MyBatis

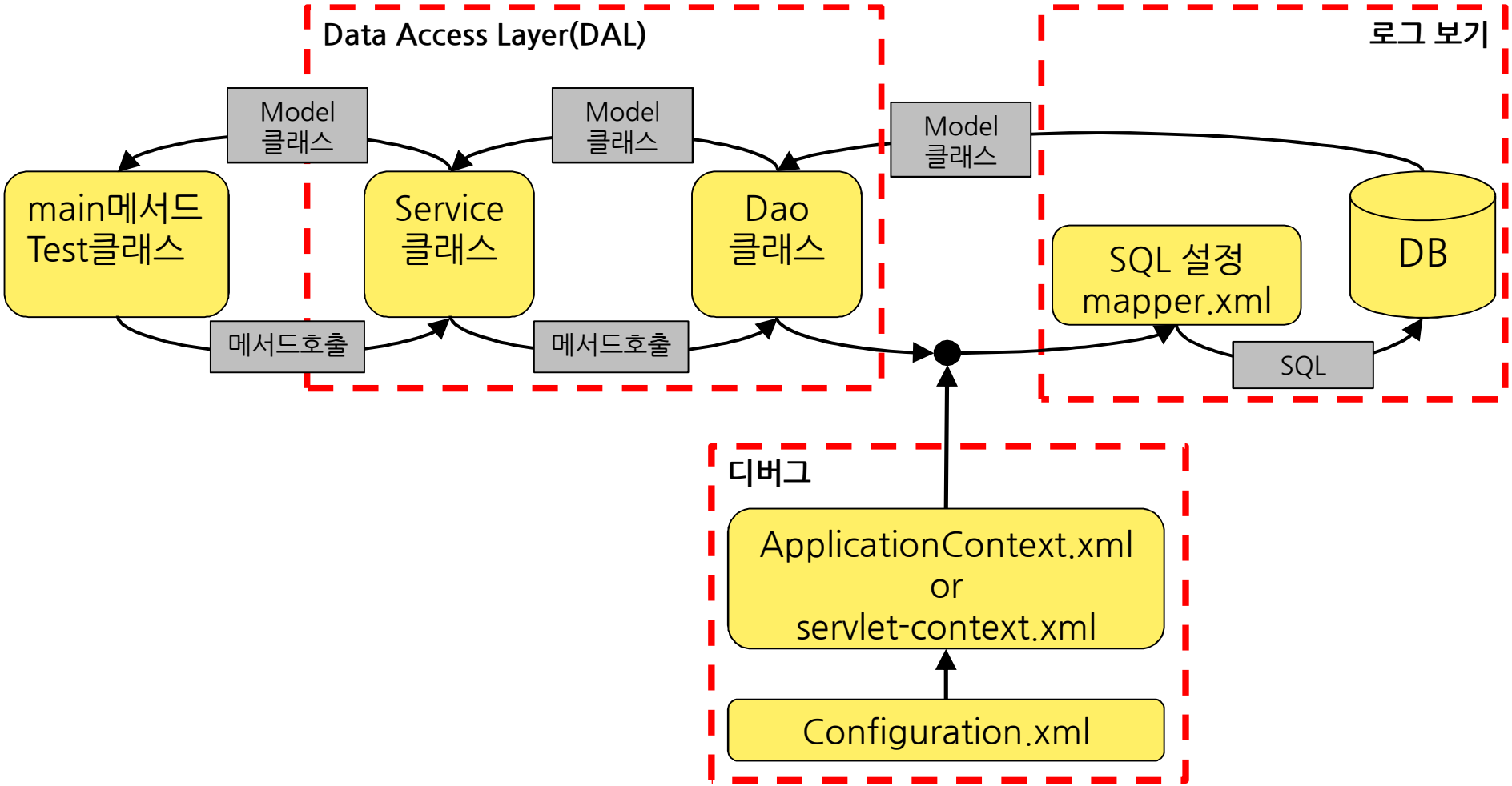
1. gradle로 프로젝트 생성
   - build.gradle 수정
   - gradle >> refresh dependency
   - gradle >> refresh all 실행
2. 데이터베이스 테이블 생성
3. src/main/resources 폴더에 아래 파일 추가
   - log4j.properties 파일 생성
   - ApplicationContext.xml 파일 생성
   - Configuration.xml 파일 생성
4. src/main/resources/mapper/mapperBook.xml 파일 생성
   - Configuration.xml 에 등록
5. Model 클래스 생성
6. Dao 생성
   - IDaoBook 인터페이스, DaoBook 클래스 생성
7. Service 클래스 생성
   - IServiceBook 인터페이스, ServiceBook 클래스 생성
8. JUnit Test 클래스 생성

JDBC vs MyBatis

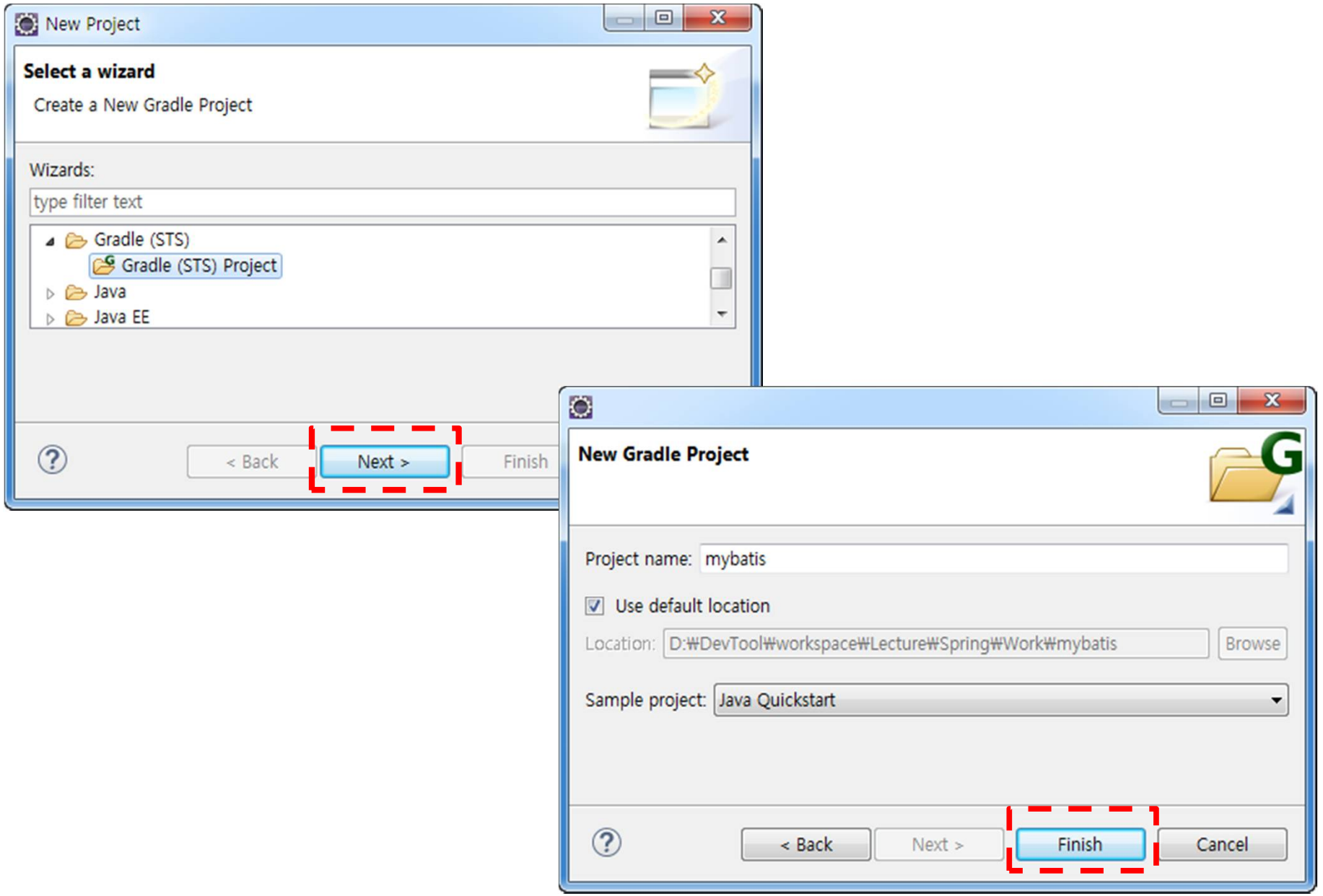
```
String query = "SELECT * FROM book WHERE bookid = ? ";

PreparedStatement stmt = conn.prepareStatement(query);
stmt.setString(1, bookname );

rs = stmt.executeQuery();
```

```
<select id="getSQLSelectEqual" parameterType="string" resultType="ModelBook">
    SELECT * FROM book
    WHERE bookname = #{bookname}
    ORDER BY bookid ASC
</select>
```

MyBatis 데이터 처리 과정



gradle로 프로젝트 생성

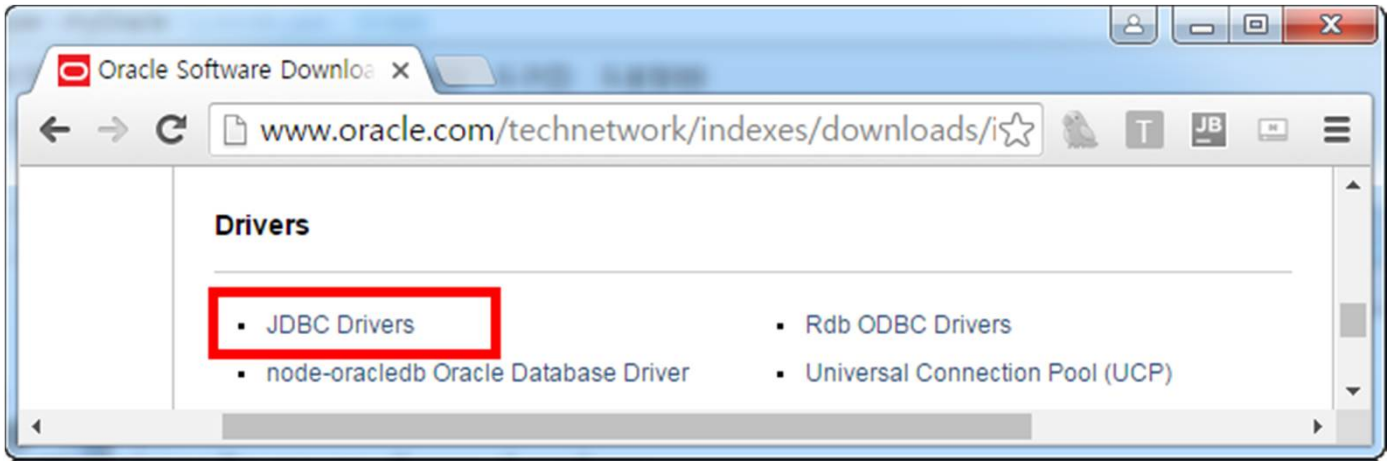


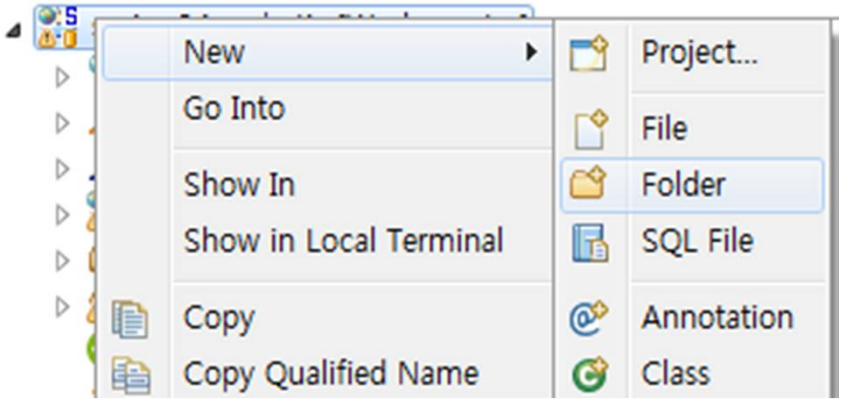
Oracle JDBC 드라이버 추가하기

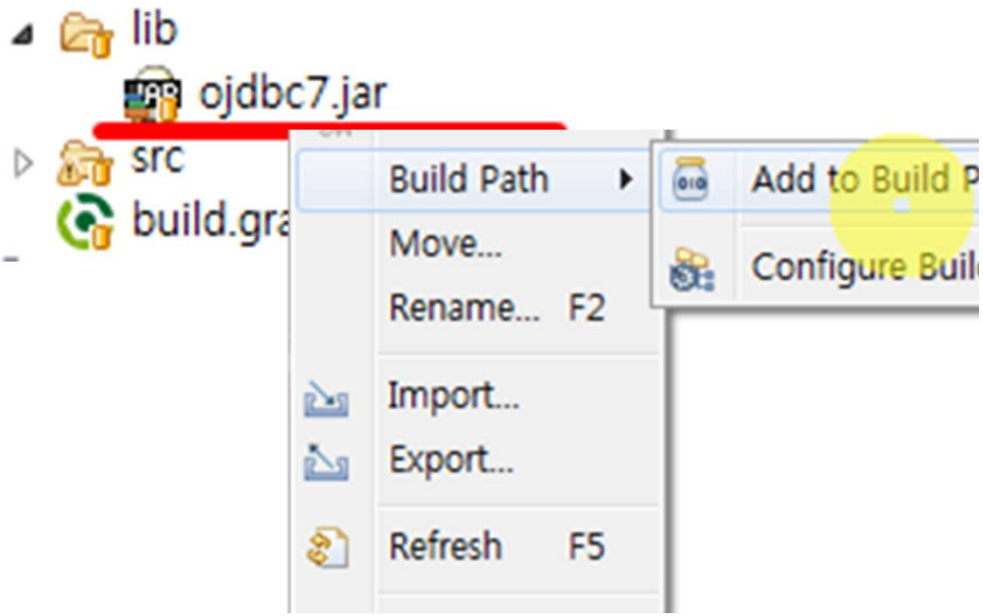
1. Oracle JDBC 드라이버 다운로드

   https://www.oracle.com/downloads/ 에서 드라이버를 다운로드 받는다. 단, jar 파일이면 압축을 풀지 마시오.
2. 프로젝트에 lib 폴더 생성
3. 다운로드 받은 ojdbc.jar를 lib 폴더로 복사한다.
4. JDBC를 Build Path에 추가하기

Oracle JDBC 드라이버 추가하기







build.gradle 수정

build.gradle

```
apply plugin: 'java'
apply plugin: 'eclipse'

sourceCompatibility = 1.8
group = 'com.lecture.database'
version = '1.0'


// 소스 인코딩 UTF-8로 지정
[compileJava, compileTestJava]*.options*.encoding = 'UTF-8'
```

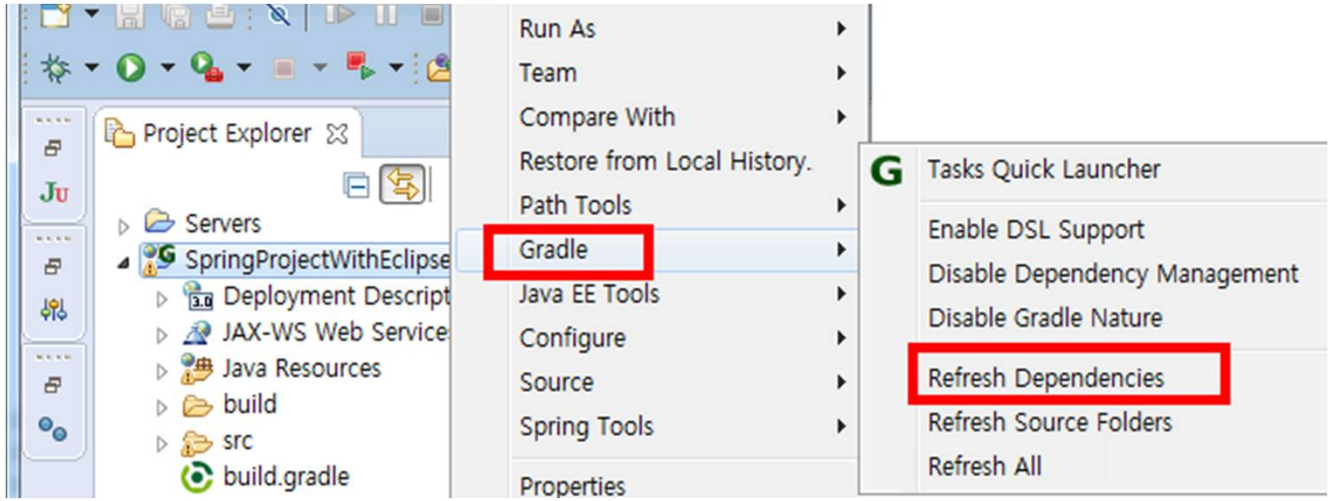
```
// log library
slf4j-api
slf4j-log4j12
log4j
log4jdbc

// spring 관련 라이브러리 추가
spring-context

// mysql connector
mysql-connector-java

// mybatis library
mybatis
mybatis-spring
```

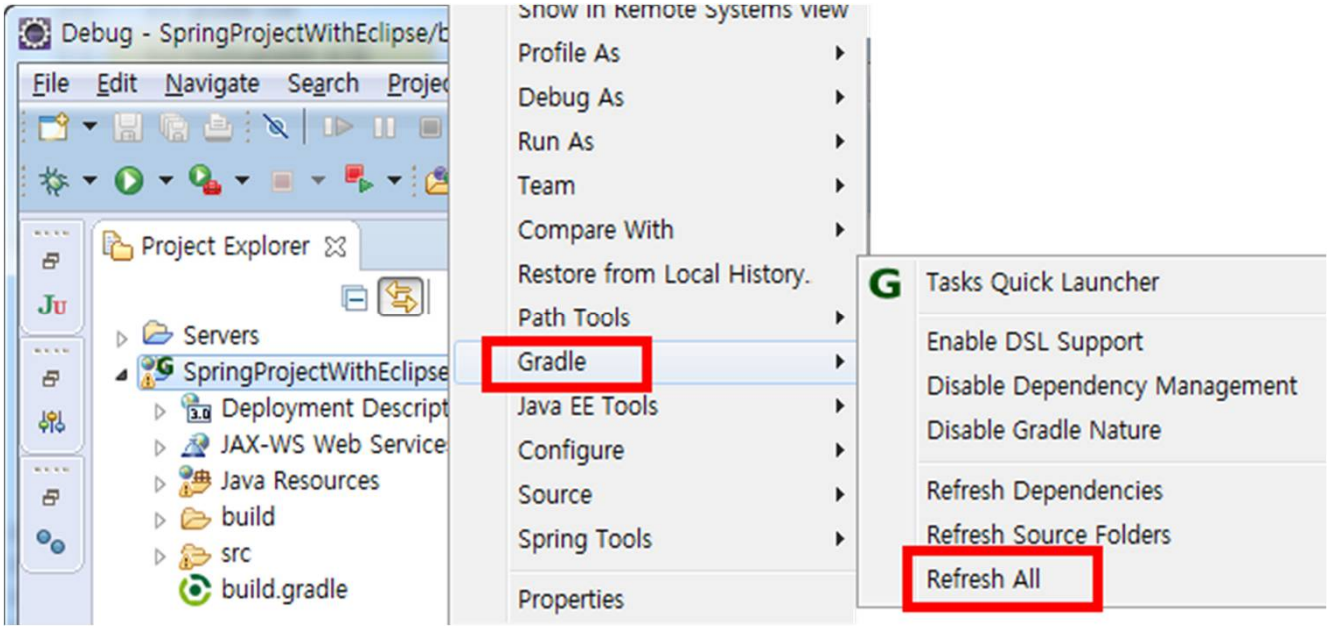
build.gradle

```
dependencies {
    // TDD를 위한 JUnit 라이브러리.
    testCompile 'junit:junit:4.+'

    compile 'commons-collections:commons-collections:3.2'

    compile 'commons-beanutils:commons-beanutils:1.9.2'

    // log library
    compile 'org.slf4j:slf4j-api:1.7.5'
    compile 'org.slf4j:slf4j-log4j12:1.7.9'
    compile 'log4j:log4j:1.2.17'
    compile 'com.googlecode.log4jdbc:log4jdbc:1.2'

    // spring 관련 라이브러리 추가
    compile 'org.springframework:spring-context:4.1.6.RELEASE'


    // mysql connector
    compile 'mysql:mysql-connector-java:5.1.34'

    // mybatis library
    compile 'org.mybatis:mybatis:3.2.8'
    compile 'org.mybatis:mybatis-spring:1.2.2'
}
```

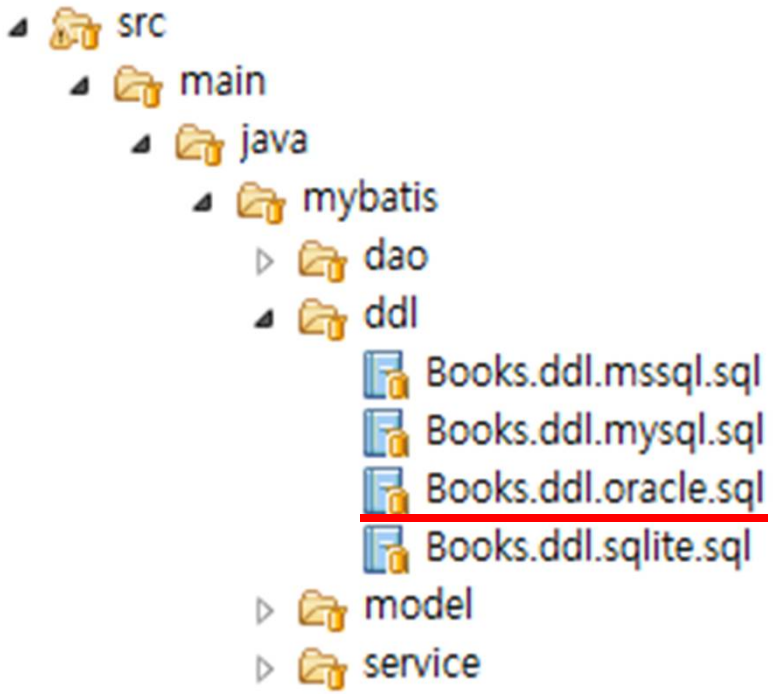
gradle refresh 실행

- build.gradle 이 수정되면 반드시 아래 과정을 반복해야 한다.





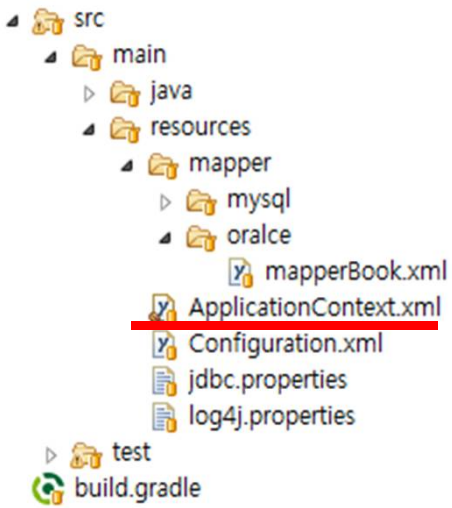
데이터베이스 테이블 생성




```
-- book 테이블 제거하기
BEGIN EXECUTE IMMEDIATE 'DROP TABLE book';
EXCEPTION WHEN OTHERS THEN NULL; END;
/


-- book 테이블 생성
CREATE TABLE book (
      book_id     NUMBER(10)  GENERATED AS IDENTITY
    , title       VARCHAR(50)
    , publisher   VARCHAR(30)
    , year        VARCHAR(10)
    , price       NUMBER(10)
    , dtm         DATE
    , use_yn      NUMBER(1)
    , authid      NUMBER(10)

    , PRIMARY KEY(book_id)
);
```

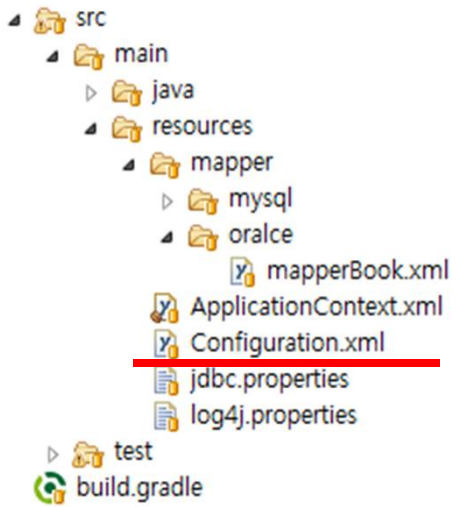
ApplicatonContext.xml 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:p="http://www.springframework.org/schema/p"

    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd

        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd">
```



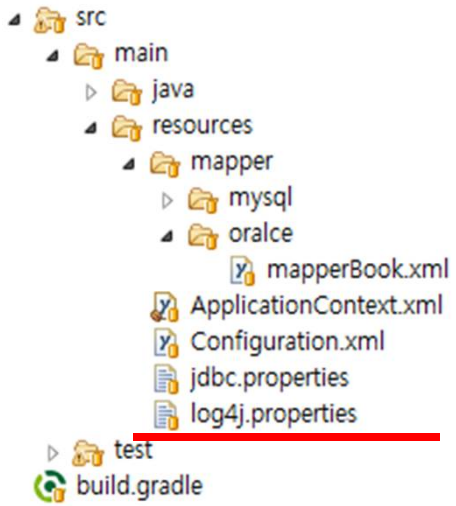
Configuration.xml 생성 및 설정

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

<configuration>
  <settings>
      <setting name="jdbcTypeForNull" value="NULL" />
  </settings>

  <!-- Model 클래스의 패키지명을 등록한다. -->
  <typeAliases>
      <package name="mybatis.model" />
  </typeAliases>

  <!-- sql 이 저장되는 xml 파일 등록: 주의 사항은 경로로 설정해야 한다. -->
  <mappers>
      <mapper resource="mapper/oracle/mapperBook.xml" />
  </mappers>

</configuration>
```



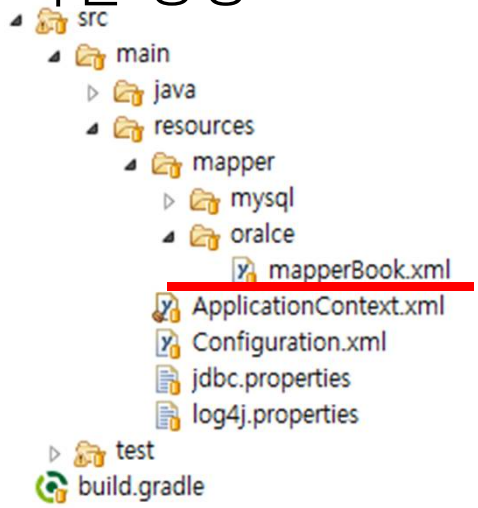
log4j.properties 생성

```
########################################
### Rules reminder:
### DEBUG < INFO < WARN < ERROR < FATAL
########################################


########################################
### Root Logger로 stout, rolling으로 셋팅
### 최상위 카테고리에 DEBUG로 레벨 설정
### appender로 stdout, rolling을 정의
########################################
log4j.rootLogger=DEBUG, stout, rolling


########################################
### console 설정
########################################

#### 콘솔에 뿌려줌
log4j.appender.stout=org.apache.log4j.ConsoleAppender

#### DEBUG 이상 레벨에서만 찍는다.
log4j.appender.stout.Threshold=DEBUG
```



mapperBook.xml 생성(1/3)

- src/main/resources/mapper/oracle/mapperBook.xml 파일 생성

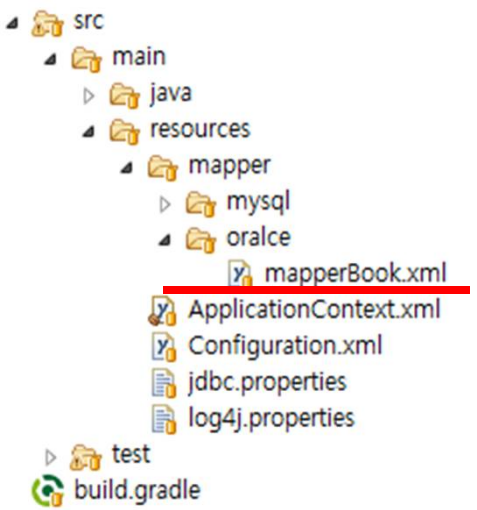



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
       "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="mapper.oracle.mapperBook">

    <select id="getSQLSelectAll" resultType="ModelBook" >
        SELECT * FROM book ORDER BY bookid ASC
    </select>

    <select id="getSQLSelectEqual" parameterType="String" resultType="ModelBook">
        SELECT * FROM book where bookname like #{bookname}
    </select>

    <select id="getSQLSelectEqual" parameterType="String" resultType="ModelBook">
        SELECT * FROM book where bookname = #{bookname}
    </select>

    <insert id="setSQLInsert" parameterType="ModelBook" >
        INSERT INTO BOOK( BOOKNAME, PUBLISHER, YEAR, PRICE, DTM, USE_YN, AUTHID )
        VALUES(#{bookname},#{publisher},#{year},#{price},#{dtm},#{use_yn},#{authid} )
    </insert>
</mapper>
```

mapperBook.xml 생성(2/3)

- src/main/resources/mapper/oracle/mapperBook.xml 파일 생성

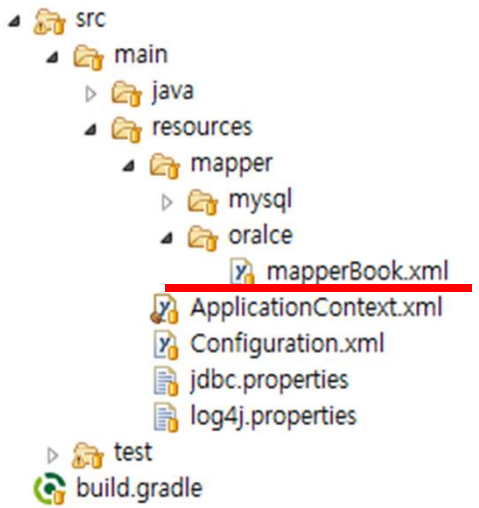
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
       "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="mapper.oracle.mapperBook">

    <update id="setSQLUpdate" parameterType="hashmap" >
         UPDATE BOOK
            SET BOOKNAME    = #{updateValue.bookname}
              , PUBLISHER   = #{updateValue.publisher}
              , YEAR        = #{updateValue.year}
              , PRICE       = #{updateValue.price}
              , DTM         = #{updateValue.dtm}
              , USE_YN      = #{updateValue.use_yn}
              , AUTHID      = #{updateValue.authid}
            WHERE 1 = 1
              <if test="searchValue.bookid != null" >
              AND BOOKID    = #{searchValue.bookid}
              </if>
              <if test="searchValue.bookname != null">
              AND BOOKNAME  = #{searchValue.bookname}
              </if>
    </update>
</mapper>
```



mapperBook.xml 생성(3/3)

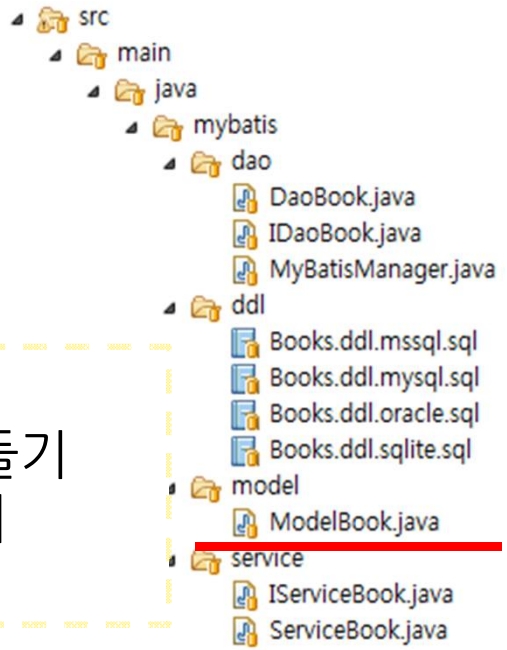
- src/main/resources/mapper/oracle/mapperBook.xml 파일 생성

```
<?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="mapper.oracle.mapperBook">

    <delete id="setSQLDelete" parameterType="ModelBook" >
         DELETE FROM BOOK
            WHERE 1 = 1
              <if test="bookid != null" >
              AND BOOKID    = #{bookid}
              </if>
              <if test="bookname != null">
              AND BOOKNAME  = #{bookname}
              </if>
    </delete>

</mapper>
```



Model 클래스 생성

```
package mybatis.model;

public class ModelBook {

    int     bookid     ;
    String  bookname   ;
    String  publisher  ;
    String  year       ;
    int     price      ;
    String  dtm        ;
    boolean use_yn     ;
    int     authid     ;

    ...
}
```



1. 멤버변수 선언
2. getter / setter 만들기
3. Constructor 만들기
4. toString 만들기

 # 모델 클래스를 만들 때는 필드 이름을 테이블 컬럼명과 동일하게 만들어야 한다.

Dao 생성

- IDaoBook 만들 때
  - mapperBook.xml 의 id 이름을 메서드 명으로 해서 만들면 편리하다

- DaoBook 클래스 만들 때

```
@Repository("daobook")
public class DaoBook implements IDaoBook {

    @Autowired
    @Qualifier("sqlSession")
    private SqlSession session;

}
```

5. Dao 클래스 생성

- DaoBook.getSQLSelectEqual() 메서드 만들기

```
@Override
public List<ModelBook> getSQLSelectEqual(String bookname) {
    List<ModelBook> result  = null;

    result = session.selectList("mapper.oracle.mapperBook.getSQLSelectEqual", bookname);

    return result;
}
```

```
<mapper namespace="mapper.oracle.mapperBook">

    <select id="getSQLSelectEqual" parameterType="String" resultType="ModelBook">
        SELECT * FROM book where bookname = #{bookname}
    </select>

</mapper>
```

5. Dao 클래스 생성

- DaoBook.setSQLInsert() 메서드 만들기

```
@Override
public int setSQLInsert(ModelBook book) {

    int result = 0 ;
    result = session.insert("mapper.oracle.mapperBook.setSQLInsert", book);
    return result;
}
```

```
<mapper namespace="mapper.oracle.mapperBook">


    <insert id="setSQLInsert" parameterType="ModelBook" >
        INSERT INTO BOOK( BOOKNAME, PUBLISHER , YEAR , PRICE , DTM , USE_YN , AUTHID)
        VALUES( #{bookname},#{publisher},#{year},#{price},#{dtm},#{use_yn},#{authid} )
    </insert>
</mapper>
```

Service 생성

- IServiceBook 만들 때
  - IDaoBook 복사해서 만들면 된다.

- ServiceBook 클래스 만들 때

```
@Service("servicebook")
public class ServiceBook implements IServiceBook {

    @Autowired
    @Qualifier("daobook")
    private IDaoBook dao;

}
```

7. JUnit Test 클래스 생성

```
public class TestServiceBook {
    private static IServiceBook service = null;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {

        // classpath 를 이용한 설정 파일 로딩
        ApplicationContext context = new
ClassPathXmlApplicationContext("classpath:ApplicationContext.xml");

        // file 을 이용한 설정 파일 로딩
        // ApplicationContext context = new
ClassPathXmlApplicationContext("file:src/main/webapp/WEB-
INF/spring/appServlet/servlet-context.xml");

        // DI를 이용한 servicebook 인스턴스 생성
        service = context.getBean("servicebook", IServiceBook.class);
    }
}
```

7. JUnit Test 클래스 생성

```
public class TestServiceBook {

    @Test
    public void testGetSQLSelectAll() {
        IServiceBook service = new ServiceBook(session);
        List<ModelBook> result = service.getSQLSelectAll();
        assertSame(result.size(), 14);
    }

    @Test
    public void testGetSQLSelectEqual() {
        List<ModelBook> result = service.getSQLSelectEqual("First SQL");
        assertSame(result.size(), 1);
    }

}
```

7. JUnit Test 클래스 생성

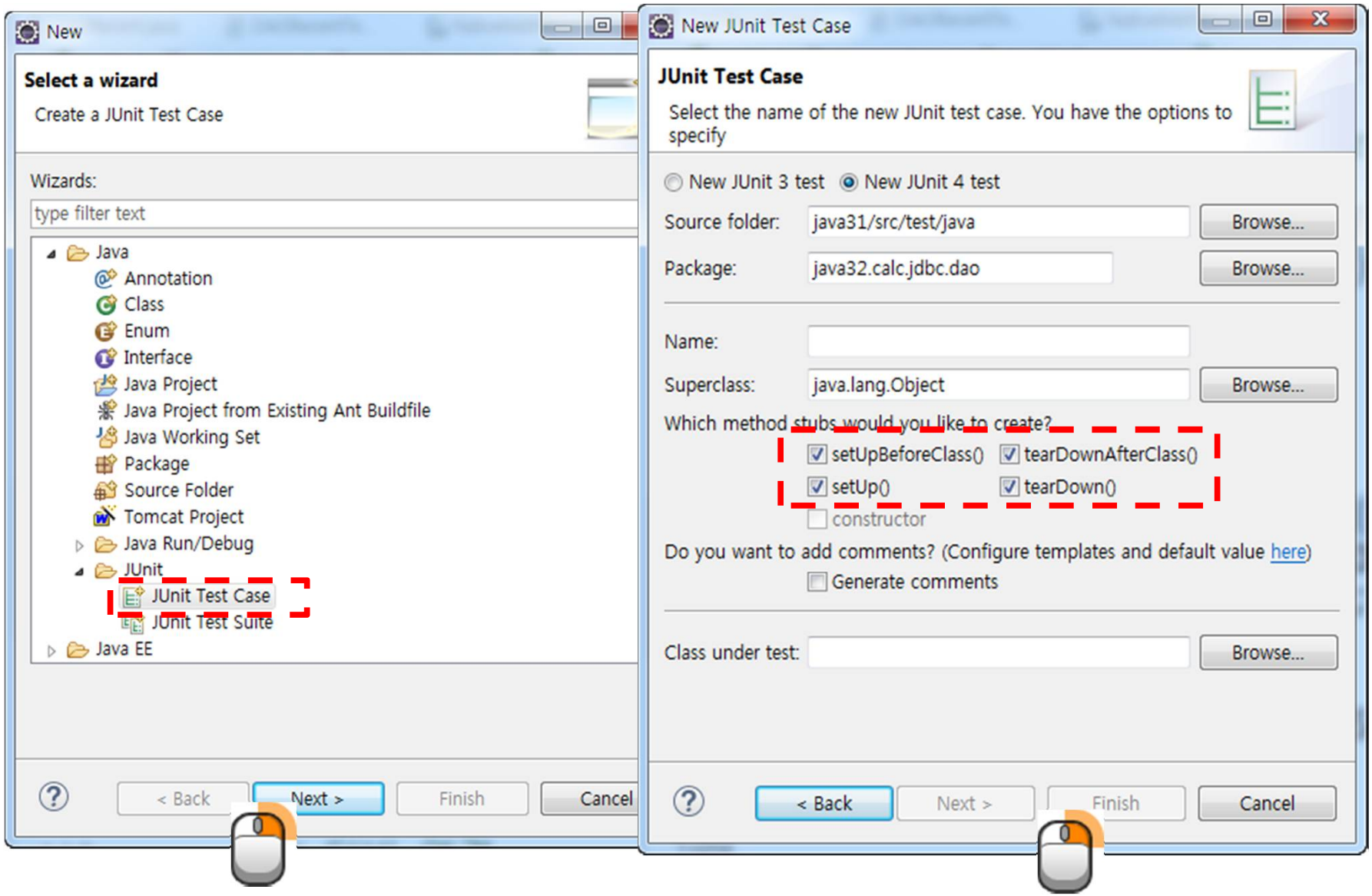
```
public class TestServiceBook {

    @Test
    public void testSetSQLInsert() {
        ModelBook book = new ModelBook();
        book.setAuthid(10);
        book.setBookname("test");
        book.setDtm("2016-11-12");
        book.setPrice(10000);
        book.setPublisher("내가");
        book.setUse_yn(true);
        book.setYear("2016");
        int result = service.setSQLInsert(book);
        assertSame(result, 1);
    }
}
```

JUnit 테스트

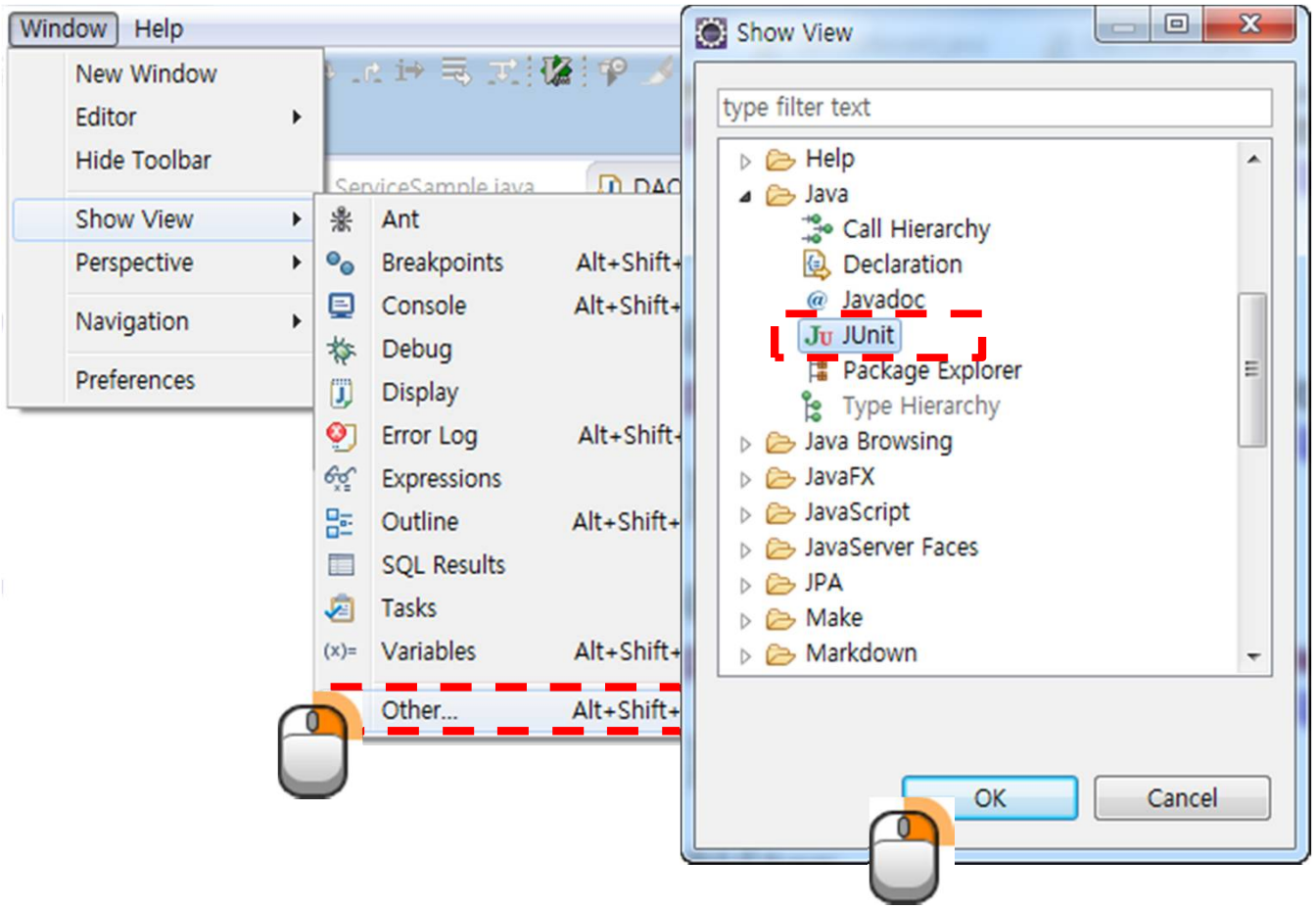
1. 단위 테스트
    1. JUnit
    2. DbUnit

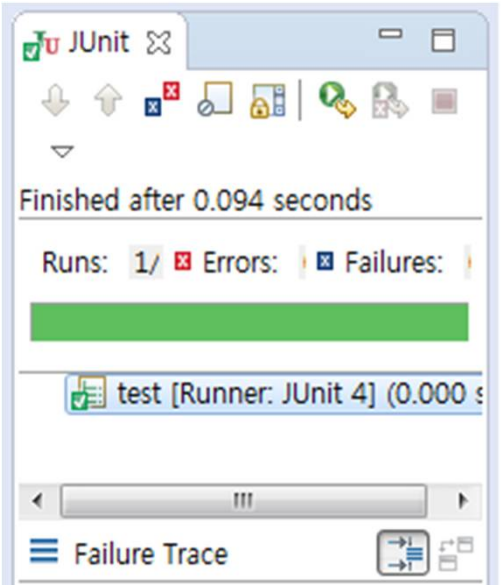
2. 통합 테스트
    1. 허드슨을 이용한 통합 테스트
    2. TeamCity를 이용한 통합 테스트

JUnit Test Case 추가



JUnit Test Case 추가

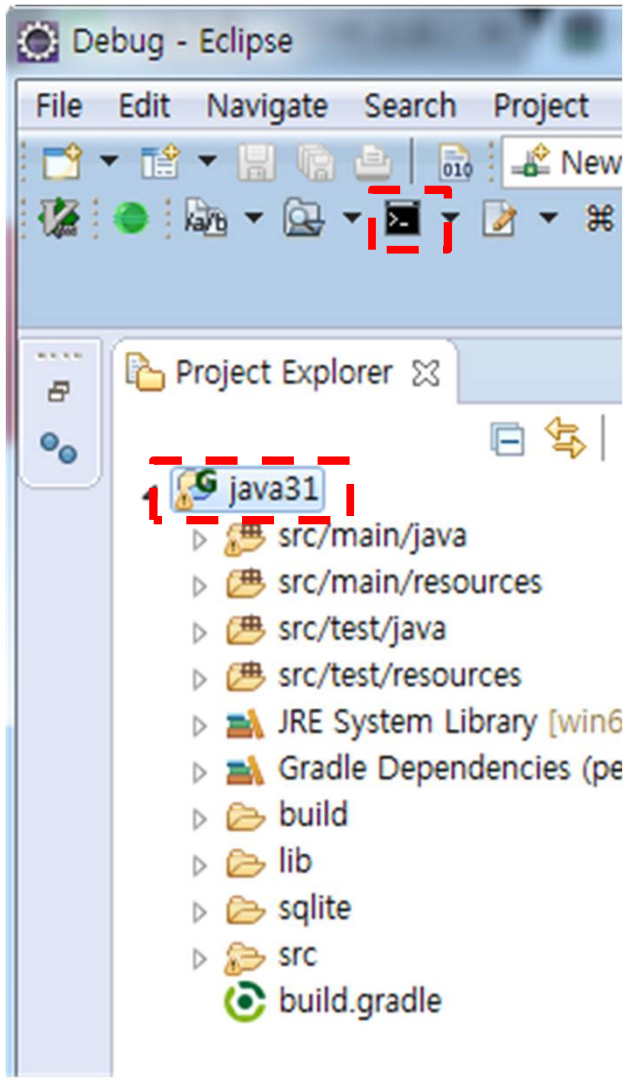


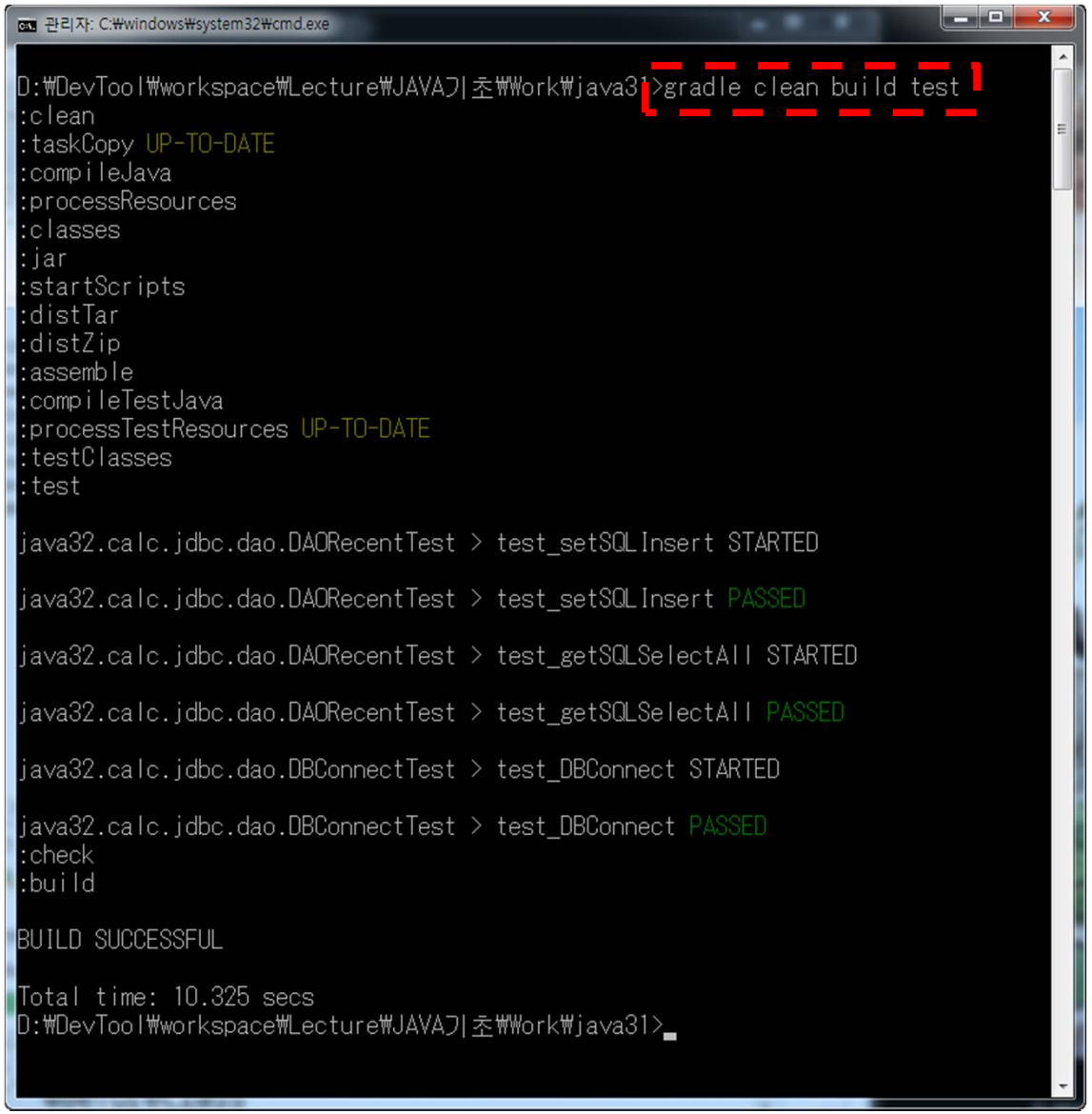


assert 메서드

- assertEquals( )
- assertTrue( )

gradle로 build와 test 하기





DbUnit 테스트

mapperBoard 오늘 진행 일정

- 1교시
  - *insertBoardList / getArticleTotalRecord*
- 2교시
  - ***getArticleList*** */ getArticle / insertArticle*
- 3교시
  - *updateArticle / deleteArticle / increaseHit*
- 4교시
  - ***getNextArticle***
- 5교시
  - ***getPrevArticle***
- 6교시
  - *getAttachFile / getAttachFileList / insertAttachFile*
- 7교시
  - *deleteAttachFile / insertComment / updateComment*
- 8교시
  - *deleteComment / getComment / getCommentList*

mapperUser 만들기

- TB_USER 테이블 생성
- mapperUser.xml 만들기
- ModelUser.java 만들기
- Configuration.xml 수정

- **IDaoUser.java** 와 DaoUser.java 만들기
- IServiceUser.java 와 ServiceUser.java 만들기
- JUnit으로 TestServiceUser.java 만들기

- mapperBoard.xml 의 getArticleList 수정
  - 기존 쿼리에 TB_User 테이블 left join 추가
- Spring DI 적용
- Git Hub 에 소스 올리기

Spring DI 적용

- DaoUser.java 수정
  - 클래스에 @Repository 적용
  - 필드에 @Autowired 적용
- ServiceUser.java 수정
  - 클래스에 @Service 적용
  - 필드에 @Autowired 적용
- ApplicationContext.xml 생성
- Confiuguration.xml 수정
- TestServiceUser.java 수정
  - 뒷장에 이어서

Spring DI 적용

- TestServiceUser.java 수정

```
public class TestServiceUser {

    private static ApplicationContext context = null;
    private static IServiceUser userservice = null;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        context     = new ClassPathXmlApplicationContext(
                                        "classpath:ApplicationContext.xml");

        userservice = context.getBean("userservice", IServiceUser.class);
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        ((ClassPathXmlApplicationContext)context).close();
    }

}
```