0. 테이블 만들기

가. 테이블 스크립트

```
CREATE TABLE member_new(
        id int PRIMARY KEY,
        password varchar2(20) not null,
        name varchar2(20) not null,
        address varcahr2(20) not null
);
```

나. 데이터 입력

1 4354 부산시 박길동

2 1234 광명시 홍길동

3 1234 울산시 김길동

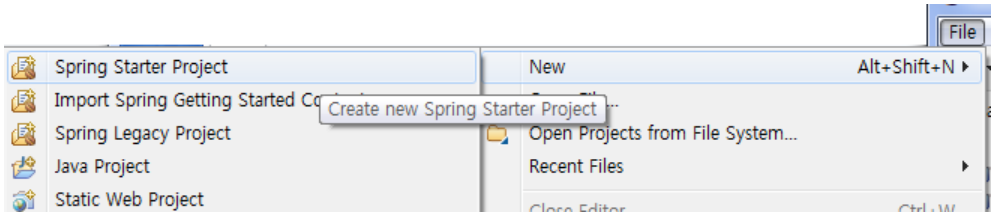
2 5644 광명시 고길동

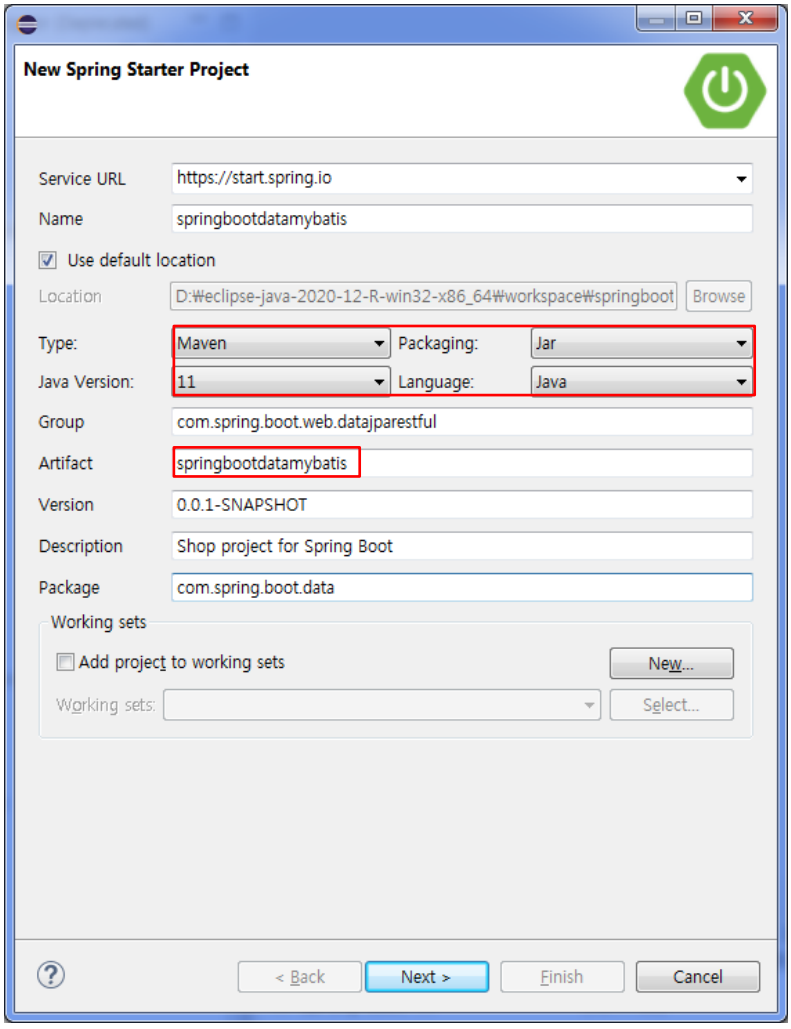
1 4354 부산시 박길동

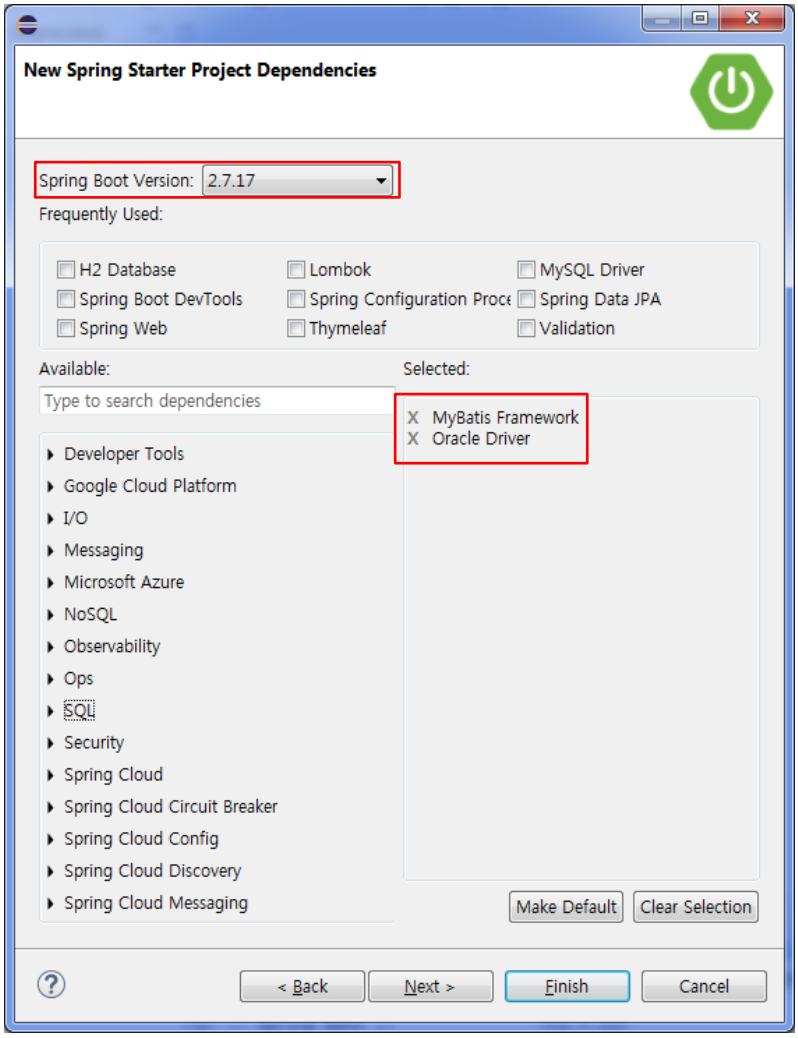
1. 개요

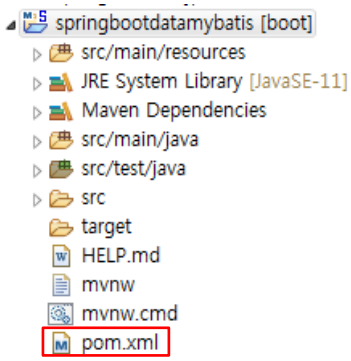
가. Spring Boot + MyBatis + Oracle

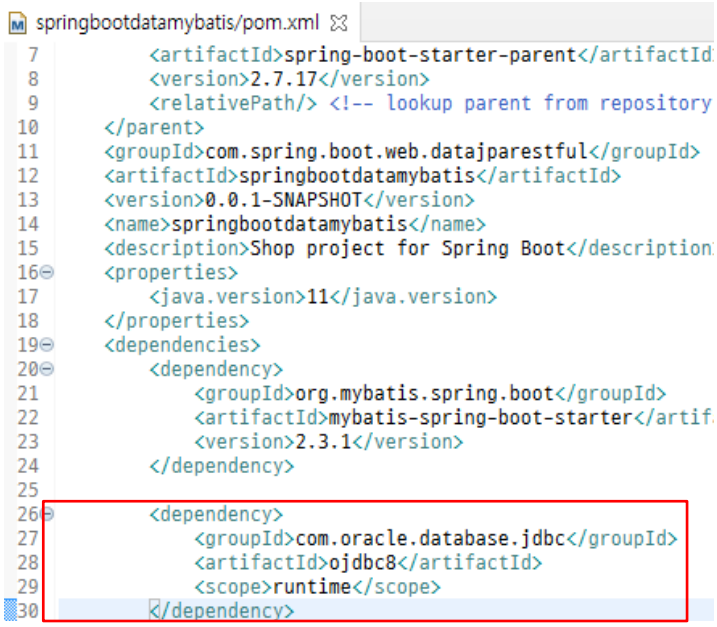
2. 프로젝트 생성






New Spring Starter Project Dependencies
Spring Boot Version: 2.7.17
Frequently Used:
H2 Database
Lombok
MySQL Driver
Spring Boot DevTools
Spring Configuration Proce
Spring Data JPA
Spring Web
Thymeleaf
Validation
Available:
Selected:
Type to search dependencies
X MyBatis Framework
X Oracle Driver
Developer Tools
Google Cloud Platform
I/O
Messaging
Microsoft Azure
NoSQL
Observability
Ops
SQL
Security
Spring Cloud
Spring Cloud Circuit Breaker
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Messaging
Make Default
Clear Selection
< Back
Next >
Finish
Cancel


springbootdatamybatis [boot]
src/main/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src/main/java
src/test/java
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml



- 변경

```
<dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2.0.4</version>
</dependency>
```

- 추가

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.24</version>
    <scope>provided</scope>
</dependency>
```

MAVEN UPDATE

3. 패키지 만들기

가. 설정관련 패키지: com.spring.boot.data.config

나. DTO 패키지: com.spring.boot.data.dto

다. DAO 패키지: com.spring.boot.data.dao(인터페이스 파일만 있음)

라. 테스트 패키지: sprintg.boot.data

마. 매퍼 패키지: 리소스 패키지 아래에 classpath:/mappers

4. 데이터 소스 설정 파일 만들기

가. 파일 위치: /src/main/resources/application.properties (이미 존재)

```
spring.datasource.username=oracle_test
spring.datasource.password=woseven
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.jdbc-url=jdbc:oracle:thin:@localhost:1521:xe
```

5. 데이터 소스 설정 클래스 만들기

```
package com.spring.boot.data.config;

import javax.sql.DataSource;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

// 어노테이션 기반 환경 구성을 도움
// 어노테이션을 붙이고 클래스 내에 하나 이상의 @Bean 메소드를 구현하면
// 스프링 컨테이너가 Bean 정의를 생성하고 런타임 시 그 Bean 들의 요청을 처리할 것을 선언
@Configuration
public class DataSourceConfig {
	// 외부 설정 파일(application.properties)을 참조할 때 쓰는 방법 중 하나
	@ConfigurationProperties(prefix="spring.datasource")
	@Bean
	public DataSource dataSource() {
		return DataSourceBuilder.create().build();
	}
}
```

6. SqlSessionFactory 빈 설정 클래스 만들기

```
package com.spring.boot.data.config;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

@Configuration
// 연결할 DAO 인터페이스를 담은 패키지를 등록
// @MapperScan(basePackages = {"com.project.mong.dao"}) 라고 정의도 가능
@MapperScan("com.spring.boot.data.dao")
public class MySQLConfig {

	// SqlSessionFactory: Oracle과 MyBatis를 연결해주는 객체
	@Bean
	public SqlSessionFactory sqlSessionFactory(DataSource dataSource)
			throws Exception {

		// SqlSessionFactoryBean: SqlSessionFactory를 생성해주는 클래스
		final SqlSessionFactoryBean sessionFactory =
					new SqlSessionFactoryBean();
		// setDataSource(): datasource를 참조
		sessionFactory.setDataSource(dataSource);

		// PathMatchingResourcePatternResolver
		// resource 위치 검색을 돕는 Spring class
		PathMatchingResourcePatternResolver resolver =
					new PathMatchingResourcePatternResolver();

		sessionFactory.setMapperLocations(resolver.getResources("classpath:mappers/*.xml"));
		sessionFactory.setTypeAliasesPackage("com.spring.boot.data.dto");

		return sessionFactory.getObject();
```

```
    }
}
```

7. DTO 클래스 만들기: lombok 적용

```
package com.spring.boot.data.dto;

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

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

8. Mapper 파일 만들기: SQL 실행문 집합

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.spring.boot.data.dao.MemberDao">
    <select id="findMember" resultType="member">
            SELECT id, password, name, address
            FROM member_new
```

```
    </select>
</mapper>
```

9. 테스트 클래스 만들기

```
package com.spring.boot.data;

import java.util.List;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.ApplicationContext;

import com.spring.boot.data.dao.MemberDao;
import com.spring.boot.data.dto.Member;

@SpringBootTest
class SpringbootdatamybatisApplicationTests {

    @Autowired
    ApplicationContext context;

    @Autowired
    private MemberDao memberDao;

    @Test
    void getFindMember() {
```

```
		List<Member> memList = memberDao.findMember();
		for(Member member : memList) {
			System.out.println(member);
		}
	}
}
```