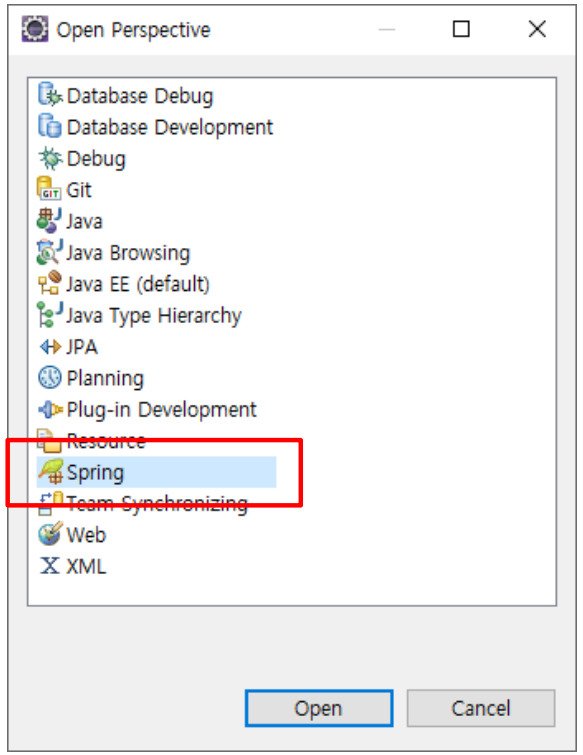
2. 스프링 MVC 프레임워크

1. 스프링 MVC 프로젝트
2. Mybatis 설정
3. JUNIT
4. SQL 로그
5. 컨트롤러와 웹페이지 작성
6. DYNAMIC WEB PROJECT 를 SPRING 프로젝트로 변경

1.1 perspective를 spring으로 변경

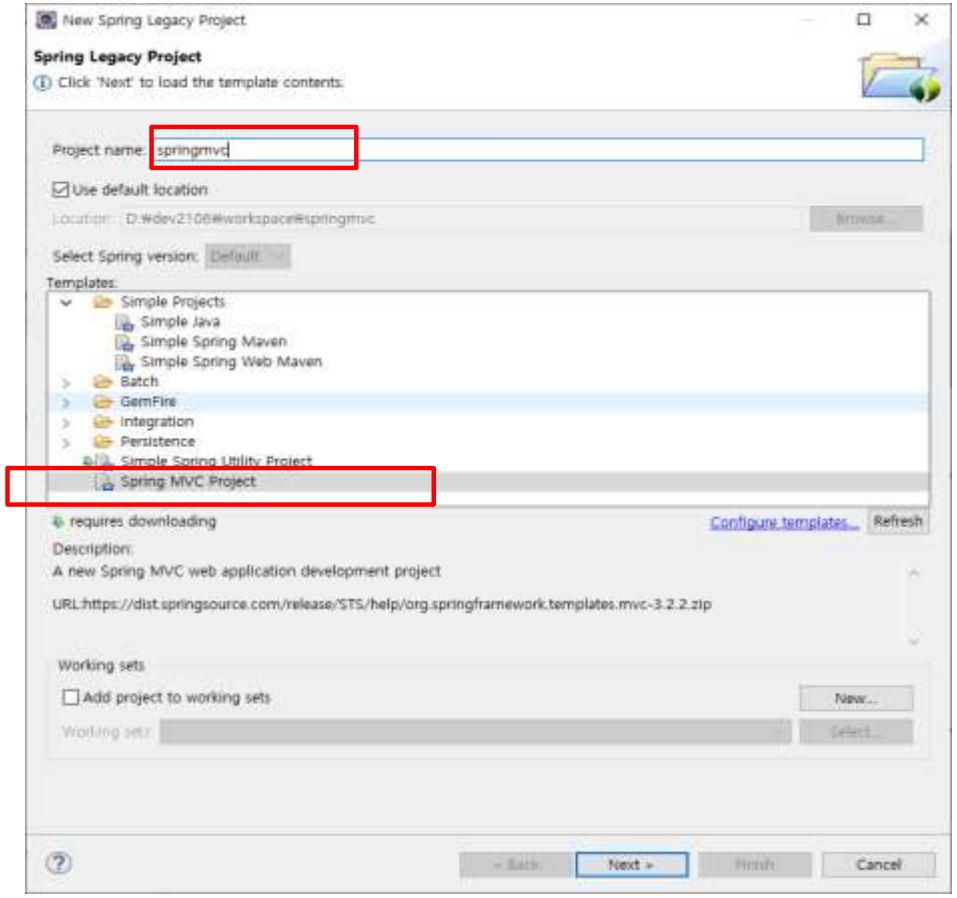
- Open Persfective

- spring 선택

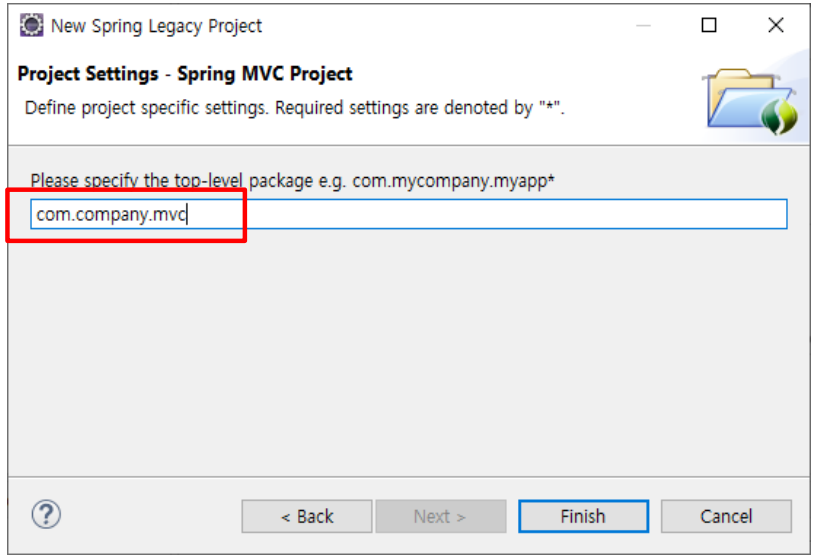


1.2 Spring Legacy Project

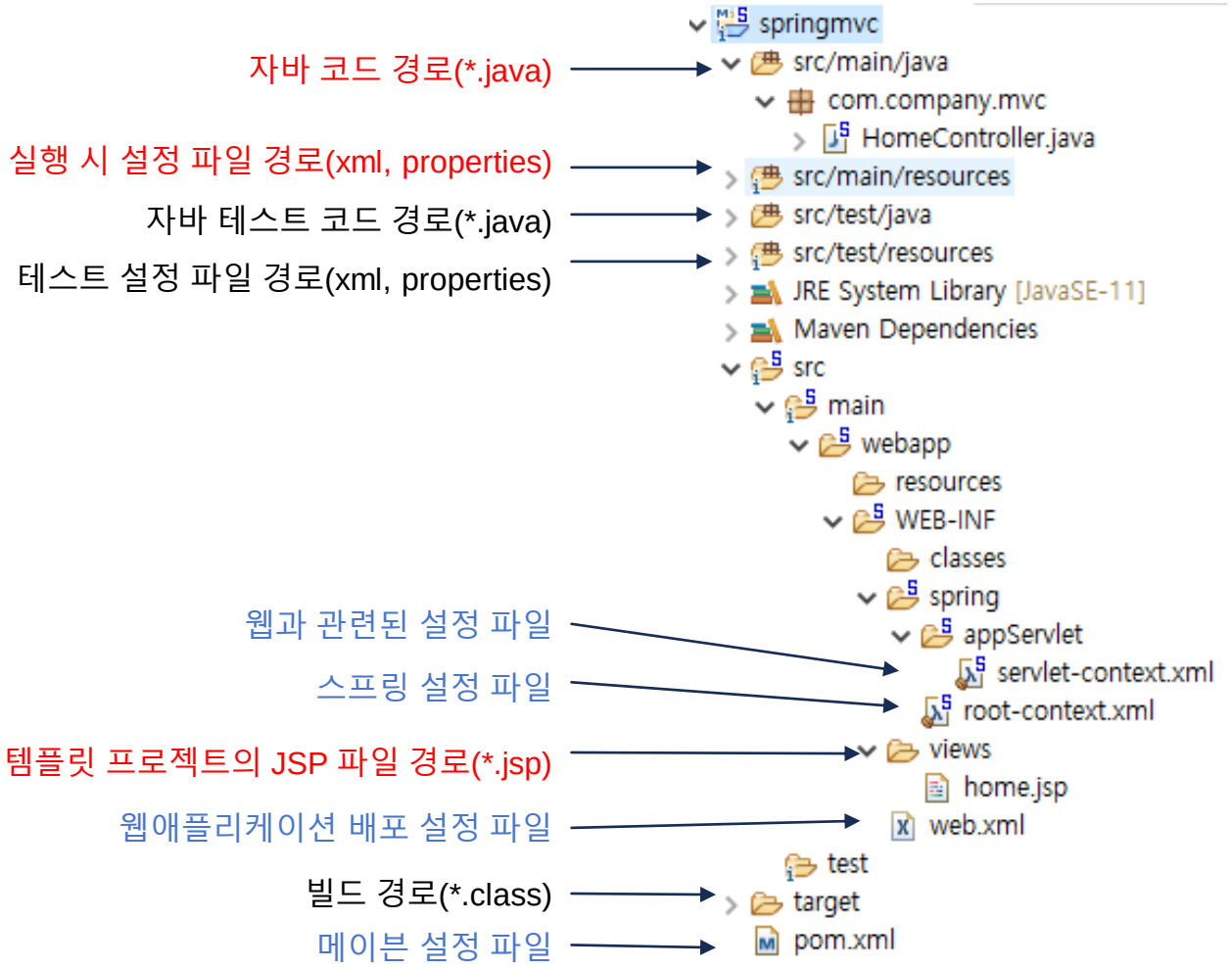
- File -> New -> spring Legacy Project
  - 템플릿에서 Spring MVC Project 선택



  - 패키지명 입력

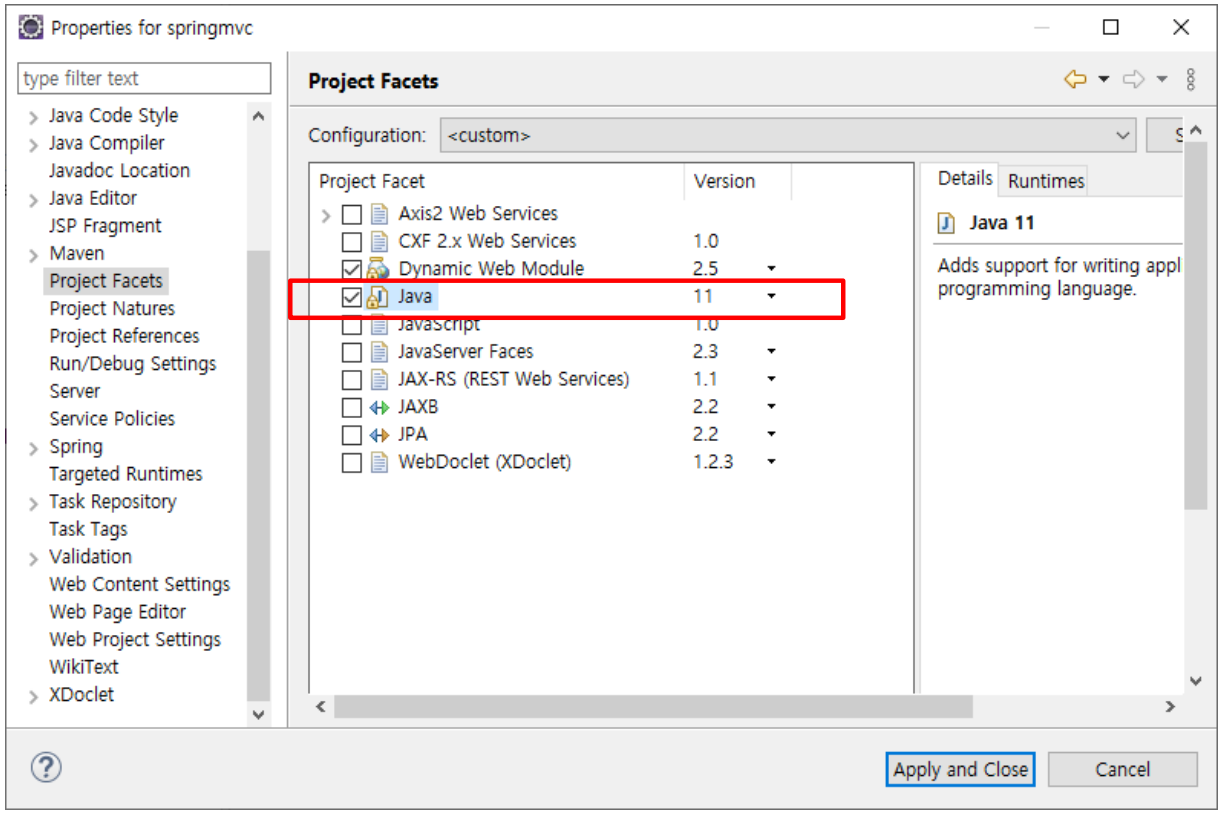


1.3 프로젝트 구조


자바 코드 경로(*.java)
실행 시 설정 파일 경로(xml, properties)
자바 테스트 코드 경로(*.java)
테스트 설정 파일 경로(xml, properties)
웹과 관련된 설정 파일
스프링 설정 파일
템플릿 프로젝트의 JSP 파일 경로(*.jsp)
웹애플리케이션 배포 설정 파일
빌드 경로(*.class)
메이븐 설정 파일
springmvc
src/main/java
com.company.mvc
HomeController.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
webapp
resources
WEB-INF
classes
spring
appServlet
servlet-context.xml
root-context.xml
views
home.jsp
web.xml
test
target
pom.xml

1.4 java version 변경

- properties -> Project facets



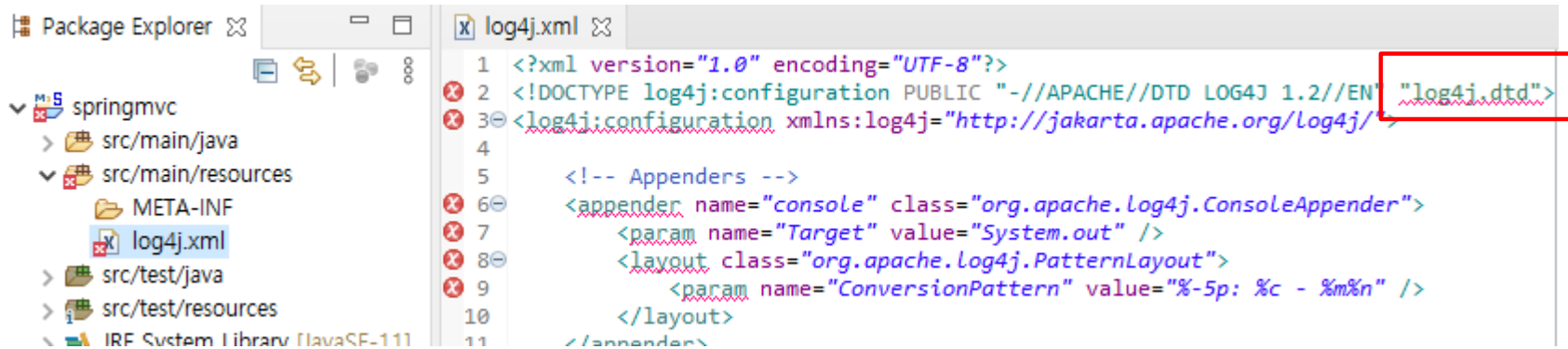
1.5 Web Module 버전 변경

- \src\main\webapp\WEB-INF\web.xml
  - namespace 변경

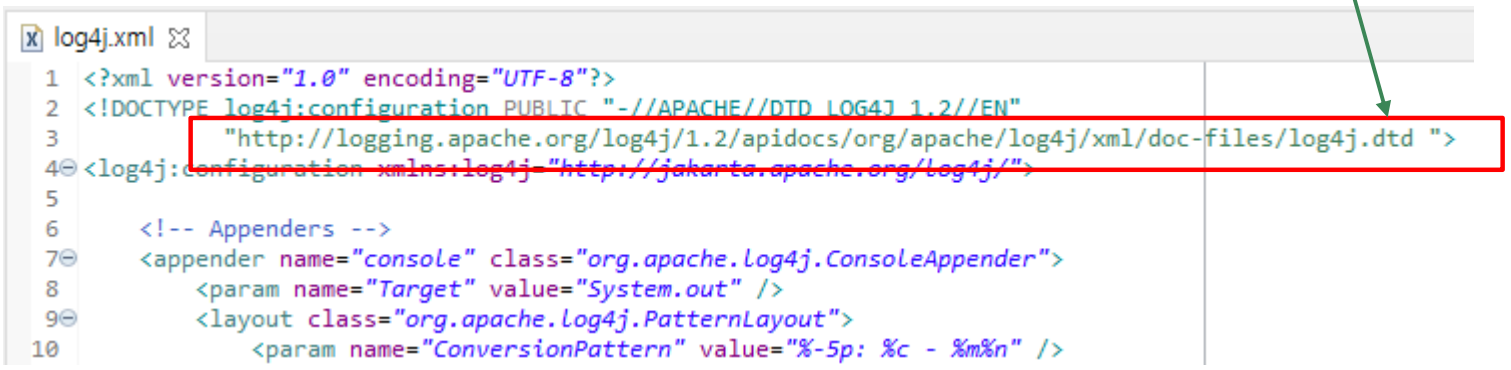
```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"    version="3.1">
```

1.6 log4j.xml dtd 경로 수정

- log4j.xml dtd 경로 수정



http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/doc-files/log4j.dtd



1.7 pom.xml 변경

- version 변경
  - java version 1.8 -> 11
  - org.springframework-version 3.1.1.RELEASE -> 5.3.16
  - org.aspectj-version 1.6 -> 1.9.0
  - log4j version 1.2.15 -> 1.2.17
  - junit version 4.7 -> 4.12
- 교체
  - servlet-api 2.5 -> 3.1.0
- <dependency> 추가
  - spring-test
  - Lombok
  - Jackson

```
<!- 기존의 servlet-api를 교체 -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

```
<!-- spring-test -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>${org.springframework-version}</version>
</dependency>

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

<!-- jackson -->
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.13.2.2</version>
</dependency>
```

2.1 커넥션 풀 설정

- pom.xml 에 <dependency> 추가
  - HikariCP
  - spring-jdbc (spring-tx 포함됨)
  - ojdbc8

```
<!-- Database connection pool -->
<dependency>
<groupId>com.zaxxer</groupId>
<artifactId>HikariCP</artifactId>
<version>5.0.1</version>
</dependency>

<!-- spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${org.sprigframework-version}</version>
</dependency>

<!-- ojdbc8 -->
<dependency>
<groupId>com.oracle.database.jdbc</groupId>
<artifactId>ojdbc8</artifactId>
<version>19.3.0.0</version>
</dependency>
```

- src\main\webapp\WEB-INF\spring\root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring
http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"   value="hr" />
  <property name="password"   value="hr" />
</bean>

<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
      destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

</beans>
```

2.2 Mybatis 설정

- pom.xml 에 <dependency> 추가
  - mybatis-spring
  - mybatis

```
<!-- mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.5.9</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>2.0.6</version>
</dependency>
```

- src\main\webapp\WEB-INF\spring\root-context.xml

```
<!-- mybatis  SqlSessionFactory -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
</bean>

<!– mapper scan -->
<mybatis-spring:scan base-package="com.company.**.mapper" />
```

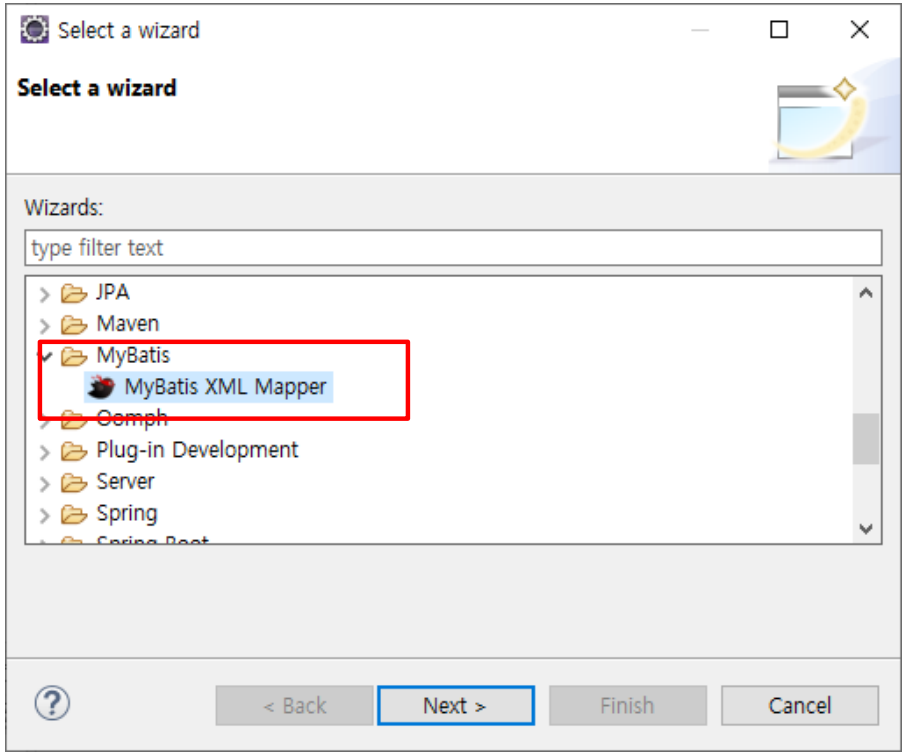
2.1 Mybatis 설정

- src\main\webapp\WEB-INF\spring\root-context.xml

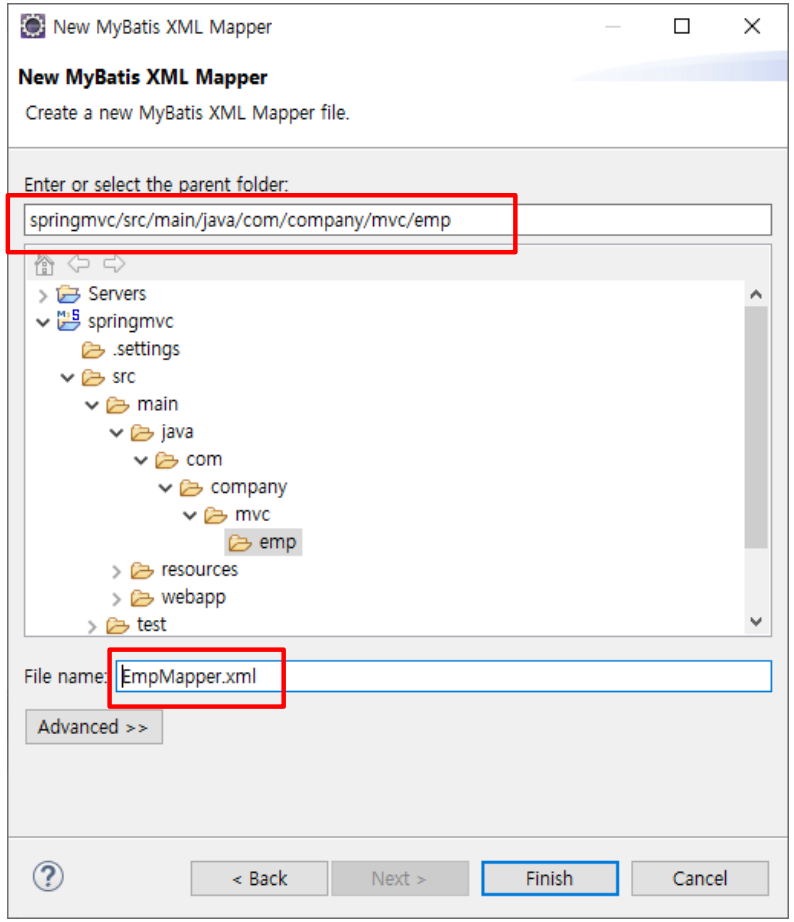
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"        value="hr" />
  <property name="password"        value="hr" />
</bean>

<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

<!-- mybatis  SqlSessionFactory -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
</bean>
</beans>
```

2.2 Sql statement xml 파일 생성

- File -> New -> Other...



- 생성위치와 파일명 입력



2.2 Sql statement xml 파일 생성

- EmpVO

```
@Data
public class EmpVO {
    String employee_id;
    String first_name;
    String last_name;
    String email;
    String hire_date;
    String job_id;
    String department_id;
    String salary;
}
```

- mapper 인터페이스

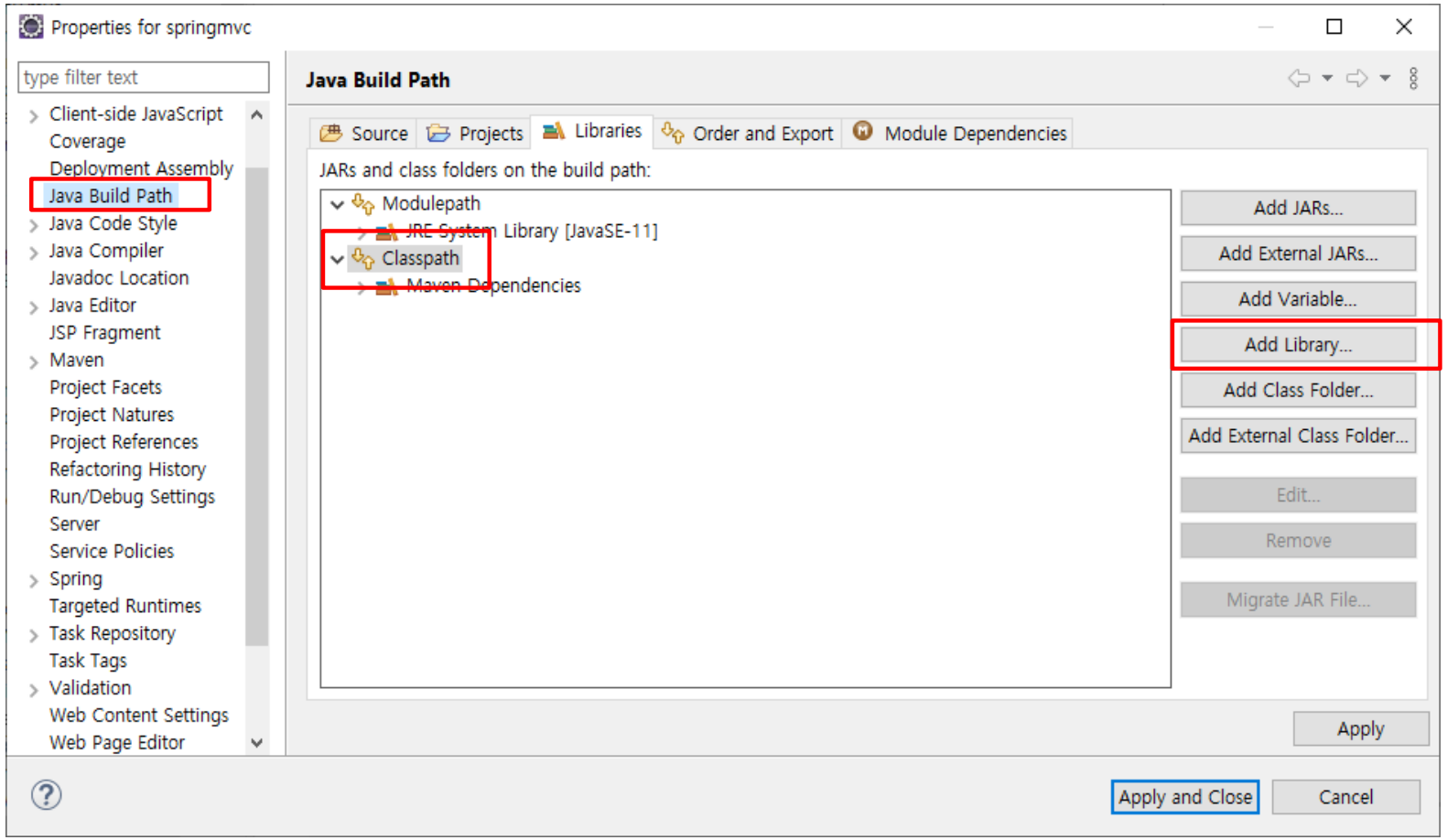
```
public interface EmpMapper {
  public EmpVO getEmp(EmpVO empVO);
}
```

- sql statmement xml 파일

```
<?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.company.mvc.emp.EmpMapper">
<select id="getEmp"
      parameterType="com.company.mvc.emp.EmpVO"
      resultType="com.company.mvc.emp.EmpVO">
SELECT  employee_id,
      first_name,
      last_name,
      email,
      hire_date,
      job_id,
      salary
 FROM employees
WHERE employee_id = #{employee_id}
</select>
</mapper>
```

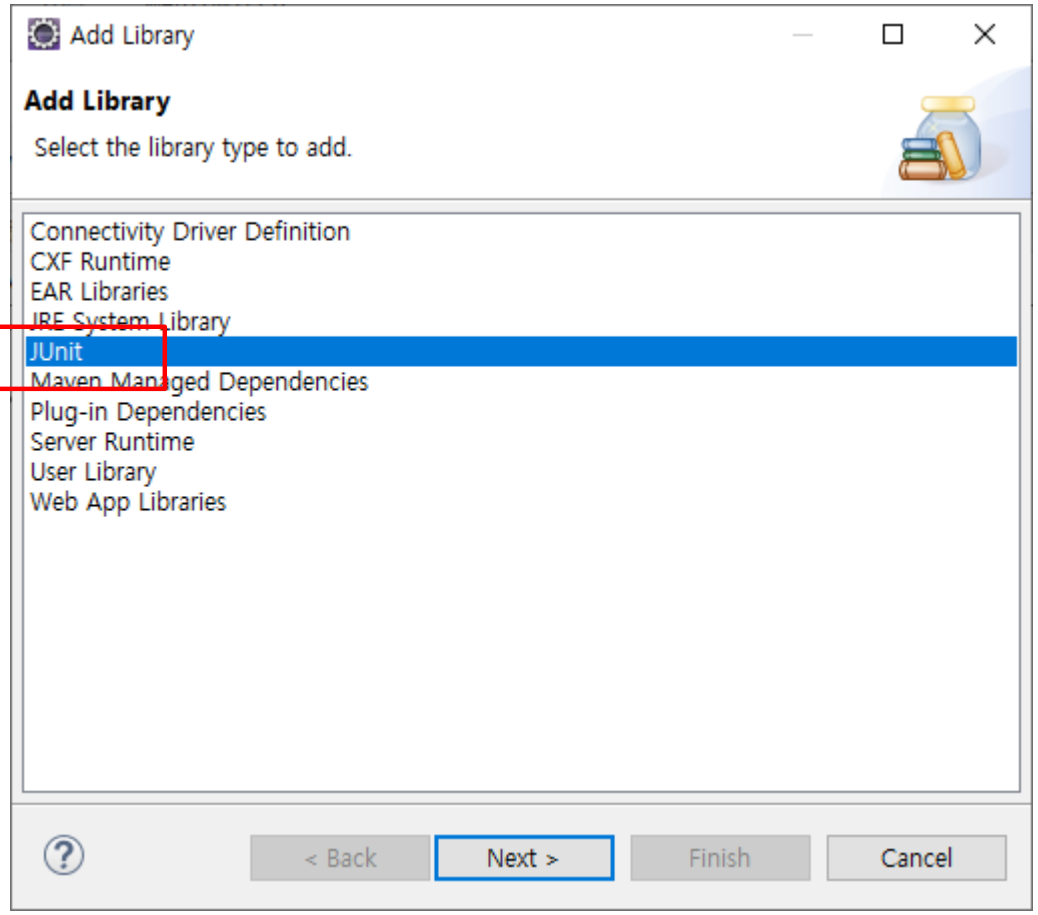
3.1 junit 라이브러리 추가

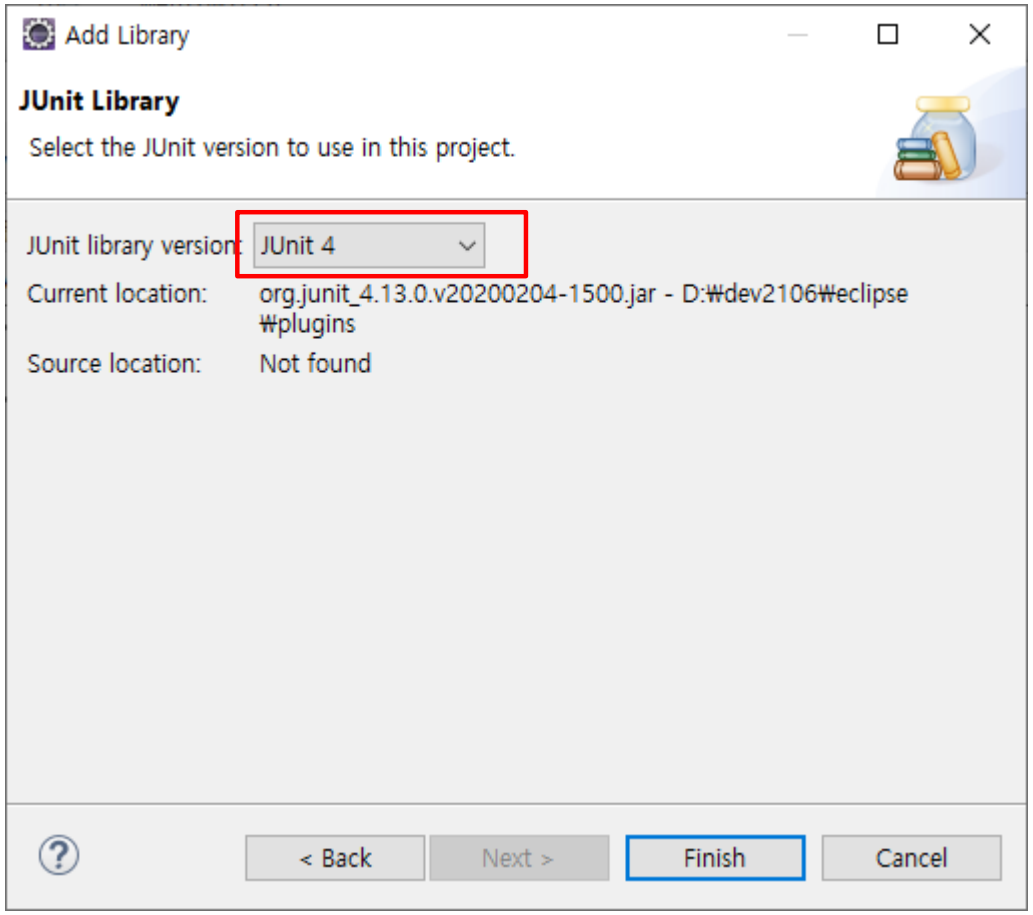
- Properties -> Java BuildPath -> Libraries 탭



3.1 junit 라이브러리 추가

- junit 라이브러리 추가





3.2 spring-test 라이브러리 추가

- Pom.xml

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
</dependency>


<dependency>
<groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
```

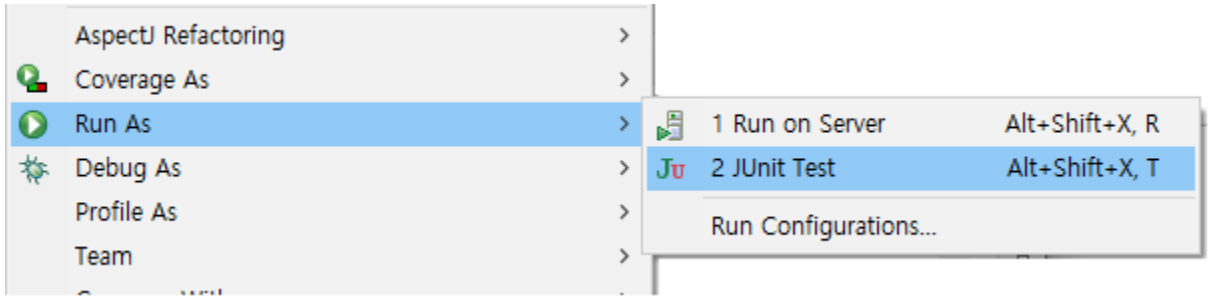
3.3 테스트 코드

- src/test/java/EmpMapperTest.java

```
package com.company.mvc.emp;

import static org.junit.Assert.assertEquals;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "file:src/main/webapp/WEB-INF/spring/root-
context.xml")
public class EmpMapperClient {

    @Autowired EmpMapper empMapper;

    @Test
    public void getEmp() {
        EmpVO vo = new EmpVO();
        vo.setEmployee_id("100");
        EmpVO findVO = empMapper.getEmp(vo);
        System.out.println(findVO.getLast_name());
        assertEquals(findVO.getLast_name(),"King");
    }
}
```

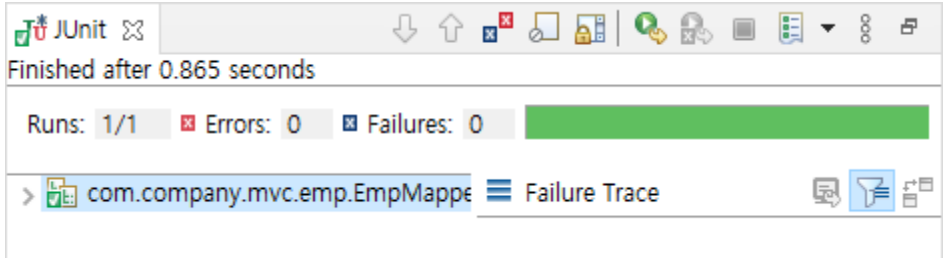
3.3 junit 테스트

- jUnit Test

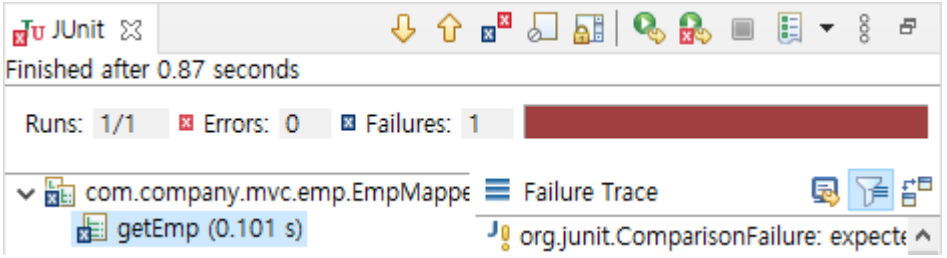


- 실행결과

테스트 성공



테스트 실패



3.4 jUnit 개요와 특징

- jUnit의 개요
  - Java에서 독립된 단위테스트(unit Test)를 지원해주는 프레임워크
  - 단위테스트란 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차, 즉 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차

- jUnit의 특징
  - TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Eric Gamma가 작성
  - 단정(assert) 메서드로 테스트 케이스의 수행결과를 판별한다.
    - 예) assertEquals(예상값, 실제값)
  - jUnit4부터는 테스트를 지원하는 어노테이션을 제공한다.
    - @Test, @Before, @After
  - 각 @Test 메서드가 호출될 때마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.
  - 결과는 성공(녹색), 실패(붉은색)중 하나로 표시

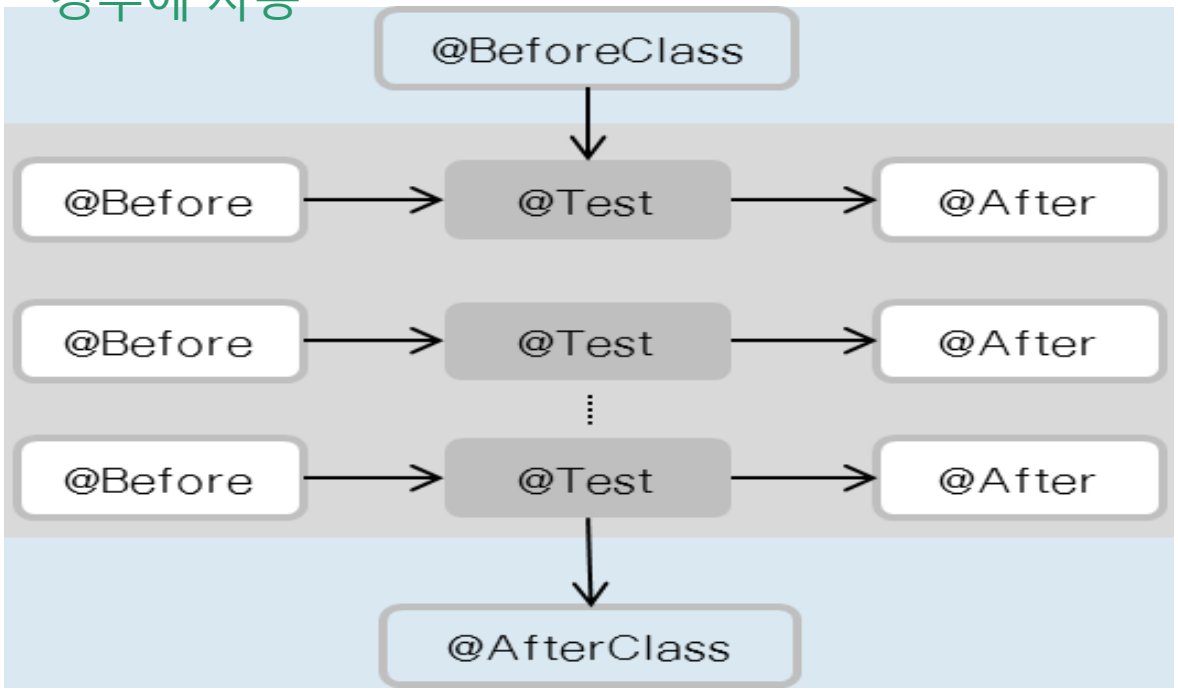
3.5 jUnit에서 테스트를 지원하는 어노테이션

- @Test
  - @Test가 선언된 메소드는 테스트를 수행하는 메소드가 된다.
  - Junit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test마다 객체를 생성한다.
- @Ignore
  - @Ignore가 선언된 메소드는 테스트를 실행하지 않게 한다.
- @Before
  - Before가 선언된 메소드는 @Test 메소드가 실행되지 전에 먼저 실행
  - @Test 메소드가 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용

3.5 jUnit에서 테스트를 지원하는 어노테이션

@Test 메소드보다 먼저 한번만 수행되어야 할 경우에 사용

@Test 메소드를 실행하기 전에 실행



@Test 메소드가 실행된 후 실행

@Test 메소드보다 나중에 한번만 수행되어야 할 경우에 사용

3.6 jUnit을 사용한 테스트

- 테스트 결과를 확인하는 단정(assert) 메서드
  - assertArrayEquals(a,b)
    - 배열 a와b가 일치함을 확인
  - assertEquals(a,b)
    - 객체 a와b의 값이 같은지 확인
  - assertSame(a,b)
    - 객체 a와b가 같은 객체임을 확인
  - assertTrue(a)
    - a가 참인지 확인
  - assertNotNull(a)
    - a객체가 null이 아님을 확인

http://junit.sourceforge.net/javadoc/org/junit/Assert.html

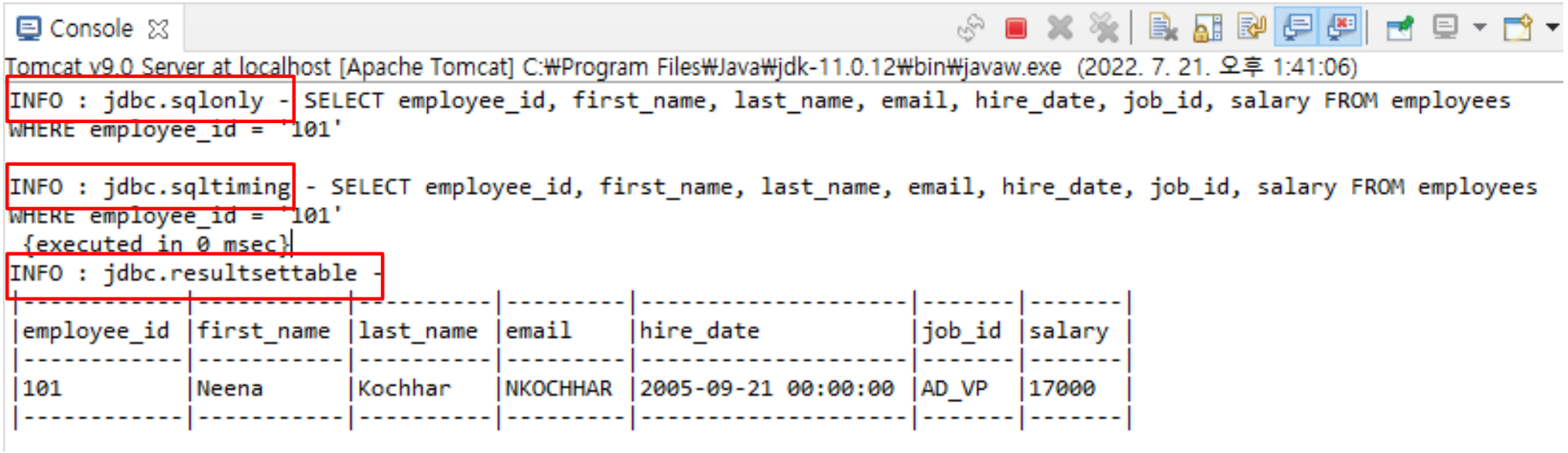
3. 6 Spring-test를 사용한 테스트

- Spring-test에서 테스트를 지원하는 어노테이션
  - @RunWith(SpringJUnit4ClassRunner.**class)**
    - @WunWith는 jUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션이다.
    - SpringJUint4ClassRunner라는 클래스를 지정해주면 ApplicationContext를 만들고 관리하는 작업을 진행해준다.
    - @RunWith 어노테이션은 각가의 테스트별로 객체가 생성되더라도 싱글톤(singleton)의 Application Context를 보장한다.
  - @ContextConfiguration
    - 스프링 빈(Bean) 설정 파일의 위치를 지정할 때 사용되는 어노테이션이다.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath:spring/*-context.xml")
public class BoardClient {
  @Autowired ApplicationContext context;
  @Test
  public void dataSourceTest() throws SQLException {
      DataSource ds = (DataSource) context.getBean("dataSource");
      System.out.println(ds.getConnection());
 }
```

4.1 sql 로그 보기

- PreparedStatement에서 파라미터가 대입된 쿼리 내용과 실행결과를 볼 수 있다.



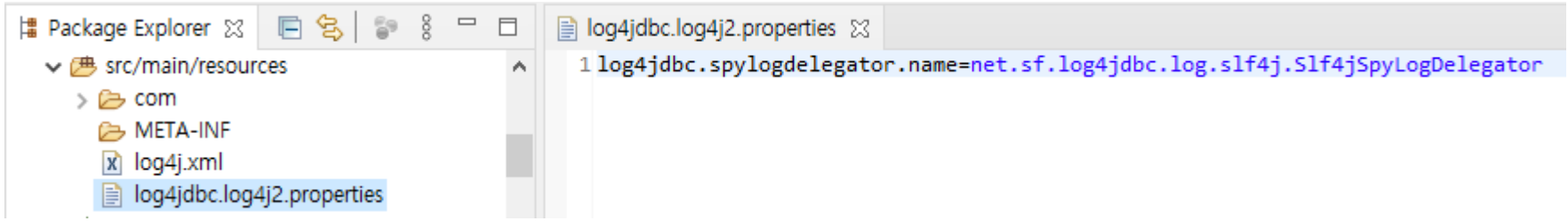
4.1 sql 로그 보기

- pom.xml
  - log4jdbc-log4j2 라이브러리 추가

```
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4.1</artifactId>
  <version>1.16</version>
</dependency>
```

- 로그 설정파일 추가
  - log4jdbc.log4j2.properties

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```



4.2 sql 로그 보기

- JDBC 드라이버와 URL 정부 수정
  - src/main/webapp/WEB-INF/spring/root-context.xml

```
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
<!-
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
-->
  <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy" />
  <property name="jdbcUrl"    value="jdbc:log4jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"         value="hr" />
  <property name="password"         value="hr" />
</bean>
```

4.2 sql 로그 보기

- src/main/resources/log4j.xml
  - 로그 레벨
    - trace < debug < info < warn < error < fetal
    - 지정된 레벨 이하는 출력 안됨
  - 루트 로그 레벨 설정
    - 패키지별 별도 지정이 없으면 루트 레벨을 적용함

```
<!-- Root Logger -->
<root>
  <priority value="info" />
  <appender-ref ref="console" />
</root>
```

- 패키지별 로그 레벨 설정

```
<logger name="jdbc.sqlonly">
  <level value="info" />
</logger>

<logger name="jdbc.sqltiming">
  <level value="info" />
</logger>

<logger name="jdbc.resultsettable">
  <level value="info" />
</logger>

<logger name="jdbc.audit">
  <level value="warn" />
</logger>

<logger name="jdbc.resultset">
  <level value="warn" />
</logger>
```

5.1 컨트롤러와 웹페이지 작성

- Controller

```
@Controller
public class EmpController {

  @Autowired EmpMapper empMapper;

  @GetMapping("/emp")
  public String emp(Model model, EmpVO empVO) {
     model.addAttribute("emp", empMapper.getEmp(empVO));
     return "emp";
  }
}
```

- 테스트
  - tomcat 서버 시작하고 브라우저에서 URL 입력

```
http://localhost/web/emp?employee_id=100
```

- 뷰페이지
  - src/main/webapp/WEB-INF/views/emp.jsp

```
<body>
<h3>사원조회</h3>
<div>사번: ${emp.employee_id}</div>
<div>이름: ${emp.first_name}</div>
<div>입사일자: ${emp.hire_date}</div>
<div>급여: ${emp.salary}</div>
</body>
```

6.1 Dynamic Web Project 를 Spring 프로젝트로 변경

- Maven 프로젝트로 변경
  - Configure 컨텍스트메뉴 -> convert to maven project
- Spring 프로젝트로 변경
  - Spring 컨텍스트메뉴 -> add Spring Project Nature
- Spring 라이브러리 설치
  - https://mvnrepository.com/ Spring context 검색
  - 5.3.16 버전 선택하여 pom.xml 에 복사
  - Maven Dependencies에서 jar 파일이 추가되었는지 확인
- Spring 설정파일 추가
  - File 메뉴 -> new -> Spring Bean Configuration File
  - XSD namespace 정의에서 context 선택