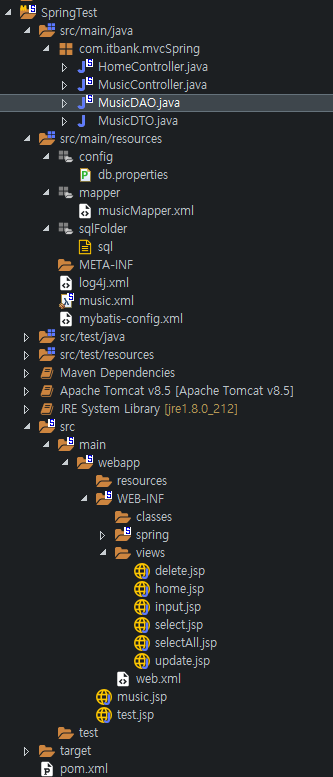
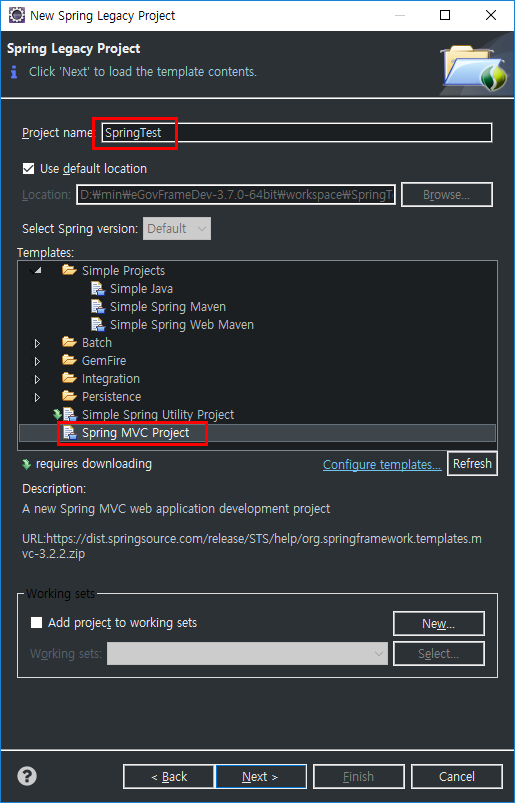
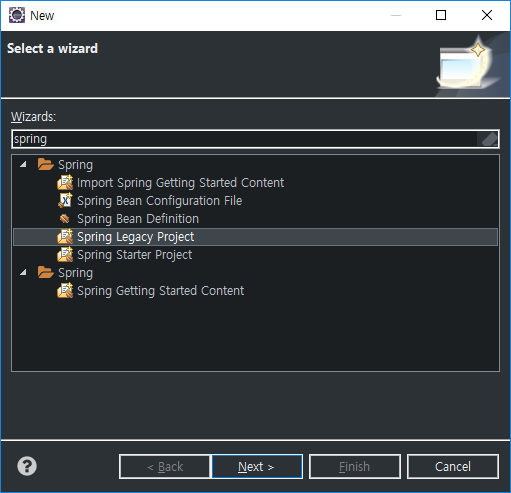
민병호

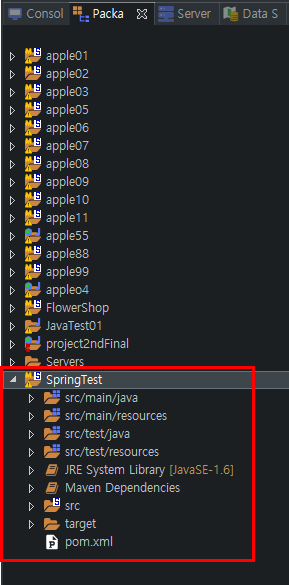
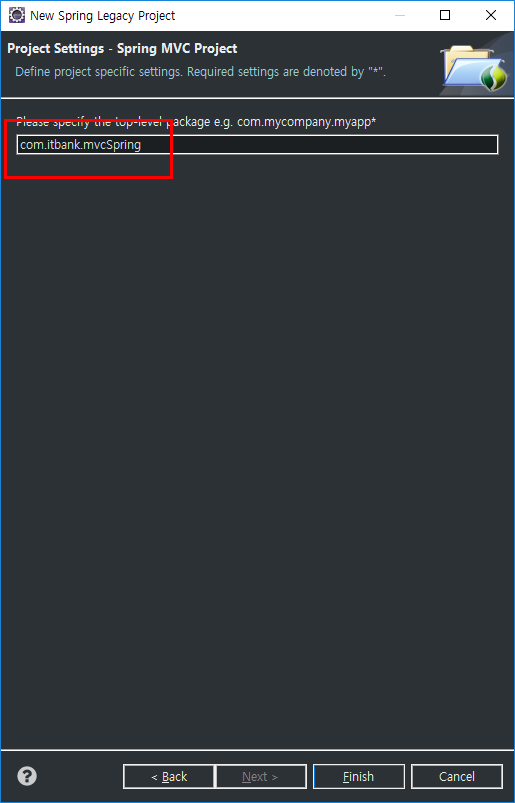
-------- 프로젝트 트리



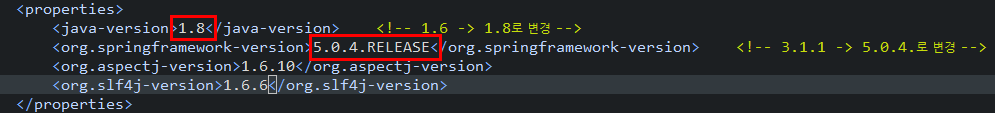
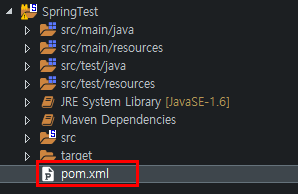
-------- 개발환경 구축하기

MVC프로젝트 생성

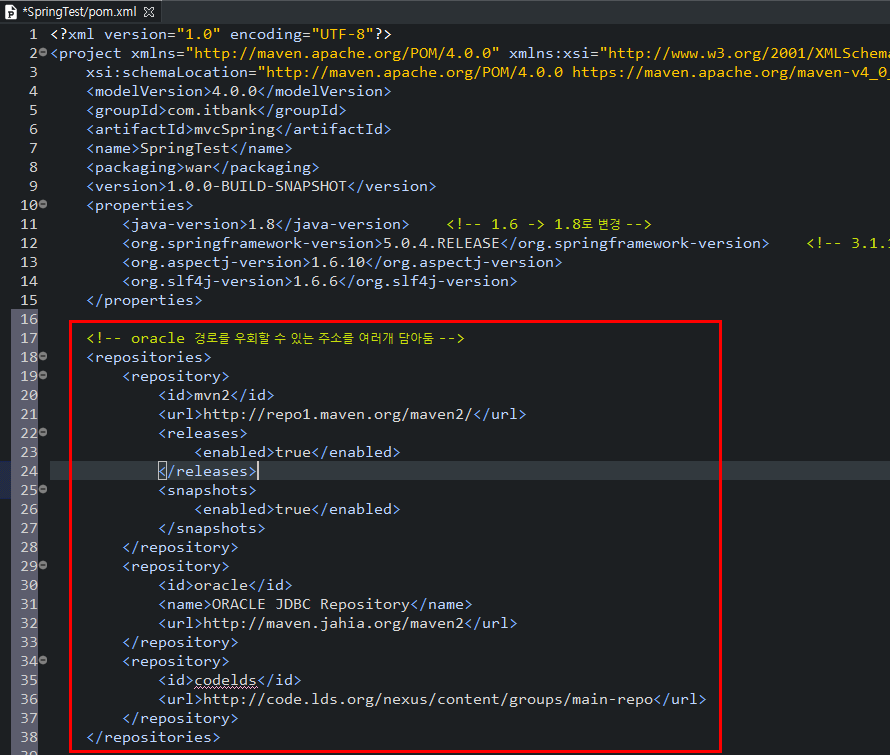


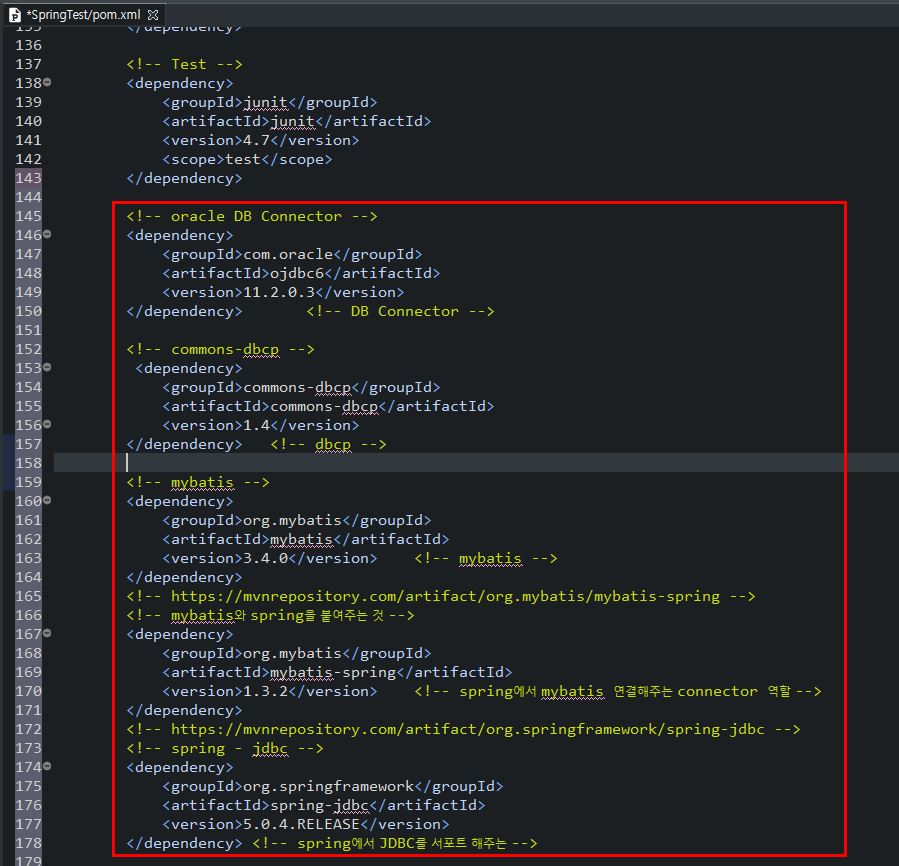


POM.XML 셋팅

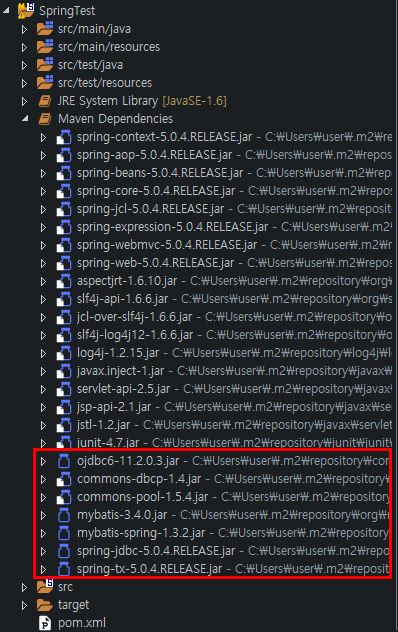
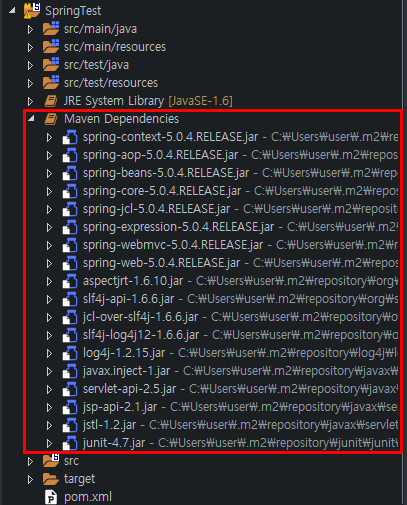


POM.XML에 repositories, dependency 설정

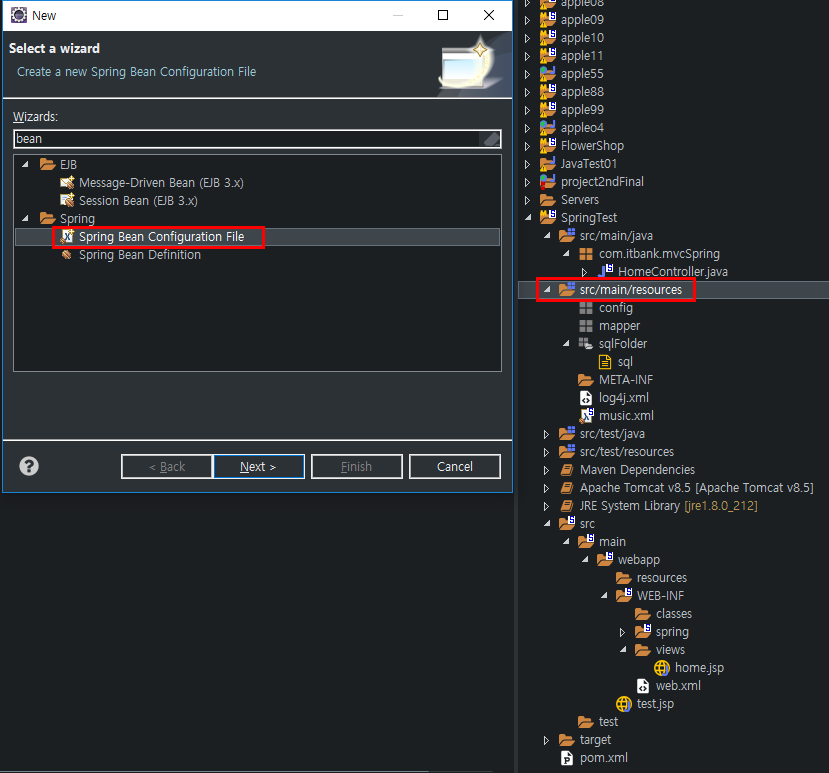


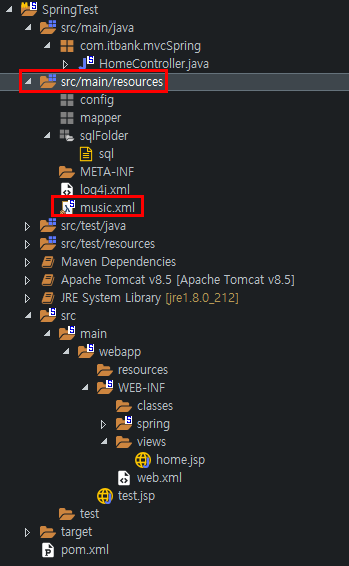
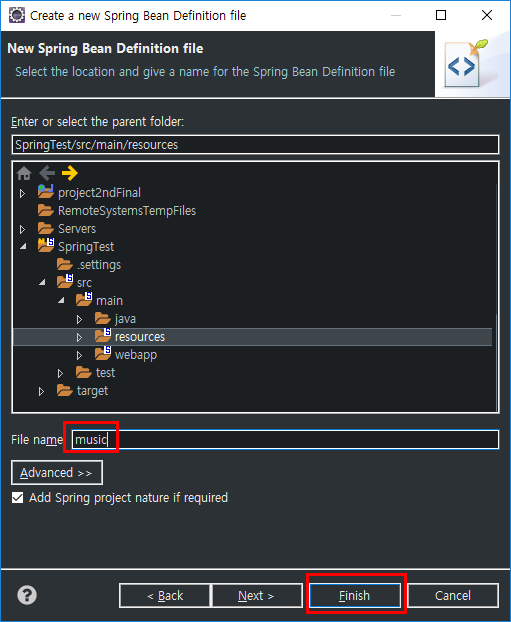


POM.XML 셋팅 전 / 후 Maven라이브러리



Oracle커넥터 설정

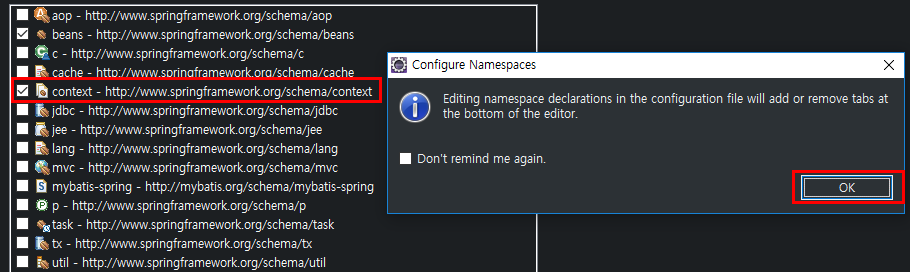


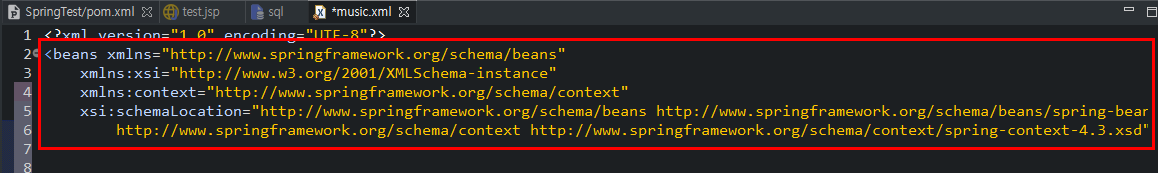


music.xml( context 파일 ) 설정

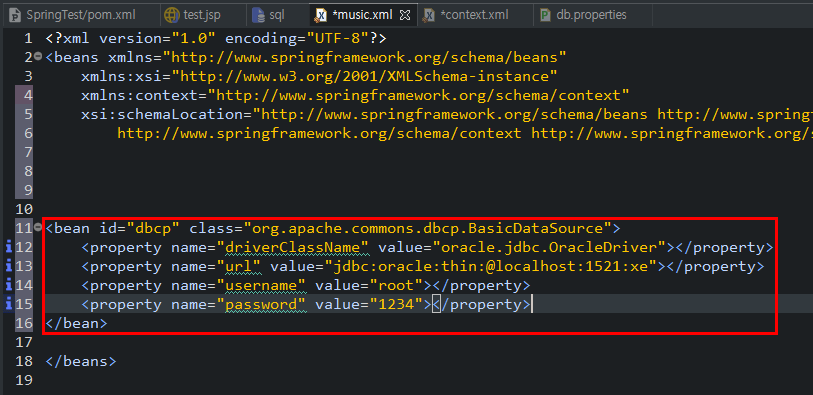
* Namespaces에서 context 지정



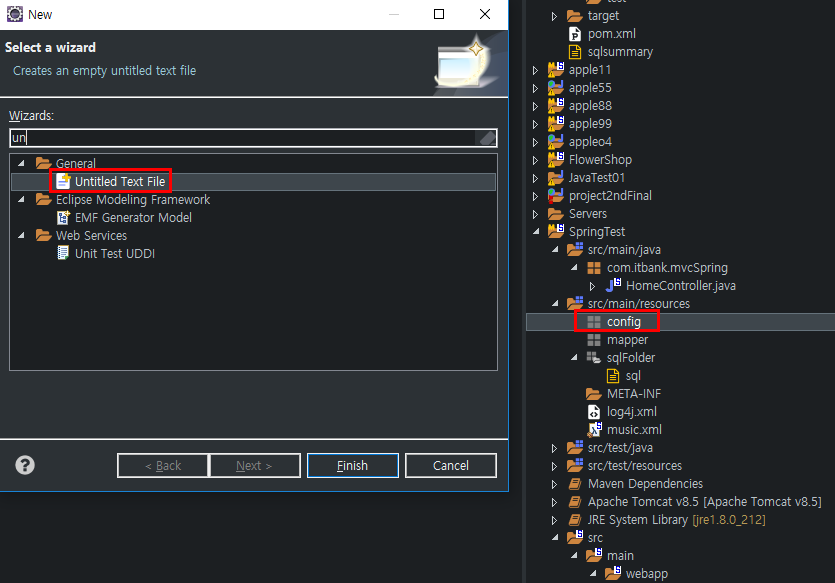


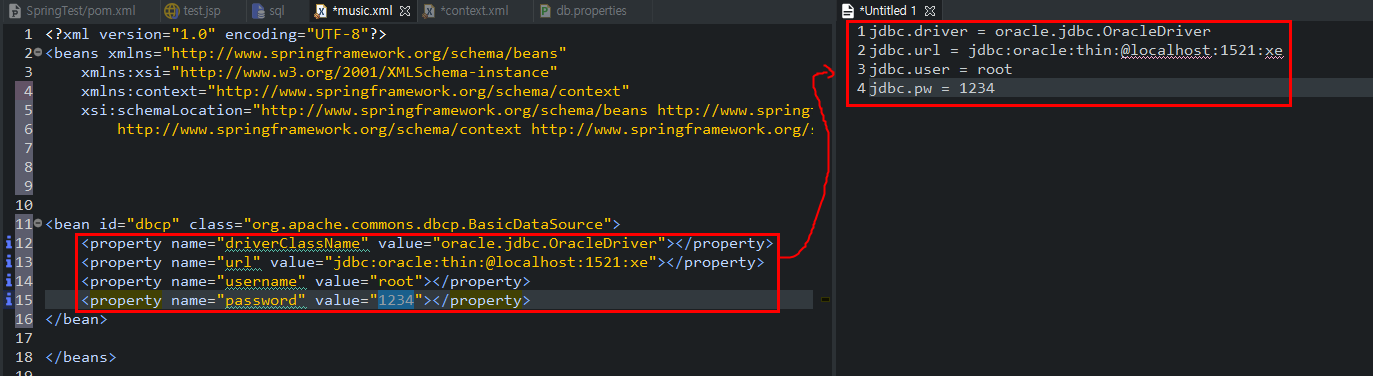


Oracle Driver 설정( DBCP )

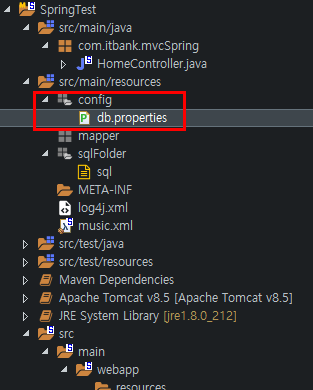
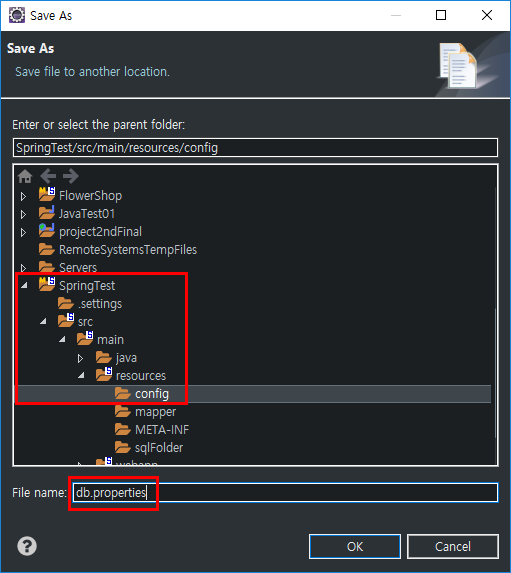


* dbcp value값들 문서에서 끌어와서 사용하게끔 셋팅

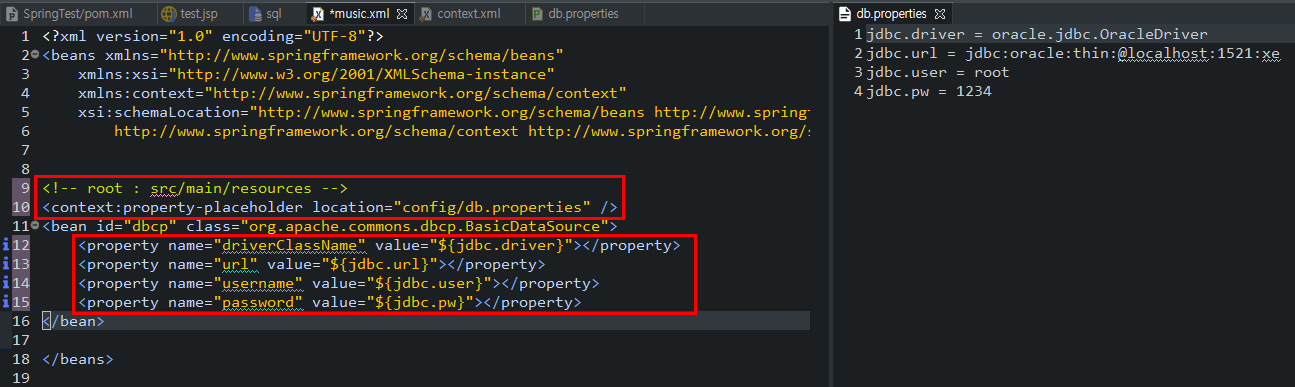




* untitle문서 이름 지정( db.properties )후 저장

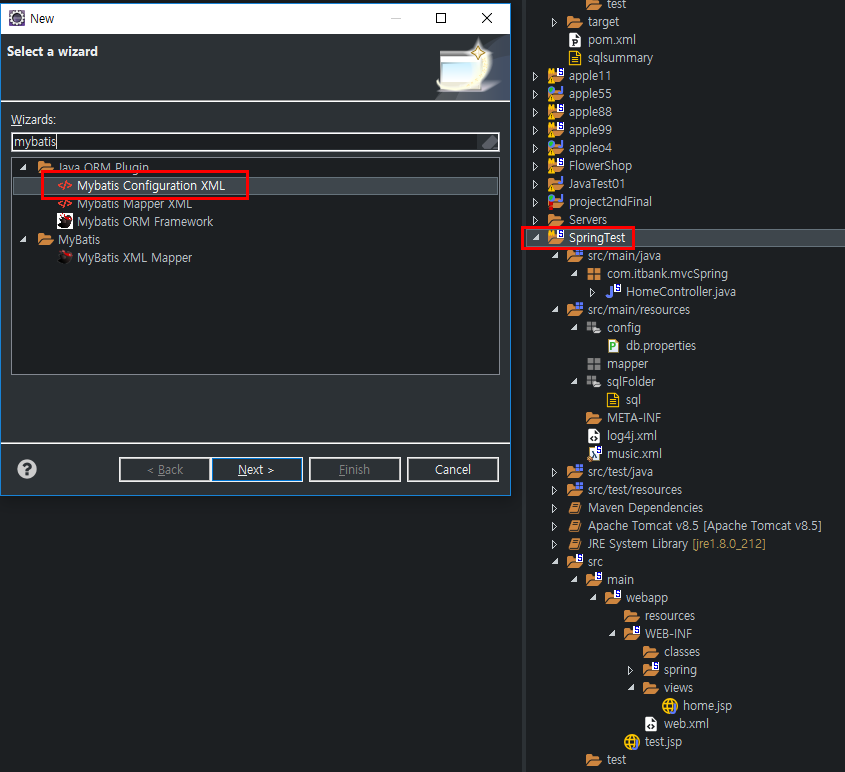


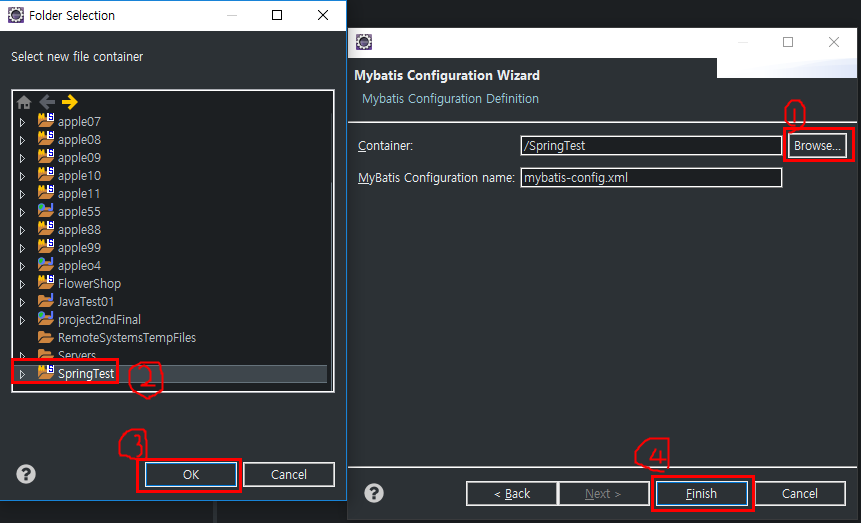
* db.properties의 내용을 가져오기 위해 <context:property:property-placeholder/> 활용
* value= “ ” 쌍따옴표 안에 EL활용 적용

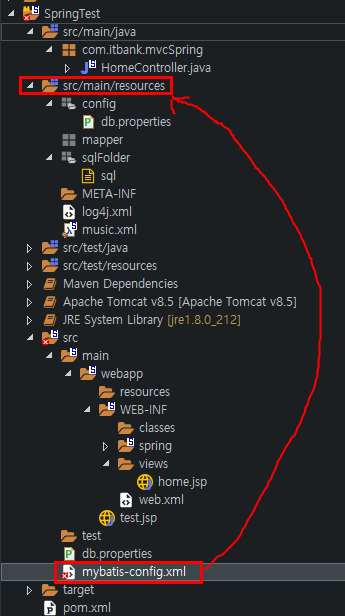


Mybatis Configuration XML 생성 및 설정

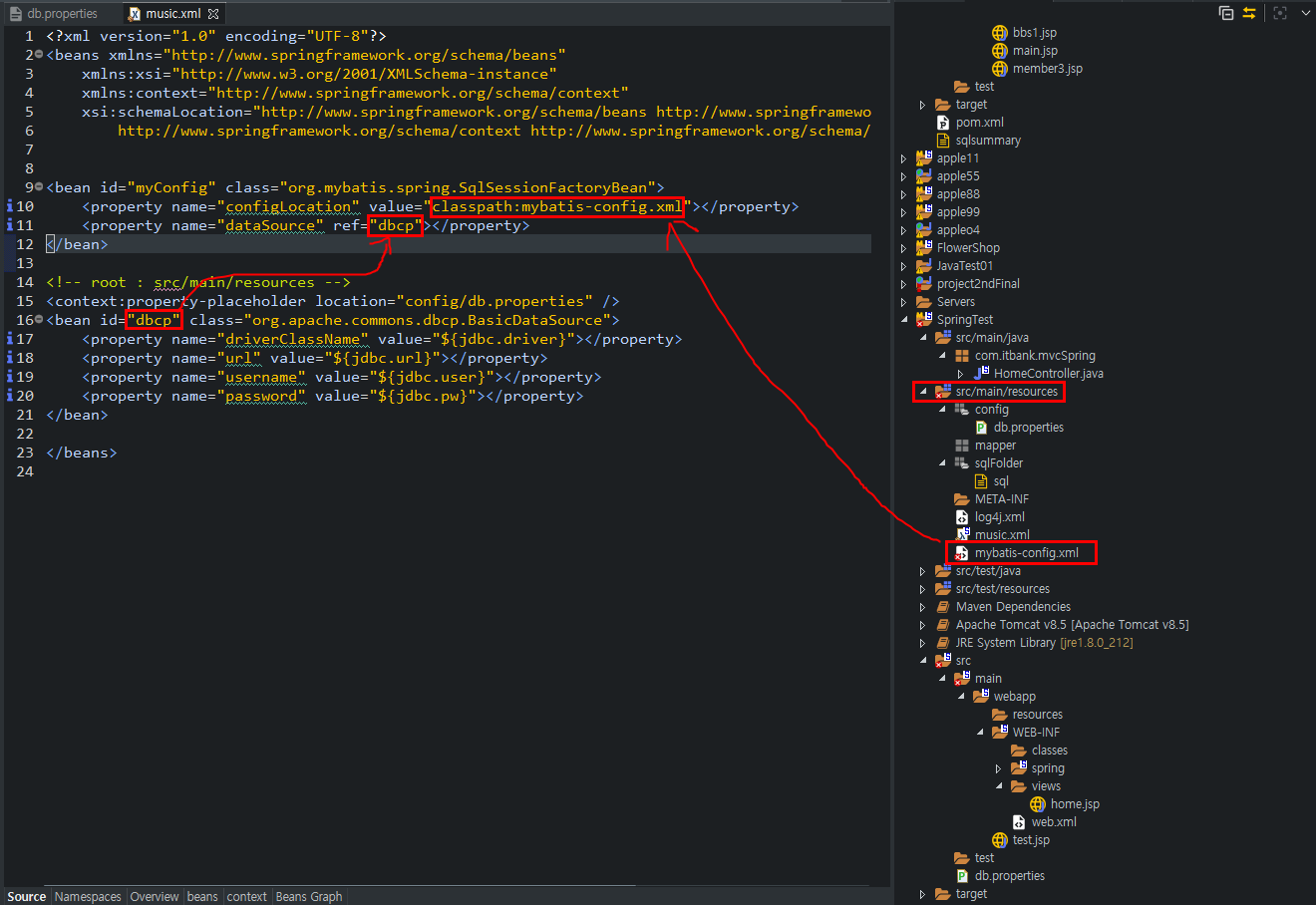
* Mybatis Configuration XML 생성

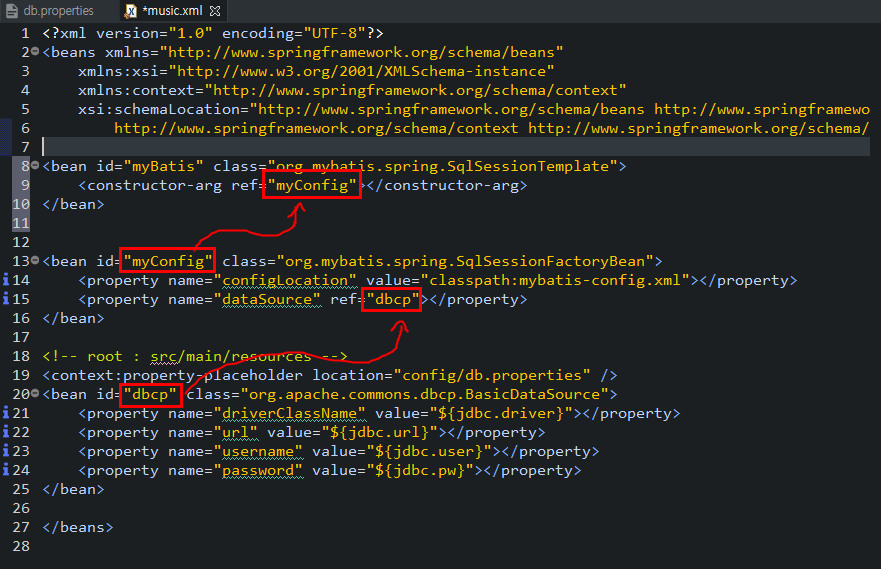




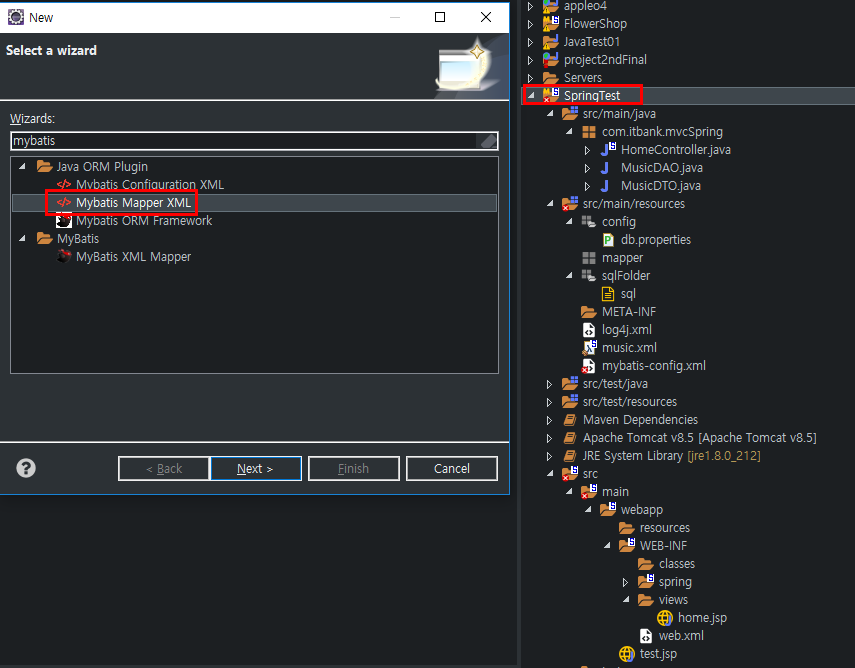


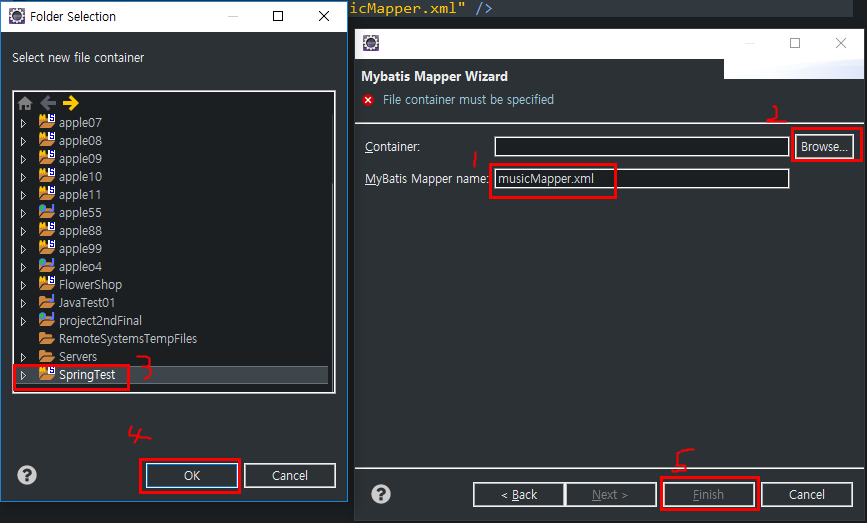
* Mybatis Configuration XML 설정

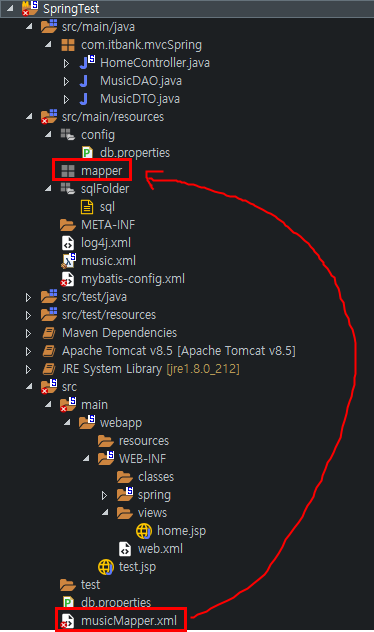




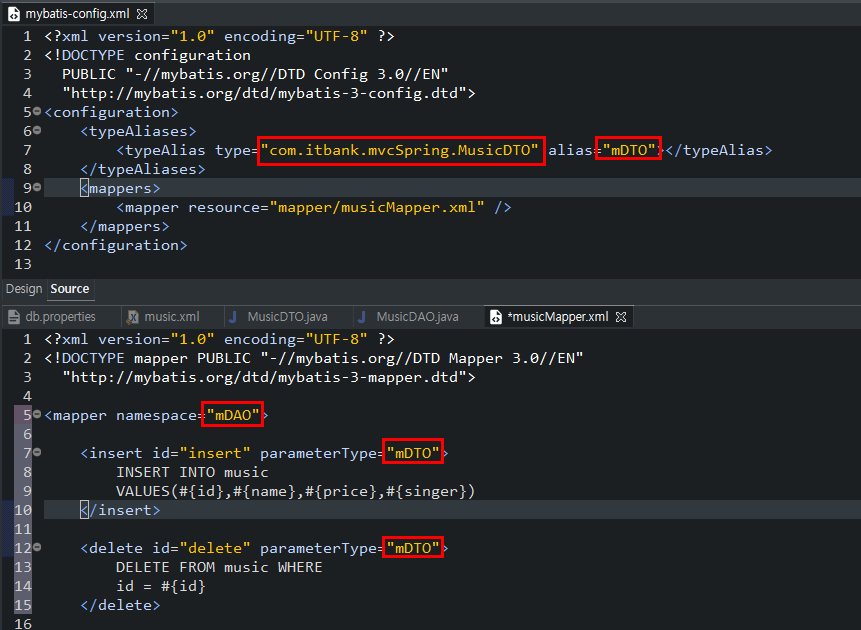
* Mybatis Mapper XML 생성 및 설정



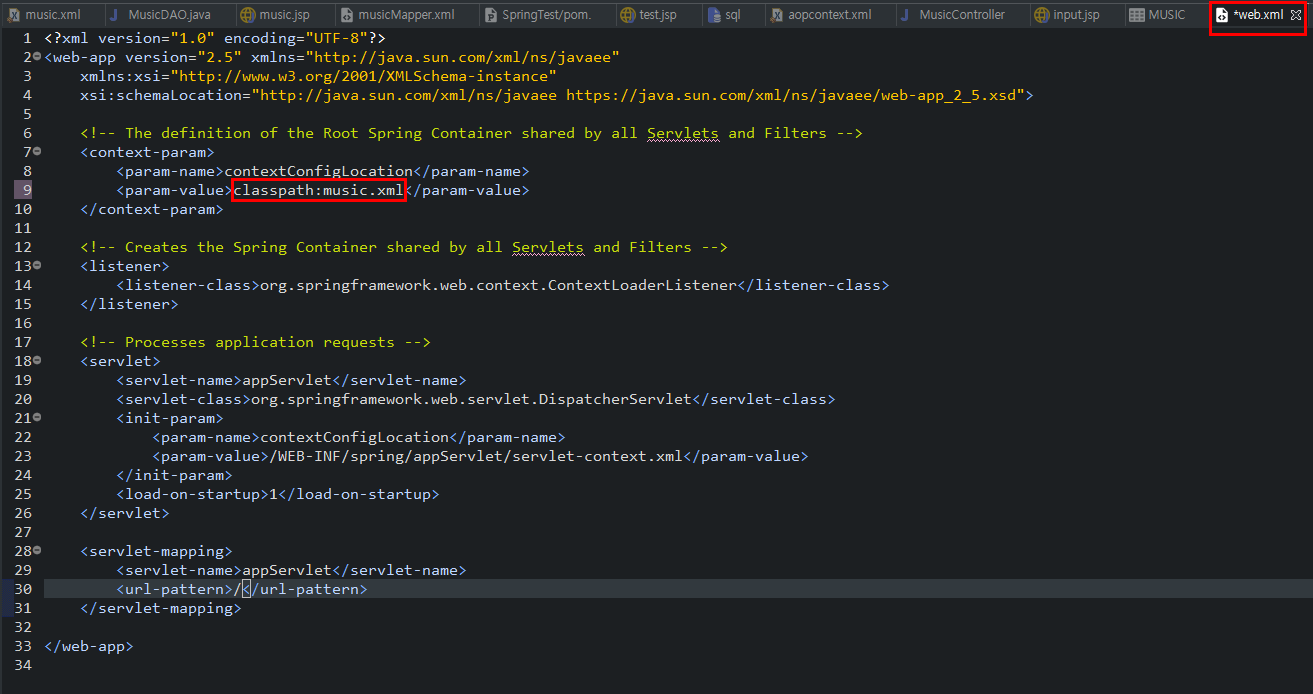




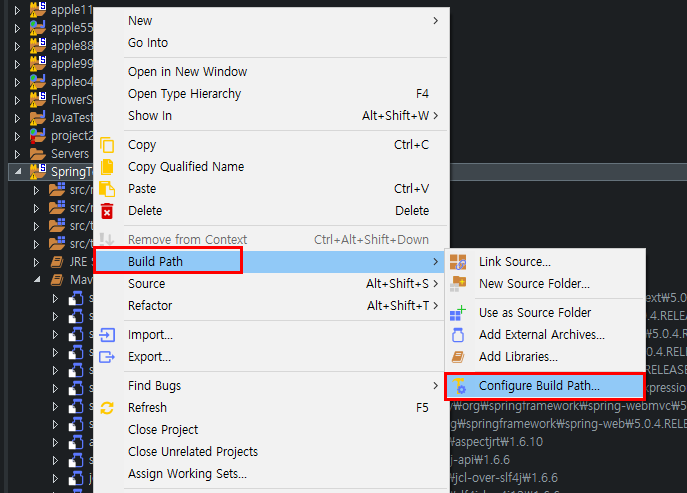
* Mybatis Configuration XML, Mybatis Mapper XML 상호 설정



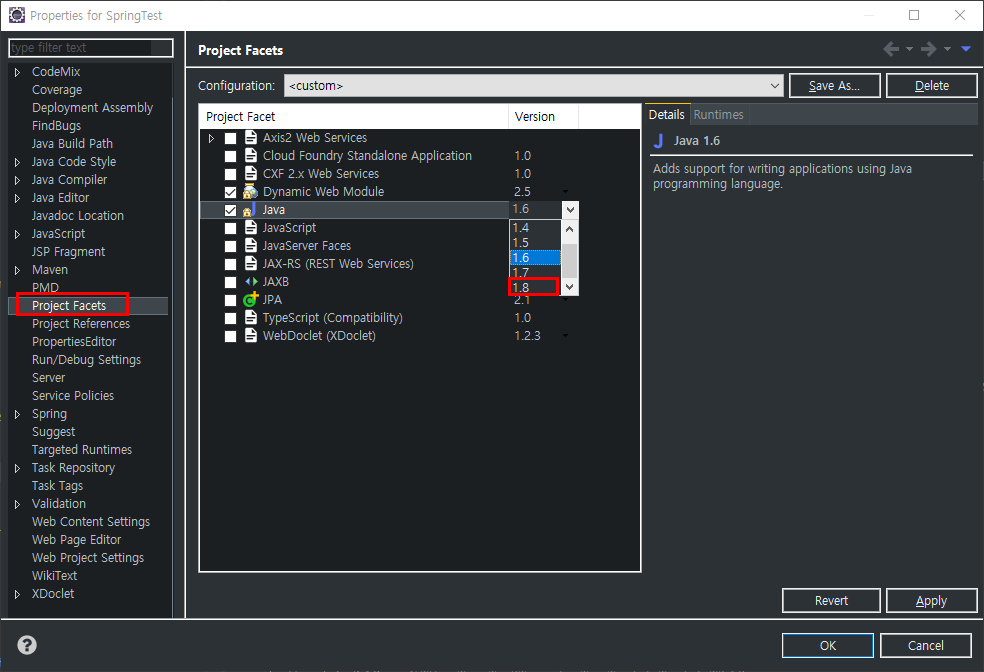
web.xml 셋팅 : <context-param>태그 안의 value의 경로를 새로 만든 musci.xml context파일로 변경



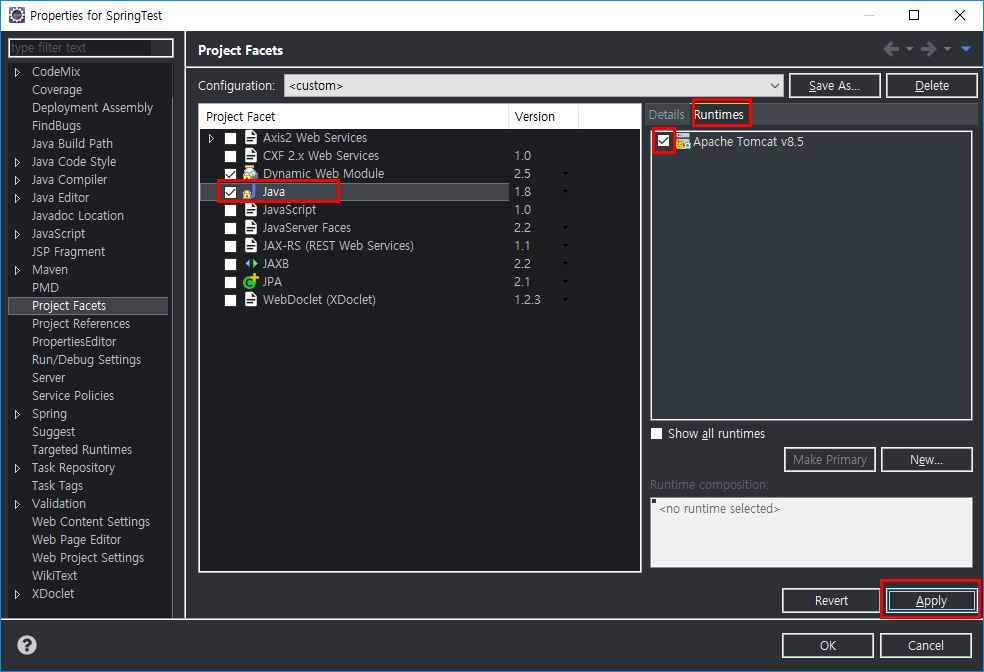
톰킷 설정 및 test.jsp로 테스트



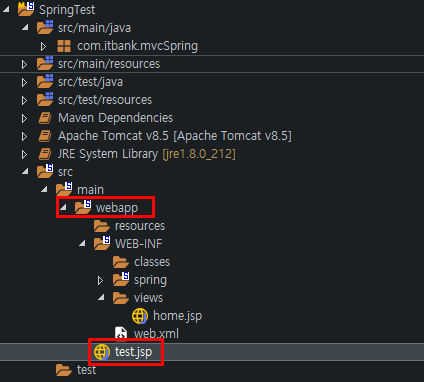
* java version 설정( 1.6 → 1.8 )

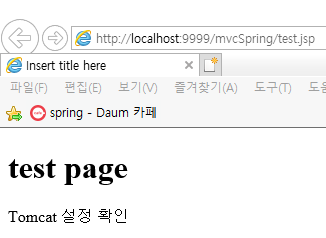
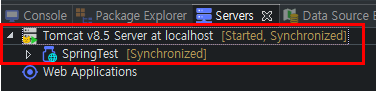


* tomcat 추가후 Apply

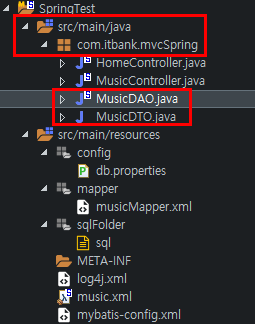


* test.jsp 파일 생성 및 테스트

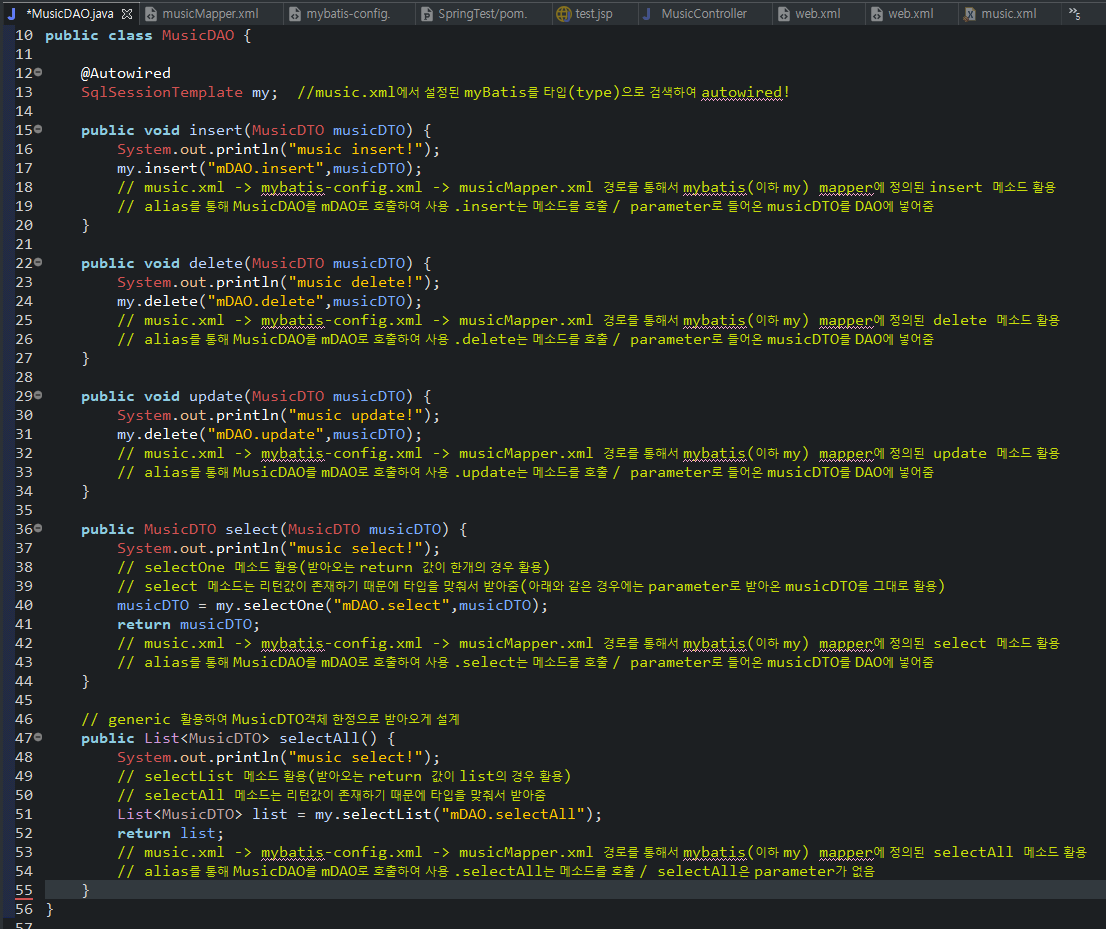




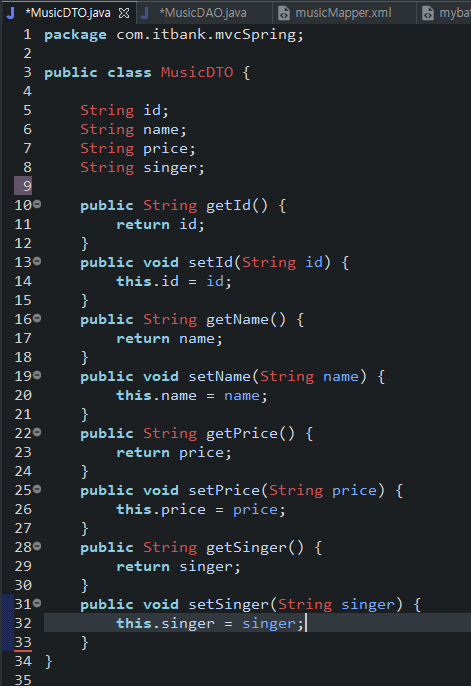
DAO, DTO 위치



DAO 삽입, 삭제, 수정, 검색, 전체검색 기능



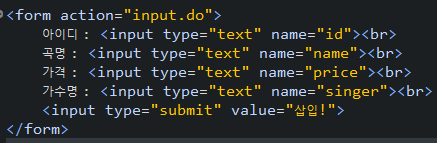
DTO 변수( id, name, price, singer )



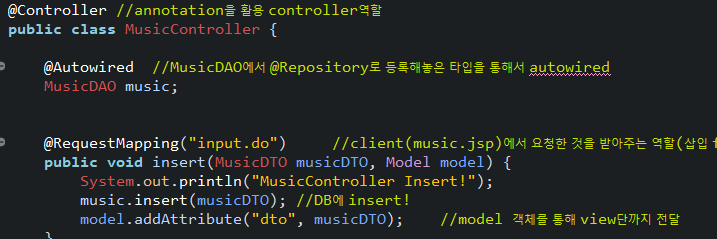
각 과정 설명

* Input를 예로 들어 설명( 모든 기능 흐름은 비슷하기 때문에)

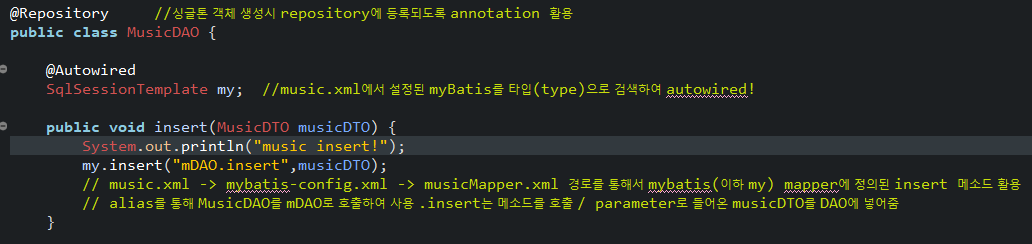
Client( webapp/music.jsp ) → input.do 요청 → Front Controller(이하 FC) →

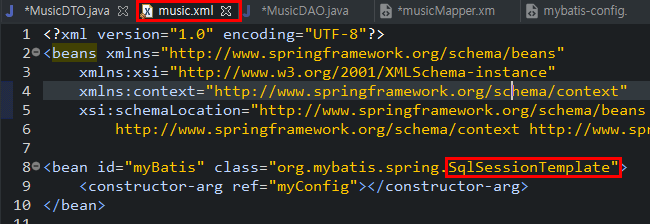


MusicController.java에서 RequestMapping값이 input.do인 메소드를 호출 → 종속 함수 실행

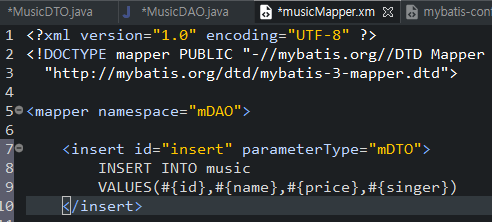


→ MusicDAO 객체인 music에서 insert메소드 호출 → 종속함수 실행





→ SqlSessionTemplate을 autowired한 my라는 객체명으로 music.xml에서 설정한 myBatis 실행 insert 호출 →



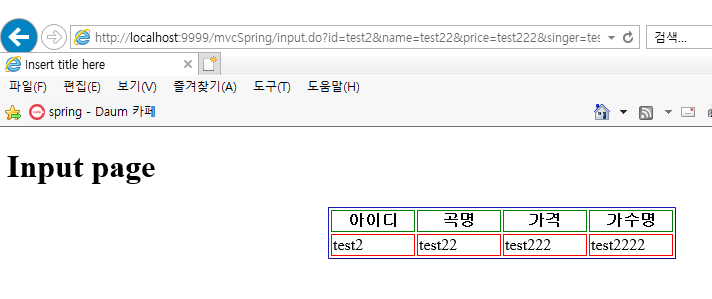
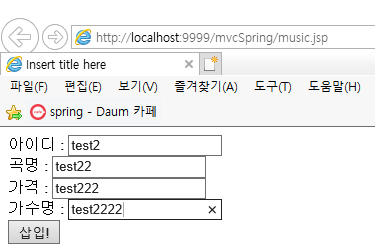
→ my.insert( parameter1, parameter2 ) → parameter1 : mDAO.insert는 musicMapper.xml에서 mDAO로 정의된 객체와 insert메소드를 실행 / parameter2 : 입력값 → Mapper에 작성된 sql문 실행

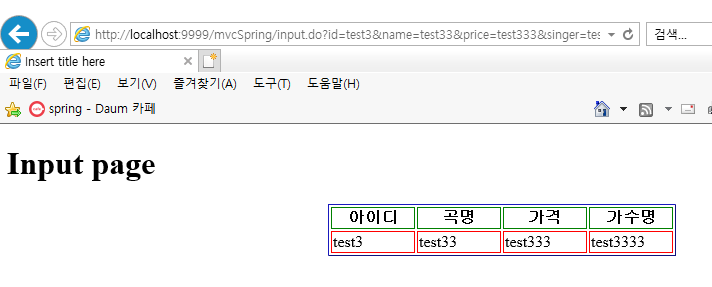
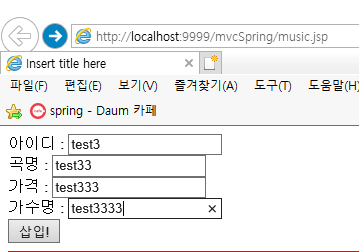
console 창의 결과 : MusicController Insert 작동 후 music insert 작동



-------- 실행결과

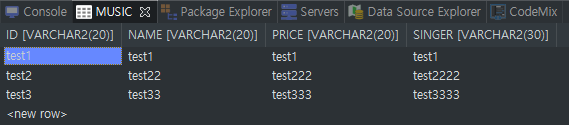
Input(Insert 기능)



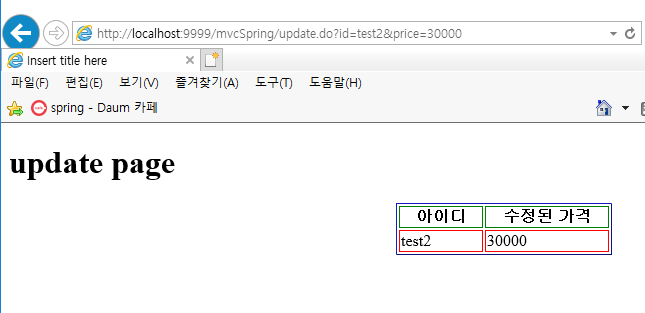
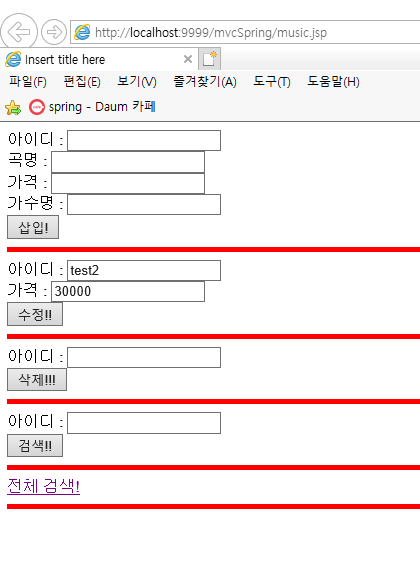


Update

* DB 내용

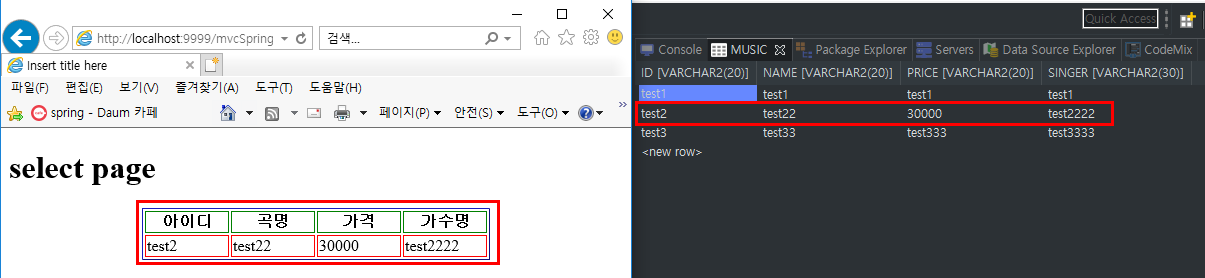
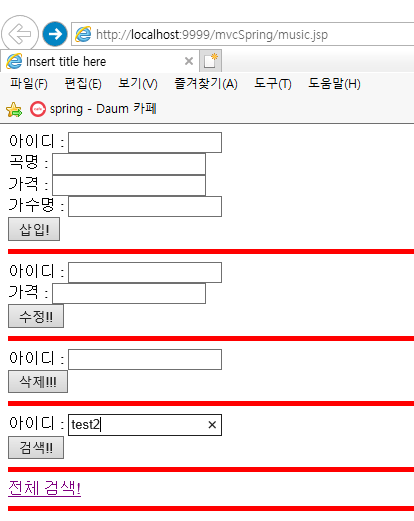


* update( id : test2, price : test222 → price : 30000 )



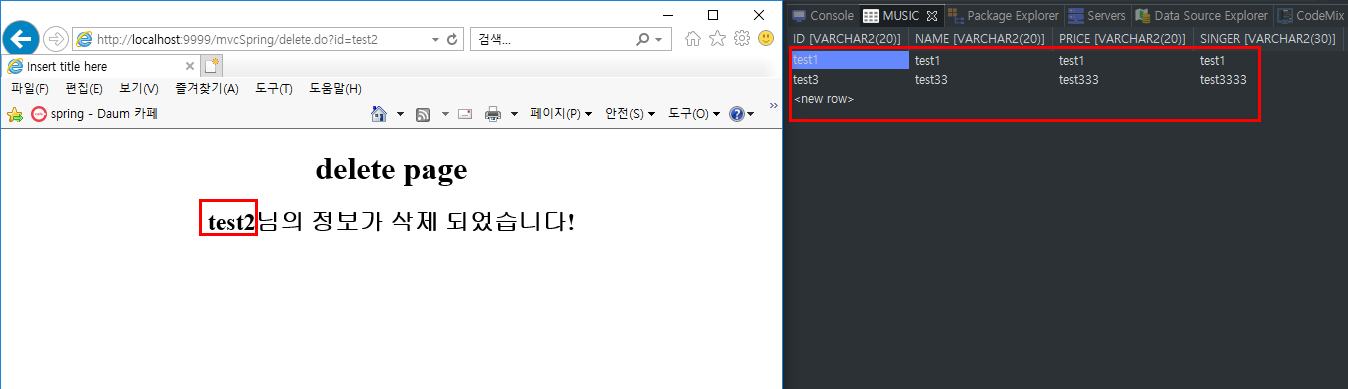
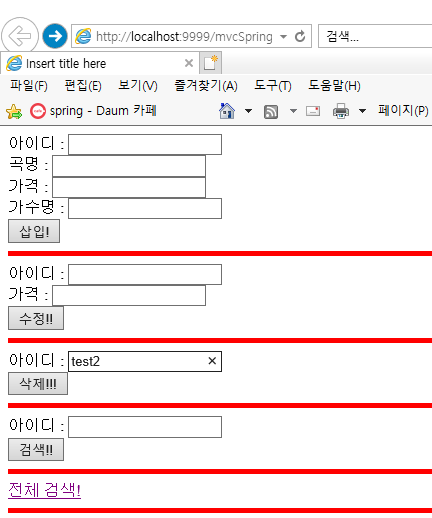
Select

* id : test2 검색

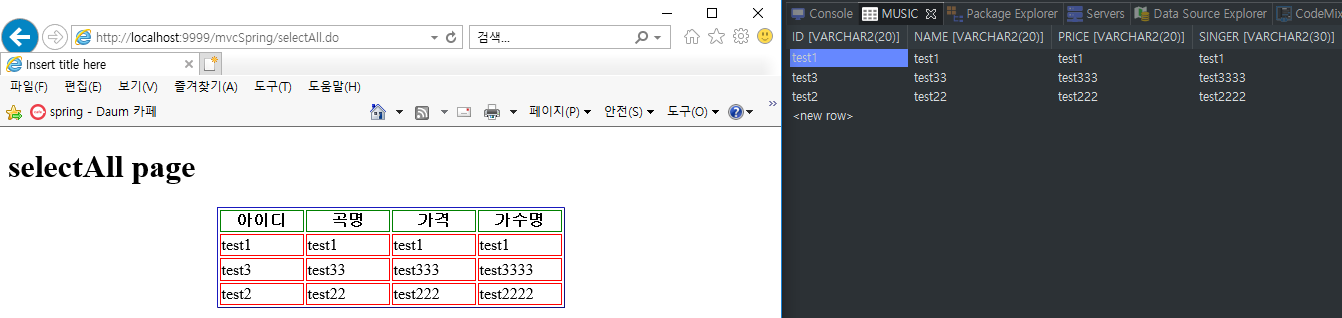
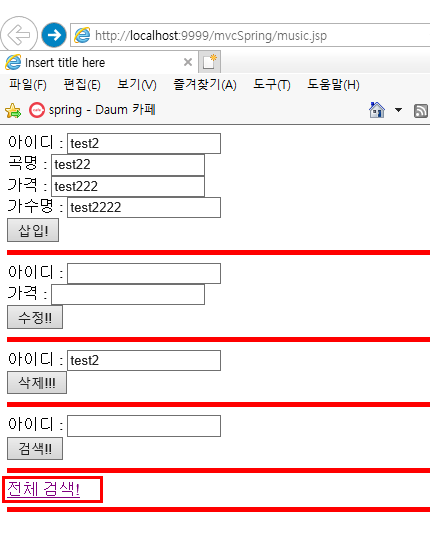


Delete

* id : test2 삭제



SelectAll



-------- 소스코드

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.itbank</groupId>

<artifactId>mvcSpring</artifactId>

<name>SpringTest</name>

<packaging>war</packaging>

<version>1.0.0-BUILD-SNAPSHOT</version>

<properties>

<java-version>1.8</java-version> <!-- 1.6 -> 1.8로 변경 -->

<org.springframework-version>5.0.4.RELEASE</org.springframework-version> <!-- 3.1.1 -> 5.0.4.로 변경 -->

<org.aspectj-version>1.6.10</org.aspectj-version>

<org.slf4j-version>1.6.6</org.slf4j-version>

</properties>

<!-- oracle 경로를 우회할 수 있는 주소를 여러개 담아둠 -->

<repositories>

<repository>

<id>mvn2</id>

<url>http://repo1.maven.org/maven2/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>oracle</id>

<name>ORACLE JDBC Repository</name>

<url>http://maven.jahia.org/maven2</url>

</repository>

<repository>

<id>codelds</id>

<url>http://code.lds.org/nexus/content/groups/main-repo</url>

</repository>

</repositories>

<dependencies>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework-version}</version>

<exclusions>

<!-- Exclude Commons Logging in favor of SLF4j -->

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework-version}</version>

</dependency>

<!-- AspectJ -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>${org.aspectj-version}</version>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${org.slf4j-version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.15</version>

<exclusions>

<exclusion>

<groupId>javax.mail</groupId>

<artifactId>mail</artifactId>

</exclusion>

<exclusion>

<groupId>javax.jms</groupId>

<artifactId>jms</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jdmk</groupId>

<artifactId>jmxtools</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jmx</groupId>

<artifactId>jmxri</artifactId>

</exclusion>

</exclusions>

<scope>runtime</scope>

</dependency>

<!-- @Inject -->

<dependency>

<groupId>javax.inject</groupId>

<artifactId>javax.inject</artifactId>

<version>1</version>

</dependency>

<!-- Servlet -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>2.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- Test -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.7</version>

<scope>test</scope>

</dependency>

<!-- oracle DB Connector -->

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0.3</version>

</dependency> <!-- DB Connector -->

<!-- commons-dbcp -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency> <!-- dbcp -->

<!-- mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.0</version> <!-- mybatis -->

</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->

<!-- mybatis와 spring을 붙여주는 것 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.2</version> <!-- spring에서 mybatis 연결해주는 connector 역할 -->

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->

<!-- spring - jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>5.0.4.RELEASE</version>

</dependency> <!-- spring에서 JDBC를 서포트 해주는 -->

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-eclipse-plugin</artifactId>

<version>2.9</version>

<configuration>

<additionalProjectnatures>

<projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>

</additionalProjectnatures>

<additionalBuildcommands>

<buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>

</additionalBuildcommands>

<downloadSources>true</downloadSources>

<downloadJavadocs>true</downloadJavadocs>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<source>1.6</source>

<target>1.6</target>

<compilerArgument>-Xlint:all</compilerArgument>

<showWarnings>true</showWarnings>

<showDeprecation>true</showDeprecation>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>1.2.1</version>

<configuration>

<mainClass>org.test.int1.Main</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

music.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:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="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-4.3.xsd">

<bean id="myBatis" class="org.mybatis.spring.SqlSessionTemplate">

<constructor-arg ref="myConfig"></constructor-arg>

</bean>

<bean id="myConfig" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="configLocation" value="classpath:mybatis-config.xml"></property>

<property name="dataSource" ref="dbcp"></property>

</bean>

<!-- root : src/main/resources -->

<context:property-placeholder location="classpath:config/db.properties" />

<bean id="dbcp" class="org.apache.commons.dbcp.BasicDataSource">

<property name="driverClassName" value="${jdbc.driver}"></property>

<property name="url" value="${jdbc.url}"></property>

<property name="username" value="${jdbc.user}"></property>

<property name="password" value="${jdbc.pw}"></property>

</bean>

</beans>

db.properties

jdbc.driver = oracle.jdbc.OracleDriver

jdbc.url = jdbc:oracle:thin:@localhost:1521:xe

jdbc.user = root

jdbc.pw = 1234

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:music.xml</param-value>

</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- Processes application requests -->

<servlet>

<servlet-name>appServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>appServlet</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

mybatis-config.xml

<?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>

<typeAliases>

<typeAlias type="com.itbank.mvcSpring.MusicDTO" alias="mDTO"></typeAlias>

</typeAliases>

<mappers>

<mapper resource="mapper/musicMapper.xml" />

</mappers>

</configuration>

musicMapper.xml

<?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="mDAO">

<insert id="insert" parameterType="mDTO">

INSERT INTO music

VALUES(#{id},#{name},#{price},#{singer})

</insert>

<select id="select" parameterType="mDTO" resultMap="rsMap">

SELECT \*

FROM music

WHERE id = #{id}

</select>

<resultMap type="mDTO" id="rsMap">

<result property="id" column="id" />

<result property="name" column="name" />

<result property="price" column="price" />

<result property="singer" column="singer" />

</resultMap>

<select id="selectAll" resultMap="rsMap">

SELECT \* FROM music

</select>

<delete id="delete" parameterType="mDTO">

DELETE FROM music WHERE

id = #{id}

</delete>

<update id="update" parameterType="mDTO">

UPDATE music

SET

price=#{price}

WHERE

id = #{id}

</update>

</mapper>

sql

create table music(

id varchar2(20),

name varchar2(20),

price varchar2(20),

singer varchar2(30)

)

select \* from music

Music.Controller.java

package com.itbank.mvcSpring;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller //annotation을 활용 controller역할

public class MusicController {

@Autowired //MusicDAO에서 @Repository로 등록해놓은 타입을 통해서 autowired

MusicDAO music;

@RequestMapping("input.do") //client(music.jsp)에서 요청한 것을 받아주는 역할(삽입 form에서 insert.do를 호출하면 이 컨트롤러에서 이 RequesMapping된 메소드를 실행)

public void insert(MusicDTO musicDTO, Model model) {

System.out.println("MusicController Insert!");

music.insert(musicDTO); //DB에 insert!

model.addAttribute("dto", musicDTO); //model 객체를 통해 view단까지 전달

}

@RequestMapping("update.do") //client(music.jsp)에서 요청한 것을 받아주는 역할(수정 form에서 update.do를 호출하면 이 컨트롤러에서 이 RequesMapping된 메소드를 실행)

public void update(MusicDTO musicDTO, Model model) {

System.out.println("MusicController Update!");

music.update(musicDTO); //DB에 update!

model.addAttribute("dto", musicDTO); //model 객체를 통해 view단까지 전달

}

@RequestMapping("delete.do") //client(music.jsp)에서 요청한 것을 받아주는 역할(삭제 form에서 delete.do를 호출하면 이 컨트롤러에서 이 RequesMapping된 메소드를 실행)

public void delete(MusicDTO musicDTO, Model model) {

System.out.println("MusicController Delete!");

model.addAttribute("dto", musicDTO); //model 객체를 통해 view단까지 전달

music.delete(musicDTO);

}

@RequestMapping("select.do") //client(music.jsp)에서 요청한 것을 받아주는 역할(검색 form에서 select.do를 호출하면 이 컨트롤러에서 이 RequesMapping된 메소드를 실행)

public void select(MusicDTO musicDTO, Model model) {

System.out.println("MusicController Select!");

// MusicDTO 객체를 dto라는 명칭으로 하나 더 만들어서(프로토 타입으로) select해온 dto를 받아서 model 객체로 view단까지 전달

MusicDTO dto = music.select(musicDTO);

model.addAttribute("dto", dto);

}

@RequestMapping("selectAll.do") //client(music.jsp)에서 요청한 것을 받아주는 역할(전체 검색 form에서 selectAll.do를 호출하면 이 컨트롤러에서 이 RequesMapping된 메소드를 실행)

public void select(Model model) {

System.out.println("MusicController SelectAll!");

// MusicDTO 객체를 dto라는 명칭으로 하나 더 만들어서(프로토 타입으로) select해온 dto를 받아서 model 객체로 view단까지 전달

List<MusicDTO> list = music.selectAll();

model.addAttribute("list", list);

}

}

MusicDAO.java

package com.itbank.mvcSpring;

import java.util.List;

import org.mybatis.spring.SqlSessionTemplate;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

@Repository //싱글톤 객체 생성시 repository에 등록되도록 annotation 활용

public class MusicDAO {

@Autowired

SqlSessionTemplate my; //music.xml에서 설정된 myBatis를 타입(type)으로 검색하여 autowired!

public void insert(MusicDTO musicDTO) {

System.out.println("music insert!");

my.insert("mDAO.insert",musicDTO);

// music.xml -> mybatis-config.xml -> musicMapper.xml 경로를 통해서 mybatis(이하 my) mapper에 정의된 insert 메소드 활용

// alias를 통해 MusicDAO를 mDAO로 호출하여 사용 .insert는 메소드를 호출 / parameter로 들어온 musicDTO를 DAO에 넣어줌

}

public void delete(MusicDTO musicDTO) {

System.out.println("music delete!");

my.delete("mDAO.delete",musicDTO);

// music.xml -> mybatis-config.xml -> musicMapper.xml 경로를 통해서 mybatis(이하 my) mapper에 정의된 delete 메소드 활용

// alias를 통해 MusicDAO를 mDAO로 호출하여 사용 .delete는 메소드를 호출 / parameter로 들어온 musicDTO를 DAO에 넣어줌

}

public void update(MusicDTO musicDTO) {

System.out.println("music update!");

my.delete("mDAO.update",musicDTO);

// music.xml -> mybatis-config.xml -> musicMapper.xml 경로를 통해서 mybatis(이하 my) mapper에 정의된 update 메소드 활용

// alias를 통해 MusicDAO를 mDAO로 호출하여 사용 .update는 메소드를 호출 / parameter로 들어온 musicDTO를 DAO에 넣어줌

}

public MusicDTO select(MusicDTO musicDTO) {

System.out.println("music select!");

// selectOne 메소드 활용(받아오는 return 값이 한개의 경우 활용)

// select 메소드는 리턴값이 존재하기 때문에 타입을 맞춰서 받아줌(아래와 같은 경우에는 parameter로 받아온 musicDTO를 그대로 활용)

musicDTO = my.selectOne("mDAO.select",musicDTO);

return musicDTO;

// music.xml -> mybatis-config.xml -> musicMapper.xml 경로를 통해서 mybatis(이하 my) mapper에 정의된 select 메소드 활용

// alias를 통해 MusicDAO를 mDAO로 호출하여 사용 .select는 메소드를 호출 / parameter로 들어온 musicDTO를 DAO에 넣어줌

}

// generic 활용하여 MusicDTO객체 한정으로 받아오게 설계

public List<MusicDTO> selectAll() {

System.out.println("music select!");

// selectList 메소드 활용(받아오는 return 값이 list의 경우 활용)

// selectAll 메소드는 리턴값이 존재하기 때문에 타입을 맞춰서 받아줌

List<MusicDTO> list = my.selectList("mDAO.selectAll");

return list;

// music.xml -> mybatis-config.xml -> musicMapper.xml 경로를 통해서 mybatis(이하 my) mapper에 정의된 selectAll 메소드 활용

// alias를 통해 MusicDAO를 mDAO로 호출하여 사용 .selectAll는 메소드를 호출 / selectAll은 parameter가 없음

}

}

MusicDTO.java

package com.itbank.mvcSpring;

public class MusicDTO {

String id;

String name;

String price;

String singer;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPrice() {

return price;

}

public void setPrice(String price) {

this.price = price;

}

public String getSinger() {

return singer;

}

public void setSinger(String singer) {

this.singer = singer;

}

}

webapp단

music.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<form action="input.do">

아이디 : <input type="text" name="id"><br>

곡명 : <input type="text" name="name"><br>

가격 : <input type="text" name="price"><br>

가수명 : <input type="text" name="singer"><br>

<input type="submit" value="삽입!">

</form>

<hr size="3" color="red">

<form action="update.do">

아이디 : <input type="text" name="id"><br>

가격 : <input type="text" name="price"><br>

<input type="submit" value="수정!!">

</form>

<hr size="3" color="red">

<form action="delete.do">

아이디 : <input type="text" name="id"><br>

<input type="submit" value="삭제!!!">

</form>

<hr size="3" color="red">

<form action="select.do">

아이디 : <input type="text" name="id"><br>

<input type="submit" value="검색!!">

</form>

<hr size="3" color="red">

<a href="selectAll.do">전체 검색!</a>

<hr size="3" color="red">

</body>

</html>

test.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>test page</h1>

<p>Tomcat 설정 확인</p>

</body>

</html>

View단

input.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>Input page</h1>

<center>

<table border="1" bordercolor="blue">

<tr bordercolor="green">

<th width="80">아이디</th>

<th width="80">곡명</th>

<th width="80">가격</th>

<th width="80">가수명</th>

</tr>

<tr bordercolor="red">

<td>${dto.id }</td>

<td>${dto.name }</td>

<td>${dto.price }</td>

<td>${dto.singer }</td>

</tr>

</table>

</center>

</body>

</html>

delete.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<center>

<h1>delete page</h1>

<h2>${dto.id }님의 정보가 삭제 되었습니다!</h2>

</center>

</body>

</html>

update.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>update page</h1>

<center>

<table border="1" bordercolor="blue" background="gray">

<tr bordercolor="green">

<th width="80">아이디</th>

<th width="120">수정된 가격</th>

</tr>

<tr bordercolor="red">

<td>${dto.id }</td>

<td>${dto.price }</td>

</tr>

</table>

</center>

</body>

</html>

select.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>select page</h1>

<center>

<table border="1" bordercolor="blue" background="lime">

<tr bordercolor="green">

<th width="80">아이디</th>

<th width="80">곡명</th>

<th width="80">가격</th>

<th width="80">가수명</th>

</tr>

<tr bordercolor="red">

<td>${dto.id }</td>

<td>${dto.name }</td>

<td>${dto.price }</td>

<td>${dto.singer }</td>

</tr>

</table>

</center>

</body>

</html>

selectAll.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>selectAll page</h1>

<center>

<table border="1" bordercolor="blue" background="lime">

<tr bordercolor="green">

<th width="80">아이디</th>

<th width="80">곡명</th>

<th width="80">가격</th>

<th width="80">가수명</th>

</tr>

<c:forEach var="dto" items="${list}">

<tr bordercolor="red">

<td>${dto.id }</td>

<td>${dto.name }</td>

<td>${dto.price }</td>

<td>${dto.singer }</td>

</tr>

</c:forEach>

</table>

</center>

</body>

</html>