto productDto = productService.findById(Integer.valueOf(id));
        modelMap.addAttribute("product", productDto);
        return "productOneView";
    }
}
```

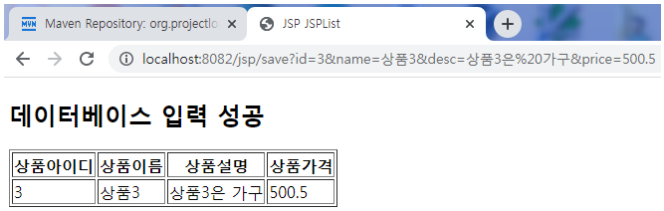
5) 테스트



6) 표준 저장(=변경)/단일행조회/멀티행조회

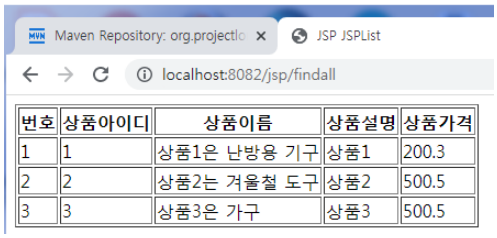
가. 저장(=변경):

http://localhost:8082/jsp/save?id=3&name=%EC%83%81%ED%92%883&desc=%EC%83%81%ED%92%883%EC%9D%80%20%EA%B0%80%EA%B5%AC&price=500.5



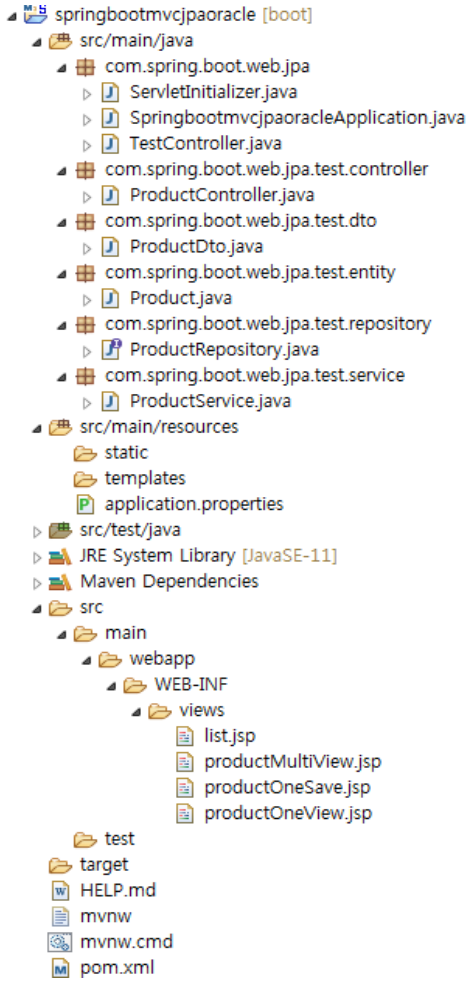
나. 전체 조회

http://localhost:8082/jsp/findall



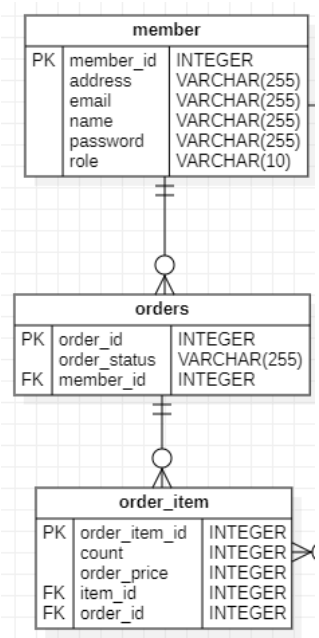
번호	상품아이디	상품이름	상품설명	상품가격
1	1	상품1은 난방용 기구	상품1	200.3
2	2	상품2는 겨울철 도구	상품2	500.5
3	3	상품3은 가구	상품3	500.5

다. 최종완성 모습



8.4. 스프링 부트 웹 + JPA(단일 테이블) 프로젝트 만들기

1) 대상 테이블



2) 테이블/데이터 내용

가. member

정렬... 필터:

	MEMB...	ADDRE...	EMAIL	NAME	PASS...	ROLE
1	5	송파구	songpa@www.net	송파형	5	user
2	1	동작구	dongjak@www.net	동작형	1	user
3	2	서초구	seocho@www.net	서초형	2	user
4	3	강남구	gangnam@www.net	강남형	3	user
5	4	마포구	mahpho@www.net	마포형	4	admin

작업...

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	MEMBER_ID	NUMBER(38,0)	No
2	ADDRESS	VARCHAR2(255 BYTE)	Yes
3	EMAIL	VARCHAR2(255 BYTE)	Yes
4	NAME	VARCHAR2(255 BYTE)	Yes
5	PASSWORD	VARCHAR2(255 BYTE)	Yes
6	ROLE	VARCHAR2(10 BYTE)	Yes

나. orders

	ORDER_ID	ORDER_STATUS	MEMBER_ID
1	1	O	1
2	2	O	1
3	3	C	1
4	4	O	2
5	5	O	2
6	6	C	2
7	7	O	3
8	8	O	3
9	9	C	3
10	10	O	5
11	11	O	5
12	12	C	5

작업...

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	ORDER_ID	NUMBER(38,0)	No
2	ORDER_STATUS	VARCHAR2(255 BYTE)	Yes
3	MEMBER_ID	NUMBER(38,0)	Yes

다. order_item

ORDER_ITEM_ID	COUNT	ORDER_PRICE	ITEM_ID	ORDER_ID
1	2	1000	10	1
2	3	2000	11	1
3	5	3000	20	2
4	3	2000	21	2
5	2	1000	30	3
6	2	1500	31	3
7	6	3500	50	5
8	3	2500	51	5

COLUMN_NAME	DATA_TYPE	NULLABLE
ORDER_ITEM_ID	NUMBER(38,0)	No
COUNT	NUMBER(38,0)	Yes
ORDER_PRICE	NUMBER(38,0)	Yes
ITEM_ID	NUMBER(38,0)	Yes
ORDER_ID	NUMBER(38,0)	Yes

3) 패키지 만들기

- springbootmvcjpaoracle [boot]
  - src/main/java
    - com.spring.boot.web.jpa
      - ServletInitializer.java
      - SpringbootmvcjpaoracleApplication.java
      - TestController.java
    - com.spring.boot.web.jpa.join.controller
    - com.spring.boot.web.jpa.join.dto
    - com.spring.boot.web.jpa.join.entity
    - com.spring.boot.web.jpa.join.repository
    - com.spring.boot.web.jpa.join.service

4) member 개발 – DTO/ENTITY

가. MemberFormDto

```
package com.spring.boot.web.jpa.join.dto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class MemberFormDto {
 private int id;
 private String address;
 private String email;
 private String name;
 private String password;
 private String role;
}
```

나. Member

```
package com.spring.boot.web.jpa.join.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import com.spring.boot.web.jpa.join.dto.MemberFormDto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Entity
@Table(name = "member")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
```

```java
public class Member {

 @Id
 @Column(name = "member_id")
 @GeneratedValue(strategy = GenerationType.AUTO)
 private int id;

 private String address;

 @Column(unique = true)
 private String email;

 private String name;

 private String password;

 private String role;

 public static Member createMember(MemberFormDto memberFormDto) {
            Member member = new Member(memberFormDto.getId(),
                                       memberFormDto.getAddress(),
                                       memberFormDto.getEmail(),
                                       memberFormDto.getName(),
                                       memberFormDto.getPassword(),
                                       memberFormDto.getRole());
            return member;
 }
}
```

5) orders 개발 – DTO/ENTITY

가. OrdersDto

```
package com.spring.boot.web.jpa.join.dto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class OrdersDto {
		private int order_id;
		private String order_status;
		private String member_id;
}
```

나. Orders

```
package com.spring.boot.web.jpa.join.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Table(name = "orders")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Orders {
		@Id
		@Column(name = "order_id")
		@GeneratedValue
		private int id;
		private String order_status;

		@ManyToOne(fetch = FetchType.LAZY)
		@JoinColumn(name = "member_id")
```

```
        private Member member;

        @OneToMany(mappedBy = "orders",
                cascade = CascadeType.ALL,
                orphanRemoval = true, fetch = FetchType.LAZY)
        private List<OrderItem> orderItems = new ArrayList<OrderItem>();
}
```

6) orderItem – 개발

가. OrderItemDto

```
package com.spring.boot.web.jpa.join.dto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class OrderItemDto {
 private int order_item_id;
 private int count;
 private int order_price;
 private int item_id;
 private int order_id;
}
```

나. OrderItem

```
package com.spring.boot.web.jpa.join.entity;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import lombok.Getter;
import lombok.Setter;

@Entity
@Table(name = "order_item")
@Getter
@Setter
public class OrderItem {

		@Id
		@Column(name = "order_item_id")
		@GeneratedValue
		private int id;
		private int count;
		private int order_price;
		private int item_id;
		@ManyToOne(fetch = FetchType.LAZY)
		@JoinColumn(name = "order_id")
		private Orders orders;
}
```

7) Repository **인터페이스** 만들기

가. MemberRepository

```
package com.spring.boot.web.jpa.join.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.spring.boot.web.jpa.join.entity.Member;

public interface MemberRepository
		extends JpaRepository<Member, Integer>{

}
```

나. OrdersRespository

```
package com.spring.boot.web.jpa.join.repository;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
import com.spring.boot.web.jpa.join.entity.Orders;

public interface OrdersRepository
		extends JpaRepository<Orders, Integer>{

}
```

다. OrderItemRepository

```
package com.spring.boot.web.jpa.join.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.spring.boot.web.jpa.join.entity.OrderItem;

public interface OrderItemRepository
		extends JpaRepository<OrderItem, Integer>{

}
```

8) 서비스 클래스 만들기

가. OrdersService

```
package com.spring.boot.web.jpa.join.service;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityNotFoundException;
import javax.transaction.Transactional;
import org.springframework.stereotype.Service;

import com.spring.boot.web.jpa.join.entity.Member;
import com.spring.boot.web.jpa.join.entity.OrderItem;
import com.spring.boot.web.jpa.join.entity.Orders;
import com.spring.boot.web.jpa.join.entity.ReturnDataList;
import com.spring.boot.web.jpa.join.repository.OrdersRepository;

import lombok.RequiredArgsConstructor;
```

```
@Service
@Transactional
@RequiredArgsConstructor
public class OrdersService {

 private final OrdersRepository ordersRepository;

 public List<ReturnDataList> getOrderAndOthersList(int id) {
		List<ReturnDataList> returnDataLists =
				new ArrayList<ReturnDataList>();

		Orders orders = ordersRepository.findById(id)
				.orElseThrow(EntityNotFoundException::new);

		List<OrderItem> orderItemsList = orders.getOrderItems();

		Member member = orders.getMember();

		returnDataLists.add(orders);
		returnDataLists.add(member);

		for(OrderItem orderItem : orderItemsList) {
			returnDataLists.add(orderItem);
		}

		return returnDataLists;
 }

}
```

9) 컨트롤러 만들기

```
package com.spring.boot.web.jpa.join.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.spring.boot.web.jpa.join.entity.Member;
import com.spring.boot.web.jpa.join.entity.OrderItem;
import com.spring.boot.web.jpa.join.entity.Orders;
import com.spring.boot.web.jpa.join.entity.ReturnDataList;
import com.spring.boot.web.jpa.join.service.OrdersService;

import lombok.RequiredArgsConstructor;
```

```java
@Controller
@RequiredArgsConstructor
public class OrdersController {

 private final OrdersService ordersService;

 @GetMapping(value = "/getOrderInfo")
 public String findById(@RequestParam("id") String id,
                        ModelMap modelMap) {
            List<ReturnDataList> returnDataLists =
                                ordersService.getOrderAndOthersList(Integer.valueOf(id));
            System.out.println("리턴사이즈: " + returnDataLists.size());

            Orders orders = (Orders)returnDataLists.get(0);
            Member member = (Member)returnDataLists.get(1);

            List<OrderItem> orderItemsList = new ArrayList<OrderItem>();
            for(int i = 2; i < 4; i++) {
                        orderItemsList.add((OrderItem)returnDataLists.get(i));
            }

            modelMap.addAttribute("orders", orders);
            modelMap.addAttribute("member", member);
            modelMap.addAttribute("orderitemlist", orderItemsList);
            modelMap.addAttribute("msg", "조회 성공");

            return "orderMultiView";
 }
}
```

10) 뷰 만들기

orderMultiView.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
     pageEncoding="UTF-8" isELIgnored="false"   %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html>
<html>
 <head>
            <meta charset="UTF-8">
            <title>JSP JSPList</title>
 </head>
```

```
<body>
        <h3>${msg}</h3>
        <table border="1">
                <tr>
                        <th>주문번호</th>
                        <th>주문상태</th>
                </tr>
                <tr>
                        <td>${orders.id}</td>
                        <td>${orders.order_status}</td>
                </tr>
        </table>

        <br>

        <table border="1">
                <tr>
                        <th>회원번호</th>
                        <th>회원주소</th>
                        <th>이메일</th>
                        <th>회원이름</th>
                        <th>회원암호</th>
                        <th>회원역할</th>
                </tr>
                <tr>
                        <td>${member.id}</td>
                        <td>${member.address}</td>
                        <td>${member.email}</td>
                        <td>${member.name}</td>
                        <td>${member.password}</td>
                        <td>${member.role}</td>
                </tr>
        </table>

        <br>

        <table border="1">
                <tr>
                        <th>인덱스</th>
                        <th>주문상품번호</th>
                        <th>주문상품갯수</th>
                        <th>주문상품가격</th>
                        <th>상품아이디</th>
                </tr>

                <c:forEach var="orderitem" items="${orderitemlist}" varStatus="idx">
                <tr>
                        <td>${idx.index + 1}</td>
                        <td>${orderitem.id}</td>
```

```
                <td>${orderitem.count}</td>
                <td>${orderitem.order_price}</td>
                <td>${orderitem.item_id}</td>
            </tr>
            </c:forEach>
        </table>

 </body>

</html>
```