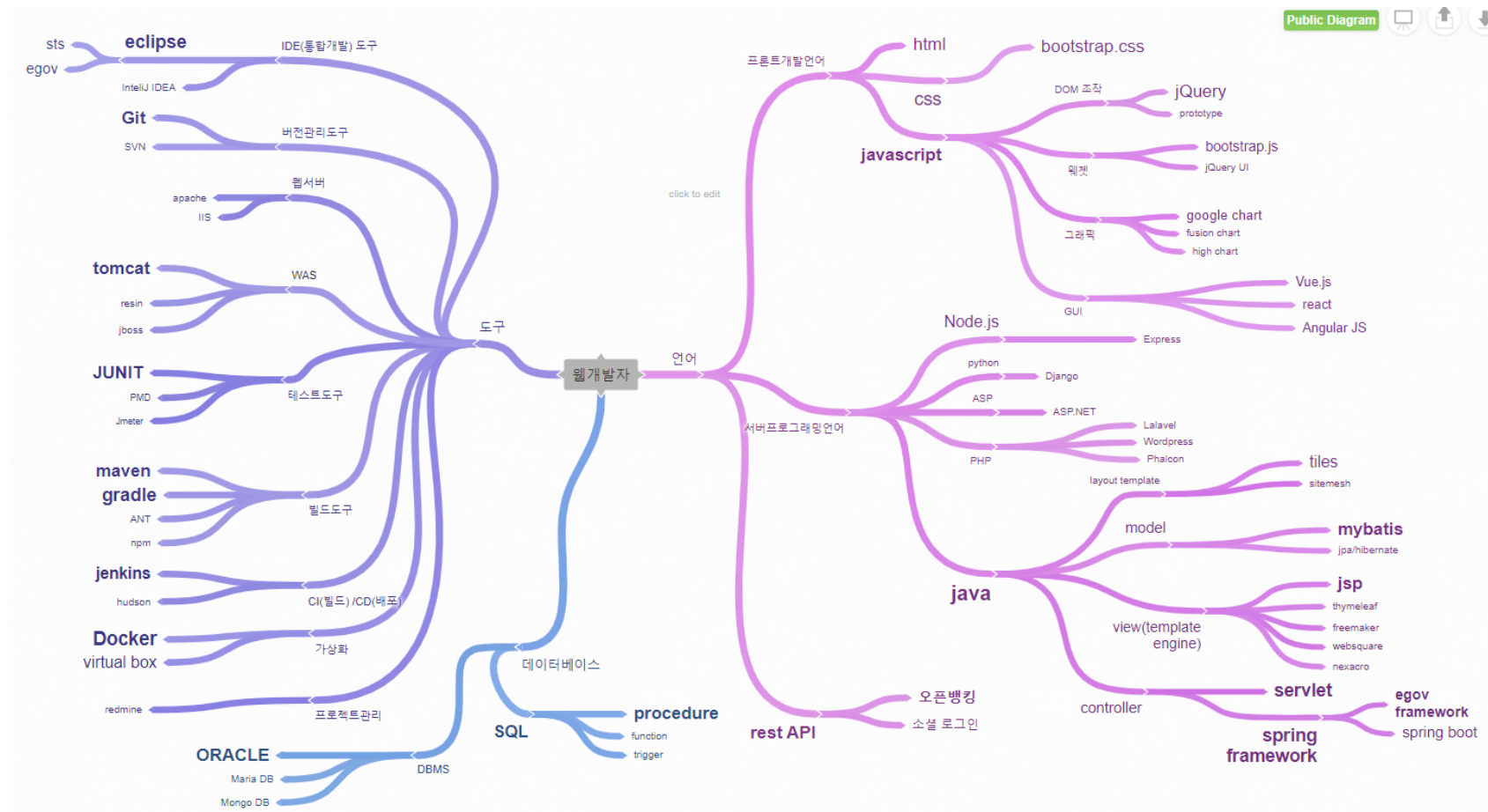


|. 스프링 프레임워크 개요

1. 스프링 프레임워크 소개
2. 개발환경 구축
3. 스프링 프로젝트

로드맵

■ <https://coggle.it/diagram/YCnltleANbfWWIbYG/t/웹개발자>



I.1 프레임워크의 장점

- 빠른구현시간
 - 비기능업무(성능, 보안, 확장성, 안정성등 공통 로직)는 프레임워크가 제공
 - 개발자들은 비즈니스 로직만 구현하면 됨
- 쉬운관리
 - 같은 프레임워크가 적용된 애플리케이션들은 아키텍처가 같으므로 관리가 수월.
 - 유지보수 인력과 시간을 줄일 수 있음
 - 프레임워크를 사용하면 숙련된 개발자와 초급 개발자가 생성한 코드가 비슷해짐.
- 검증된 아키텍처의 재사용과 일관성 유지

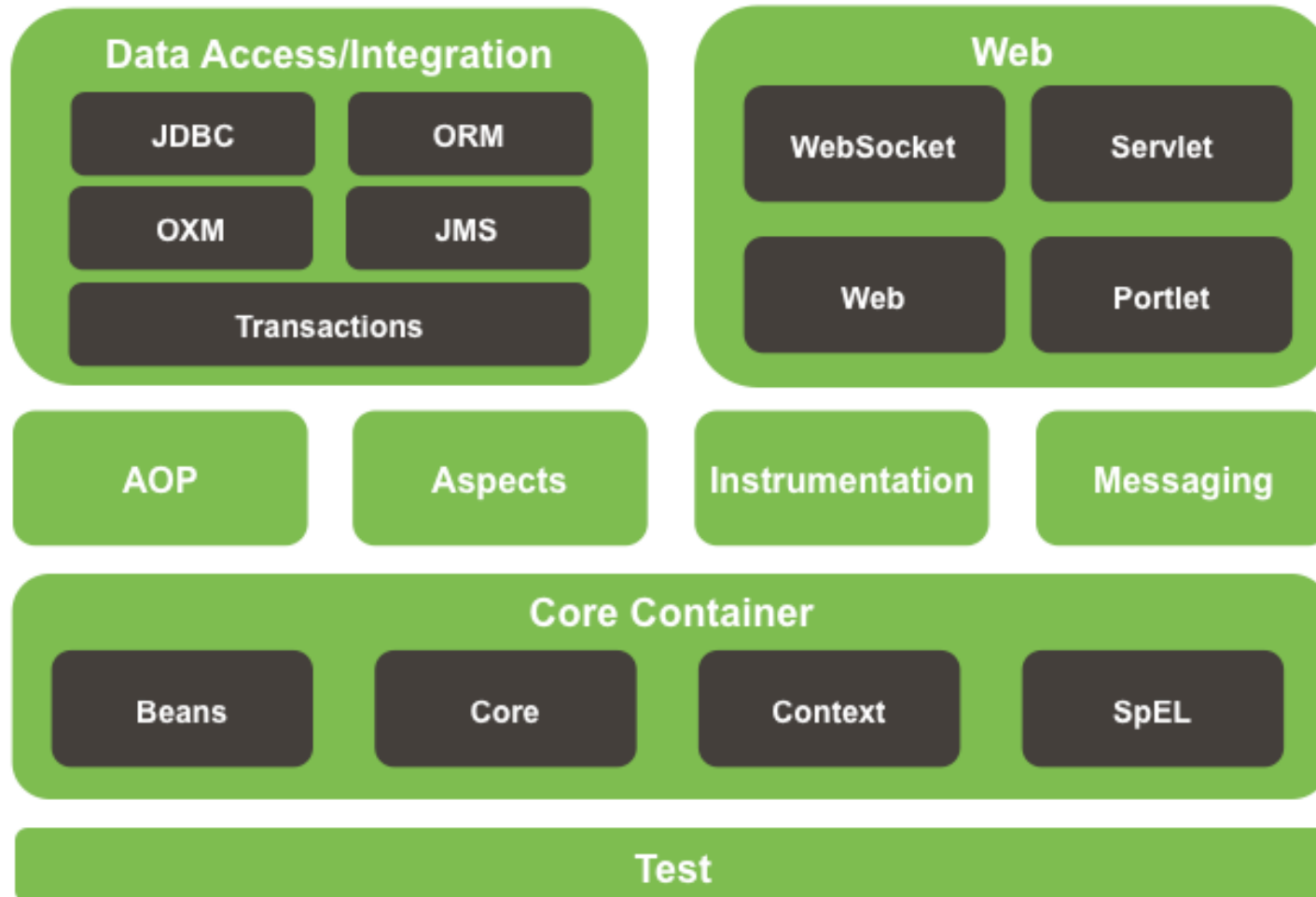
I.2 Spring 프레임워크 개요

- 오픈소스
 - Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크이다.
 - 스프링 프레임워크 개발자는 로드존슨 2002년 발표했으며 유지보수는 Pivotal (VMware 자회사)에서 함.
- 솔루션 제공
 - 디자인 패턴과 마찬가지로 반복적으로 발견되는 문제를 해결하기 위한 특화된 솔루션을 제공
- 테스트 환경
 - junit를 이용하여 쉬운 테스트 환경을 제공한다.
- 전자정부 표준프레임워크의 기반 기술
- POJO(Plain Old Java Object)
 - Spring 컨테이너에 저장되는 Java객체는 특정한 인터페이스를 구현하거나, 특정 클래스를 상속받지 않아도 된다.

I.3 Spring 프레임워크 특징

- 컨테이너
 - Java 객체의 LifeCycle을 관리하며, Spring 컨테이너로부터 필요할 객체를 가져와 사용할 수 있다.
- IoC
 - 제어의 역전. 인스턴스 생성부터 소멸까지의 인스턴스 생명주기 관리를 개발자가 아닌 컨테이너가 대신 해줌.
 - DI(Dependency Injection) 객체 간의 의존관계를 설정
- AOP(Aspect Oriented Programming)
 - Spring은 트랜잭션이나 로깅, 보안과 같이 공통적으로 필요로 하는 모듈들을 실제 핵심 모듈에서 분리해서 적용할 수 있다.

1.4 Spring 프레임워크 모듈



1.4 Spring 프레임워크 모듈

- 스프링 프레임워크는 주요기능으로 DI, AOP, MVC, JDBC 모듈 등을 제공한다.

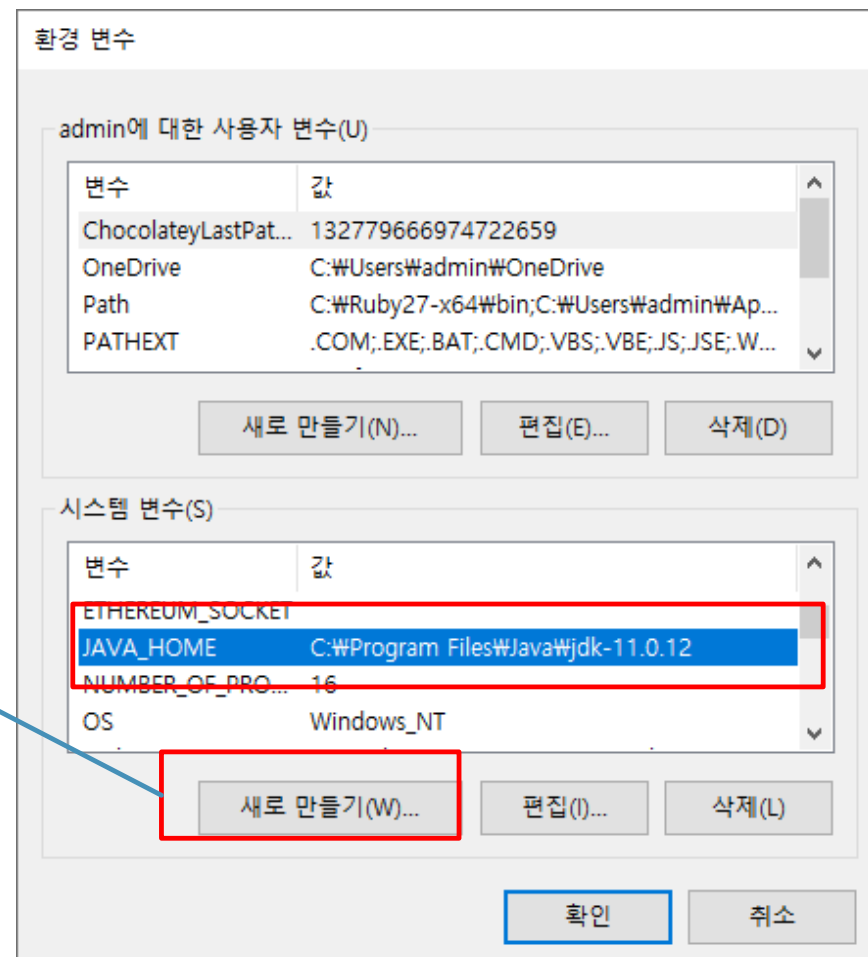
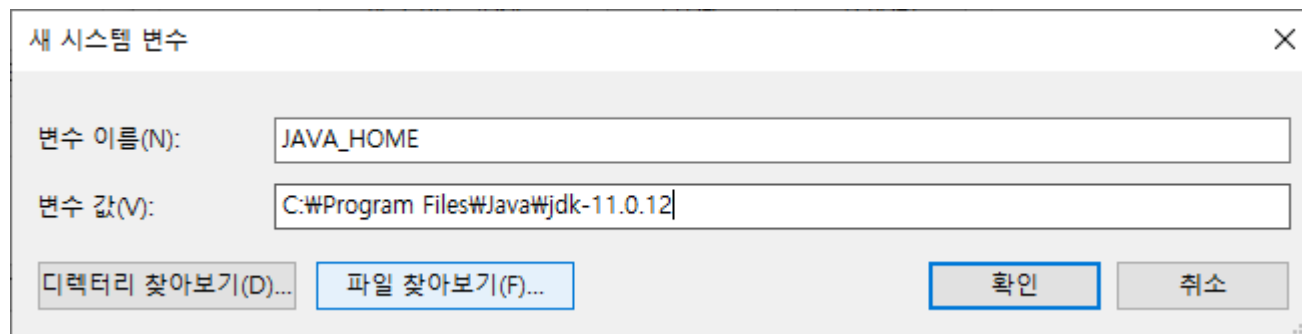
스프링 모듈	기능
spring-core	스프링의 핵심인 DI(Dependency Injection)와 IoC(Inversion of Control)를 제공
spring-aop	AOP구현 기능 제공
spring-jdbc	데이터베이스를 쉽게(적은 양의 코드) 다룰 수 있는 기능 제공
spring-tx	스프링에서 제공하는 트랜잭션 관련 기능 제공
spring-webmvc	스프링에서 제공하는 컨트롤러(Controller)와 뷰(View)를 이용한 스프링MVC 구현 기능 제공

2.1 프로그램 설치

- JDK 11
 - <http://www.oracle.com>
- Tomcat 9.0
 - <http://tomcat.apache.org>
- Eclipse 4.X
 - <https://www.eclipse.org/downloads/packages/>

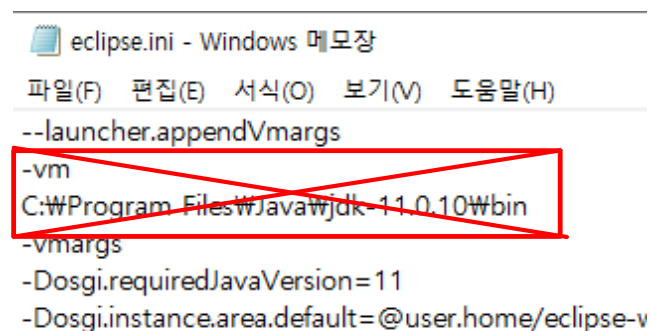
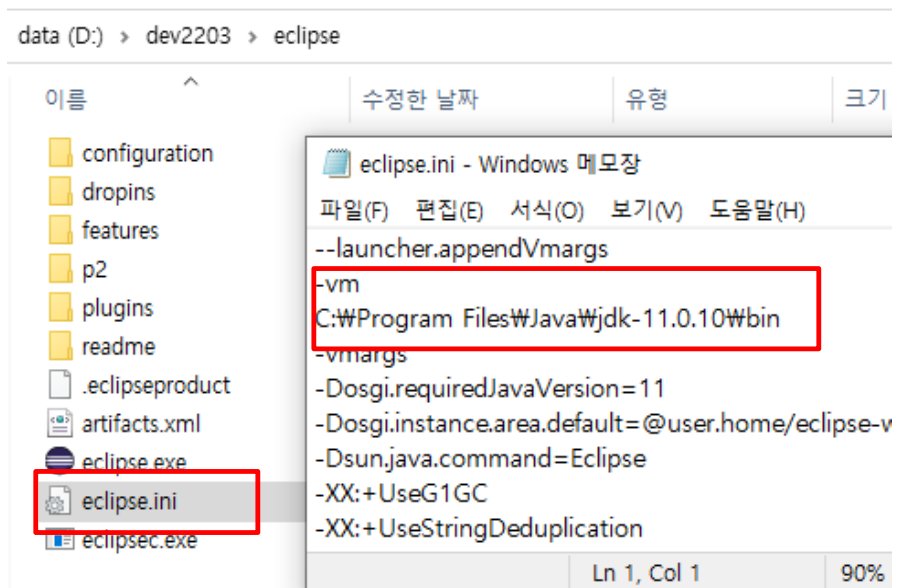
2.2 시스템 환경변수 추가

- 시스템 환경 변수 편집 실행
- [고급] 탭에서 환경 변수 클릭
- 새로 만들기 클릭
 - 변수이름: JAVA_HOME
 - 변수 값: C:\Program Files\Java\jdk-11.0.12



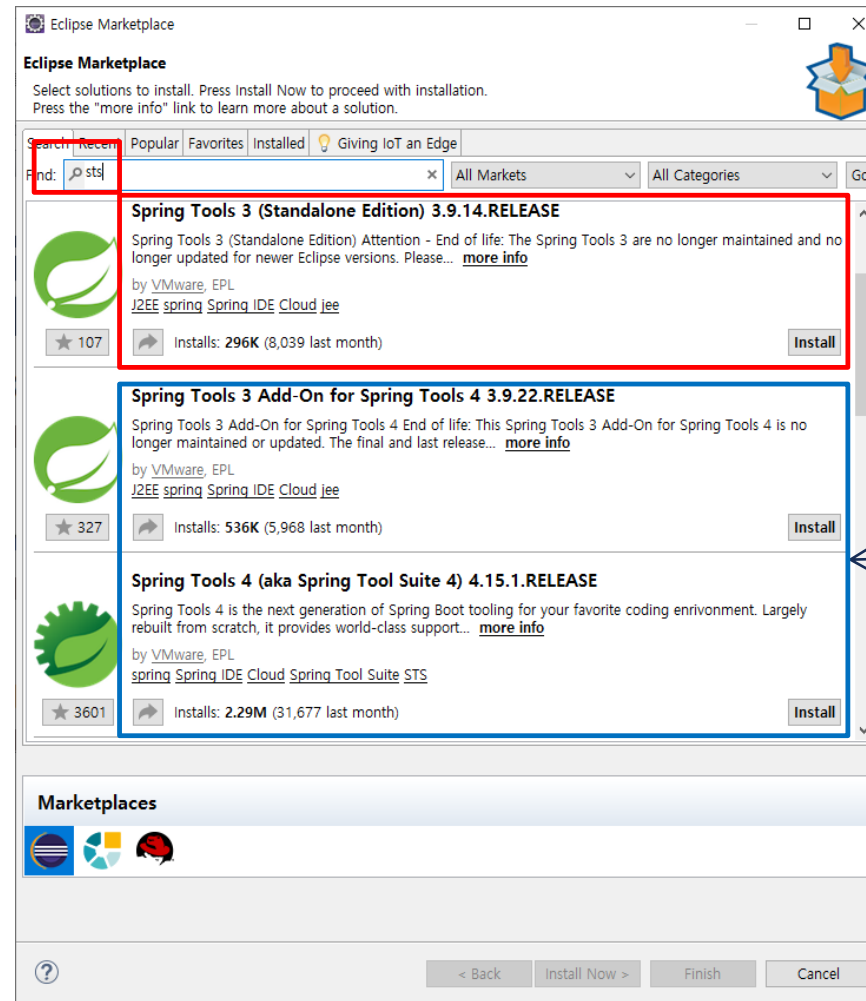
2.3 Eclipse java 실행경로 지정

- eclipse.ini
 - -vm 옵션 변경하거나 삭제



2.4 eclipse STS 플러그인 설치

- Eclipse Marketplace에서 sts 검색
- Spring Tools 3 설치
 - Spring Legacy : xml 기반, AOP 지원
 - eclipse용 스프링 도구

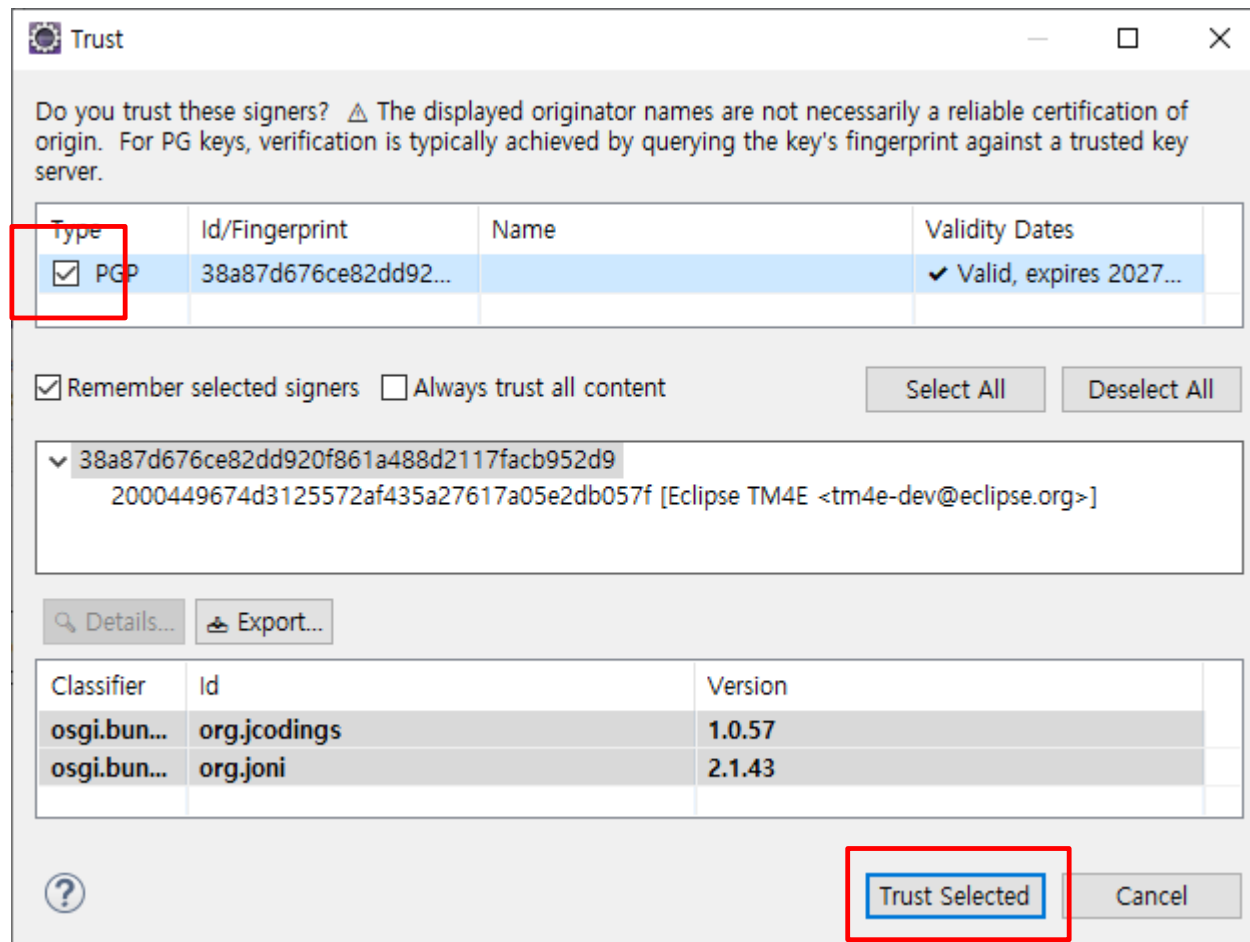


spring Tools 4는 Spring Boot 만 지원함

spring tools 3 Add-On을 설치해야 spring legacy를 사용할 수 있음

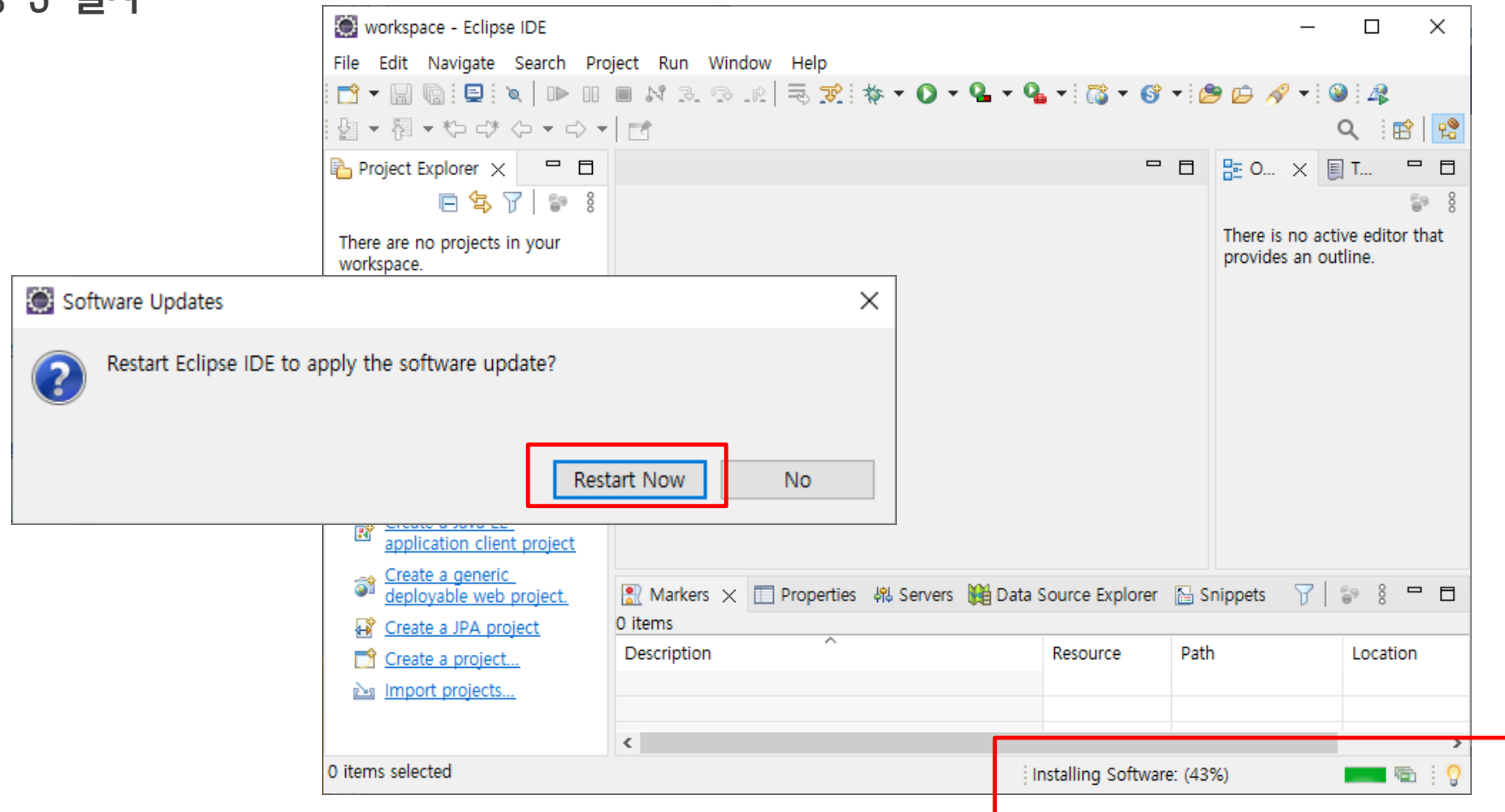
2.4 eclipse STS 플러그인 설치

■ Spring Tools 3 설치



2.4 eclipse STS 플러그인 설치

■ Spring Tools 3 설치

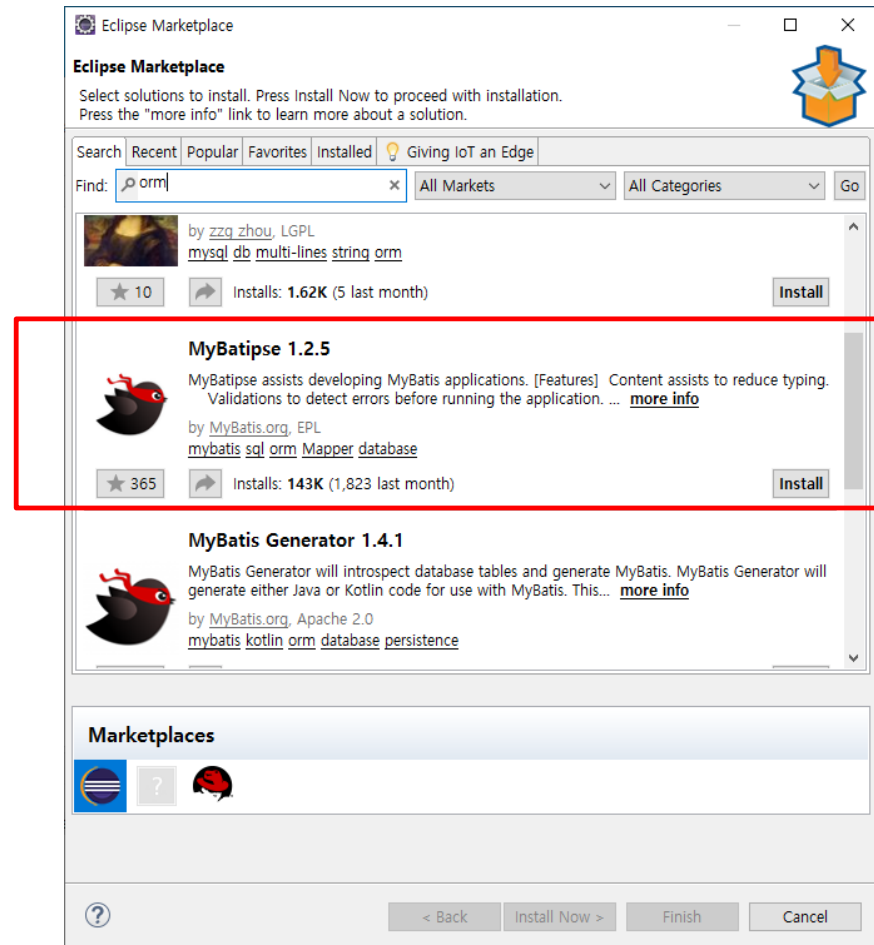


2.5 ORM 플러그인 설치

- Mybatis와 관련된 복잡한 XML 설정 파일들을 자동으로 만들고 관리

- 설치순서

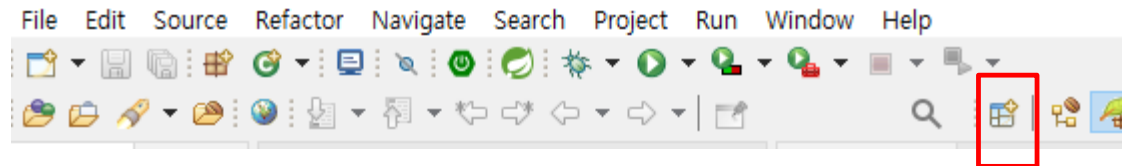
- MarketPlace에서 ORM 검색
- MyBatipse Install
- install anyway
- restart



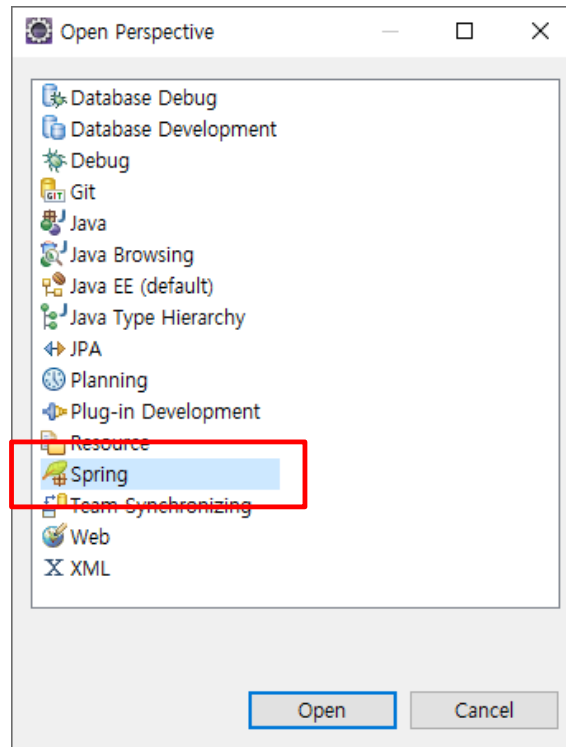
3.1 Spring MVC 프로젝트

- perspective를 spring으로 변경

- Open Persfective

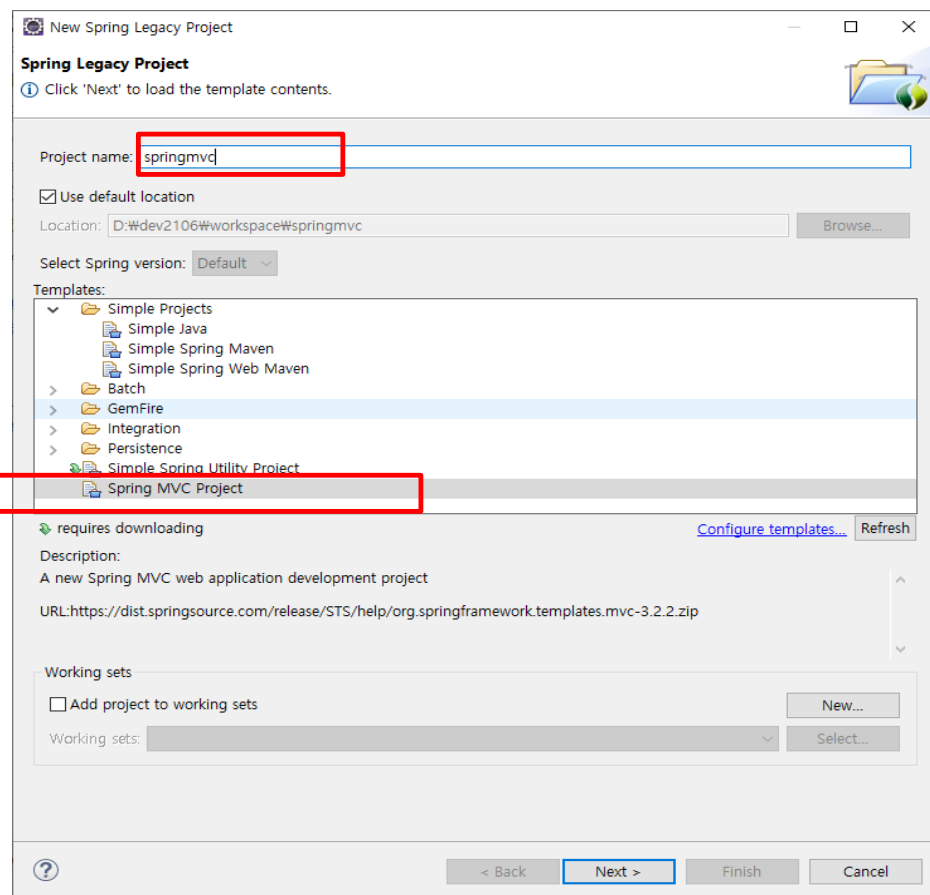


- spring 선택

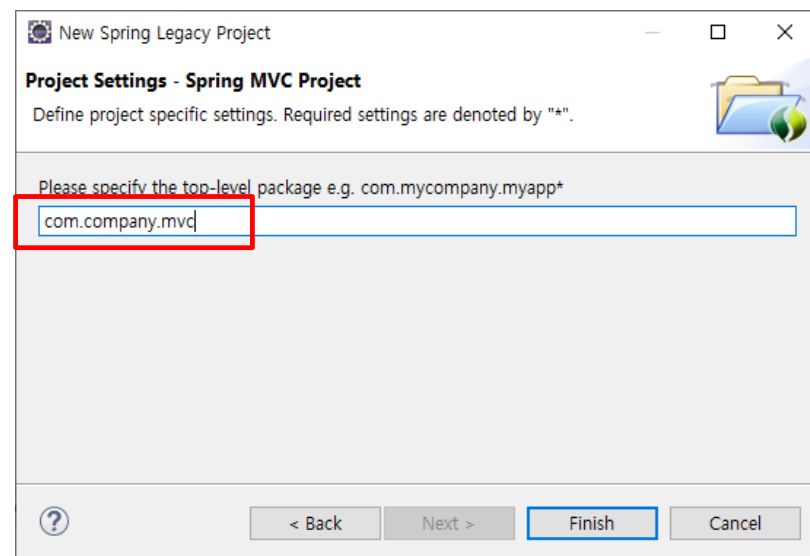


3.1 Spring MVC 프로젝트

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

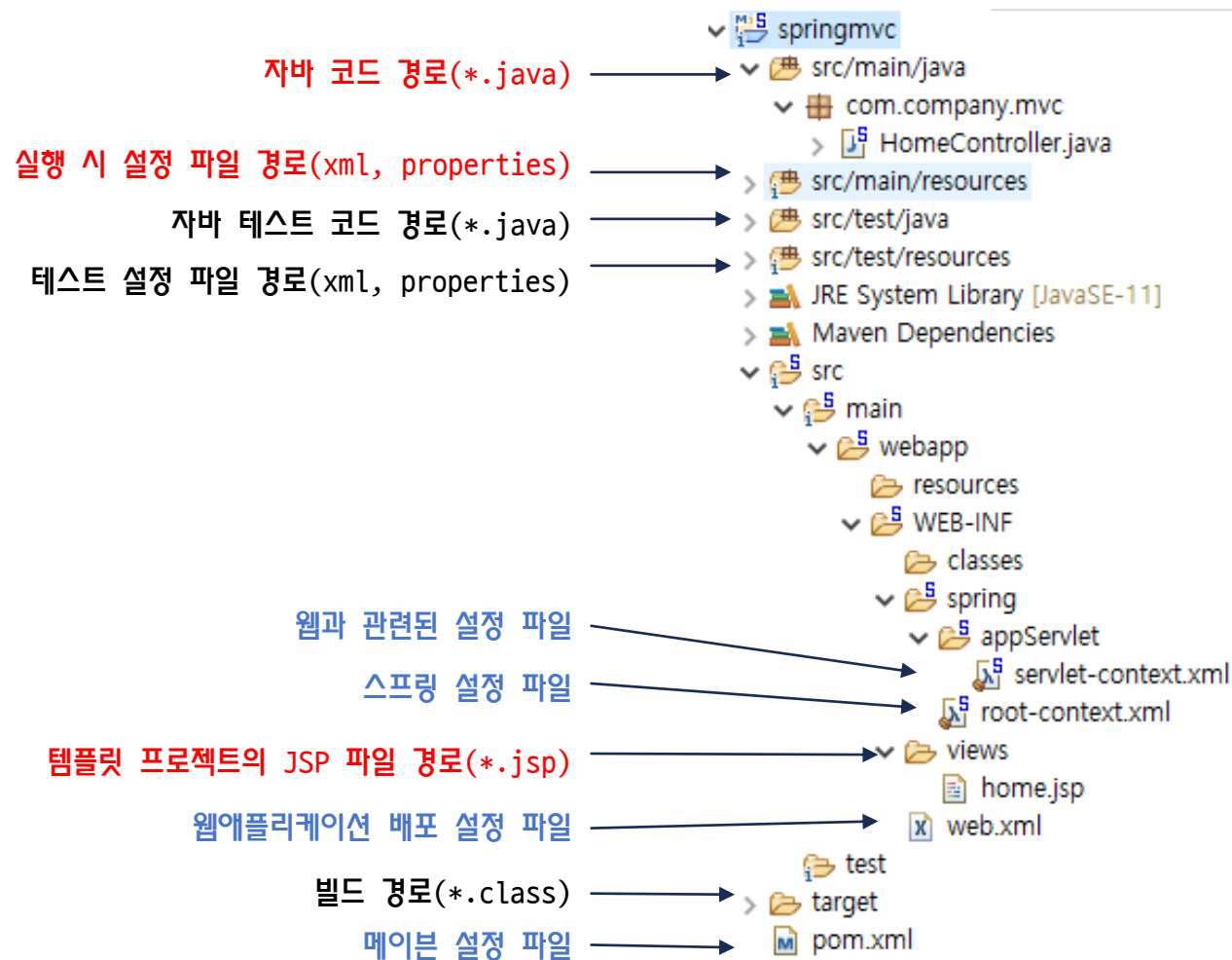


- 패키지명 입력



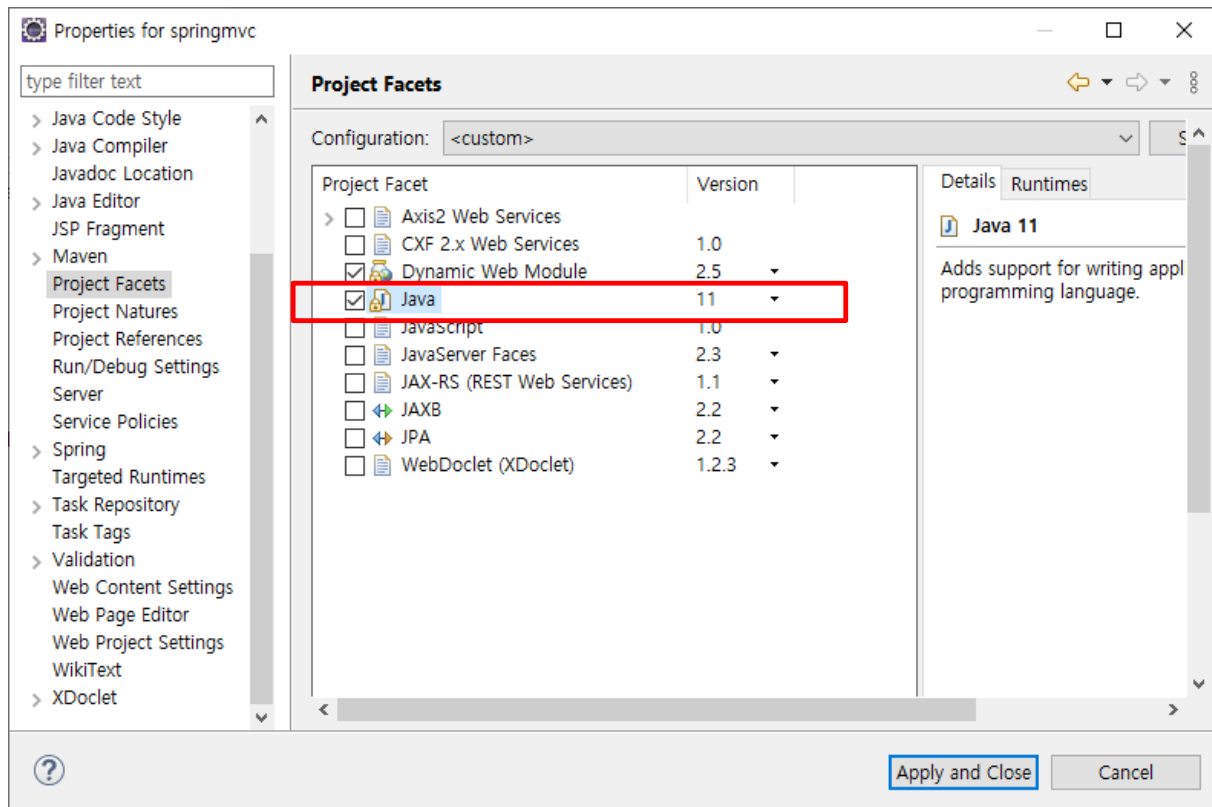
3.1 Spring MVC 프로젝트

■ 프로젝트 구조



3.1 Spring MVC 프로젝트

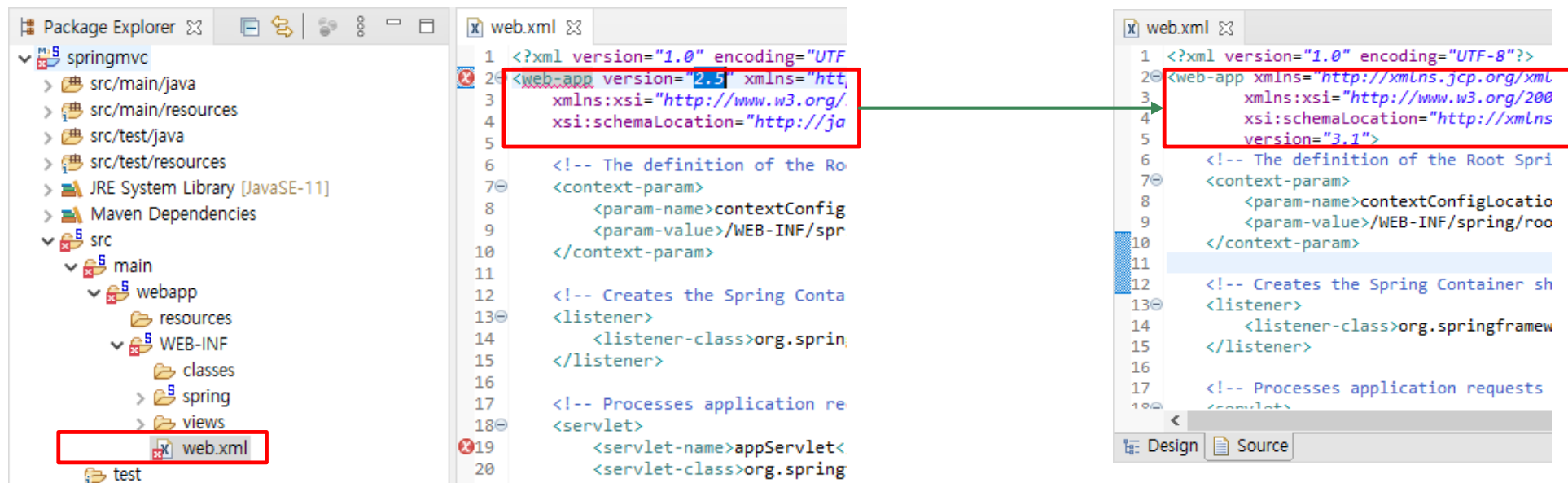
■ java version 변경



3.1 Spring MVC 프로젝트

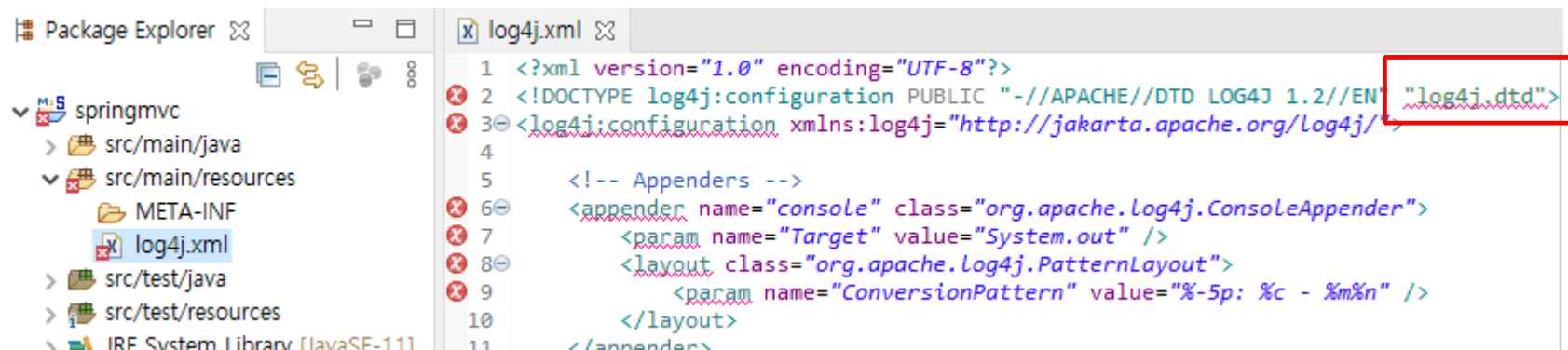
- Web Module 버전 변경
 - 프로젝트\src\main\webapp\WEB-INF\web.xml

```
<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">
```

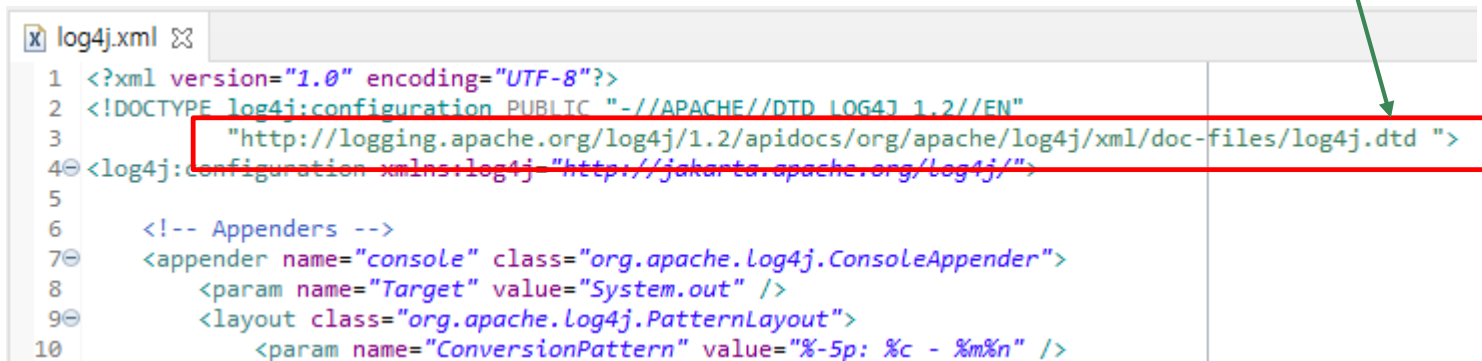


3.1 Spring MVC 프로젝트

■ log4j.xml dtd 경로 수정



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



3.1 Spring MVC 프로젝트

- pom.xml 변경
 - version 변경
 - java version 1.8 -> 11
 - org.springframework-version 3.1.1.RELEASE -> 5.3.16
 - log4j version 1.2.15 -> 1.2.17
 - junit version 4.7 -> 4.12
 - <dependency> 추가
 - spring-test
 - Lombok
 - Jackson

```
<!-- 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>
```

3.2 Mybatis 연동

- pom.xml 에 <dependency> 추가
 - HikariCP
 - spring-jdbc
 - ojdbc8
 - mybatis
 - mybatis-spring

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

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

<!-- 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>
```

3.2 Mybatis 연동

- 마이바티스 설정 : root-context.xml 파일위치:src\main\webapp\WEB-INF\spring

```
<?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>
```


3.2 Mybatis 연동

■ 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

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

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

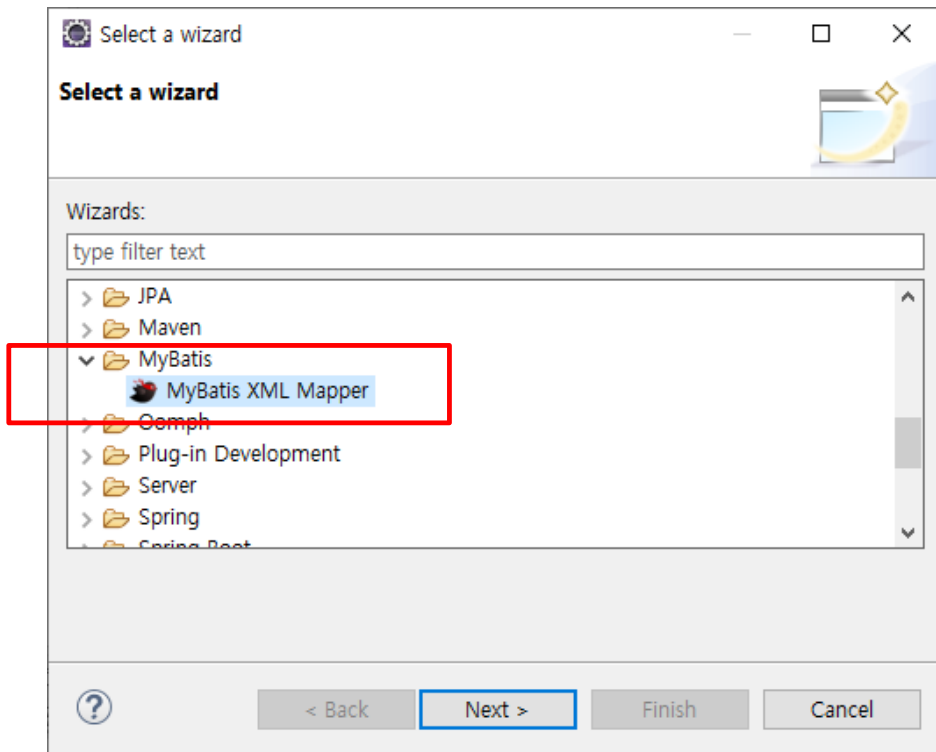
■ sql statment xml 파일

```
<?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">

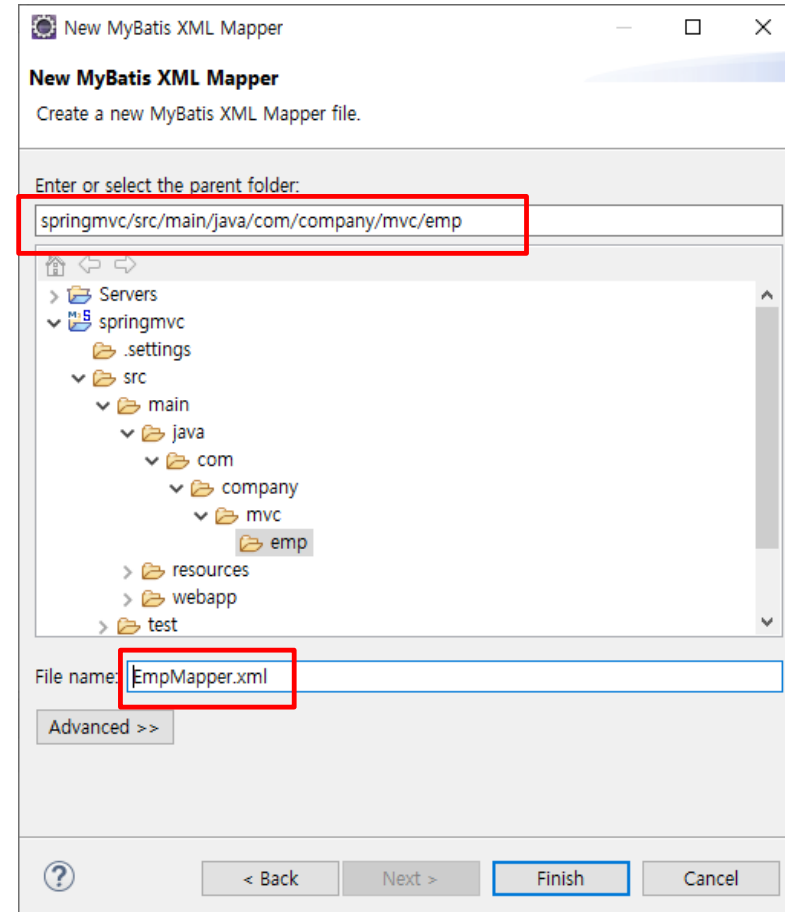
    <select id="getEmp"
        parameterType="com.company.mvc.emp.EmpVO"
        resultType="com.company.mvc.emp.EmpVO">
SELECT    employee_id,
          first_name,
          last_name,
          email,
          hire_date,
          job_id,
          salary
FROM      employees
WHERE     employee_id =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

3.2 Mybatis 연동

- Sql statement xml 파일 생성
 - File -> New -> Other...

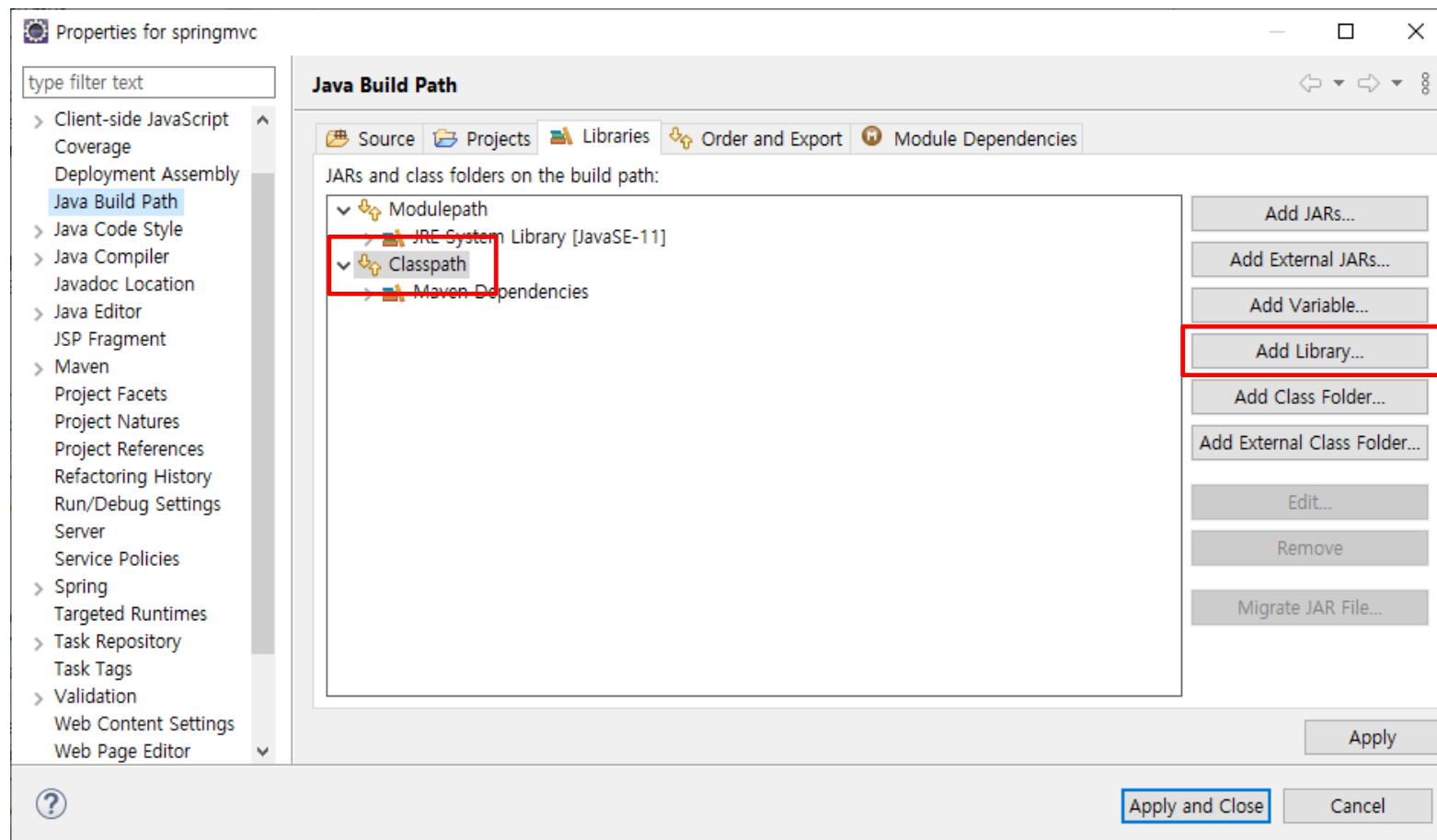


- 생성위치와 파일명 입력



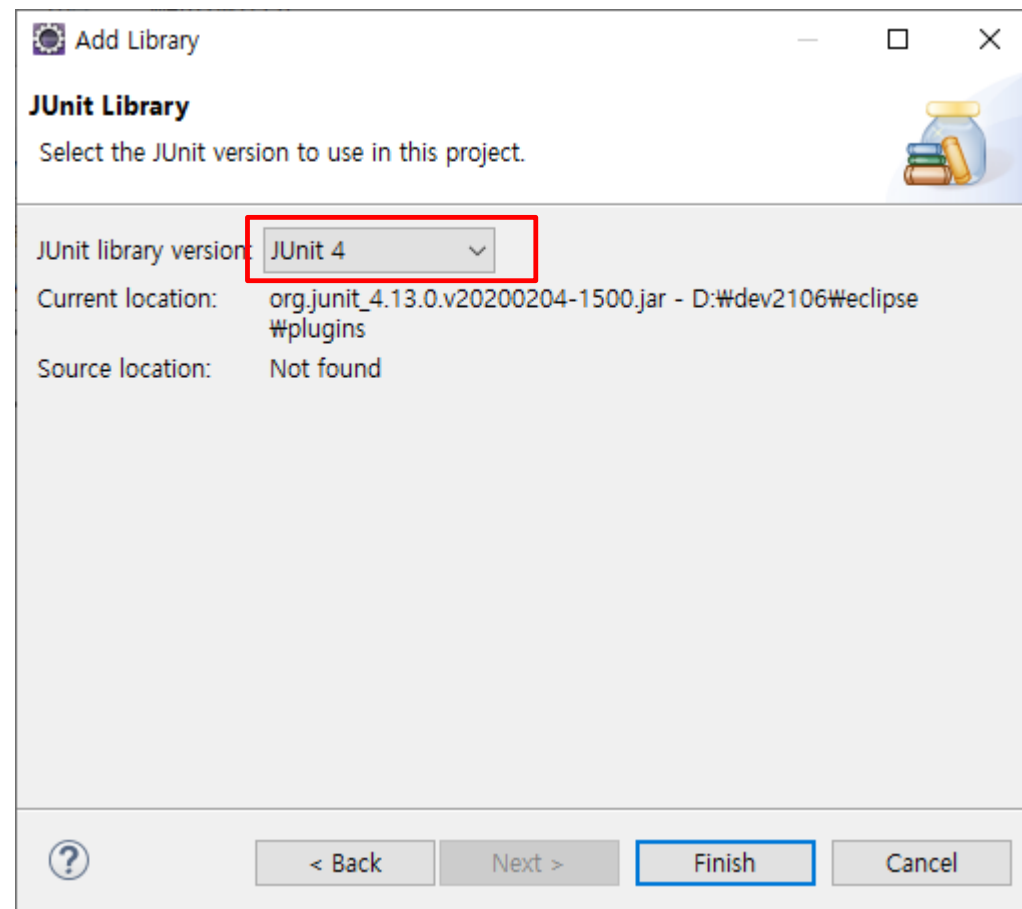
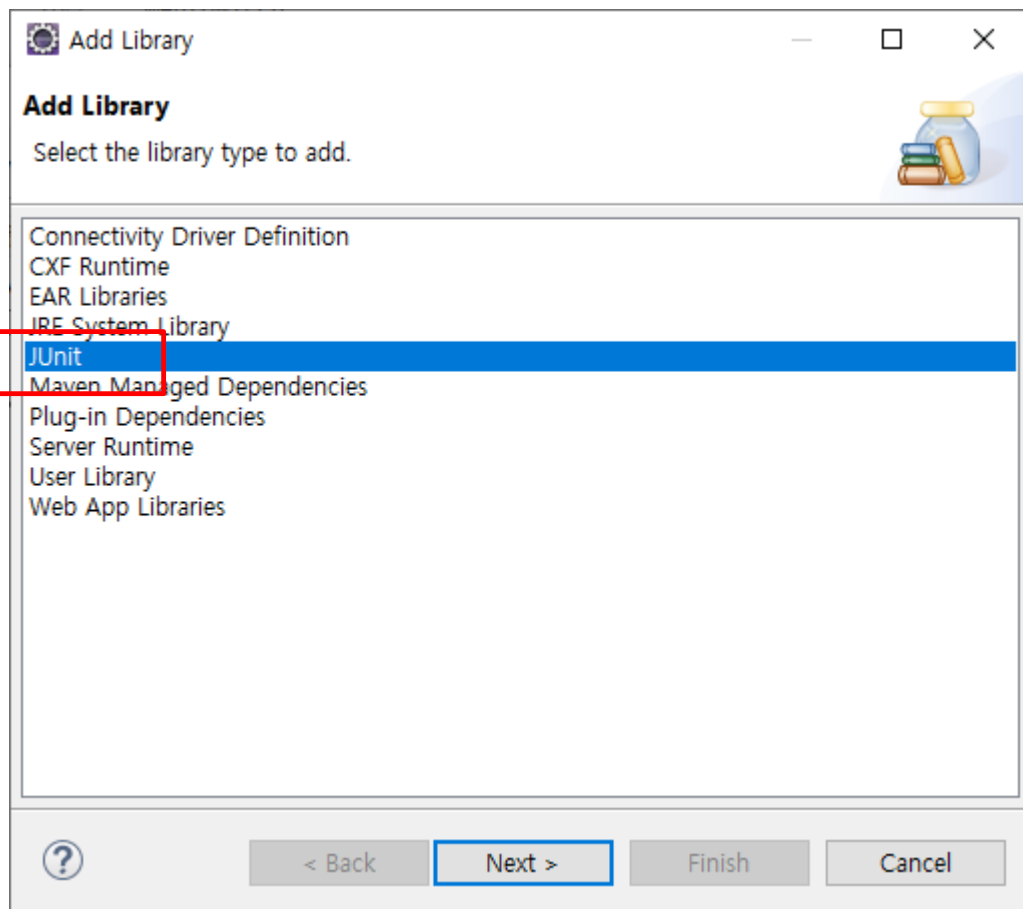
3.3 junit 테스트

■ junit 라이브러리 추가



3.3 junit 테스트

■ junit 라이브러리 추가



3.3 junit 테스트

■ 테스트 코드

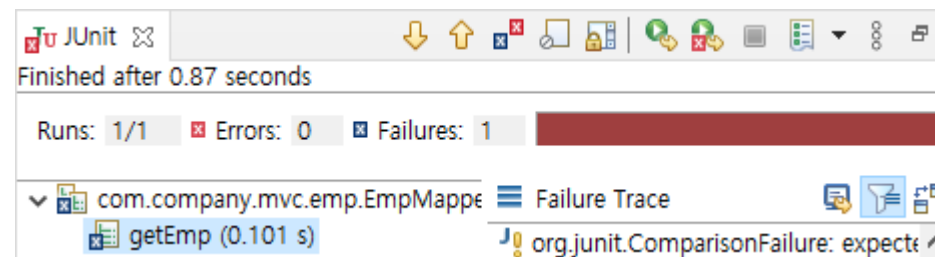
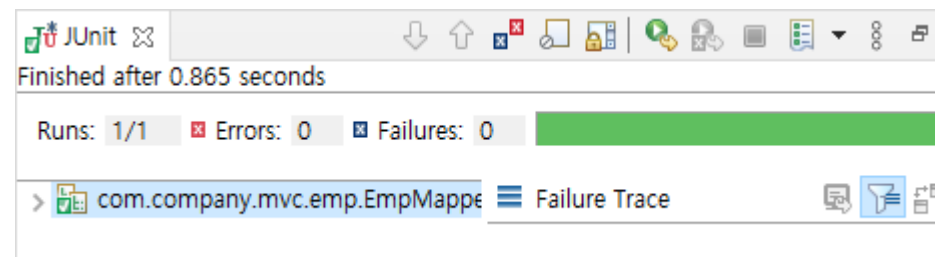
```
package com.company.mvc.emp;

import static org.junit.Assert.assertEquals;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

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

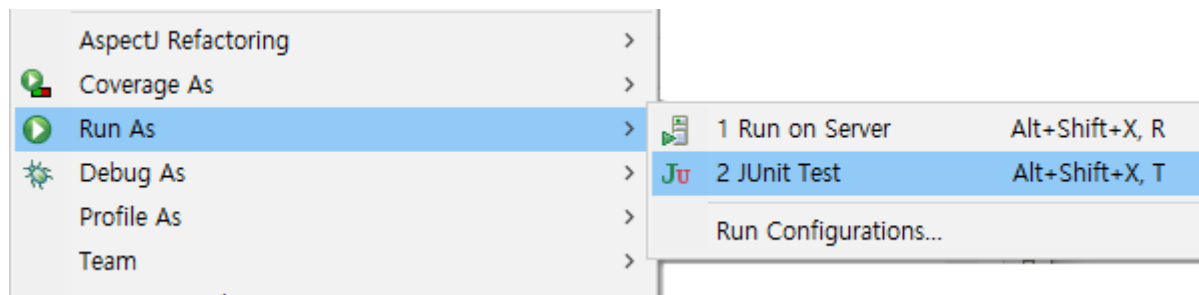
    @Autowired
    EmpMapper empMapper;

    @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");
    }
}
```

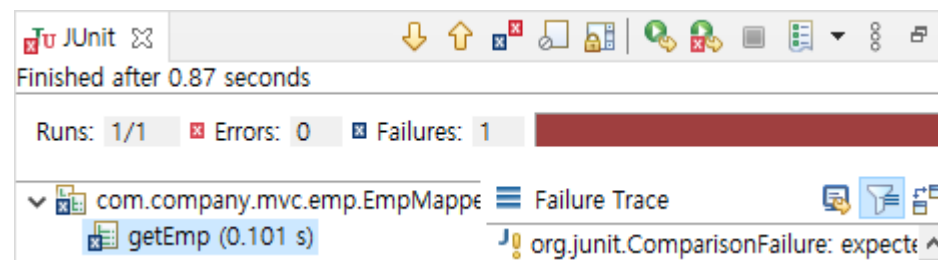
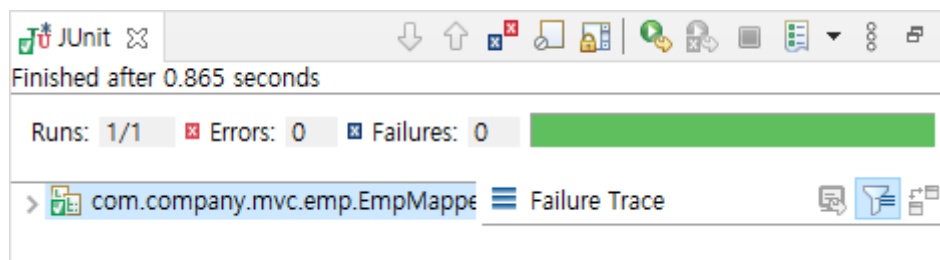


3.3 junit 테스트

■ JUnit Test

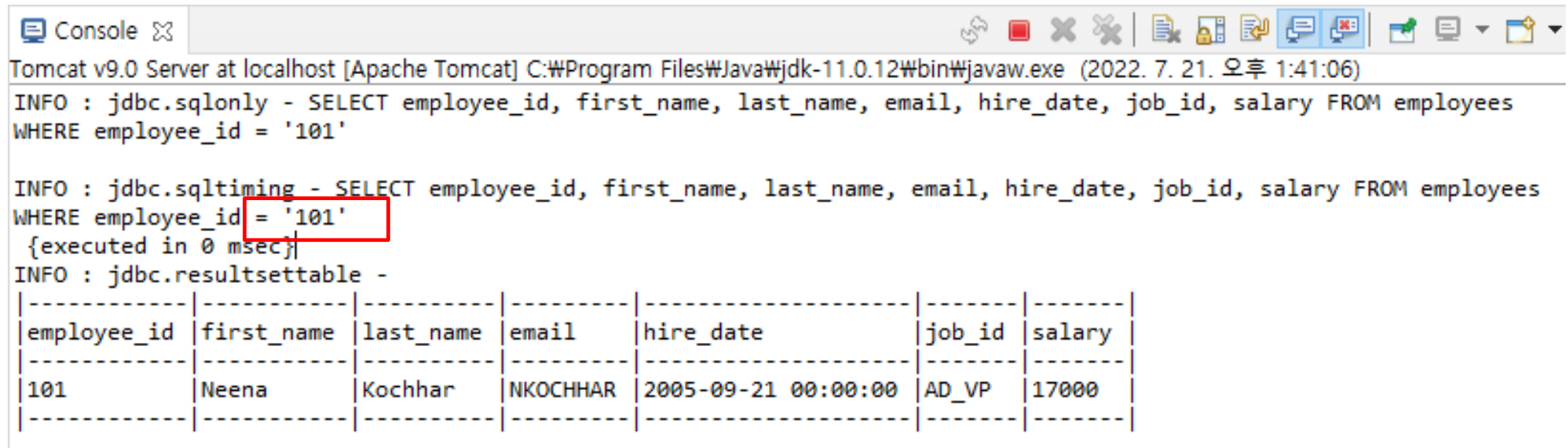


■ 실행결과



3.4 sql 로그 보기

- PreparedStatement에서 파라미터가 대입된 쿼리 내용과 실행결과를 볼 수 있다.



```

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (2022. 7. 21. 오후 1:41:06)
INFO : jdbc.sqlonly - SELECT employee_id, first_name, last_name, email, hire_date, job_id, salary FROM employees
WHERE employee_id = '101'

INFO : jdbc.sqltiming - SELECT employee_id, first_name, last_name, email, hire_date, job_id, salary FROM employees
WHERE employee_id = '101'
{executed in 0 msec}
INFO : jdbc.resultsettable -

```

employee_id	first_name	last_name	email	hire_date	job_id	salary
101	Neena	Kochhar	NKOCHHAR	2005-09-21 00:00:00	AD_VP	17000

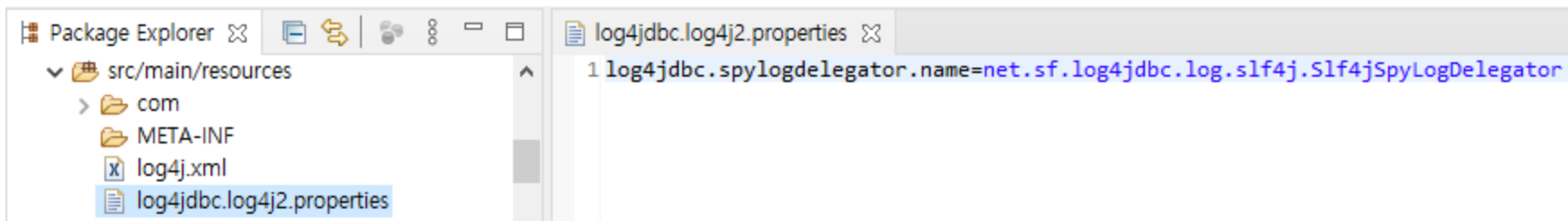
3.4 sql 로그 보기

■ 라이브러리 추가

■ log4jdbc-log4j2

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

■ 로그 설정파일 추가



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


3.4 sql 로그 보기

- JDBC 드라이버와 URL 정보 수정

- 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>
```

3.4 sql 로그 보기

■ 로그 레벨 설정

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

■ 로그가 출력안되게 하려면

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

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

3.5 컨트롤러와 웹페이지 작성

■ Controller

```
@Controller
public class EmpController {

    @Autowired EmpMapper empMapper;

    @RequestMapping(value = "/emp", method = RequestMethod.GET)
    public String emp(Locale locale, Model model, EmpVO empVO) {
        model.addAttribute("emp", empMapper.getEmp(empVO));
        return "emp";
    }
}
```

■ 테스트

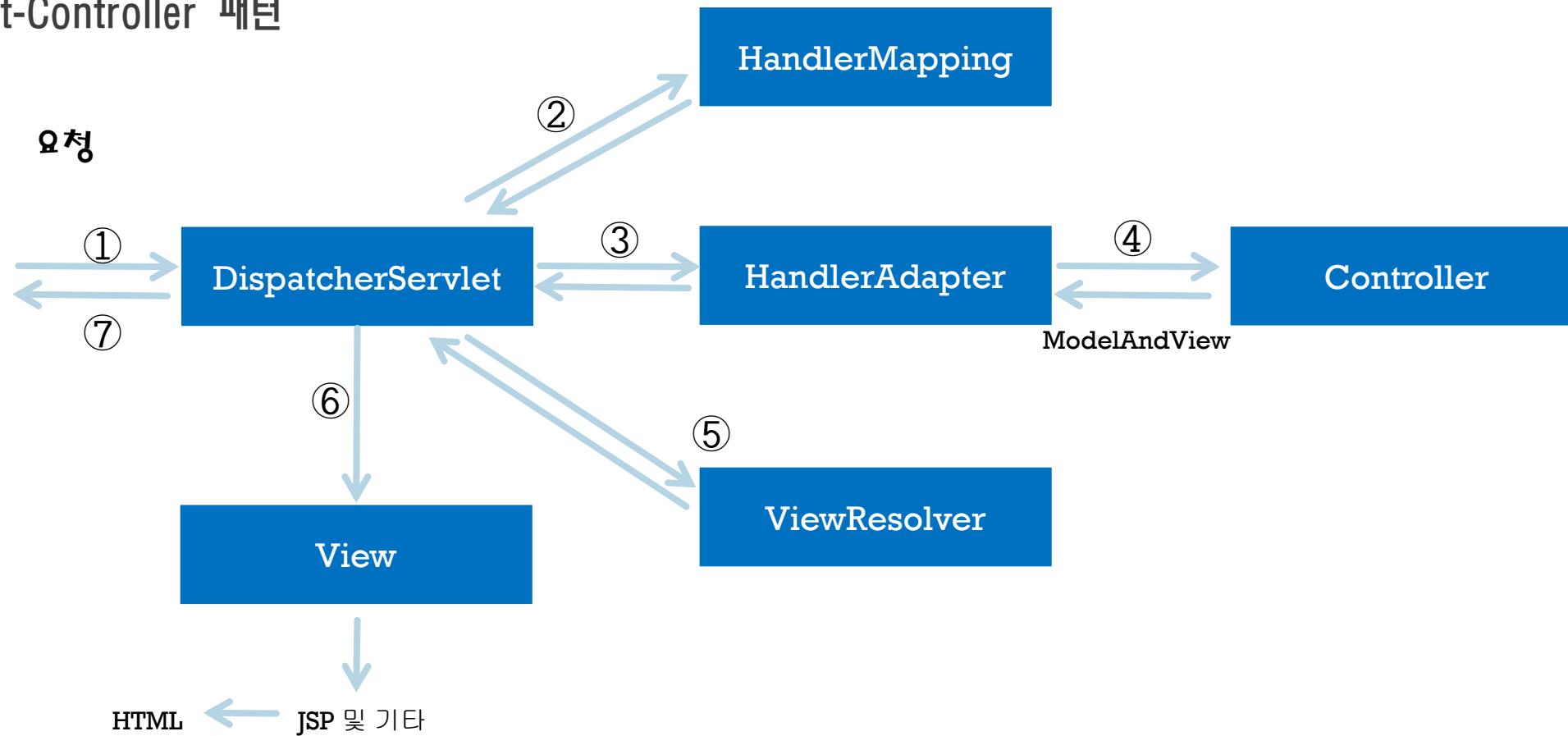
- tomcat 서버 시작
- 브라우저에서 `http://localhost/web/emp?employee_id=100`

■ jsp : 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>
```

3.6 Spring MVC 구조

■ Front-Controller 패턴



3.6 Spring MVC 구조

1. 사용자의 모든 요청은 Front-Controller인 DispatcherServlet을 통해서 처리
2. HandlerMapping은 Request의 처리를 담당하는 컨트롤러를 찾기 위해서 존재.
 - @RequestMapping 어노테이션 참조
3. 컨트롤러를 찾았다면 HandlerAdapter를 이용해서 컨트롤러를 동작
4. Controller는 실제 요청을 처리하는 로직을 작성.
 - 이때 view에 전달할 데이터는 Model 객체에 담아서 전달
5. ViewResolver를 컨트롤러가 반환한 결과를 어떤 View를 통해서 처리할지 해석
6. View는 실제로 응답 보내야 하는 데이터를 jsp등을 이용해서 생성하는 역할을 함
 - 만들어진 응답은 DispatcherServlet을 통해서 전송

3.7 Maven

- 빌드 도구
- 의존모듈(jar) 관리
 - 라이브러리 다운로드 자동화. 필요한(의존성 있는) 라이브러리를 하나씩 다운로드 받을 필요가 없다.
 - 중앙 repository 서버에서 필요한 jar파일을 다운받아 의존 모듈을 관리한다.
- 다운로드 받아서 로컬에 저장
 - C:\Users\user\.m2\repository
- 참고사이트
 - <https://javacan.tistory.com/entry/MavenBasic>

3.7 configuration metadata

- Annotation-based configuration
 - spring 2.5부터
- XML-based configuration
 - 루트 엘리먼트 <beans/> 안에 <bean/> 엘리먼트를 이용하여 설정
- Java-based configuration
 - spring 3.0부터
 - @Configuration
 - uses @Bean-annotated methods within a @Configuration class.