



MyBatis

1. 데이터베이스 테이블 생성
2. gradle로 프로젝트 생성
 - build.gradle 수정
 - gradle >> refresh 실행
3. src/main/resources 폴더에 아래 파일 추가
 - log4j.properties 파일 생성
4. 스프링 설정 파일 생성
 - ApplicationContext.xml 파일 생성
 - Configuration.xml 파일 생성
5. src/main/resources/mapper/mapperBook.xml 파일 생성
 - mapperBook.xml 파일을 Configuration.xml 에 등록
6. Model 클래스 생성
7. IBook 인터페이스,
8. DaoBook 클래스 생성
9. ServiceBook 클래스 생성
10. JUnit Test 클래스 생성



JDBC vs MyBatis

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

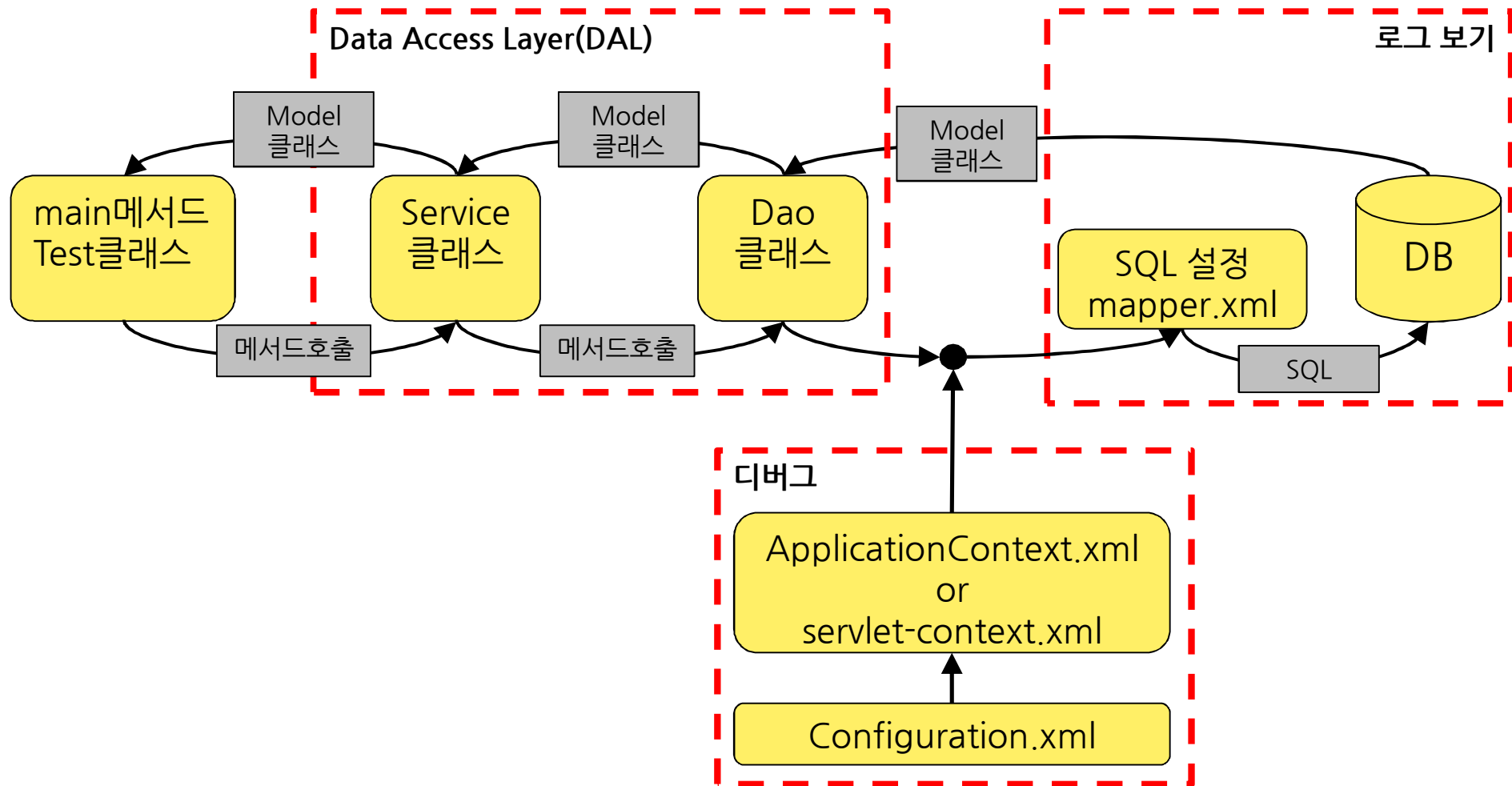
```
PreparedStatement stmt = conn.prepareStatement(query);  
stmt.setString(1, bookname );
```

```
rs = stmt.executeQuery();
```

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

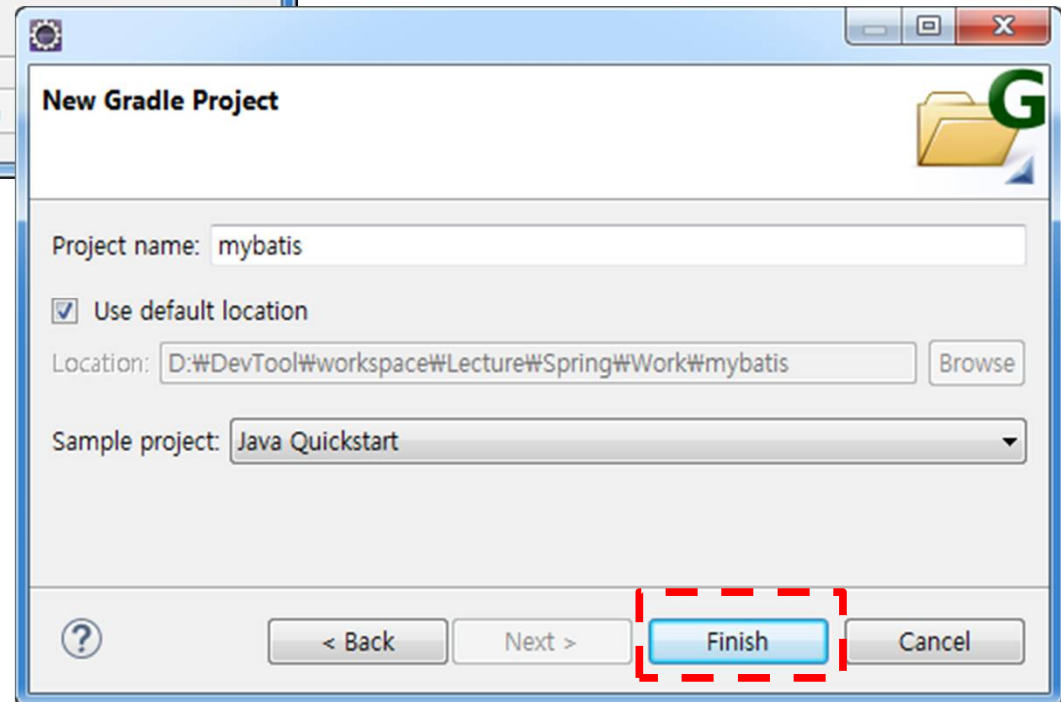
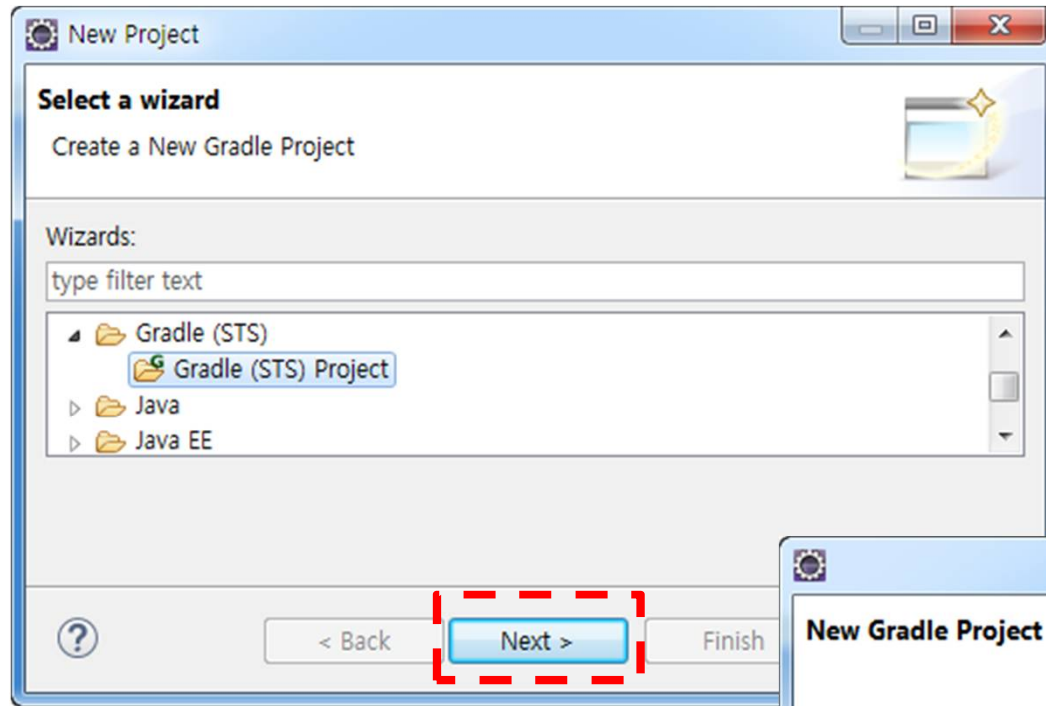


MyBatis 데이터 처리 과정





gradle로 프로젝트 생성



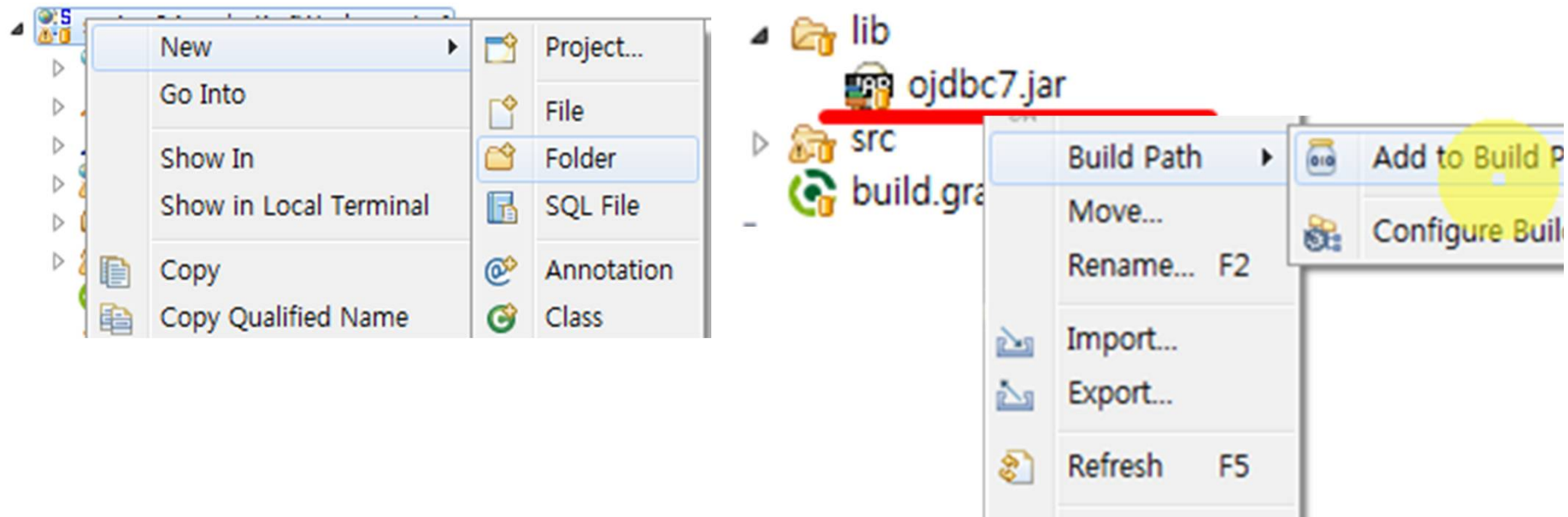
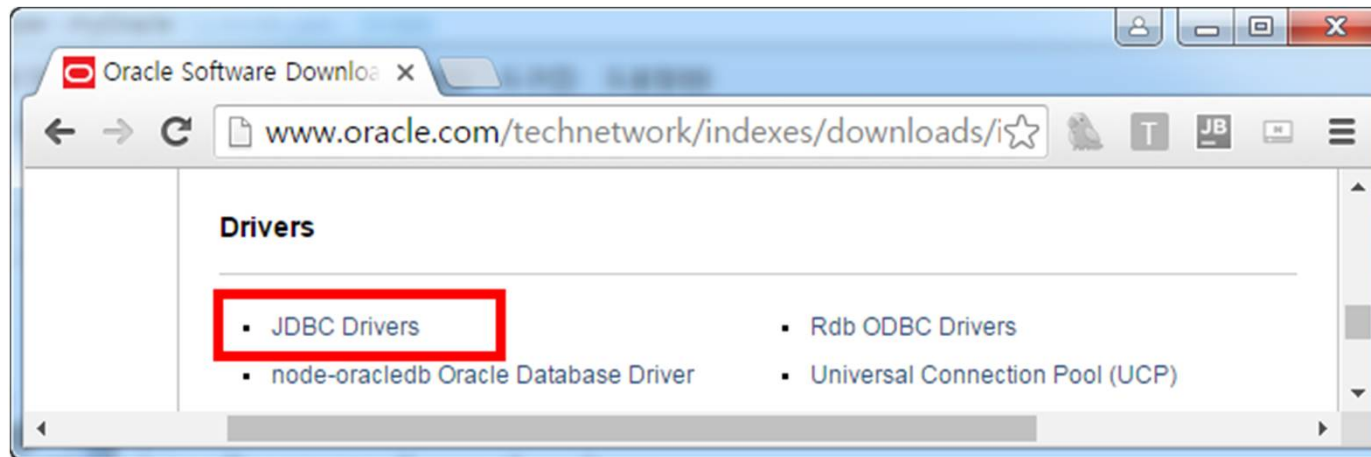


Oracle JDBC 드라이버 추가하기

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 ✕
66 dependencies {
67
68     compile fileTree(dir: "lib", include: ["*.jar"])
69
70     // log 라이브러리
71     compile 'org.slf4j:slf4j-api:1.7.12'
72     compile 'org.slf4j:slf4j-log4j12:1.7.21'
73     compile 'org.slf4j:log4jdb3:1.1'
74     compile 'log4j:log4j:1.2.17'
75     compile 'com.googlecode.log4jdbc:log4jdbc:1.2'
76
77     // spring 라이브러리
78     compile "org.springframework:spring-core:4.1.7.RELEASE"
79     compile "org.springframework:spring-beans:4.1.7.RELEASE"
80     compile "org.springframework:spring-context:4.1.7.RELEASE"
81     compile "org.springframework:spring-webmvc:4.1.7.RELEASE"
82     compile "org.springframework:spring-aop:4.1.7.RELEASE"
83     compile "org.springframework:spring-jdbc:4.1.7.RELEASE"
84     compile "org.springframework:spring-tx:4.1.7.RELEASE"
85     compile "org.springframework:spring-aspects:4.1.7.RELEASE"
86
87     // mysql 라이브러리
88     compile 'mysql:mysql-connector-java:5.1.38'
89
90     // mybatis 라이브러리
91     compile 'org.mybatis:mybatis:3.2.8'
92     compile 'org.mybatis:mybatis-spring:1.2.2'
93
94     // Apache Commons package
95     compile "org.apache.commons:commons-lang3:3.4"
96     compile "org.apache.commons:commons-dbcp2:2.0"
97
98     testCompile 'junit:junit:4.12'
99 }
```

```
// log library
slf4j-api
slf4j-log4j12
log4jdbc3
log4j
log4jdbc

// spring 라이브러리
spring-core
spring-beans
spring-context
spring-webmvc
spring-aop
spring-jdbc
spring-tx
spring-aspects

// mysql connector
mysql-connector-java

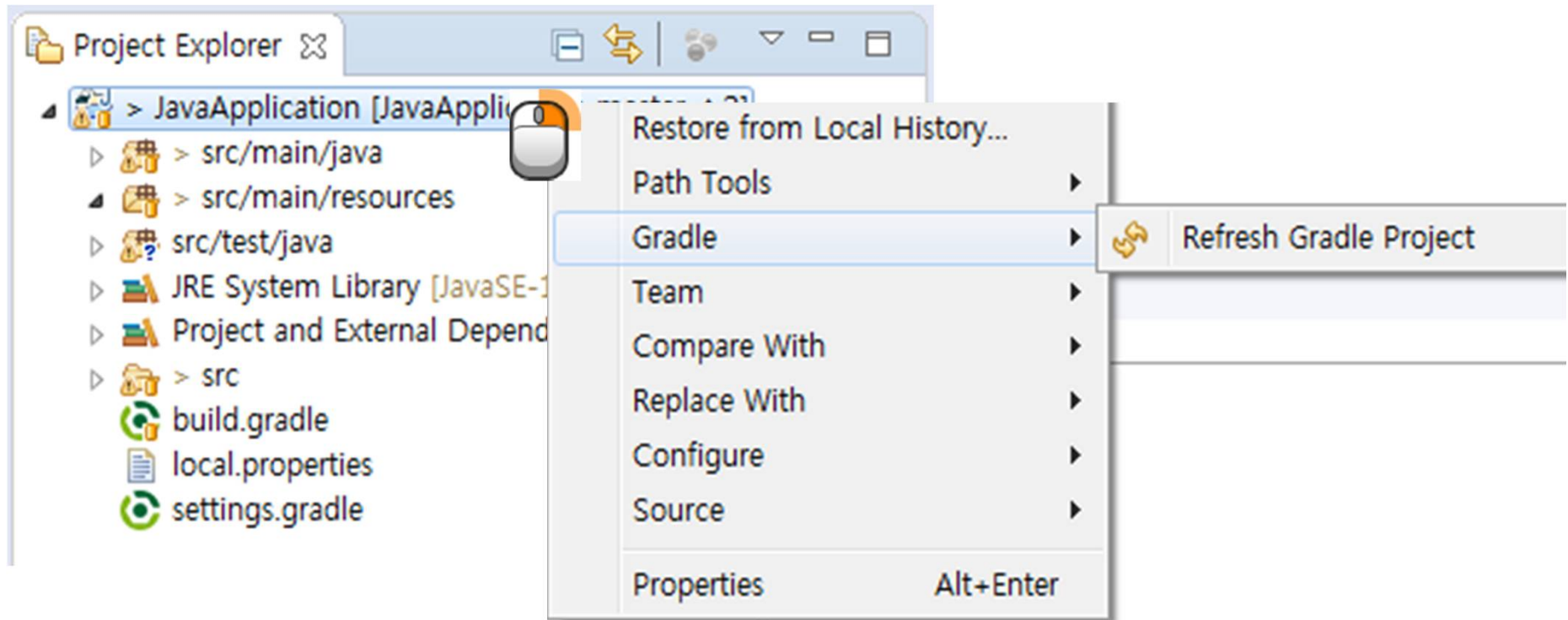
// mybatis library
mybatis
mybatis-spring

// Apache Commons package
commons-lang3
commons-dbcp2
```



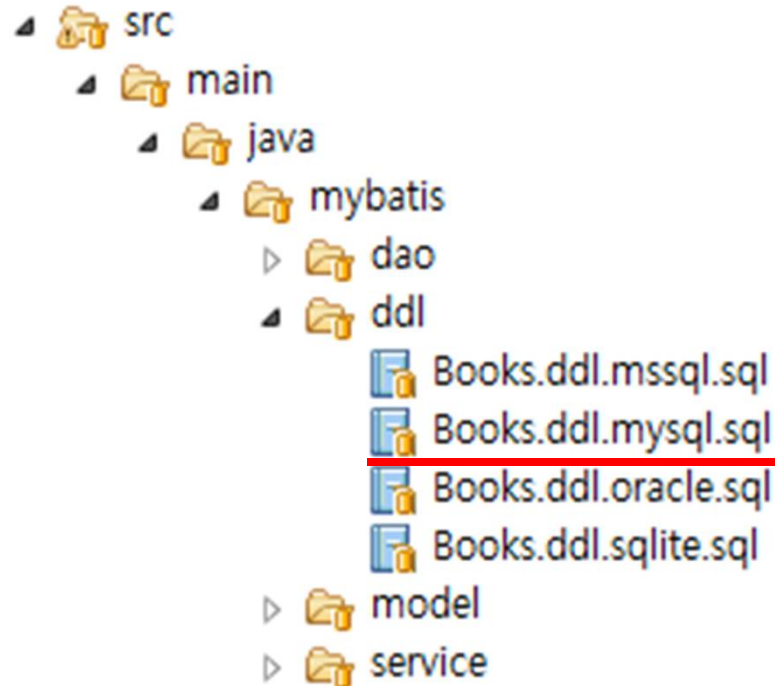
gradle refresh 실행

- 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)
);
```



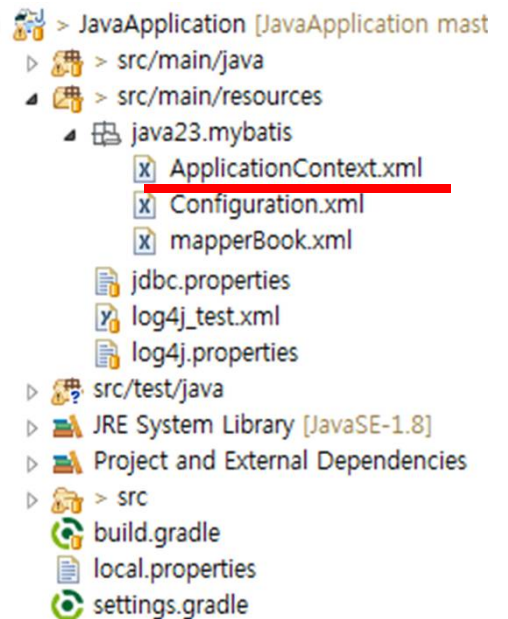
ApplicatonContext.xml 생성

```
<?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">
```





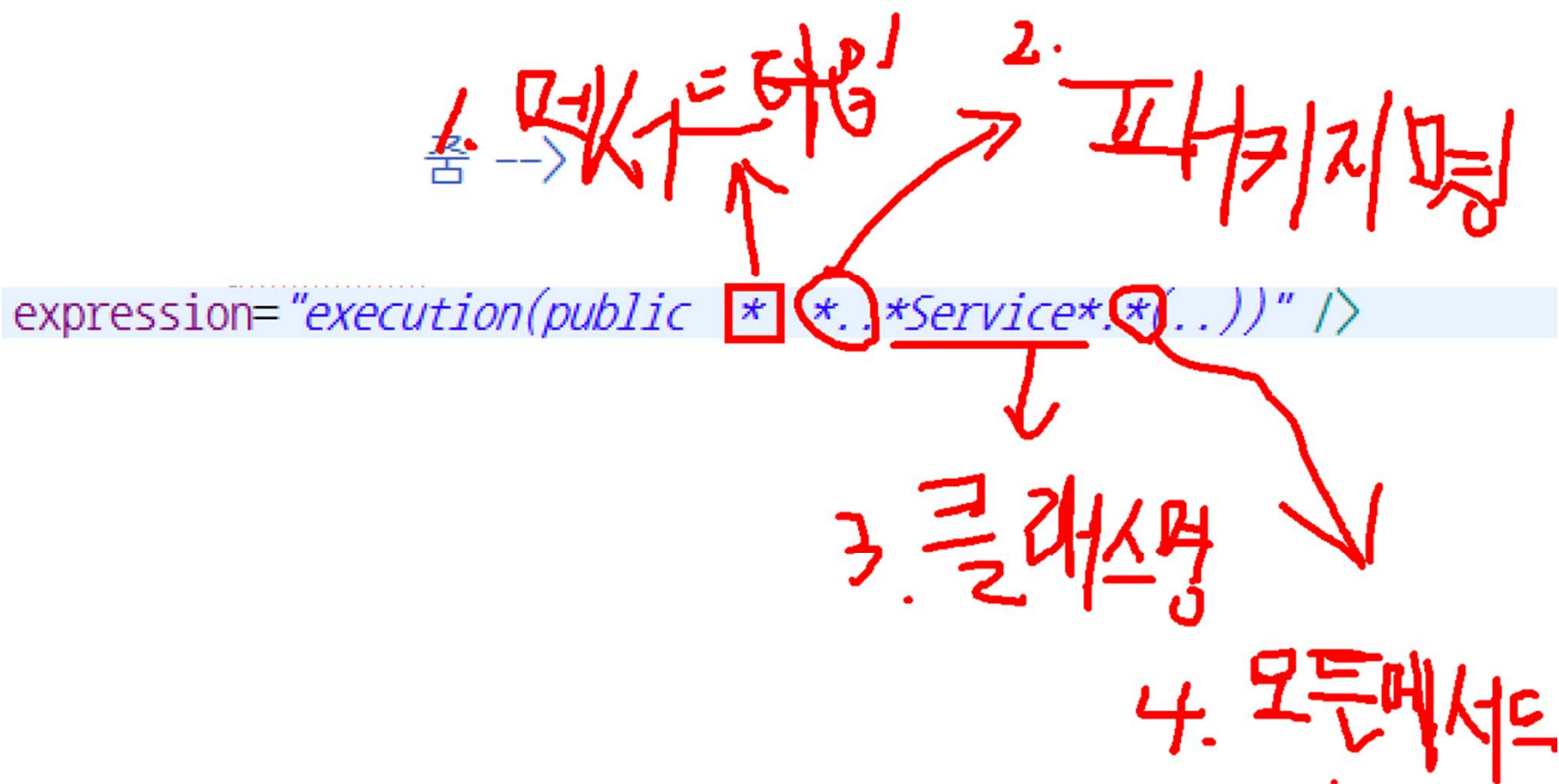
트랜잭션 설정

```
<!-- step4. 트랜잭션 설정 : commit, rollback -->
<beans:bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <beans:property name="dataSource" ref="dataSource" />
</beans:bean>
<!-- step4.1 실질적으로 트랜잭션 advice가 어디서 필요한지 알려줌 -->
<aop:config proxy-target-class="true">
    <!-- 패키지 mybatis. 이하의 모든 메서드에 pointcut을 걸음. -->
    <aop:pointcut id="controllerTx"
        expression="execution(public * *..*Service*.*(..))" />
    <aop:advisor pointcut-ref="controllerTx" advice-ref="txAdvice" />
</aop:config>
<!-- step4.2 Transaction 메서드 설정 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="insert*"          rollback-for="RuntimeException" />
        <tx:method name="update*"          rollback-for="RuntimeException" />
        <tx:method name="delete*"          rollback-for="RuntimeException" />
        <tx:method name="select*"          read-only="true" />
    </tx:attributes>
</tx:advice>
```



트랜잭션 설정

expression="execution(public * *..*Service*.*(..))"





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">
```

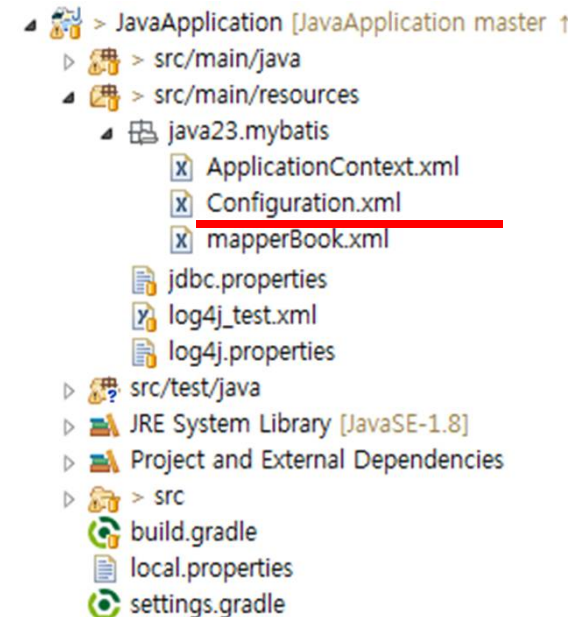
```
<configuration>
```

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

```
    <!-- Model 클래스의 패키지명을 등록한다. -->
    <typeAliases>
        <package name="java23.mybatis.model" />
    </typeAliases>
```

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

#####

console 설정

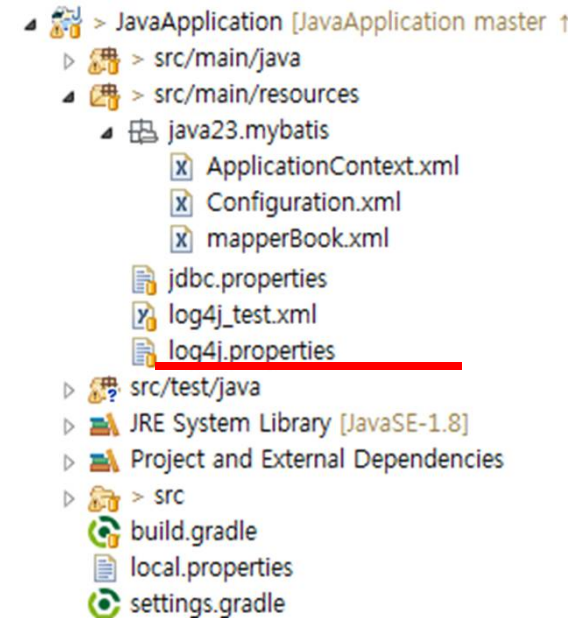
#####

콘솔에 뿌려줌

log4j.appender.stout=org.apache.log4j.ConsoleAppender

DEBUG 이상 레벨에서만 찍는다.

log4j.appender.stout.Threshold=DEBUG





mapper 파일 생성

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



mapperBook.xml 생성(1/3)

```
<?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">

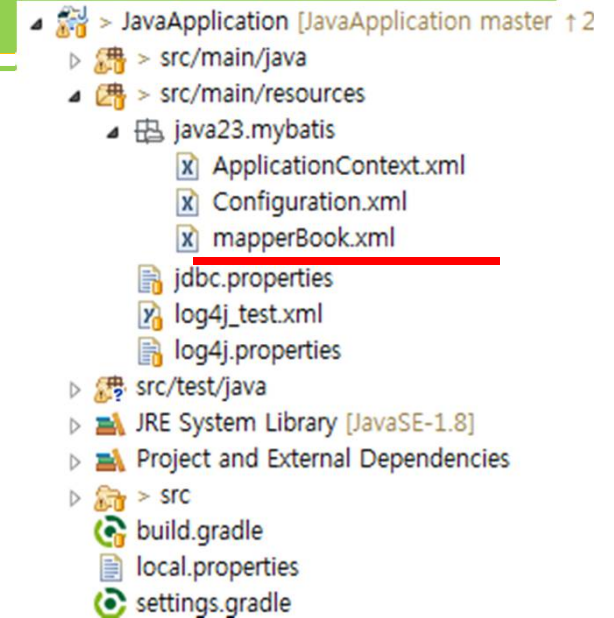
<mapper namespace="mapper.mapperBook">

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

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

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

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





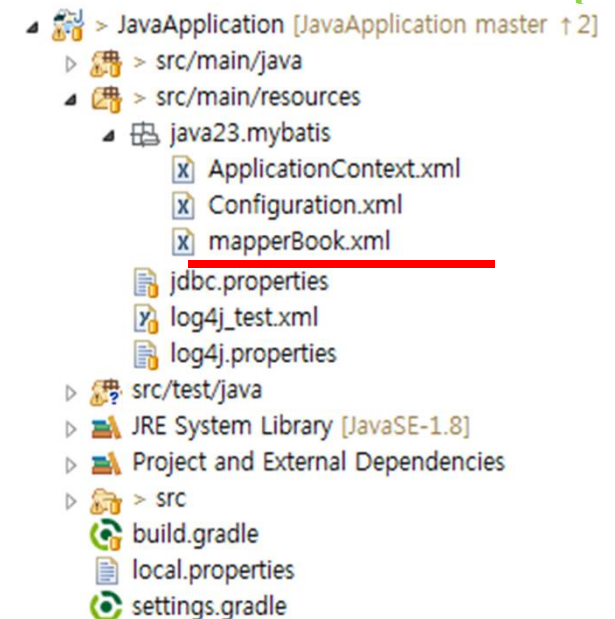
mapperBook.xml 생성(2/3)

- src/main/resources/mapper/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. mapperBook">

    <update id="update" 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)

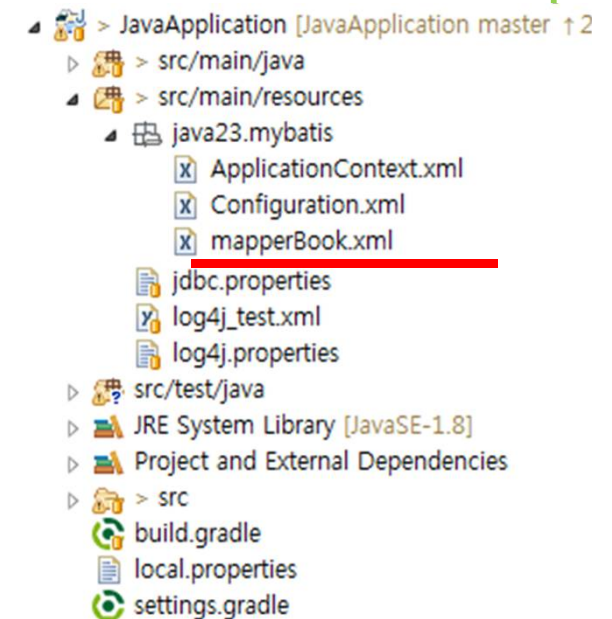
- src/main/resources/mapper/mapperBook.xml 파일 생성

```
<?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.mapperBook">

    <delete id="delete" 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>

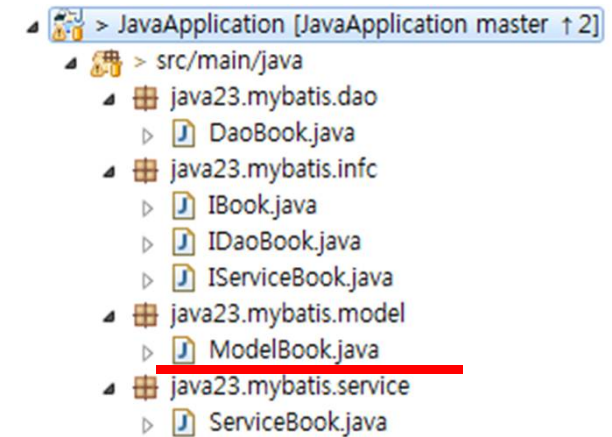
</mapper>
```





Model 클래스 생성

```
public class ModelBook {  
    private Integer bookid    = null ;  
    private String  bookname  = "" ;  
    private String  publisher = "" ;  
    private String  year      = "" ;  
    private Integer price     = null ;  
    private Date    dtm       = null ;  
    private Boolean use_yn     = null ;  
    private Integer authid    = null ;  
}
```

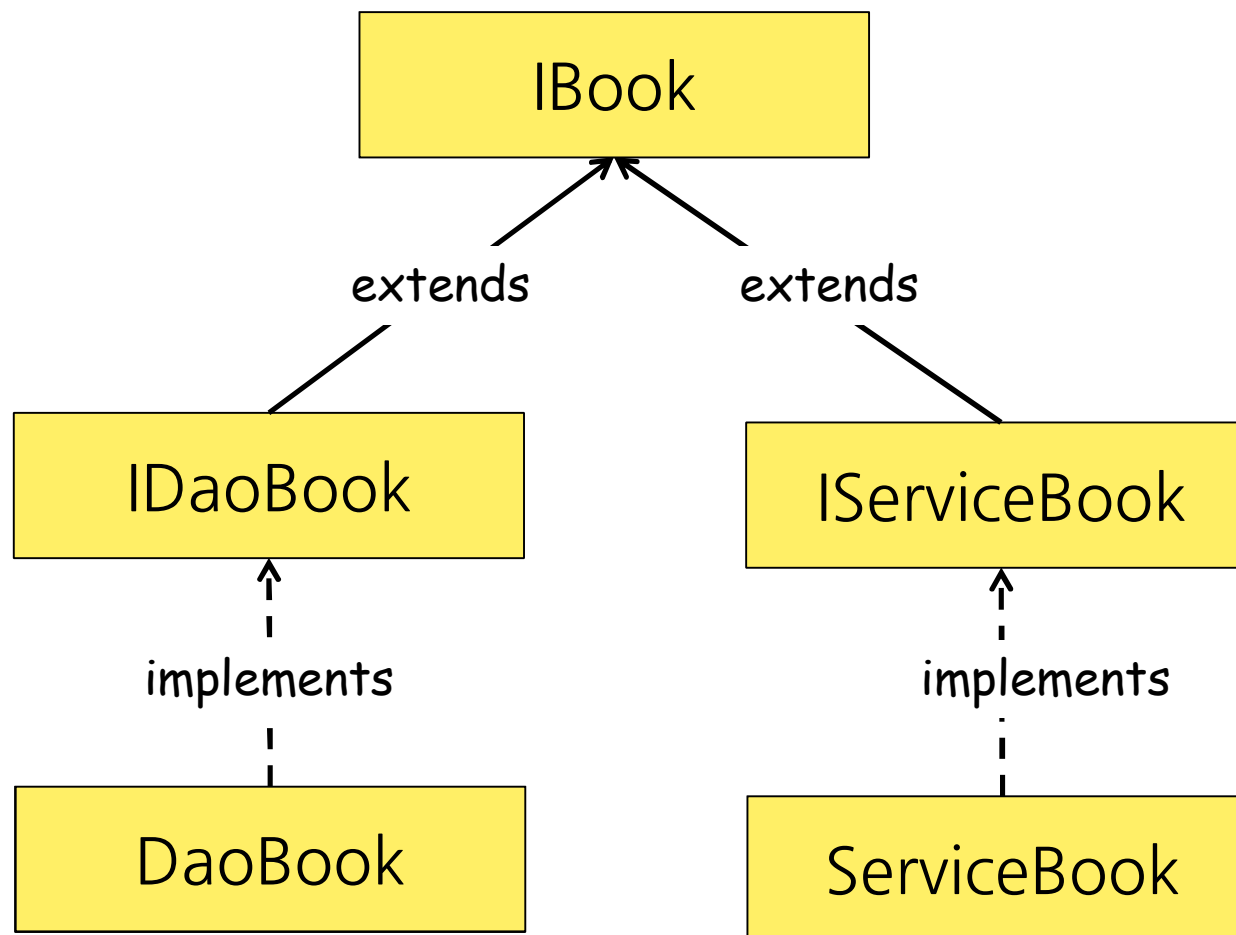


1. 필드 선언
2. getter / setter 만들기
3. Constructor 만들기
4. toString 만들기

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



Book 인터페이스 구조





IBook 인터페이스 만들기

```
public interface IBook {
```

```
    int getCount(ModelBook book) throws Exception;
```

```
    int getMaxBookid() throws Exception;
```

```
    List<ModelBook> selectAll(ModelBook book) throws Exception;
```

```
    List<ModelBook> selectLike(ModelBook book) throws Exception;
```

```
    List<ModelBook> selectEqual(ModelBook book) throws Exception;
```

```
    int insert(ModelBook book) throws Exception;
```

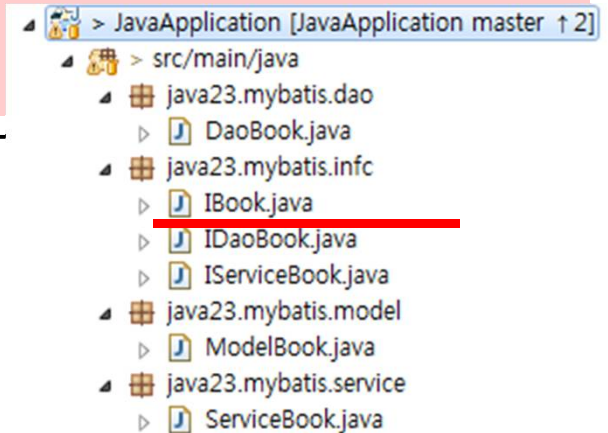
```
    int insertMap(String bookname, Date dtm, int authid) throws Exception;
```

```
    int update(ModelBook wherebook, ModelBook setbook ) throws Exception;
```

```
    int delete(ModelBook book) throws Exception;
```

```
    List<ModelBook> selectDynamic(ModelBook book) throws Exception;
```

```
}
```





Dao 클래스 생성

@Repository("daobook")

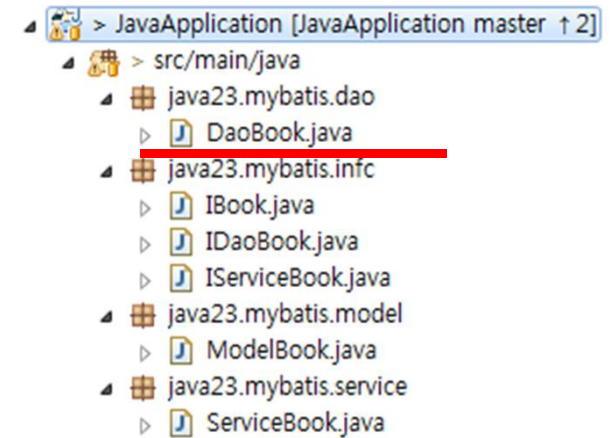
public class DaoBook implements IBook {

@Autowired

@Qualifier("sqlSession")

private SqlSession session;

}



빨간 글씨 부분을 추가.

SqlSession 은 java.sql.Connection과 같은 역할



DaoBook.selectEqual() 메서드

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

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

    return result;
}
```

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

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

`</mapper>`



DaoBook.insert() 메서드

```
@Override
public int insert(ModelBook book) {
    int result = 0 ;
    result = session.insert("mapper.mapperBook.insert", book);
    return result;
}
```

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

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




DaoBook.insertMap() 메서드

```
@Override
public int insertMap(String bookname, Date dtm, int authid) throws Exception {
    int result = -1;
    Map<String, Object> map = new HashMap<>();
    map.put("bookname", bookname);
    map.put("dtm", dtm);
    map.put("authid", authid);
    result = session.insert("mapper.mapperBook.insertMap", map);
    return result;
}
```

<mapper namespace="mapper.mapperBook">

```
<insert id="insert" parameterType="ModelBook">
    INSERT INTO BOOK( BOOKNAME , DTM , AUTHID )
    VALUES( #{bookname},#{dtm},#{authid} )
</insert>
```

</mapper>



아래의 Dao 메서드를 작성하시오

- getCount(ModelBook book)
- getMaxBookid()
- selectAll(ModelBook book)
- selectLike(ModelBook book)
- update(ModelBook wherebook, ModelBook setbook)
- delete(ModelBook book)
- selectDynamic(ModelBook book)



Service 클래스 생성

@Service("servicebook")

public class **ServiceBook** implements IServiceBook {

private static Logger *logger* = LoggerFactory.getLogger(**ServiceBook**.class);

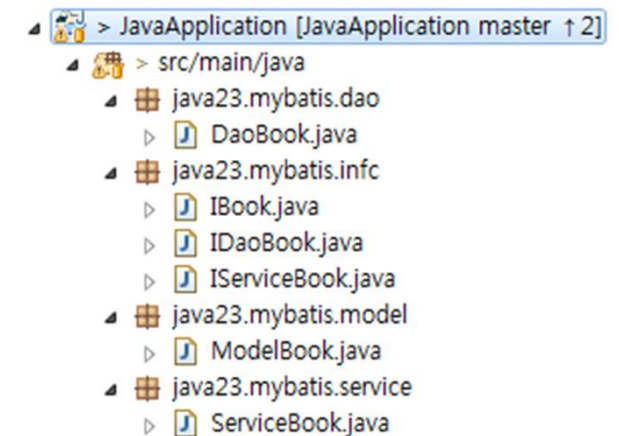
@Autowired

@Qualifier("daobook")

private IDaoBook dao;

}

빨간 글씨 부분을 추가.





ServiceBook.selectEqual() 메서드

```
@Override
public List<ModelBook> selectEqual(ModelBook book) throws Exception
{

    List<ModelBook> result = null;

    try {
        result = dao.selectEqual(book);
    } catch (Exception e) {
        logger.error( "selectEqual " + e.getMessage() );
    }

    return result;
}
```



ServiceBook.insert() 메서드

```
@Override
public int insert(ModelBook book) throws Exception {

    int result = -1;

    try {
        result = dao.insert(book);
    } catch (Exception e) {
        logger.error( "insert " + e.getMessage() );
    }

    return result;
}
```



ServiceBook.insertMap() 메서드

```
@Override
public int insertMap(String bookname, Date dtm, int authid)
    throws Exception {
    int result = -1;
    try {
        result = dao.insertMap(bookname, dtm, authid);
    } catch (Exception e) {
        logger.error( "insertMap " + e.getMessage() );
    }
    return result;
}
```

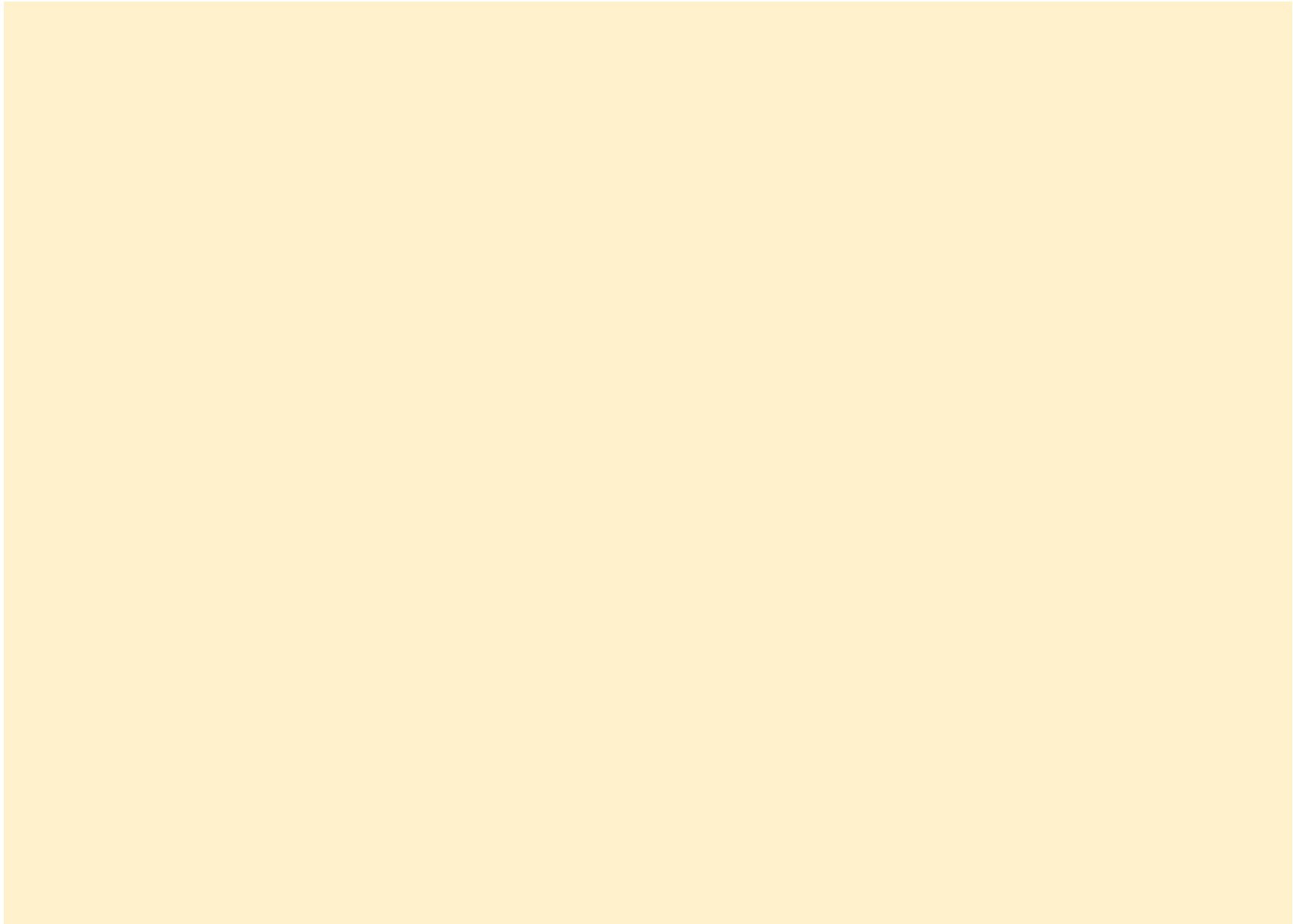


아래의 Service 메서드를 작성하시오

- getCount(ModelBook book)
- getMaxBookid()
- selectAll(ModelBook book)
- selectLike(ModelBook book)
- update(ModelBook wherebook, ModelBook setbook)
- delete(ModelBook book)
- selectDynamic(ModelBook book)

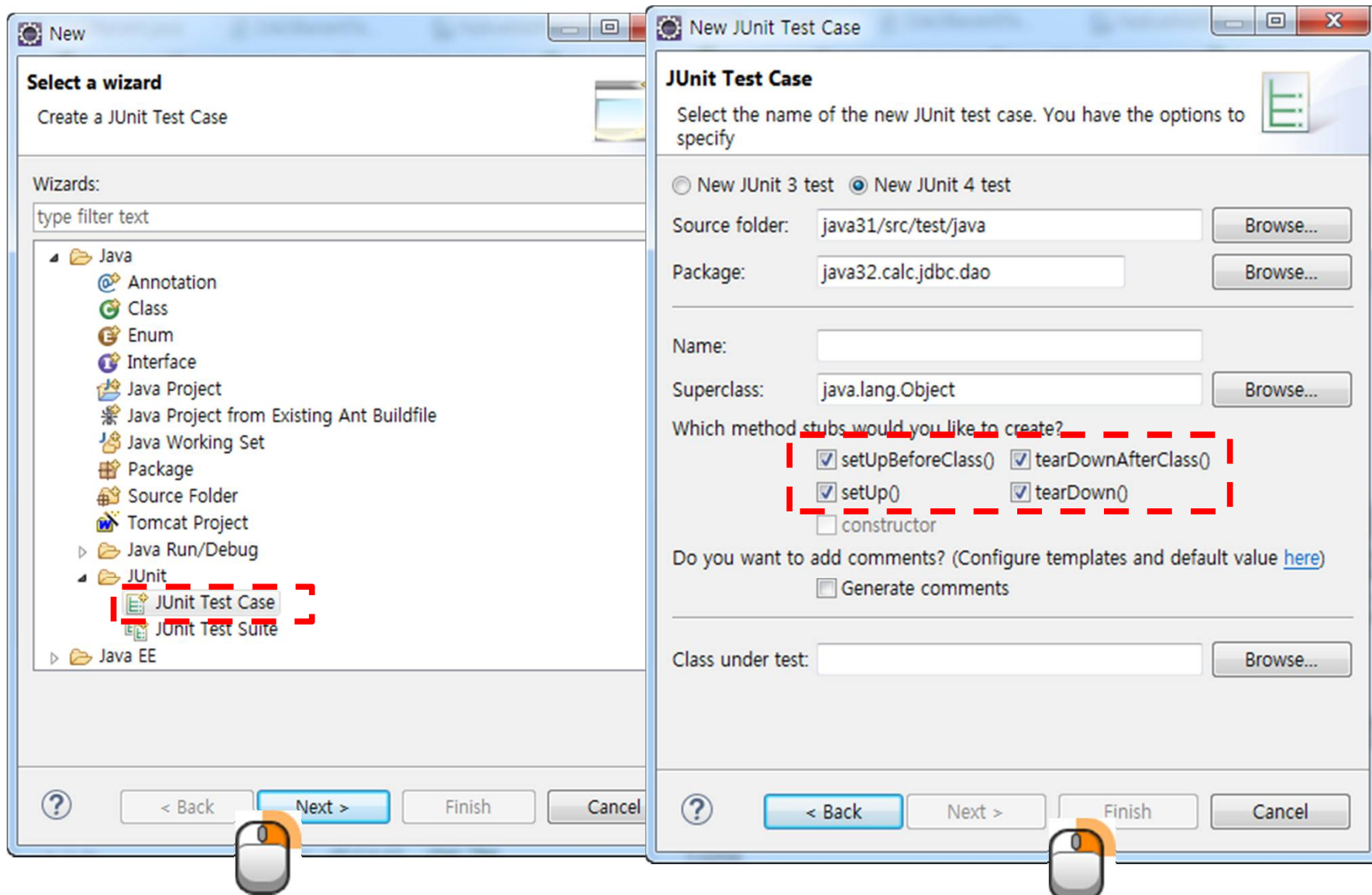


JUnit 테스트





JUnit Test Case 추가





JUnit Test Case 추가

The image shows a sequence of steps to add a JUnit test case in an IDE. On the left, the 'Window' menu is open, and 'Show View' is selected. This opens the 'Show View' dialog box in the center, which lists various views. The 'JUnit' view is highlighted with a red dashed box. On the right, the JUnit test runner output is shown, indicating a successful test run.

Show View Dialog:

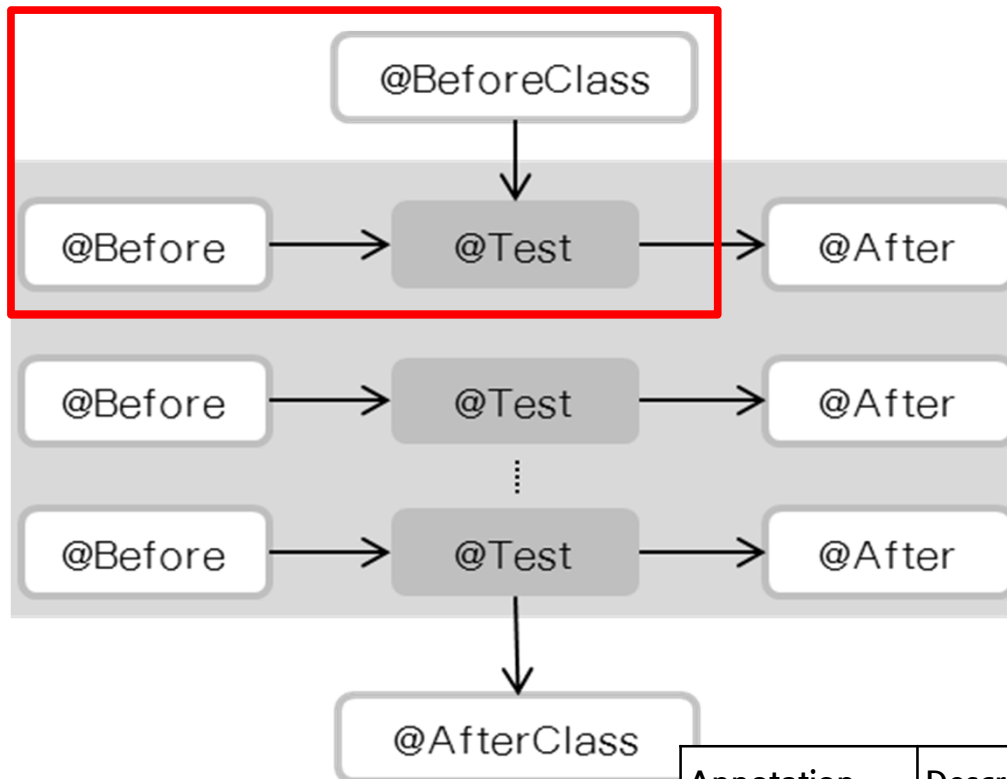
- type filter text
- Help
- Java
 - Call Hierarchy
 - Declaration
 - @ Javadoc
 - JUnit**
 - Package Explorer
 - Type Hierarchy
- Java Browsing
- JavaFX
- JavaScript
- JavaServer Faces
- JPA
- Make
- Markdown

JUnit Test Runner Output:

- JUnit
- Finished after 0.094 seconds
- Runs: 1/ Errors: Failures:
- test [Runner: JUnit 4] (0.000 s)
- Failure Trace



JUnit annotations



Annotation	Description
@Test	The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.
@BeforeClass	Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class.
@Before	Several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before each Test method.



JUnit assert 주요 메서드

메서드명	설명
<code>assertNull(a);</code>	객체 A가 null이 아님을 확인한다.
<code>assertNotNull(a);</code>	객체 A가 null이 아님을 확인한다.
<code>assertEquals(a, b);</code>	객체 A와 B가 일치함을 확인한다.
<code>assertNotEquals(a, b);</code>	
<code>assertSame(a, b);</code>	객체 A와 B가 같은 값을 확인한다.
<code>assertNotSame(a, b);</code>	
<code>assertTrue(a);</code>	조건 A가 참인가를 확인한다.
<code>assertFalse(a);</code>	조건 A가 참인가를 확인한다.
<code>assertArrayEquals(a, b);</code>	배열 A와 B가 일치함을 확인한다.

단정문 assertEquals와 assertSame의 차이점

- assertEquals는 두 값이 같은지를 비교하는 단정문.
- assertSame은 두 객체가 동일한 객체인지 주소값으로 비교하는 단정문.



실행 순서

```
ServiceBookTest.java
22 id setUpBeforeClass() throws Exception {
23
24
25 path 를 이용한 설정 파일 로딩
26 ionContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
27
28 을 이용한 설정 파일 로딩
29 cationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
30 o.File log4jfile = new java.io.File("log4j.properties");
31 ationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
32
33 이용한 servicebook 이스터시 생성
34 = context.getBean("servicebook");
35 eption e) {
36 tackTrace();
37
38
39
40
41
42
43 ctAll() throws Exception {
44 ok = new ModelBook();
45 ok> result = service.selectAll(t);
46 result.size(), 9);
47
```

```
ServiceBook.java
1 package java23.mybatis.service;
2
3 import org.slf4j.Logger;
4
5 @Service("servicebook")
6 public class ServiceBook implements IServiceBook {
7     private static Logger logger = LoggerFactory.getLogger(ServiceBook.class);
8
9     @Autowired
10    @Qualifier("daobook")
11    private IDaoBook dao;
12
13
```

```
DaoBook.java
1 package java23.mybatis.dao;
2
3 import org.apache.ibatis.session.SqlSession;
4
5 @Repository("daobook")
6 public class DaoBook implements IDaoBook {
7
8     @Autowired
9     @Qualifier("sqlSession")
10    private SqlSession session;
11
12    @Override
13
```

```
ApplicationContext.xml
50 <beans:property name="username" value="root" />
51 <beans:property name="password" value="1234" />
52
53 <beans:property name="defaultAutoCommit" value="true" />
54 <beans:property name="poolPreparedStatements" value="true" />
55 <beans:property name="cacheState" value="true" />
56 </beans:bean>
57
58 <!-- SessionFactory 설정 :: MyBatis가 사용하는 SessionFactory 설정 -->
59 <beans:bean id="sqlSessionFactory" class="org.apache.ibatis.session.SqlSessionFactory" />
60 <beans:property name="dataSource" ref="dataSource" />
61 <beans:property name="configLocation" value="classpath:mybatis-config.xml" />
62
63 <!-- mybatis 디렉토리에 xml 파일만 -->
64 <!-- <beans:property name="mapperLocations" value="classpath:mybatis/*.xml" /> -->
65 </beans:bean>
66
67 <!-- MyBatis의 CRUD 템플릿을 사용할 수 -->
68 <beans:bean id="sqlSession" class="org.apache.ibatis.session.SqlSession" />
69 <beans:constructor-arg index="0" ref="sqlSessionFactory" />
70 </beans:bean>
71
72
73 </beans>
74
```




Map을 이용하는 경우의 실행 순서

ServiceBookTest.java

```
84 @Test
85 public void insertMap() throws Exception {
86     int result = service.insertMap("test", Date.valueOf("2017-01-01"), 10);
87     assertEquals(result, 1);
88 }
```

ServiceBook.java

```
79 @Override
80 public int insertMap(String bookname, Date dtm, int authid) throws Exception {
81     int result = -1;
82     try {
83         result = dao.insertMap(bookname, dtm, authid);
84     } catch (Exception e) {
85         logger.error("insertMap " + e.getMessage());
86     }
87     return result;
88 }
```

DaoBook.java

```
61 @Override
62 public int insertMap(String bookname, Date dtm, int authid) throws Exception {
63     int result = -1;
64     Map<String, Object> map = new HashMap<>();
65     map.put("bookname", bookname);
66     map.put("dtm", dtm);
67     map.put("authid", authid);
68     result = session.insert("mapper.mapperBook.insertMap", map);
69     return result;
70 }
```

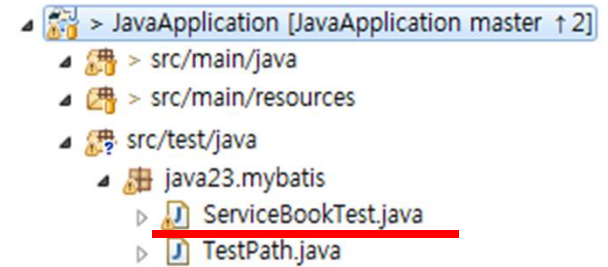
mapperBook.xml

```
23
24 <insert id="insertMap" parameterType="hashmap">
25     INSERT INTO BOOK( BOOKNAME, DTM, AUTHID)
26     VALUES( #{bookname},#{dtm},#{authid})
27 </insert>
```



7. JUnit Test 클래스 생성

```
public class ServiceBookTest {  
    private static ServiceBook 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/resources/ApplicationContext.xml")  
        ;  
  
        // DI를 이용한 servicebook 인스턴스 생성  
        service = context.getBean("servicebook", ServiceBook.class);  
    }  
}
```





7. JUnit Test 클래스 생성

```
public class TestServiceBook {
```

```
    @Test
```

```
    public void selectAll() {
```

```
        ModelBook book = new ModelBook();
```

```
        List<ModelBook> result = service.selectAll(book);
```

```
        assertEquals(result.size(), 14);
```

```
    }
```

```
    @Test
```

```
    public void selectEqual() {
```

```
        ModelBook book = new ModelBook();
```

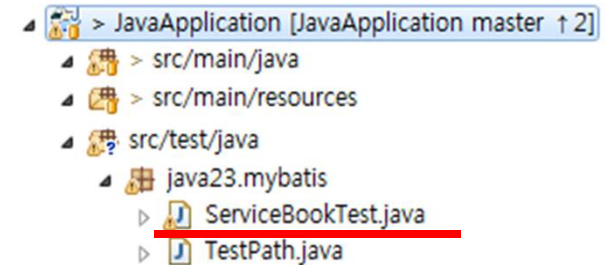
```
        book.setBookname("First SQL");
```

```
        List<ModelBook> result = service.selectEqual(book);
```

```
        assertEquals(result.size(), 1);
```

```
    }
```

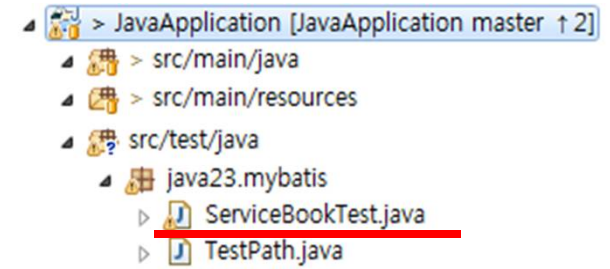
```
}
```





테스트 코드를 작성하시오.

```
public class TestServiceBook {  
  
    @Test  
    public void selectLike() {  
  
        // 테스트 코드를 작성하시오.  
  
    }  
}
```



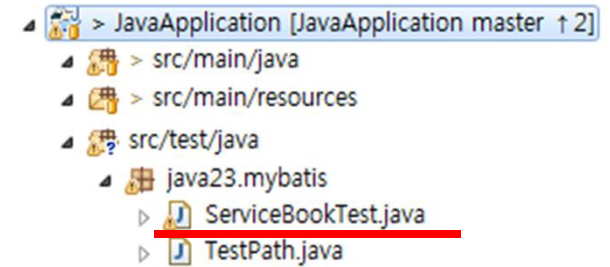


7. JUnit Test 클래스 생성

```
public class TestServiceBook {

    @Test
    public void insert() {
        ModelBook book = new ModelBook();
        book.setBookname("test");
        book.setDtm(Date.valueOf("2016-11-12"));
        book.setPrice(10000);
        book.setPublisher("내가");
        book.setUse_yn(true);
        book.setYear("2016");
        book.setAuthid(10);

        int result = service.insert(book);
        assertEquals(result, 1);
    }
}
```





7. JUnit Test 클래스 생성

```
public class TestServiceBook {
```

```
    @Test
```

```
    public void insertMap() {
```

```
        ModelBook book = new ModelBook();
```

```
        String bookname = "test";
```

```
        java.util.Date dtm = "2016-11-12";
```

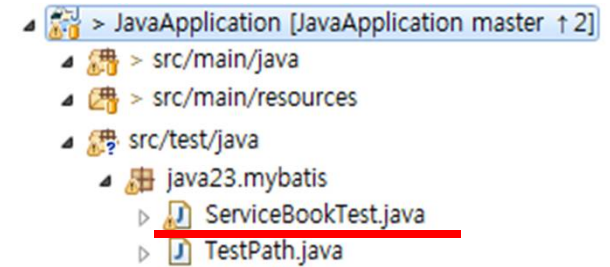
```
        int authid = 10 ;
```

```
        int result = service.insertMap( bookname, dtm, authid );
```

```
        assertEquals(result, 1);
```

```
    }
```

```
}
```





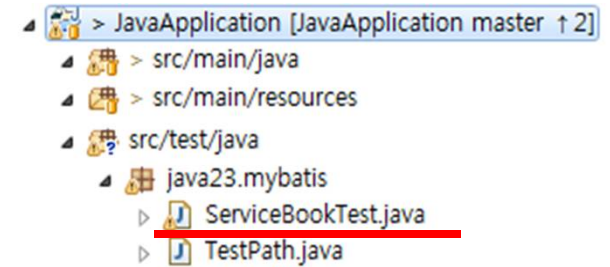
테스트 코드를 작성하시오.

```
public class TestServiceBook {

    @Test
    public void update() {
        // 테스트 코드를 작성하시오.
    }

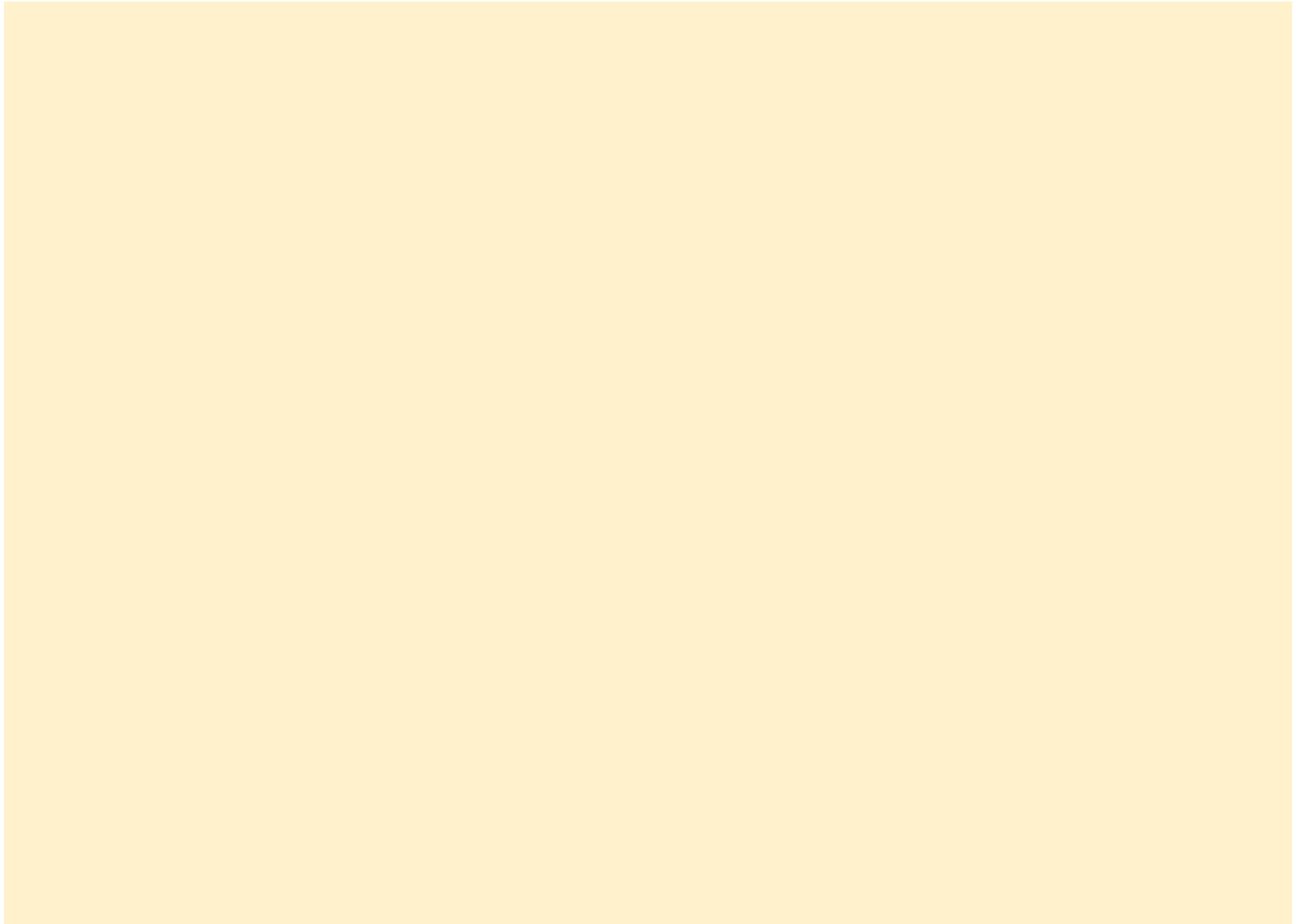
    @Test
    public void delete() {
        // 테스트 코드를 작성하시오.
    }

}
```



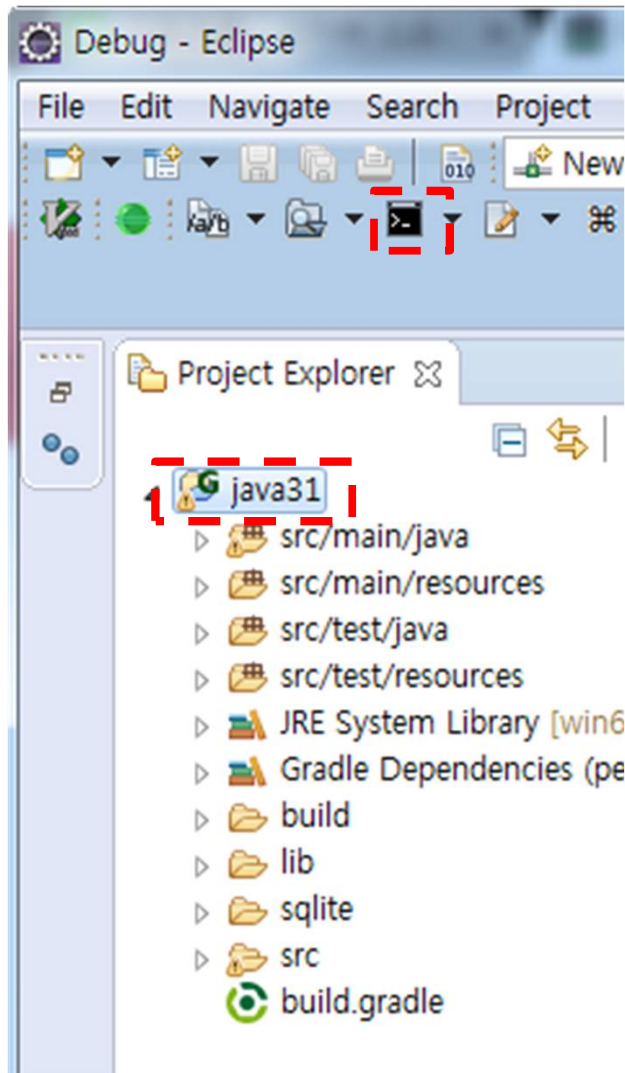


gradle





gradle로 build와 test 하기



```
관리자: C:\windows\system32\cmd.exe
D:\DevTool\workspace\Lecture\JAVA\기초\Work\java31>gradle clean build test
:clean
:taskCopy UP-TO-DATE
:compileJava
:processResources
:classes
:jar
:startScripts
:distTar
:distZip
:assemble
:compileTestJava
:processTestResources UP-TO-DATE
:testClasses
:test

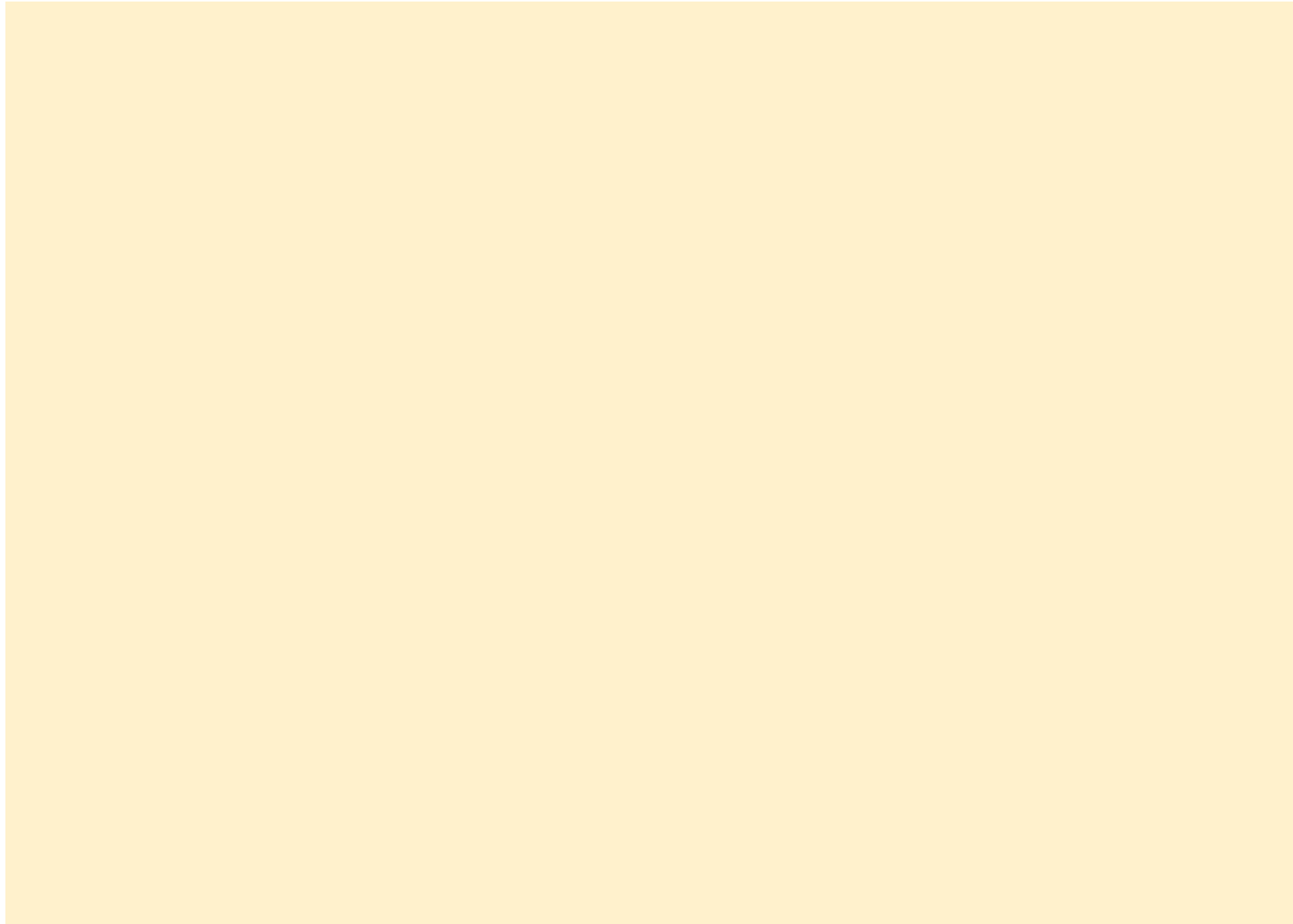
java32.calc.jdbc.dao.DAORecentTest > test_setSQLInsert STARTED
java32.calc.jdbc.dao.DAORecentTest > test_setSQLInsert PASSED
java32.calc.jdbc.dao.DAORecentTest > test_getSQLSelectAll STARTED
java32.calc.jdbc.dao.DAORecentTest > test_getSQLSelectAll PASSED
java32.calc.jdbc.dao.DBConnectTest > test_DBConnect STARTED
java32.calc.jdbc.dao.DBConnectTest > test_DBConnect PASSED
:check
:build

BUILD SUCCESSFUL

Total time: 10.325 secs
D:\DevTool\workspace\Lecture\JAVA\기초\Work\java31>
```



DbUnit 테스트





통합 테스트

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

The End



- 팀 프로젝트
- 기간: 17.05.29~17.06.02
- springboard DAL 모듈 만들기
 1. 인터페이스
 2. Dao 클래스
 3. Service 클래스
 4. JUnit 테스트 코드 작성



mapperBoard 만들기

- 테이블 생성
 - mapperBoard.xml 만들기
 - Model 만들기
 - Configuration.xml 수정
-
- 인터페이스 만들기
 - Dao 만들기
 - Service 만들기
 - 테스트 코드 만들기
-
- Spring DI 적용
 - Git Hub 에 소스 올리기



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
- Configuration.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();
    }

}
```