

Project 2

Spring Web Tutorial

게시판 구축

Kwangwoon-Univ
2018 Software Project 1

Contents

1. 개발 환경 설정과 테스트

2. 스프링 MVC

3. 스프링 + MyBatis

4. 게시판 기능 구현

5. MySQL 기본

6. 각종 Tips

1. 개발 환경 설정과 테스트

Spring Tool Suite 설치

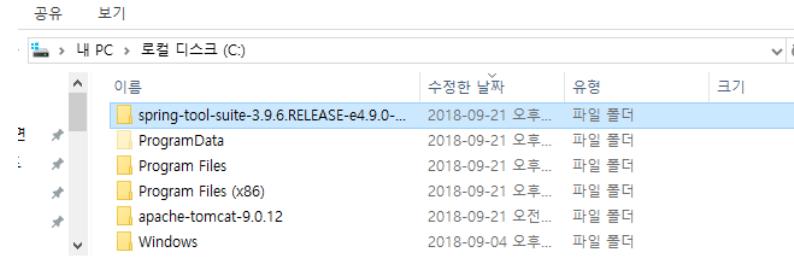
- Spring Tool Suite(STS) 설치 (사전에 JDK 설치가 필요)
 - <http://spring.io/tools/sts/all> 에 접속
- 현재 자신의 컴퓨터 사양(64bit or 32bit)에 맞는 버전으로 설치

The screenshot shows the 'Spring Tool Suite™ Downloads' page. It features the STS logo and the text 'STS 3.9.6.RELEASE New & Noteworthy'. Below this, there are download links for Windows, Mac, and Linux, all based on Eclipse 4.9.0.

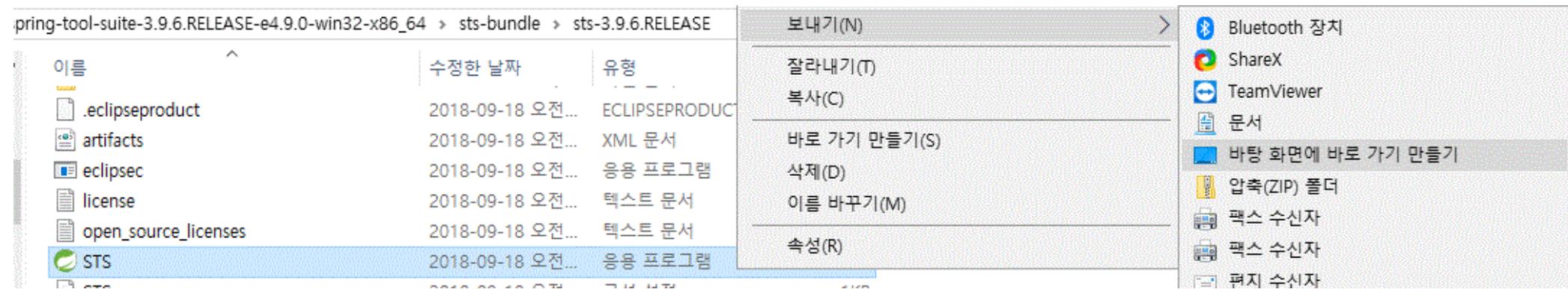
Platform	Version	Architecture	File Type	Size
Windows	Based on Eclipse 4.9.0	WIN, 32BIT	zip	403MB
		WIN, 64BIT	zip	403MB
Mac	Based on Eclipse 4.9.0			
Linux	Based on Eclipse 4.9.0			

Spring Tool Suite 설치

- 적당한 경로에 압축 해제

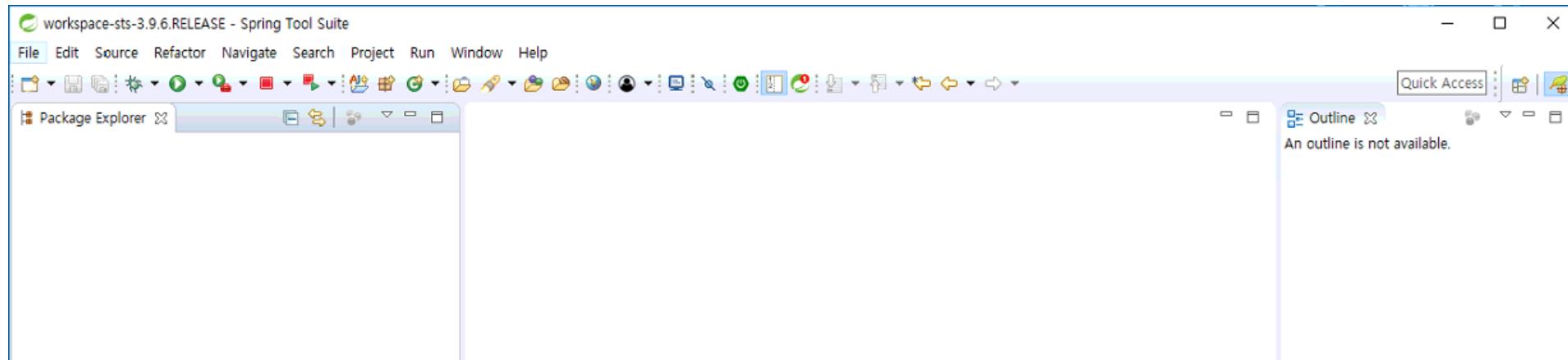
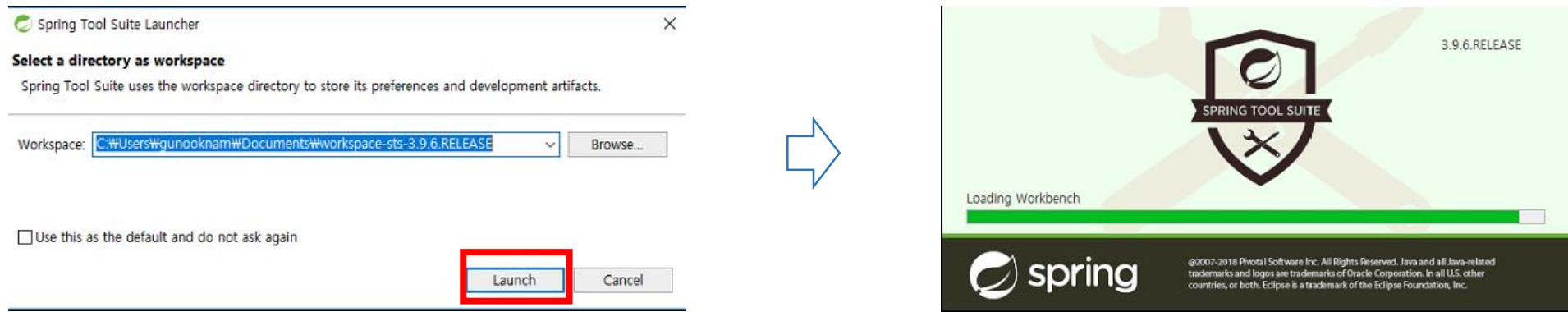


- STS 실행파일이 있는 경로에서 STS 파일을 실행
(바로가기 아이콘을 바탕화면에 만들거나, 작업 표시줄에 고정하여 사용)



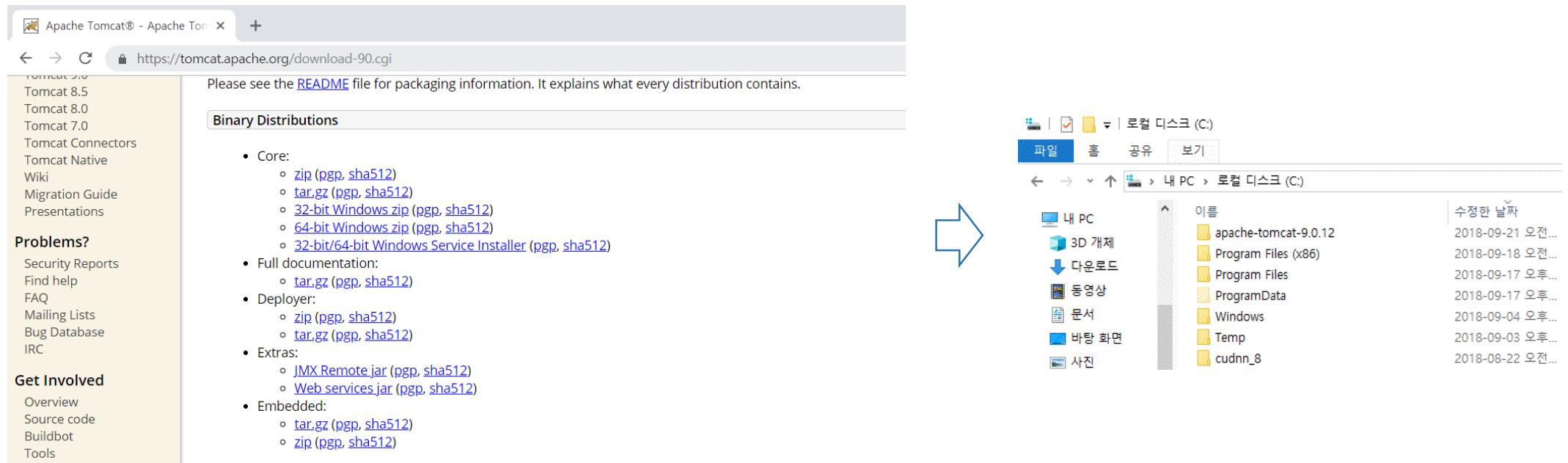
Spring Tool Suite 설치

- STS 실행 시 스프링 전용 이클립스 개발환경이 나타남



Tomcat 다운로드

- Tomcat은 WAS(web application server)로 사용
- <https://tomcat.apache.org/download-90.cgi> 접속하여 자신의 컴퓨터 사양(64bit or 32bit)에 맞는 Tomcat 9.0 다운 후 C 드라이브로 압축 해제



MySQL(Database) 설치

■ MySQL 설치

- <https://dev.mysql.com/downloads/installer/> 에 접속하여 설치

The screenshot shows the MySQL Installer download page for Windows 8.0.12. At the top, there's a header with the text "Generally Available (GA) Releases". Below it, the title "MySQL Installer 8.0.12" is displayed. A dropdown menu labeled "Select Operating System:" shows "Microsoft Windows" selected. To the right, a link says "Looking for previous GA versions?". Two download options are listed:

Installer Type	Version	File Size	Action
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.12.0.msi)	8.0.12	15.9M	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.12.0.msi)	8.0.12	273.4M	Download

Each download row also includes MD5 and Signature links.

MySQL(Database) 설치

■ 로그인 없이 바로 설치

Begin Your Download

mysql-installer-web-community-8.0.12.0.msi

[Login Now](#) or [Sign Up](#) for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system
- Comment in the MySQL Documentation

[Login »](#)

using my Oracle Web account

[Sign Up »](#)

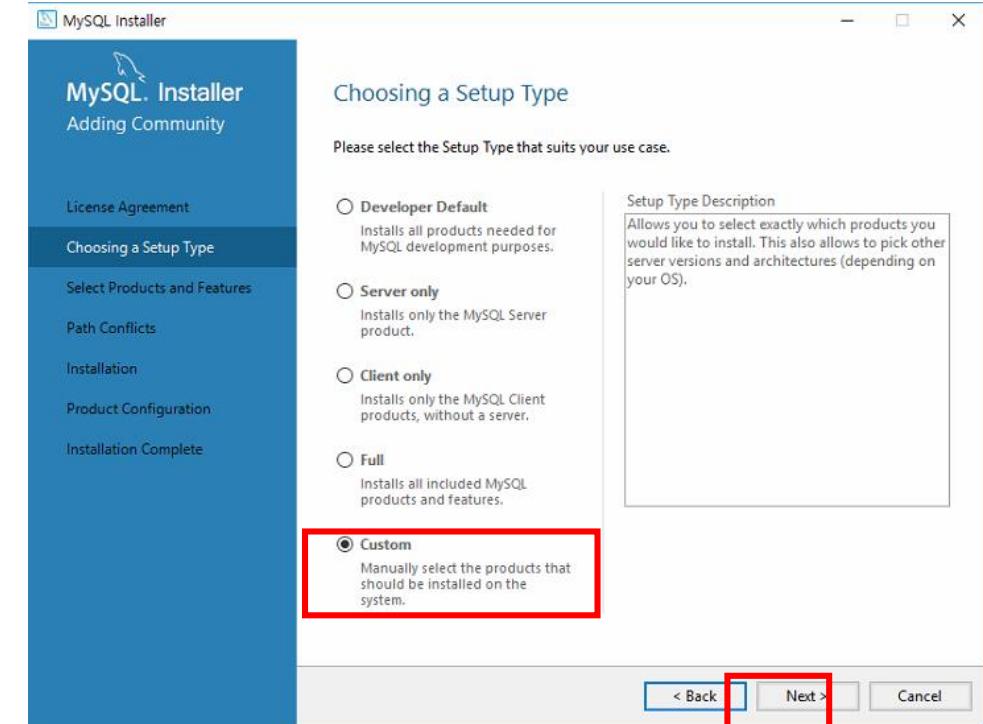
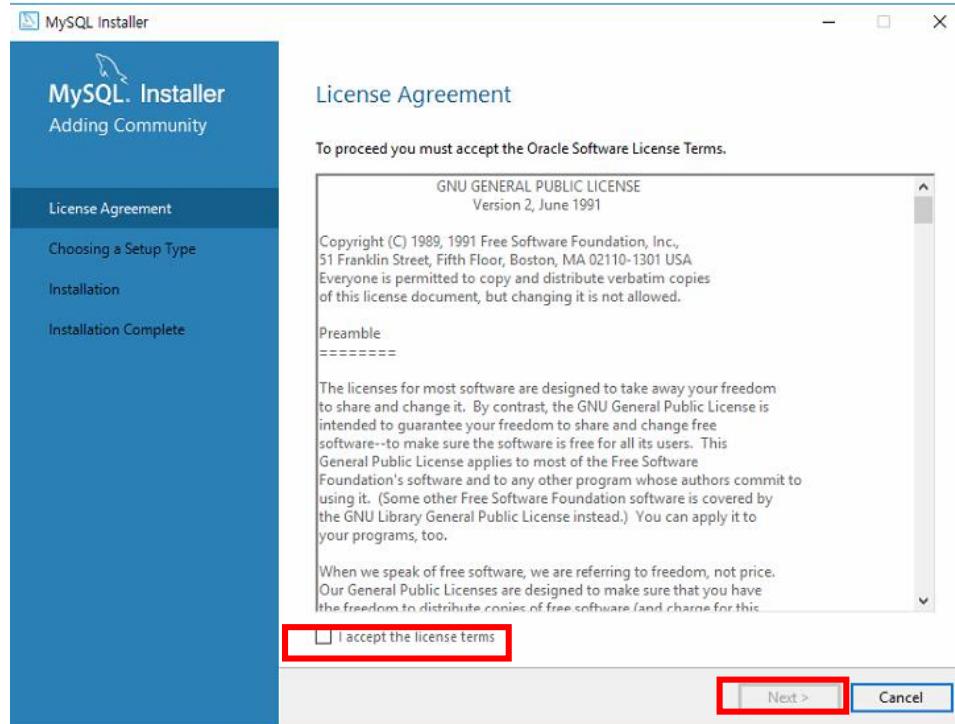
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

[No thanks, just start my download.](#)

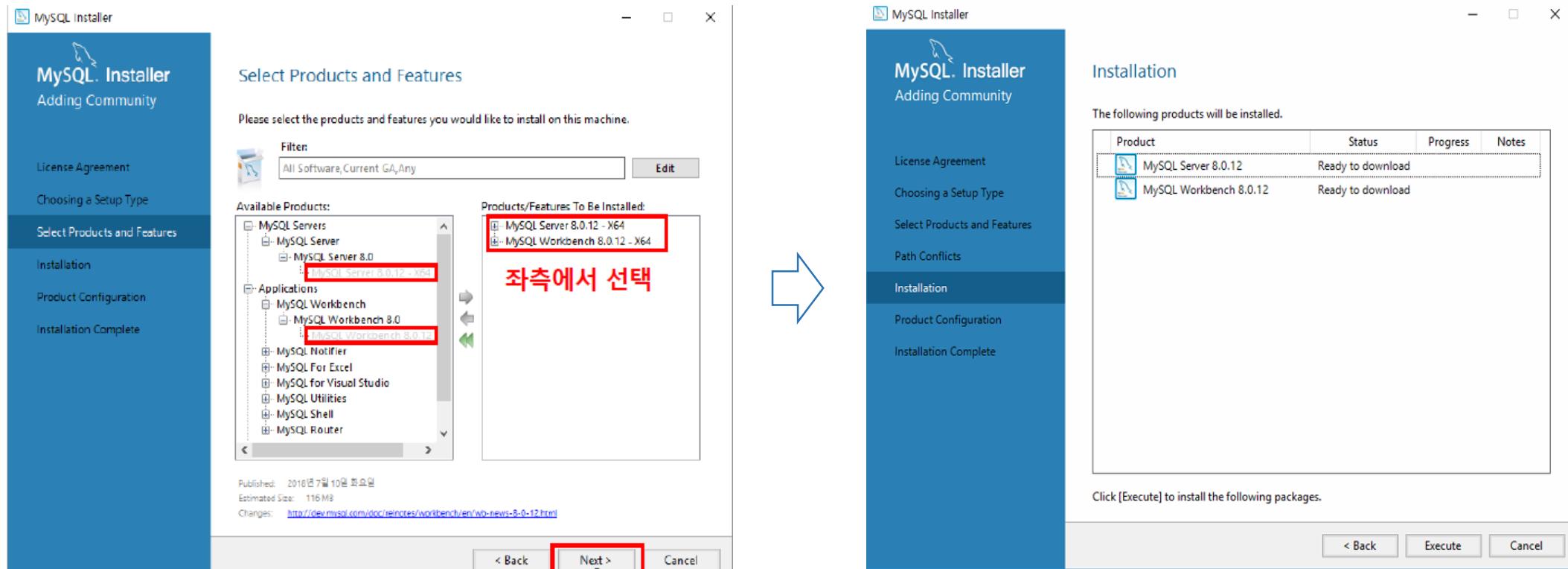
MySQL(Database) 설치

- 다운로드 후 실행
- Custom 선택



MySQL(Database) 설치

■ MySQL Server와 Workbench 설치



MySQL(Database) 설치

■ Workbench Prerequisites

- MySQL on Windows
- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Other Downloads

Download MySQL Wo

MySQL Workbench provides DBAs and developers an in

- Database Design & Modeling
- SQL Development
- Database Administration
- Database Migration

The Community (OSS) Edition is available from this page

Download source packages of LGPL libraries: [+]

MySQL Workbench Windows Prerequisites:

To be able to install and run MySQL Workbench on items are provided as links to the corresponding di

[Microsoft .NET Framework 4.5](#)

[Visual C++ Redistributable for Visual Studio 2015](#)

클릭 시 설치
페이지로 이동



Microsoft .NET Framework 4.5

Important! Selecting a language below will dynamically change the complete page content to that language.

Select Language:

[Download](#)

Visual Studio 2015용 Visual C++ 재배포 가능 패키지

중요! 아래에서 언어를 선택하면 전체 페이지 내용이 해당 언어로 신속하게 변경됩니다.

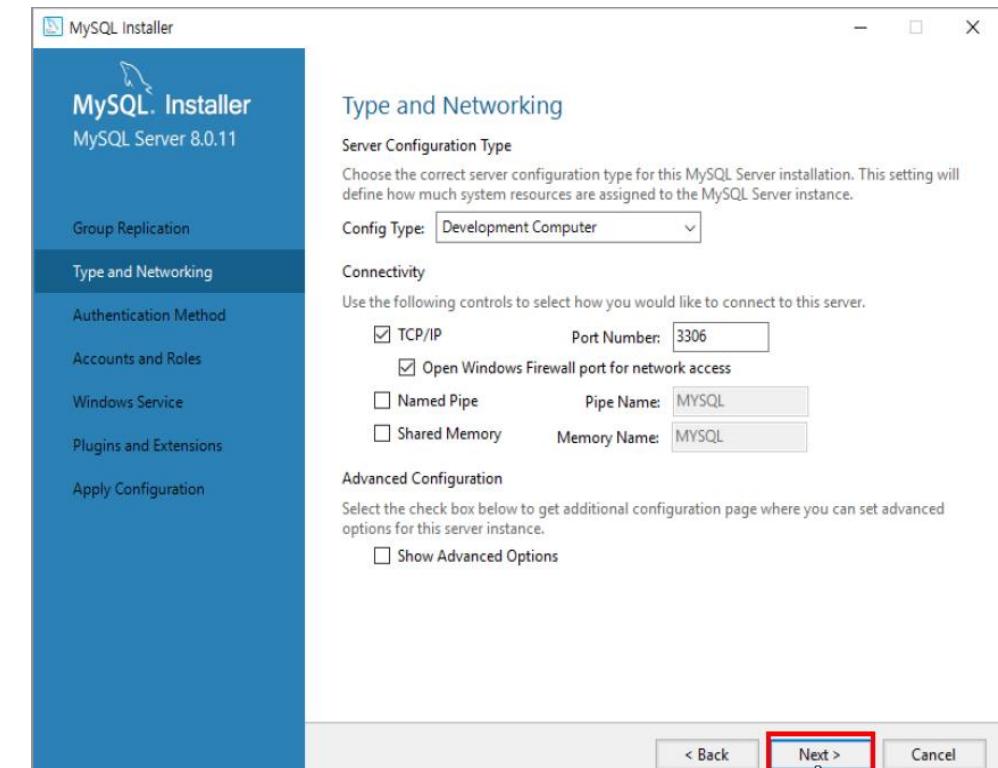
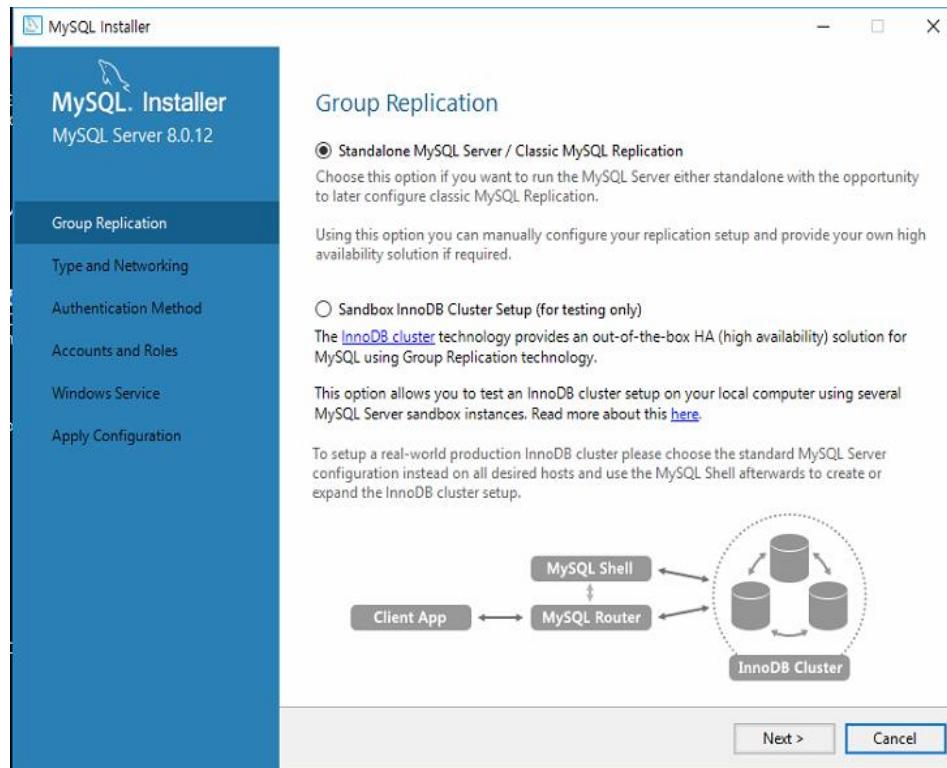
언어 선택:

[다운로드](#)

Workbench는 Prerequisites로 Microsoft .Net Framework 4.5와 Visual C++ Redistributable For Visual Studio 2015를 요구하여 이것이 기존에 설치 되지 않으면 Workbench 설치 failed 가 출력되므로 반드시 설치 필요

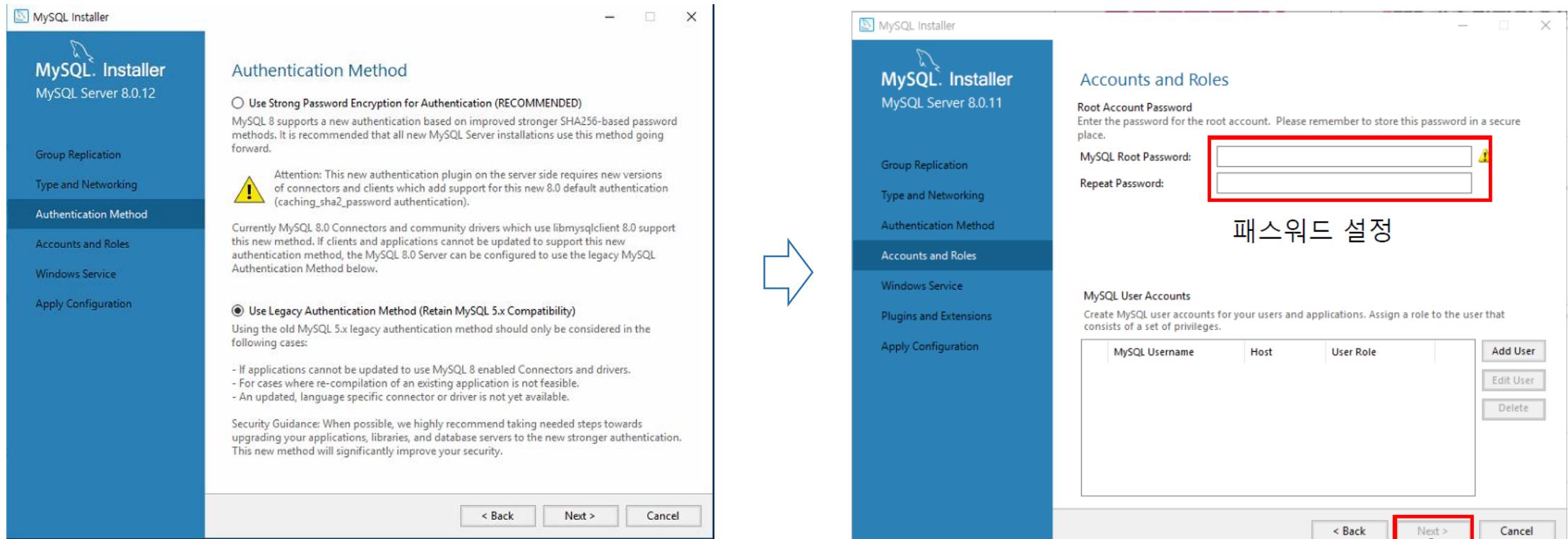
MySQL(Database) 설치

- MySQL은 Default로 Port 번호로 3306으로 셋팅



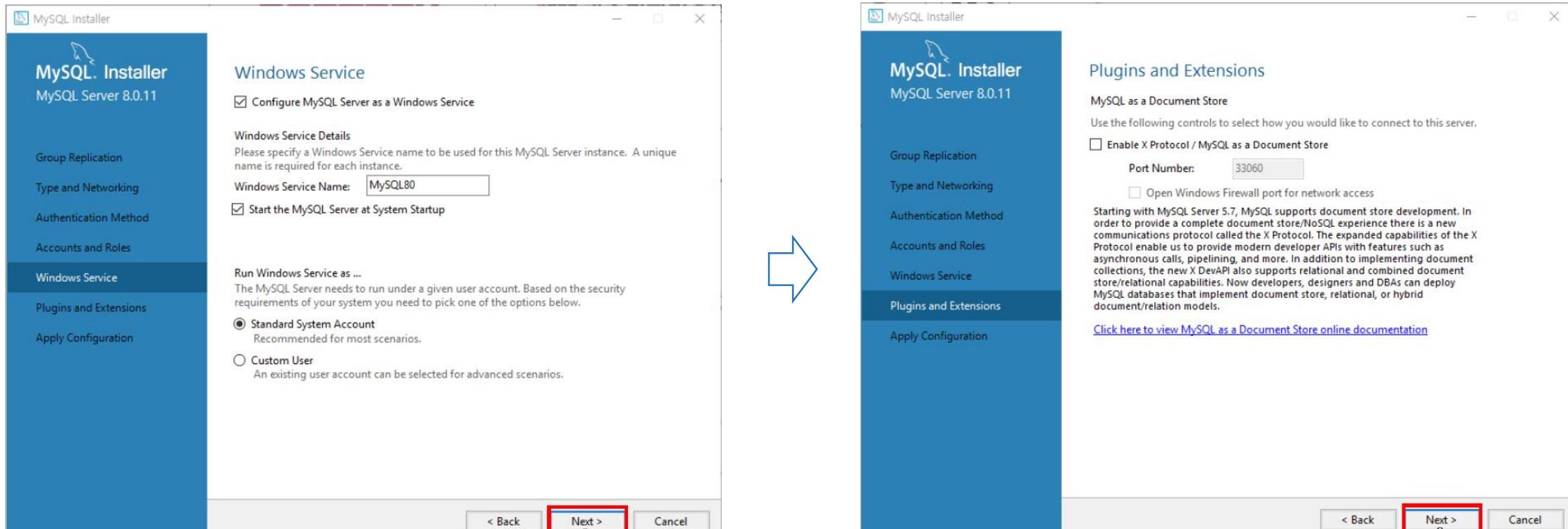
MySQL(Database) 설치

■ 비밀번호 설정



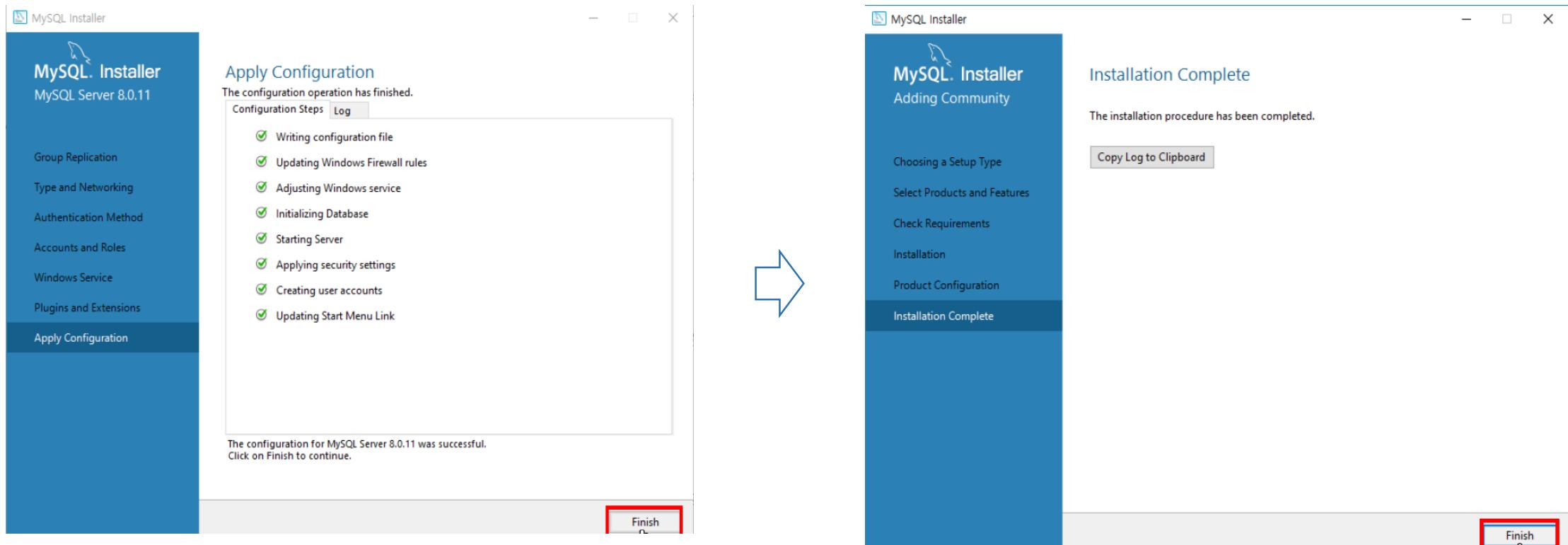
MySQL(Database) 설치

■ 계속 Next



MySQL(Database) 설치

■ 설치 완료



MySQL(Database) 설치

■ 설치된 MySQL Command Line Client 실행

- MySQL이 설치된 bin파일로 이동하여 mysql.exe -u root -p 명령어 입력 후 비밀번호 입력

```
Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql.exe -u root -p
Enter password: *****

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 537
Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

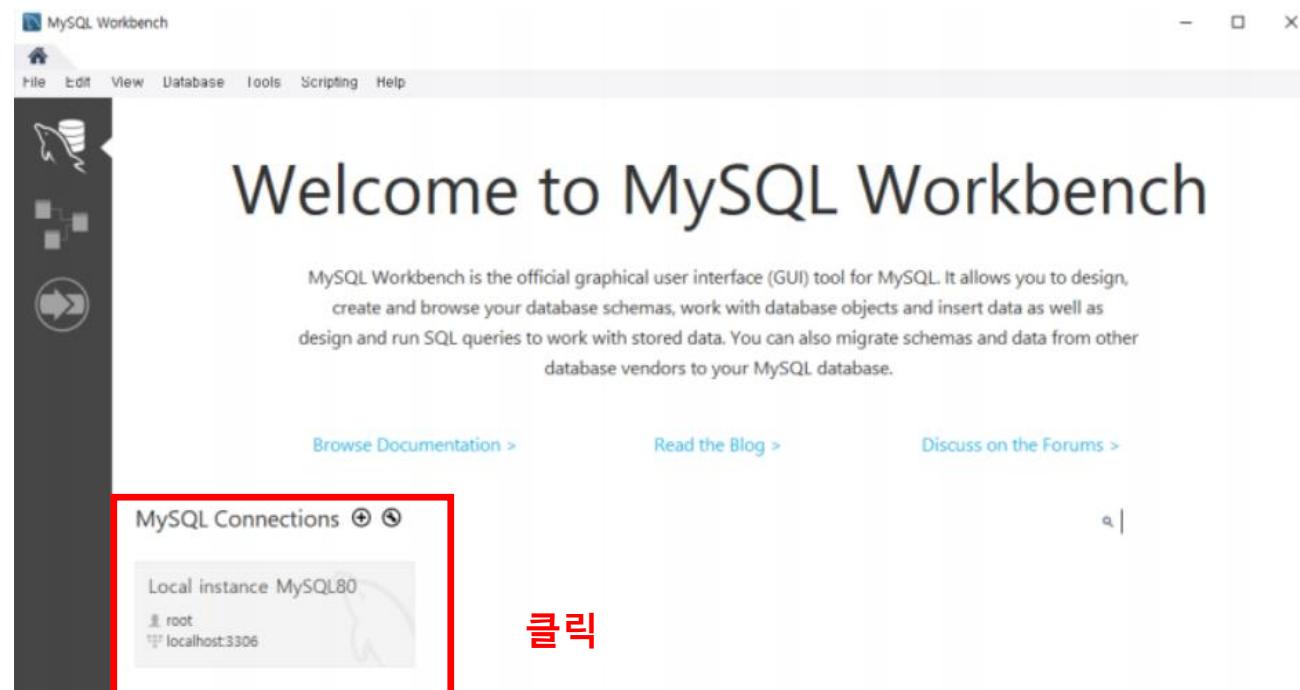
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

MySQL(Database) 설치

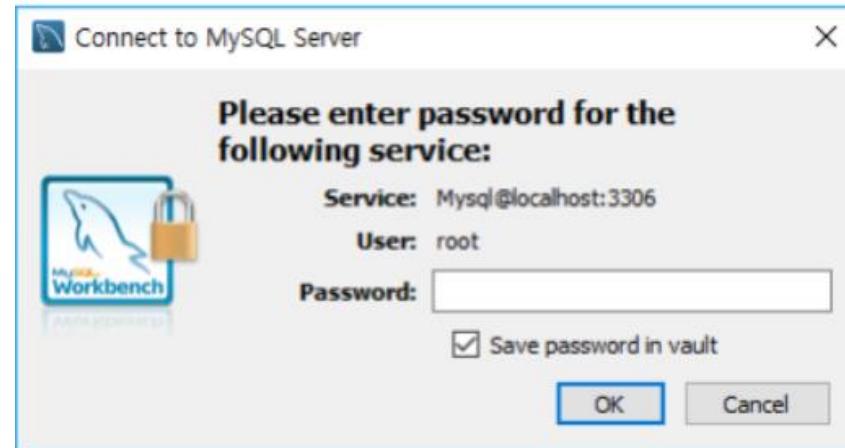
▪ MySQL Workbench

- 데이터베이스 설계 및 구현을 위한 GUI 프로그램
SQL 편집, 데이터베이스 모델링, 데이터베이스 관리, 데이터베이스 마이그레이션 등 기능 제공



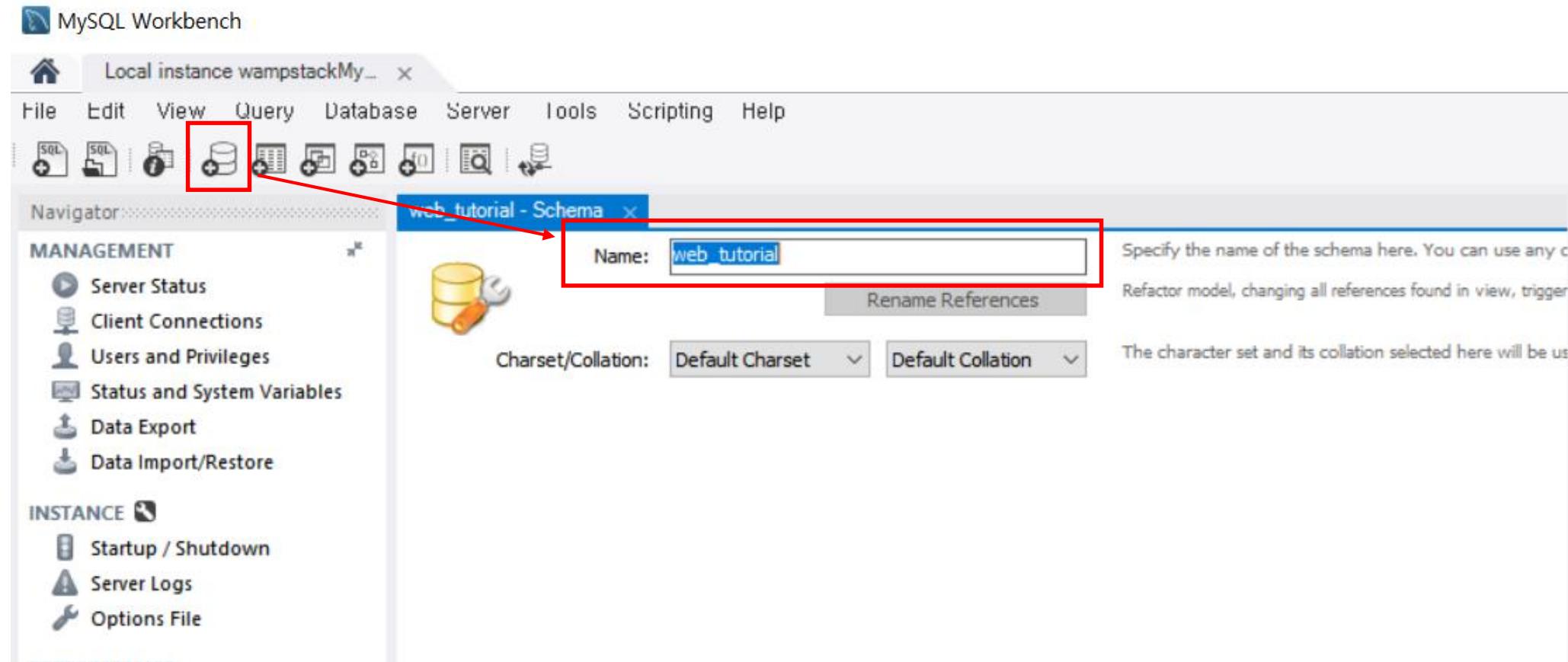
MySQL(Database) 설치

- MySQL Workbench 실행
 - 비밀번호를 입력하여 MySQL Server 접속

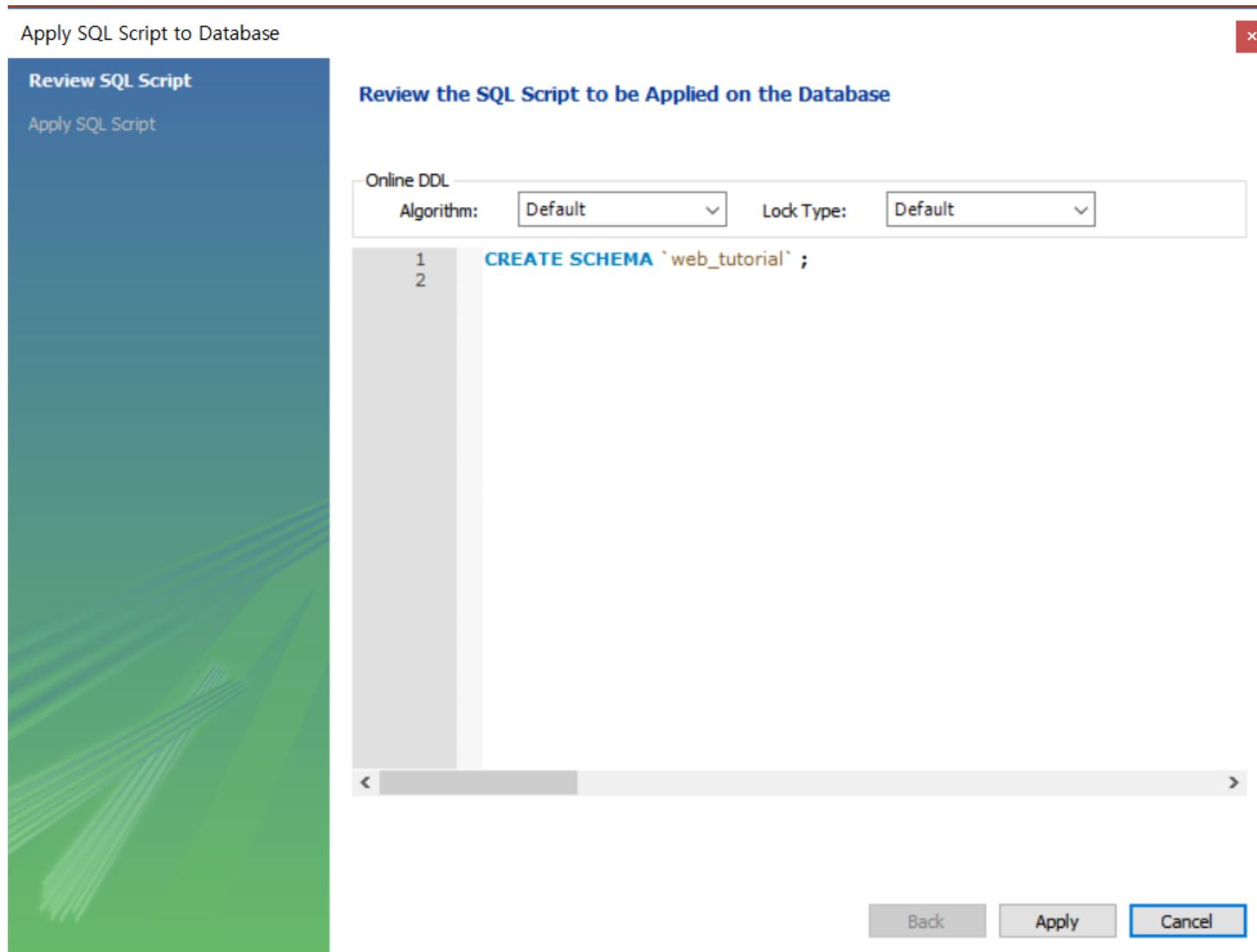


MySQL(Database) 설치

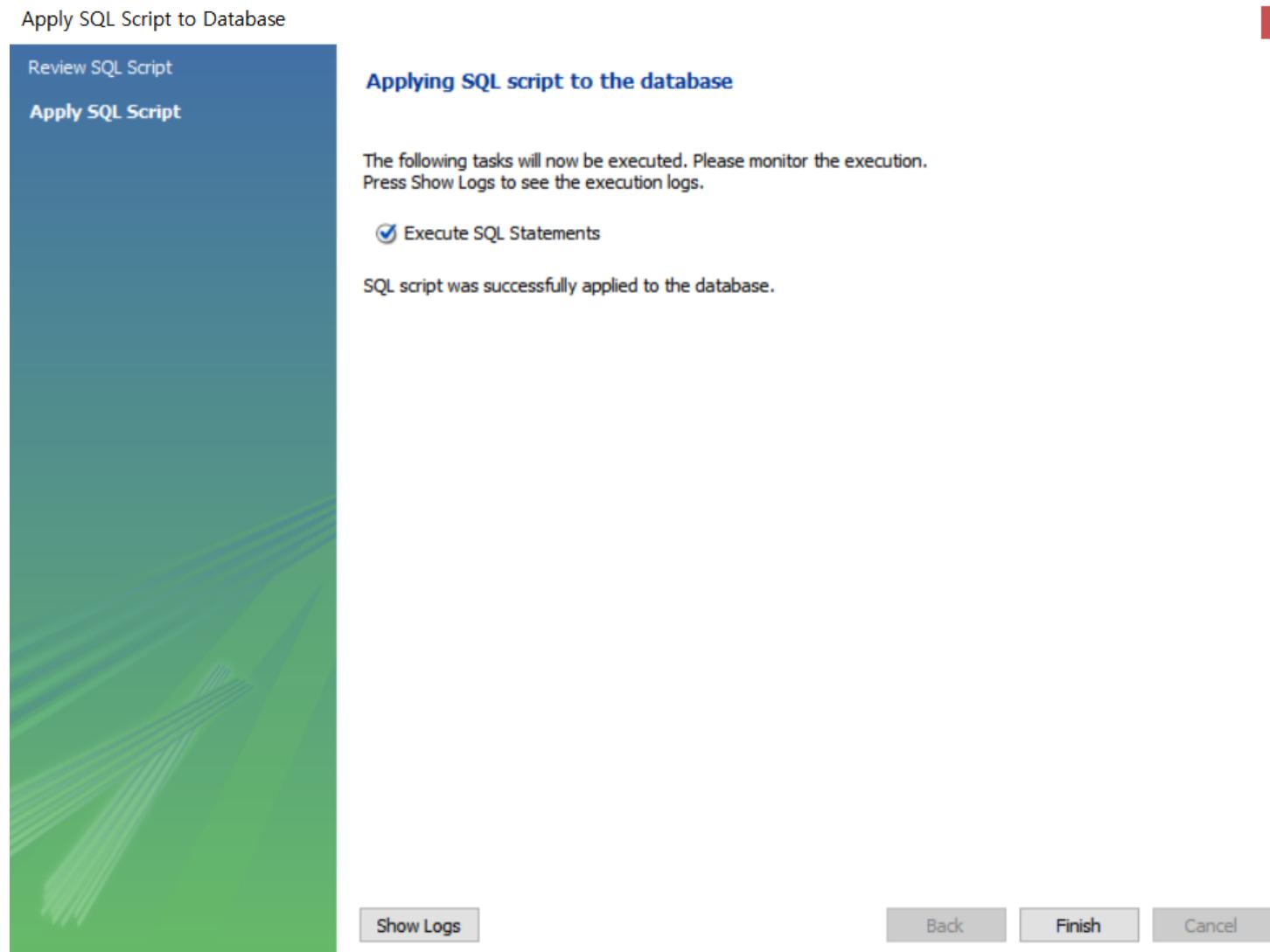
■ webTutorial Schema 생성



MySQL(Database) 설치

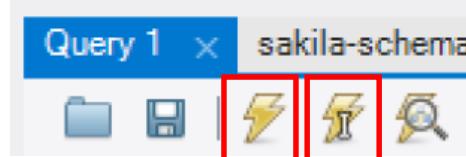


MySQL(Database) 설치



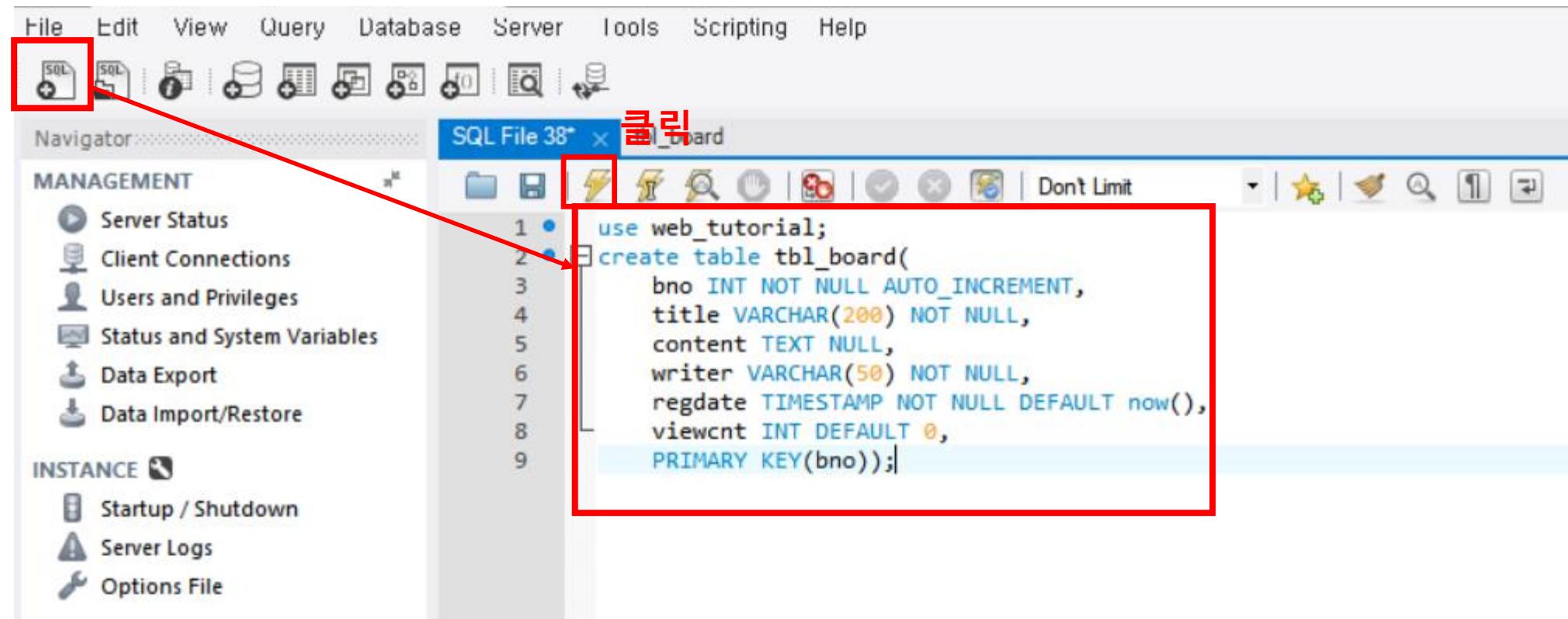
MySQL(Database) 테이블 생성

- query 실행을 위한 SQL문 작성
 - query란?
 - 한글 뜻으로 질의, 의문으로 데이터베이스에 정보를 요청하는 것
 - SQL은 Structured query language의 약자로 데이터베이스를 다루기 위한 언어
- SQL 편집기 사용법
 - 단일 명령문 실행
 - 실행하고자 하는 명령문으로 커서를 옮긴 다음 Ctrl + Enter키 또는 오른쪽 번개 클릭
 - 다중 명령문 실행
 - 실행하고자 하는 명령문들을 선택한 다음 Ctrl + Shift + Enter키 또는 왼쪽 번개 클릭



MySQL(Database) 테이블 생성

- tbl_board 테이블 생성 (쿼리문 사용법 MySQL 기본 참고)

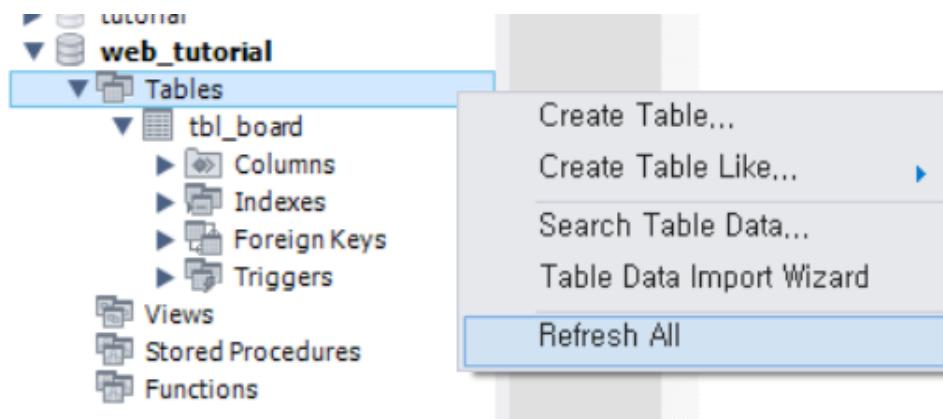


MySQL(Database) 테이블 생성

■ 결과 확인

Output			
Action Output			Message
#	Time	Action	Message
1	06:04:13	use webTutorial	0 row(s) affected
2	06:04:13	create table tbl_board(bno INT NOT NULL AUTO_INCREMENT, title VARCHAR(200) NOT NULL, content...)	0 row(s) affected

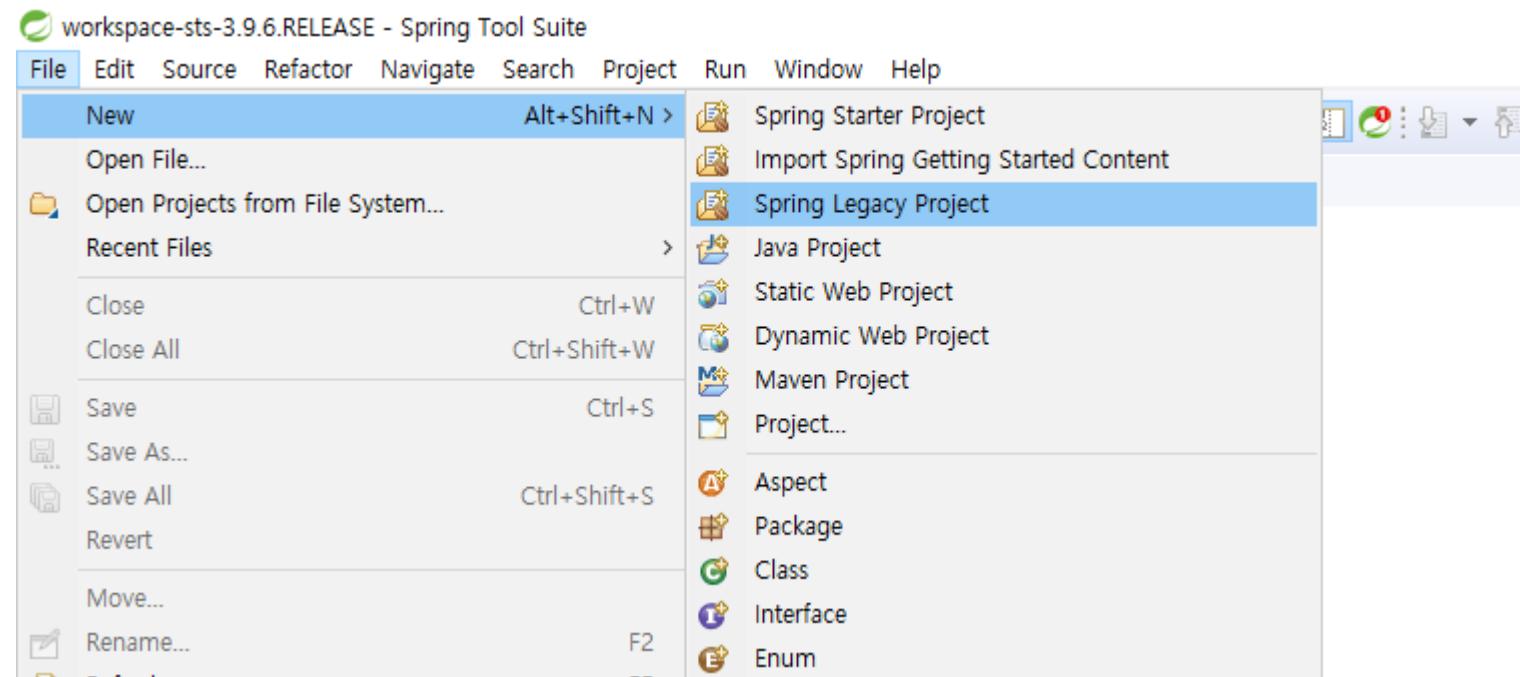
■ Tables 우클릭 -> Refresh All 하여, 생성된 테이블 확인



Spring 프로젝트 생성

■ 프로젝트 생성

- STS 실행 후 [File]->[New]->[Spring Legacy Project] 클릭



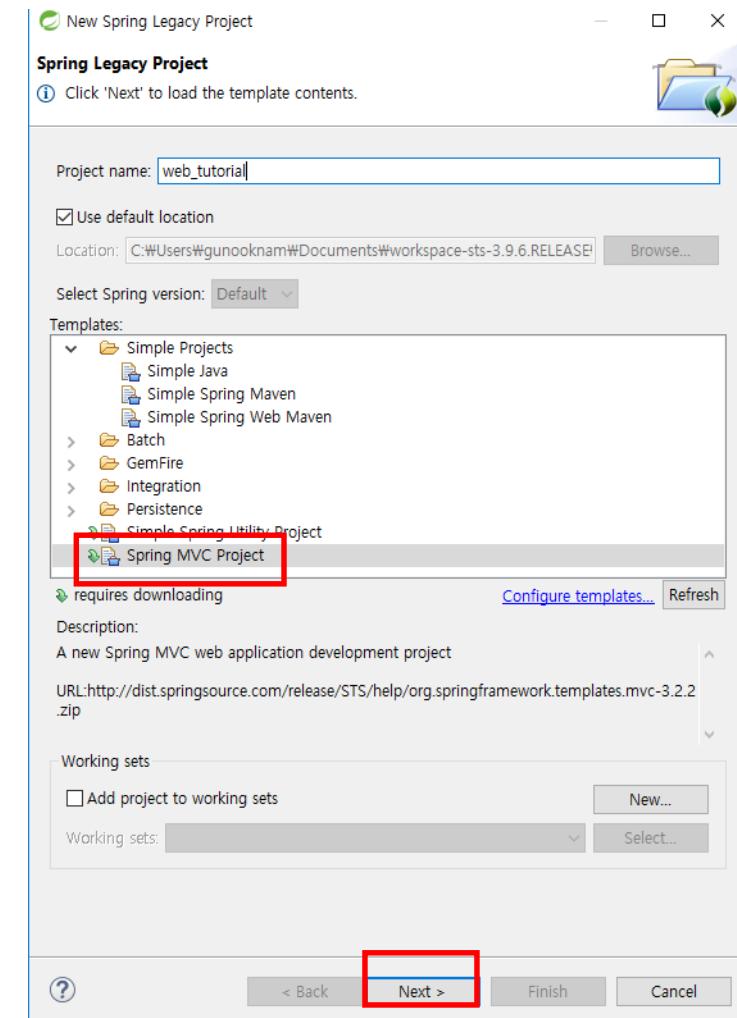
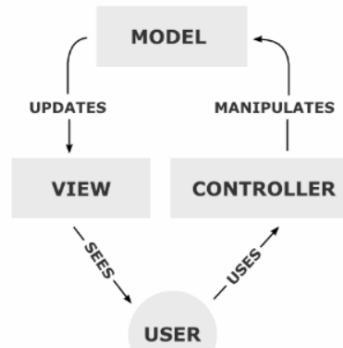
Spring 프로젝트 생성

■ 프로젝트 생성

- 프로젝트 이름 입력 "webTutorial"
- Spring **MVC Project** 선택



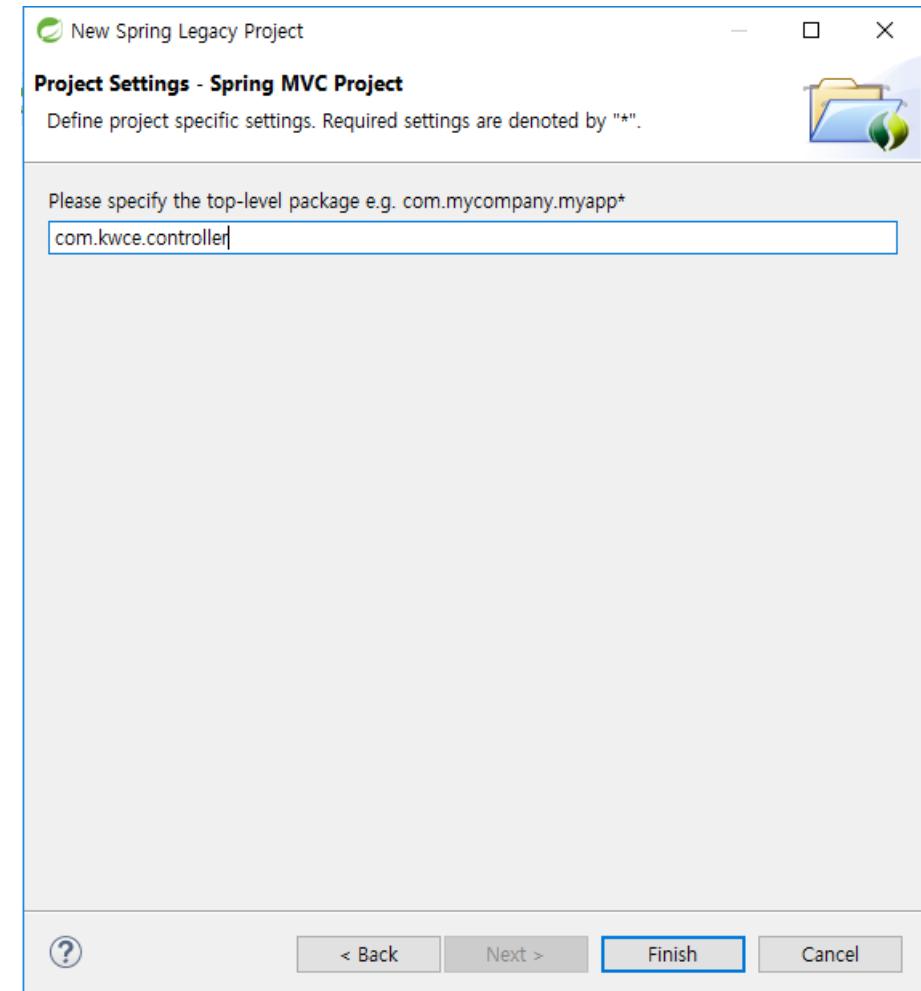
- 모델(Model) : 데이터를 처리하는 영역
- 뷰 (View) : 결과 화면을 만들어 내는 데 사용하는 자원
- 컨트롤러(Controller) : 웹의 요청을 처리하며 뷰와 모델의 중간 통신 역할



Spring 프로젝트 생성

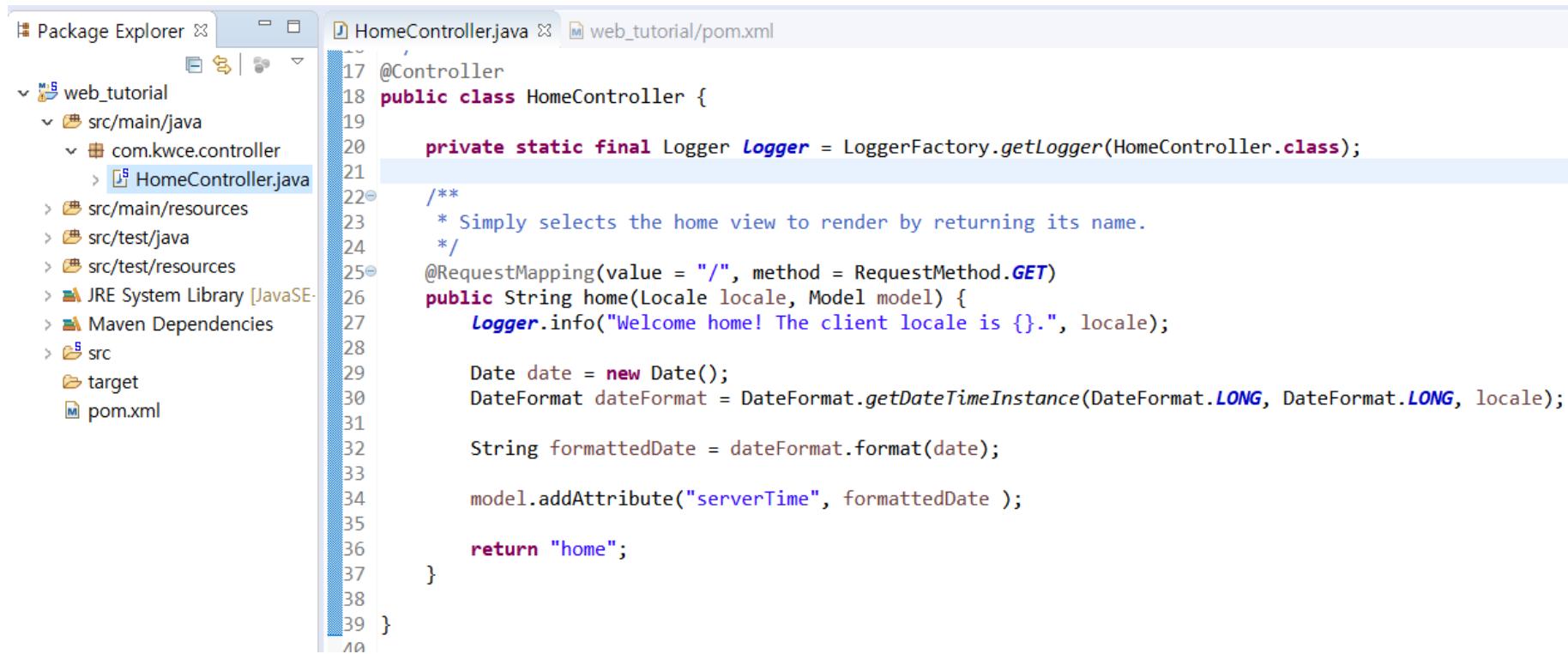
■ 프로젝트 생성

- Package 설정
- “com.kwce.controller” 입력
- Finish를 누르게 되면 프로젝트 생성과 동시에 **Maven**이 자동으로 인터넷에서 필요한 라이브러리를 다운로드(프로젝트 Build까지 약간의 시간이 소요)



Spring 프로젝트 생성

■ MVC 패턴을 위한 기본 틀 생성



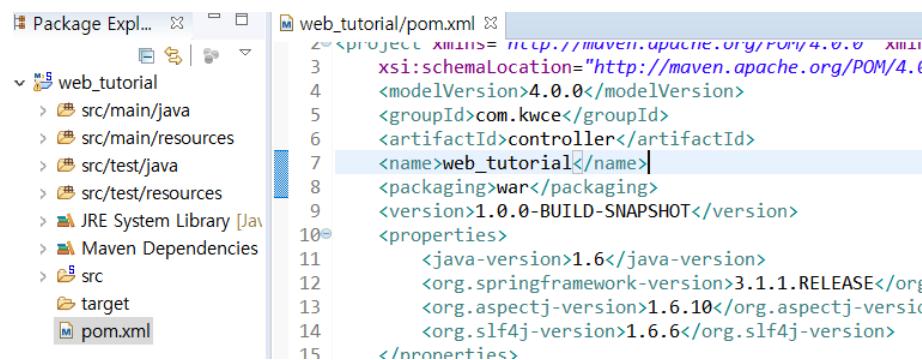
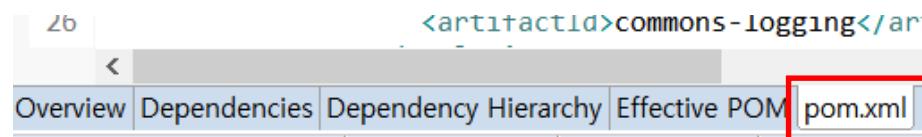
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure: a package named 'webTutorial' containing 'src/main/java/com.kwce.controller' which holds the 'HomeController.java' file. Other visible files include 'pom.xml' and various test and resource files. On the right, the code editor window shows the 'HomeController.java' code:

```
17 @Controller
18 public class HomeController {
19
20     private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
21
22     /**
23      * Simply selects the home view to render by returning its name.
24      */
25     @RequestMapping(value = "/", method = RequestMethod.GET)
26     public String home(Locale locale, Model model) {
27         logger.info("Welcome home! The client locale is {}.", locale);
28
29         Date date = new Date();
30         DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG, DateFormat.LONG, locale);
31
32         String formattedDate = dateFormat.format(date);
33
34         model.addAttribute("serverTime", formattedDate );
35
36         return "home";
37     }
38
39 }
```

Spring 프로젝트 생성

■ 프로젝트 설정 변경

- pom.xml 파일 수정
 - Java-version 1.8로 변경, springframework-version 4.3.8.RELEASE로 변경



```

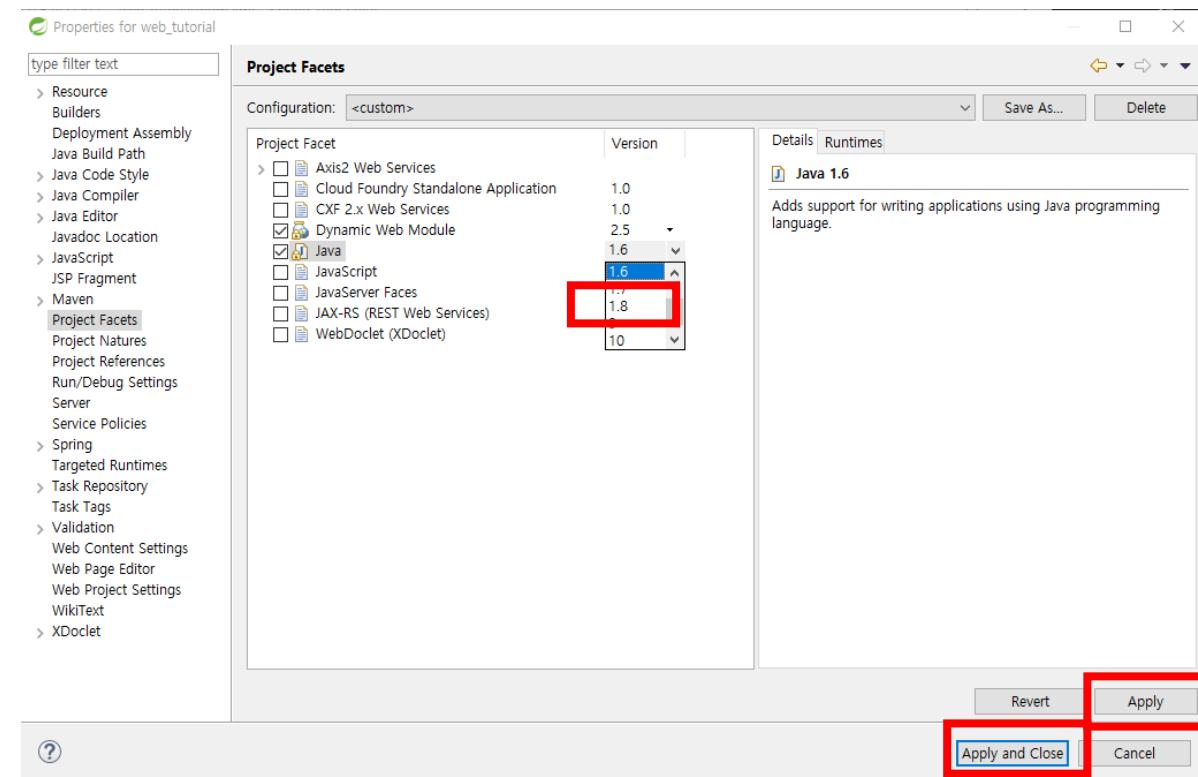
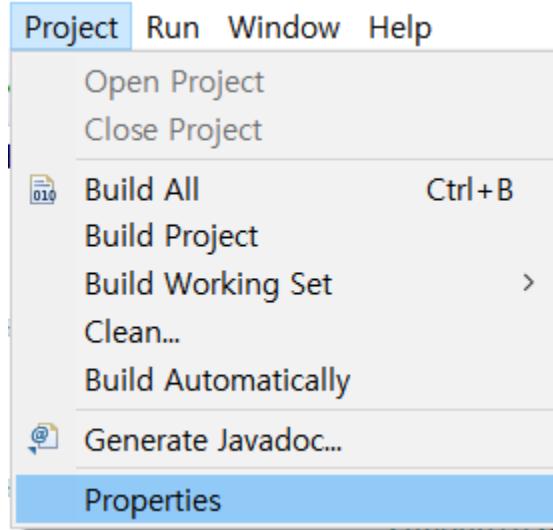
<version>1.0.0-BUILD-SNAPSHOT</version>
<properties>
  <java-version>1.8</java-version>
  <org.springframework-version>4.3.8.RELEASE</org.spring
  <org.aspectj-version>1.6.10</org.aspectj-version>
  <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
<dependencies>
  <!-- Spring -->

```

Spring 프로젝트 생성

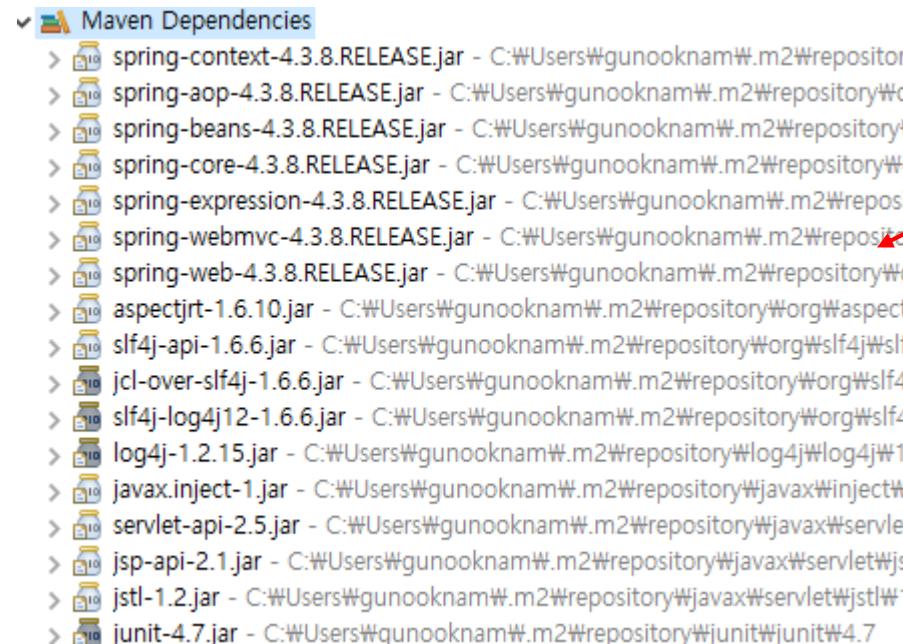
■ 프로젝트 설정 변경

- [project] -> [Properties] ->[Project Facets] 탭에서 Java version 1.8로 변경



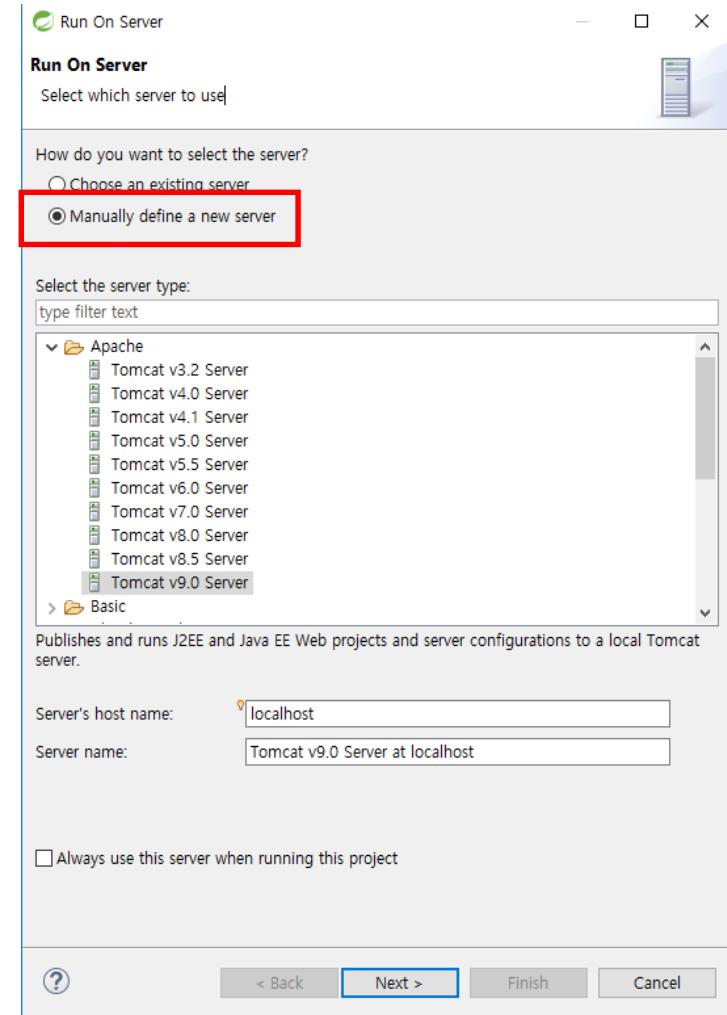
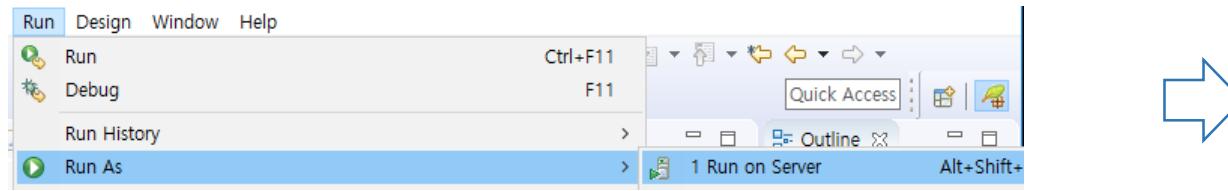
Spring 프로젝트 생성

■ MVC 프로젝트 구조 분석



Spring 프로젝트 실행

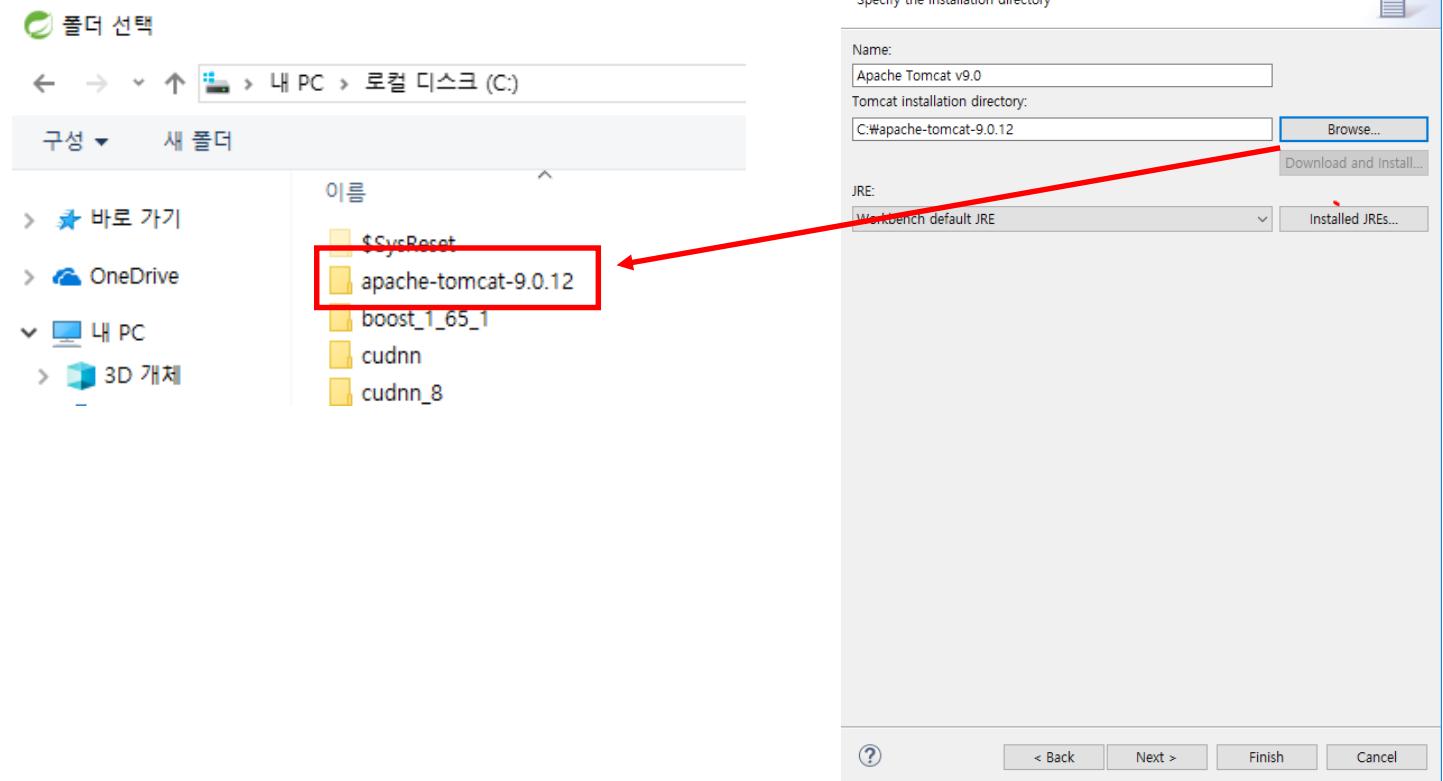
- [Run]->[Run as]->[Run on Server] 클릭



- Apache 폴더에서 이전에 다운 받은 Tomcat 9.0 선택

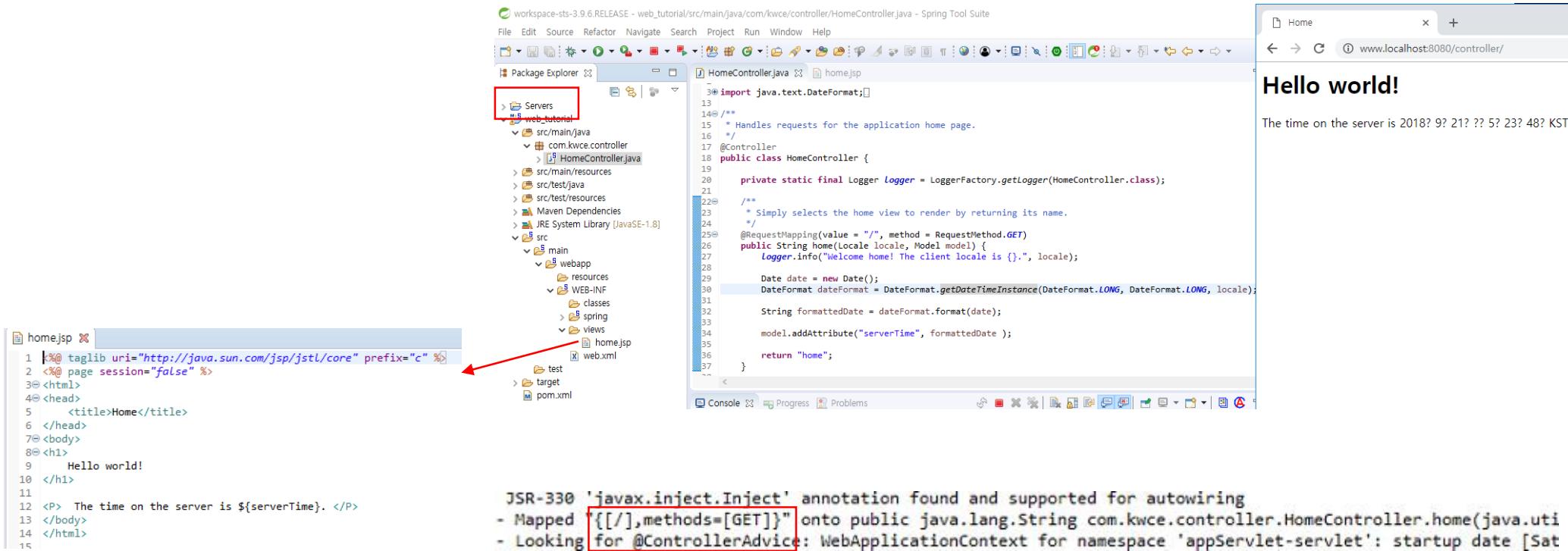
Spring 프로젝트 실행

- 이전에 압축 풀었던 C 드라이버 아래의 apache-tomcat-9.0.12 폴더를 Browse로 등록



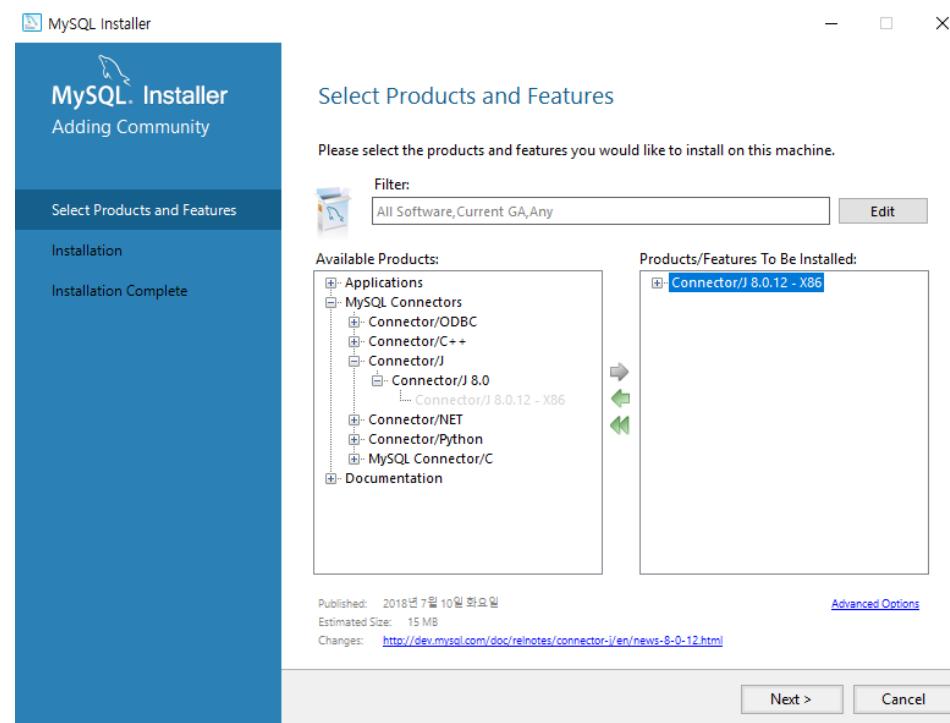
Spring 프로젝트 실행

- Finish 이후 자동으로 서버가 열리고 웹 브라우저로 www.localhost:8080/controller/에 접속 시 views 안에 home.jsp가 보여짐



DATABASE Test

- MySQL과 Spring의 연결을 테스트 하려면 JDBC 라이브러리가 필요
- MySQL에서는 “MySQL/Connector/J”라는 JDBC 라이브러리 필요
- MySQL Installer를 이용하여 “Connector/J” 추가설치



DATABASE Test

- Mavenrepository에서 MySQL Connector/J 검색
<http://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.12>
- 박스 친 부분을 복사해서 pom.xml 파일에 붙여넣기

The screenshot shows the Maven Repository interface for the MySQL Connector/J 8.0.12 artifact. It includes a search bar, a sidebar with popular categories, and detailed information about the artifact like license, categories, and usage statistics. A red box highlights the XML code snippet for the dependency:

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.12</version>
</dependency>
```

The screenshot shows an IDE displaying a Java web application project structure. The pom.xml file is open, showing the dependency section highlighted with a red box. An arrow points from the highlighted code in the pom.xml to the corresponding code on the Maven repository page.

```

110     </dependency>
111
112     <!-- Test -->
113     <dependency>
114         <groupId>junit</groupId>
115         <artifactId>junit</artifactId>
116         <version>4.7</version>
117         <scope>test</scope>
118     </dependency>
119
120     <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
121     <dependency>
122         <groupId>mysql</groupId>
123         <artifactId>mysql-connector-java</artifactId>
124         <version>8.0.12</version>
125     </dependency>
126
127
128 </dependencies>

```

pom.xml 파일에 추가된
dependency를 확인하고
Maven이 필요한 라이브러리를
바로 다운로드

DATABASE Test

- 이후에 Maven Dependencies 안에 mysql-connector-java-8.0.12.jar이 추가된 것 확인



The screenshot shows the Maven Dependencies tree on the left and the corresponding Maven POM XML code on the right. The dependency 'mysql-connector-java-8.0.12.jar' is highlighted with a red box in both the tree and the code.

```
<dependency>
    <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>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java-8.0.12 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.12</version>
</dependency>
```

DATABASE Test

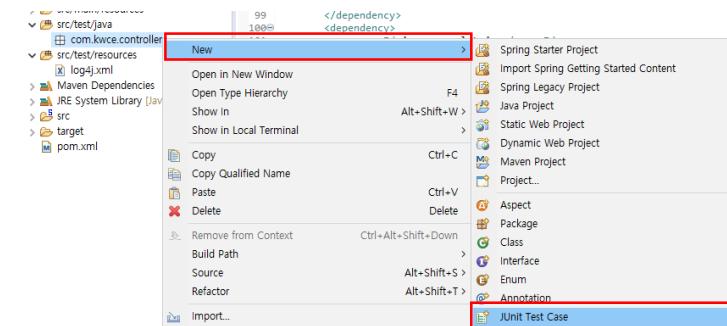
- pom.xml 파일에서 기존의 JUnit의 버전 변경
- Spring의 테스트에서는 높은 버전의 JUnit 필요

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.7</version>
    <scope>test</scope>
</dependency>
```



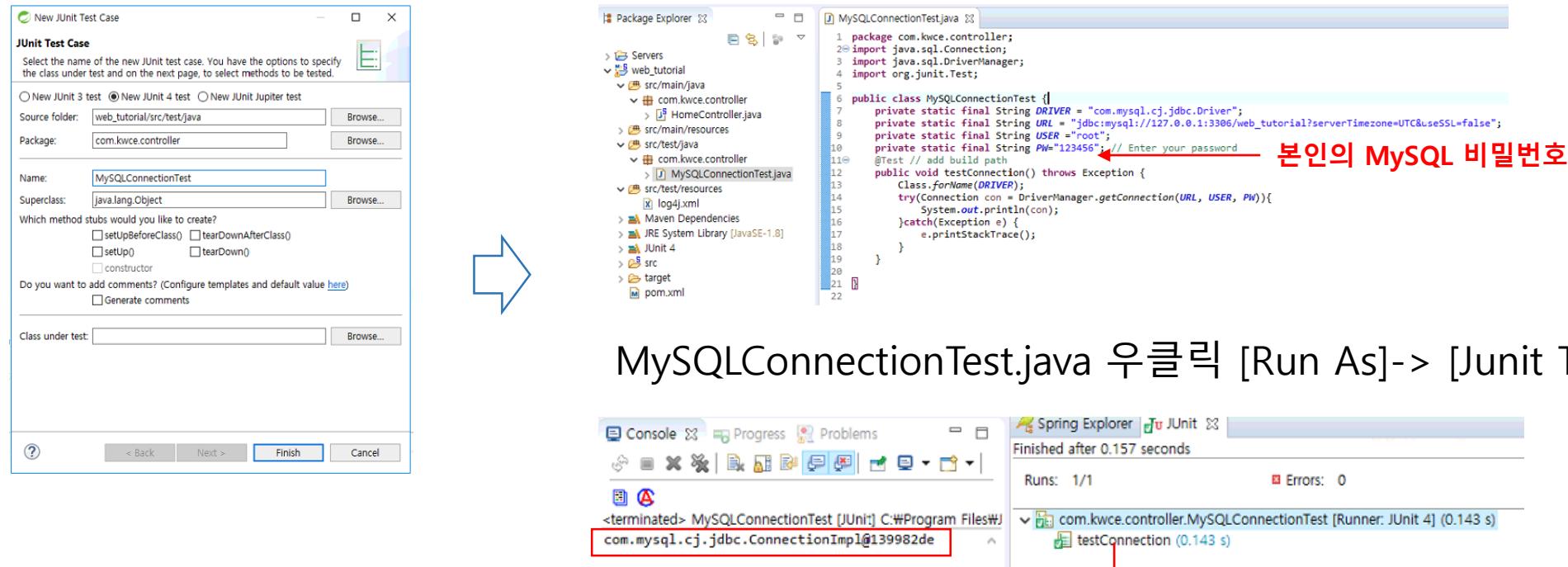
```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

- src/test/java 폴더 아래 com.kwce.controller package에 JUnit Test Code 작성

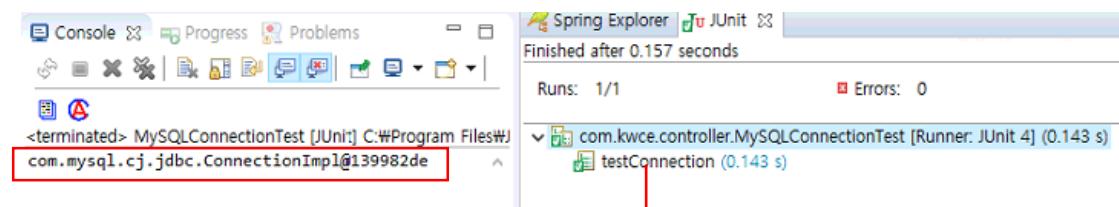


DATABASE Test

- 테스트를 하려는 Method 위에 @Test 를 쓰고 Junit으로 테스트를 진행
- MySQL과 연결이 잘 되었는지 확인



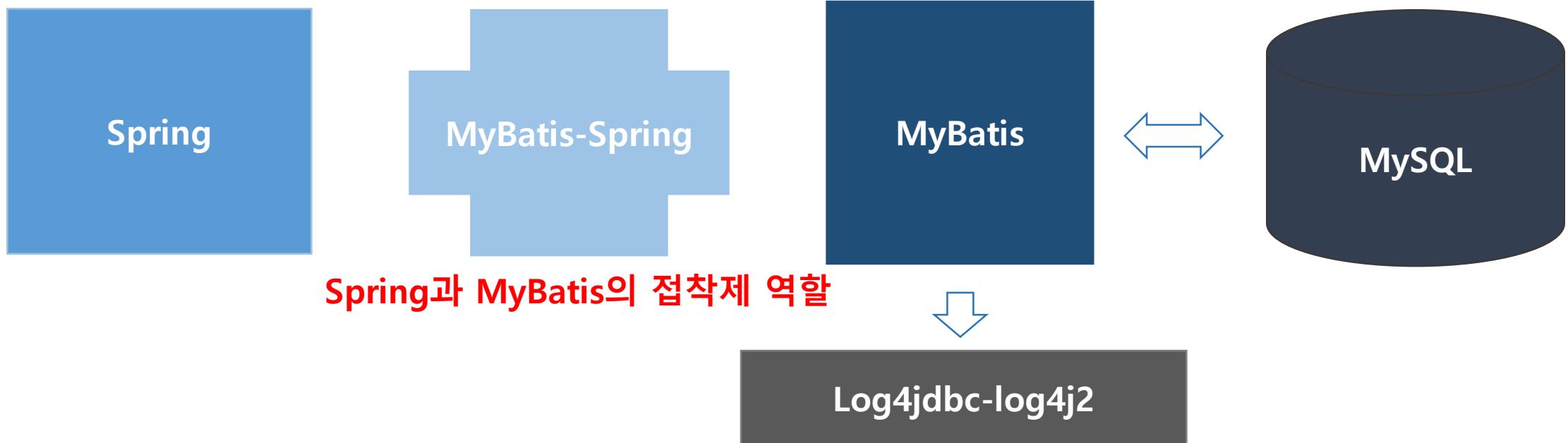
MySQLConnectionTest.java 우클릭 [Run As]-> [Junit Test] 테스트



Connection 객체 출력, Test 결과 : 성공

DATABASE Test

- Spring-jdbc, Spring-test, MyBatis, mybatis-spring 추가
- MyBatis의 로그를 보기위한 Log4jdbc 모듈 추가 (동작하는 SQL문 확인)



- Mavenrepository에서 위의 모듈들 검색 <http://mvnrepository.com/>

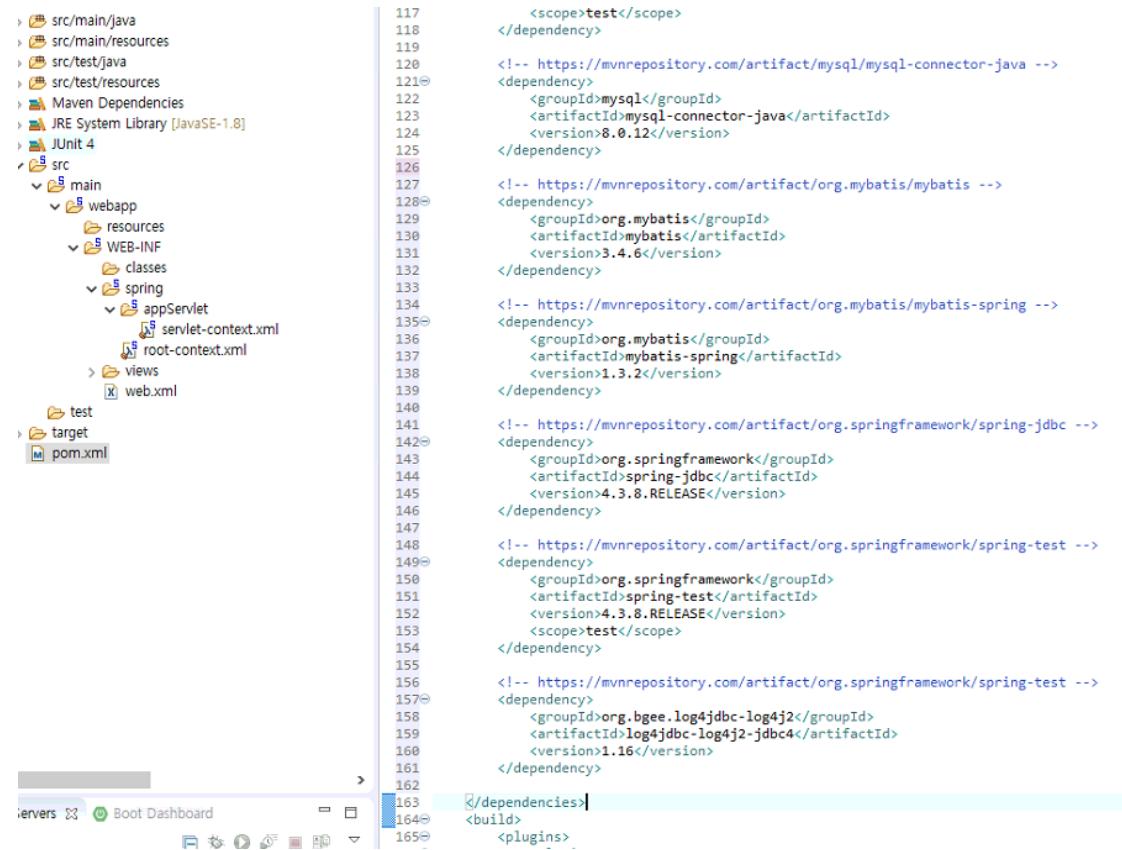


DATABASE Test

모듈	pom.xml에 추가할 dependency
MyBatis	<pre data-bbox="724 313 1512 501"><!-- https://mvnrepository.com/artifact/org.mybatis/mybatis --> <dependency> <groupId>org.mybatis</groupId> <artifactId>mybatis</artifactId> <version>3.4.6</version> </dependency></pre>
MyBatis-Spring	<pre data-bbox="724 519 1589 707"><!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring --> <dependency> <groupId>org.mybatis</groupId> <artifactId>mybatis-spring</artifactId> <version>1.3.2</version> </dependency></pre>
Spring-jdbc (스프링 프레임워크와 동일 버전 사용)	<pre data-bbox="724 740 1658 927"><!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc --> <dependency> <groupId>org.springframework</groupId> <artifactId>spring-jdbc</artifactId> <version>4.3.8.RELEASE</version> </dependency></pre> <p data-bbox="1172 894 2263 927">Property로 지정했던 \${org.springframework-version} 으로 대체 가능</p>
Spring-test (스프링 프레임워크와 동일 버전 사용)	<pre data-bbox="724 941 1658 1157"><!-- https://mvnrepository.com/artifact/org.springframework/spring-test --> <dependency> <groupId>org.springframework</groupId> <artifactId>spring-test</artifactId> <version>4.3.8.RELEASE</version> <scope>test</scope> </dependency></pre>
Log4jdbc-log4j2	<pre data-bbox="724 1186 1838 1373"><!-- https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4 --> <dependency> <groupId>org.bgee.log4jdbc-log4j2</groupId> <artifactId>log4jdbc-log4j2-jdbc4</artifactId> <version>1.16</version> </dependency></pre>

DATABASE Test

- pom.xml 파일에 모듈들 추가

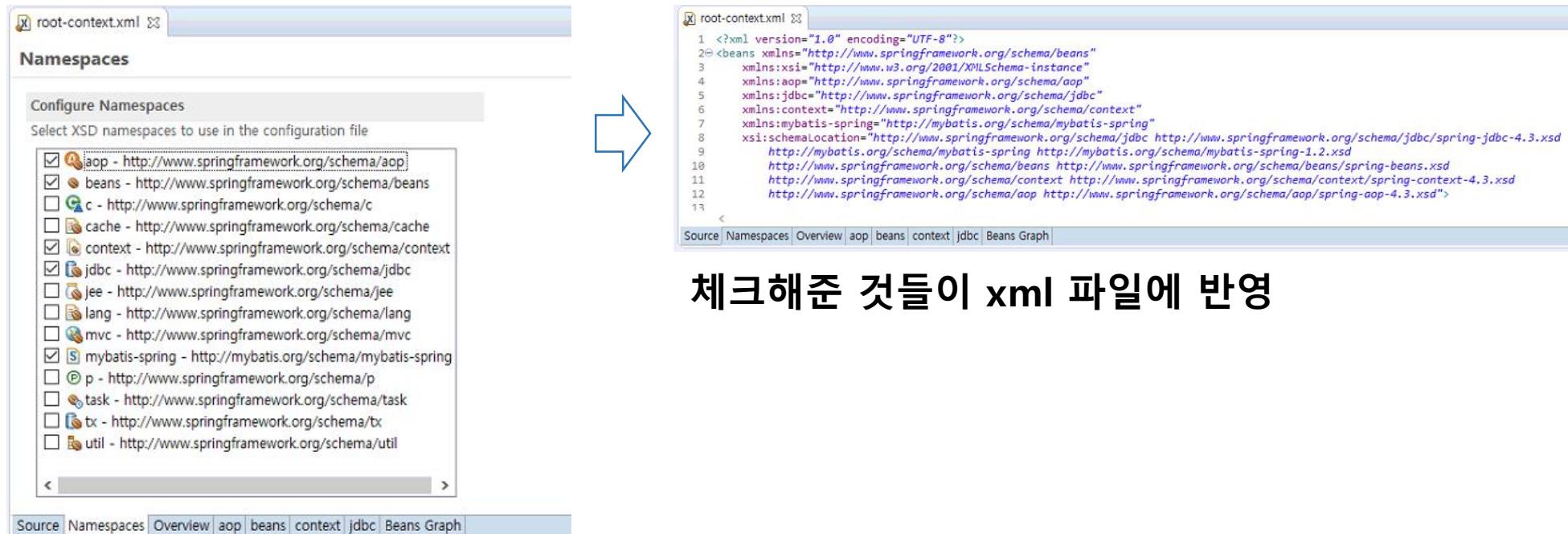


The screenshot shows the Eclipse IDE interface with the 'Servers' view open at the bottom. The project structure on the left includes 'src/main/java', 'src/main/resources', 'src/test/java', 'src/test/resources', 'Maven Dependencies', 'JRE System Library [JavaSE-1.8]', 'JUnit 4', 'src/main/webapp', 'src/main/webapp/resources', 'src/main/webapp/WEB-INF', 'src/main/webapp/WEB-INF/classes', 'src/main/webapp/WEB-INF/spring', 'src/main/webapp/WEB-INF/spring/appServlet', 'src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml', 'src/main/webapp/WEB-INF/spring/root-context.xml', 'src/main/webapp/WEB-INF/views', 'src/main/webapp/WEB-INF/web.xml', 'src/test', 'target', and 'pom.xml'. The 'pom.xml' file is open in the editor, displaying the following XML code:

```
<dependency>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.12</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>4.3.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>4.3.8.RELEASE</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
    <version>1.16</version>
</dependency>
</dependencies>
<build>
<plugins>
```

DATABASE Test

- Spring Project에서 root-context.xml 파일 수정
 - 프로젝트 내에 src/main/webapp/WEB-INF/spring/root-context.xml
 - Namespaces 탭에서 aop, context, jdbc, mybatis-spring 체크



체크해준 것들이 xml 파일에 반영

DATABASE Test

■ Spring Project에서 root-context.xml 파일 수정

```

<?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:aop="http://www.springframework.org/schema/aop"
       xmlns:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
       xsi:schemaLocation="http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd
                           http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.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-4.3.xsd
                           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"/>
        <property name="url" value="jdbc:log4jdbc:mysql://127.0.0.1:3306/webTutorial?useSSL=false&serverTimezone=UTC"/>
        <property name="username" value="root"/>
        <property name="password" value="123456"/>
    </bean>                                mysql 비밀번호
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">          데이터베이스와의 Connection 생성과 SQL 실행에
        <property name="dataSource" ref="dataSource" />          관한 Bean을 sqlSessionFactory라 지정
    </bean>
    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">          데이터베이스와 연결 후 작업이 완료되면 close()를 해야 하는데 이를
        <destroy-method="clearCache">          처리하는 작업을 SqlSessionTemplate라는 클래스가 제공
        <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
</beans>

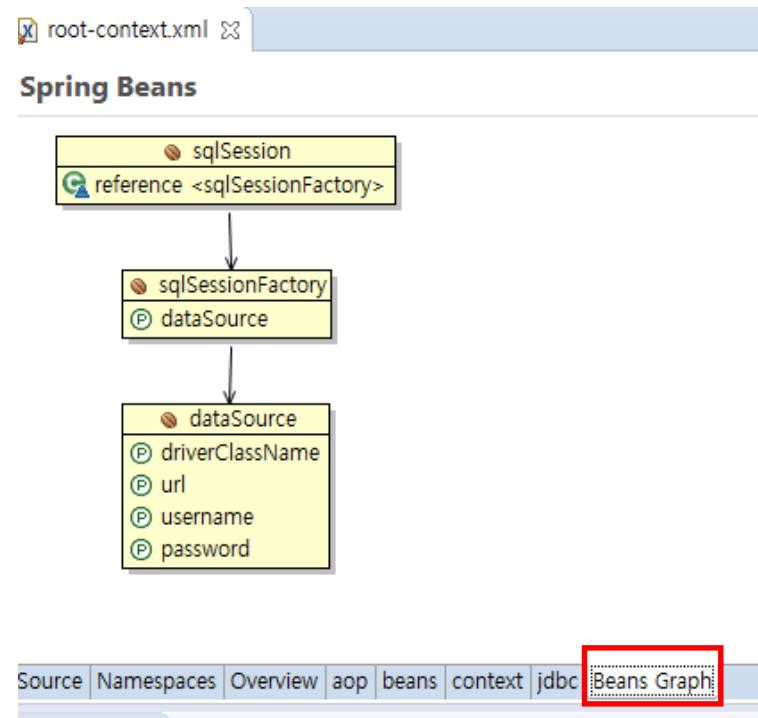
```

데이터베이스와의 Connection 생성과 SQL 실행에
관한 Bean을 sqlSessionFactory라 지정

데이터베이스와 연결 후 작업이 완료되면 close()를 해야 하는데 이를
처리하는 작업을 SqlSessionTemplate라는 클래스가 제공

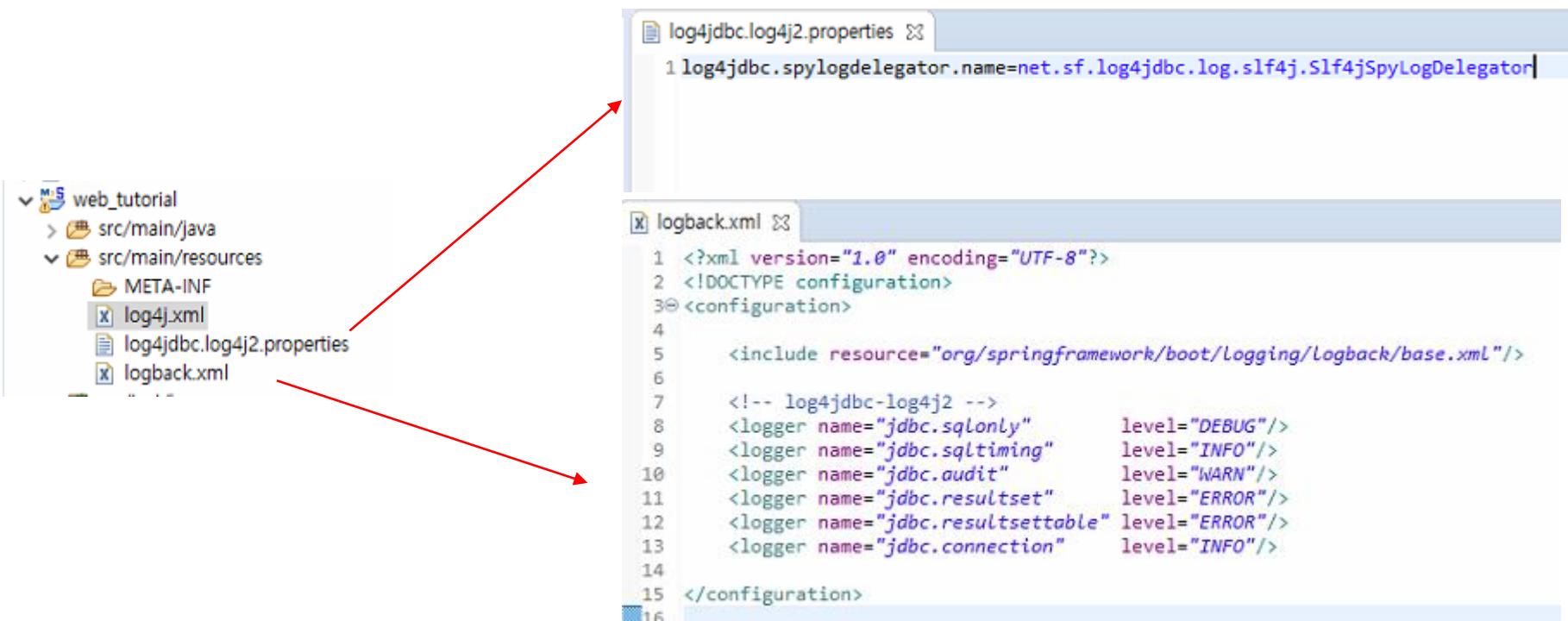
DATABASE Test

- root-context.xml 파일의 Beans Graph 탭에서 등록한 Beans에 대한 연관 관계 확인



DATABASE Test

- log4jdbc-log4j2 동작을 위해 별도의 로그 관련 설정 파일 필요
- /src/main/resources에 log4jdbc.log4j2.properties, logback.xml을 추가



DATABASE Test

- Spring-test 모듈을 사용하기 전에 pom.xml 파일에서 Servlet 버전 변경 (Tomcat의 버전에 따라 Servlet 버전도 맞춰야 함)

```
<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
```



```
<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId> javax.servlet-api</artifactId>
    <version>4.0.0</version>
    <scope>provided</scope>
</dependency>
```

DATABASE Test

■ 로그 + MySQL와의 연결 테스트

Spring-test 모듈 사용으로 WAS동작 없이 Test 가능

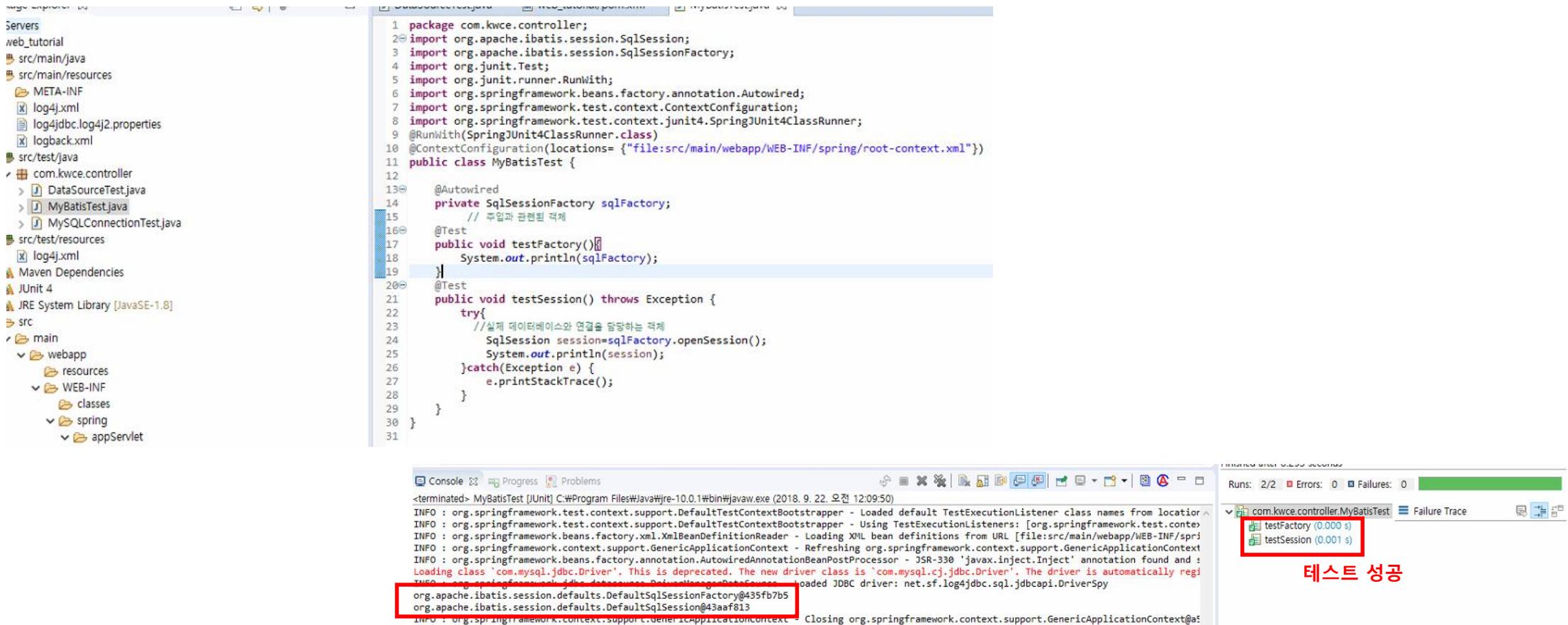
현재 테스트 코드 실행 시 스프링이 로딩되도록 하는 설정

The screenshot shows the Eclipse IDE interface with several open windows:

- Package Explorer:** Shows the project structure with packages like `src/main/java` and `src/test/java`, and files like `DataSourceTest.java`, `MyBatisTest.java`, and `MySQLConnectionTest.java`.
- DataSourceTest.java:** The code for the test class. It includes imports for `com.kwce.controller`, `java.sql.Connection`, `javax.sql.DataSource`, `org.junit.Test`, `org.junit.runner.RunWith`, `org.springframework.beans.factory.annotation.Autowired`, `org.springframework.test.context.ContextConfiguration`, and `org.springframework.test.context.junit4.SpringJUnit4ClassRunner`. The `@RunWith(SpringJUnit4ClassRunner.class)` and `@ContextConfiguration(locations = {"file:src/main/webapp/WEB-INF/spring/root-context.xml"})` annotations are highlighted with red boxes. The test method `test()` uses `@Test` and `@Autowired` to inject a `DataSource` named `ds`. It attempts to get a connection from `ds`, prints it to `System.out`, and catches any exceptions.
- Console:** Shows the terminal output of the test run. It includes log messages from Spring framework classes like `DefaultTestContextBootstrapper` and `XmlBeanDefinitionReader`, and specific JDBC driver logs. A line of text at the bottom is highlighted with a red box: `net.sf.log4jdbc.sql.jdbcapi.ConnectionSpy@649725e3`.
- Spring Explorer:** Shows the results of the test run. It indicates "Finished after 0.417 seconds" and lists "Runs: 1/1", "Errors: 0", and "Failures: 0".
- JUnit:** Shows the test results for `com.kwce.controller.DataSourceT` with a "Failure Trace" link.

DATABASE Test

■ MyBatis의 연결 테스트



The screenshot shows an IDE interface with several panes:

- Servers:** Lists the project structure, including `src/main/java`, `src/main/resources`, `src/test/java`, and `src/test/resources`.
- Code Editor:** Displays the `MyBatisTest.java` file with the following code:

```

1 package com.kwce.controller;
2 import org.apache.ibatis.session.SqlSession;
3 import org.apache.ibatis.session.SqlSessionFactory;
4 import org.junit.Test;
5 import org.junit.runner.RunWith;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.test.context.ContextConfiguration;
8 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
9 @RunWith(SpringJUnit4ClassRunner.class)
10 @ContextConfiguration(locations= {"file:src/main/webapp/WEB-INF/spring/root-context.xml"})
11 public class MyBatisTest {
12
13     @Autowired
14     private SqlSessionFactory sqlFactory;
15     // 주입과 관련된 라이브러리
16     @Test
17     public void testFactory() {
18         System.out.println(sqlFactory);
19     }
20     @Test
21     public void testSession() throws Exception {
22         try {
23             // 실제 데이터베이스와 연결을 담당하는 객체
24             SqlSession session=sqlFactory.openSession();
25             System.out.println(session);
26         }catch(Exception e) {
27             e.printStackTrace();
28         }
29     }
30 }

```

- Console:** Shows the terminal output of the test execution.
- Run:** Shows the test results: 2/2 runs, 0 errors, 0 failures.
- Output:** Shows the results of the tests: `testFactory (0.000 s)` and `testSession (0.001 s)`.

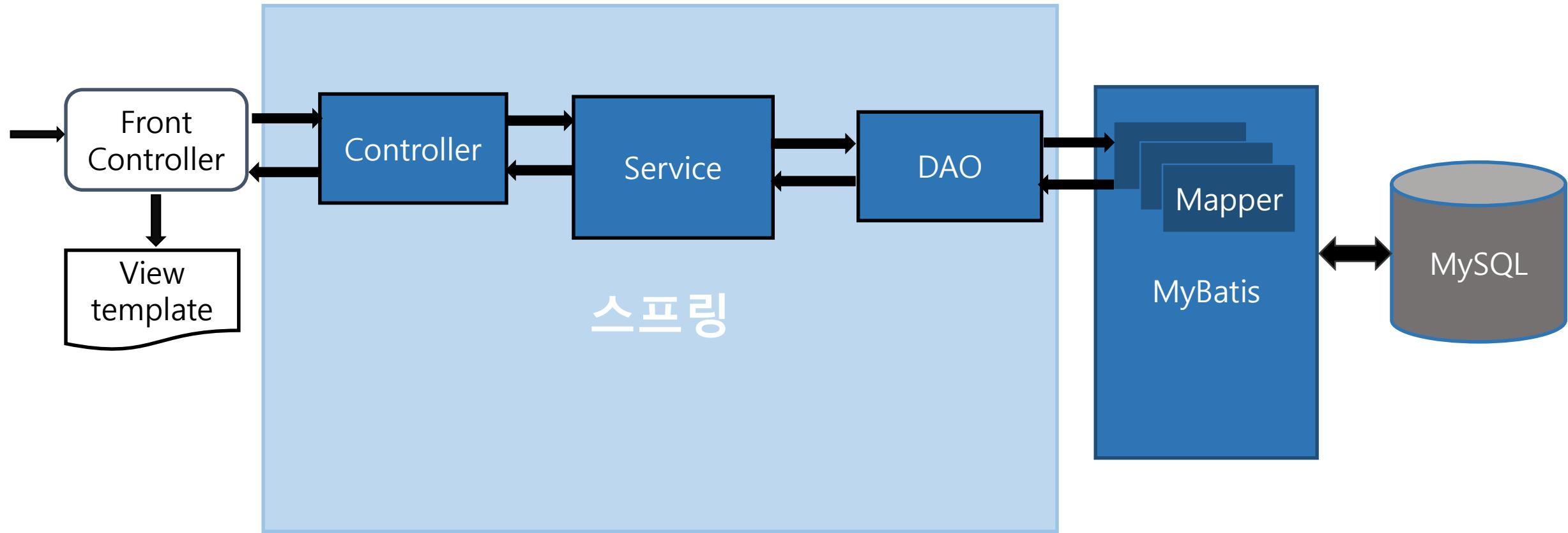
A red box highlights the output of the `testSession()` method, which includes the JDBC driver registration message.

테스트 성공

2. 스프링 MVC

스프링 MVC

- 애플리케이션에 적용할 MVC 패턴의 흐름



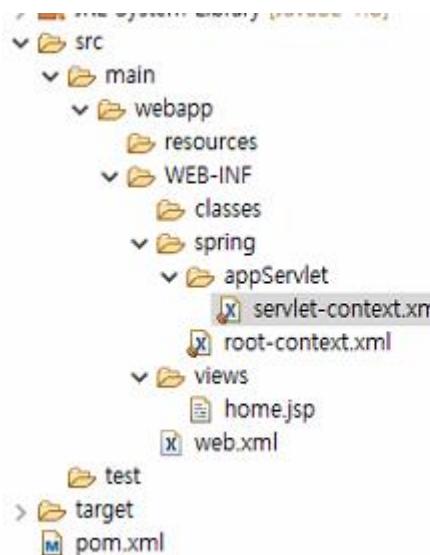
스프링 MVC

■ 스프링과 개발자가 처리해야 할 작업

스프링 MVC가 처리해 주는 작업	개발자가 직접 해야 하는 작업
URI를 분석해서 적절한 컨트롤러를 찾는 작업	특정 URI에 동작하는 컨트롤러를 설계하는 작업
컨트롤러에 필요한 메소드를 호출하는 작업	서비스 객체의 생성
컨트롤러의 결과 데이터를 뷰로 전달하는 작업	DAO 객체의 생성
적절한 뷰를 찾는 작업	컨트롤러 내에 원하는 결과를 메소드로 설계 뷰에서 전달받은 데이터의 출력

스프링 MVC

■ Spring Project의 servlet-context.xml 파일



```

servlet-context.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans:beans xmlns="http://www.springframework.org/schema/mvc"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:beans="http://www.springframework.org/schema/beans"
5   xmlns:context="http://www.springframework.org/schema/context"
6   xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
7     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">
9
10  <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
11  <!-- Enables the Spring MVC @Controller programming model -->
12  <annotation-driven />
13
14  <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${webappRoot}/resource
15  <resources mapping="/resources/**" location="/resources/" />
16
17  <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
18  <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
19    <beans:property name="prefix" value="/WEB-INF/views/" />
20    <beans:property name="suffix" value=".jsp" />
21  </beans:bean>
22
23
24  <context:component-scan base-package="com.kwce.controller" />
25
26
27
28 </beans:beans>

```

클래스 선언 위
애노테이션을 이용해서
컨트롤러 작성 가능

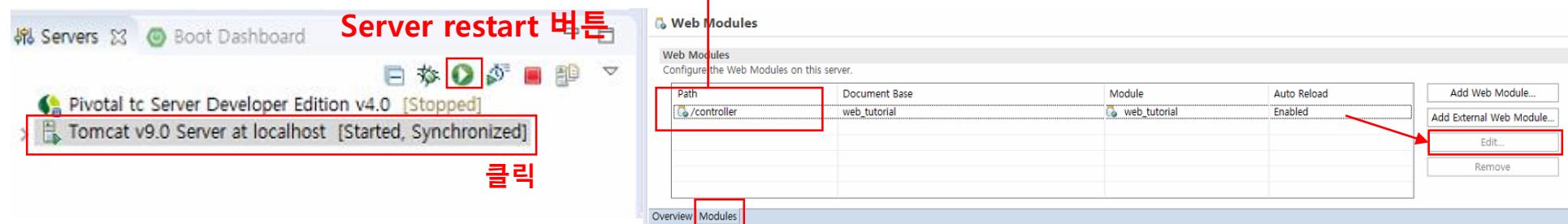
View를 어떻게 처리하겠는지 정의

컨트롤러가 /WEB-INF/views/이름.jsp
이라는 경로에서 view 파일을 가져옴

base-package 속성값에
해당되는 패키지 내부 클래스를 조사

스프링 MVC

■ URI 구조



Tomcat 을 클릭하고 modules 탭에서
컨텍스트루트 위치를 변경 가능

스프링 MVC

■ Domain Package 생성

- VO(Value Object : 데이터 그 자체로 의미 있는 것들을 담은 객체)를 담는 패키지로 사용
- 다음 예제를 위해 ExampleVO 정의

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under the 'web_tutorial' project. It includes packages like 'com.kwce.controller' and 'com.kwce.domain', and resources like 'log4j.xml' and 'logback.xml'. On the right, the code editor window shows the 'ExampleVO.java' file with the following code:

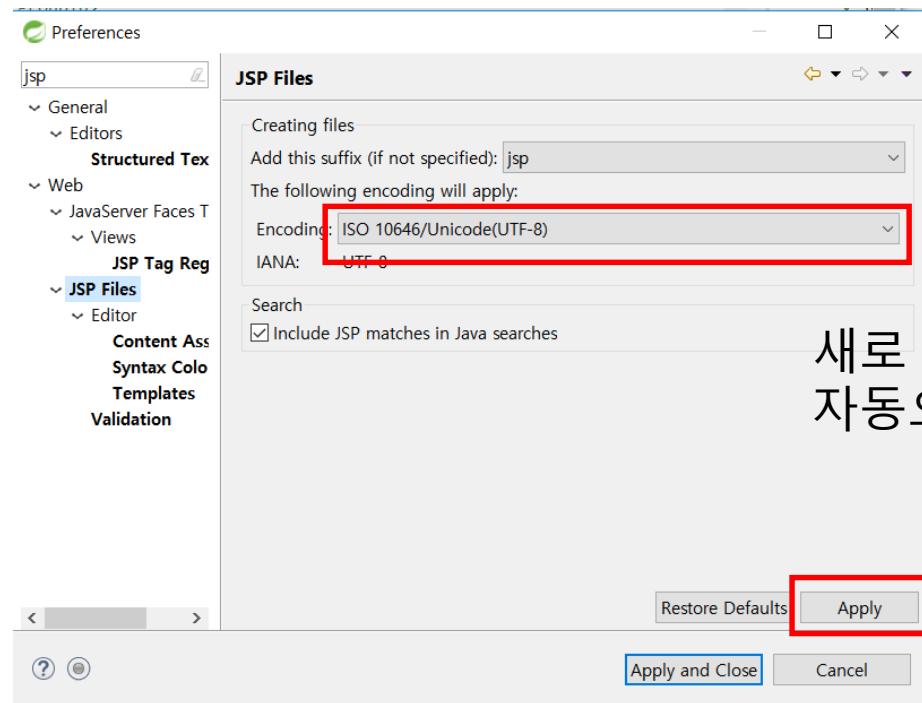
```
1 package com.kwce.domain;
2
3 public class ExampleVO {
4     private String a;
5     private int b;
6     public ExampleVO(String a,int b){
7         this.a=a;
8         this.b=b;
9     }
10    public String getA() {
11        return a;
12    }
13    public void setA(String a) {
14        this.a = a;
15    }
16    public int getB() {
17        return b;
18    }
19    public void setB(int b) {
20        this.b = b;
21    }
22 }
```

At the bottom, the Servers view shows a Pivotal tc Server Developer Edition v4.0 instance stopped and a Tomcat v9.0 Server at localhost started/restarted.

스프링 MVC

■ jsp 파일 Encoding 변경

- [Window] -> [Preferences] -> jsp 검색하여 jsp Files 탭에서 Encoding UTF-8로 변경

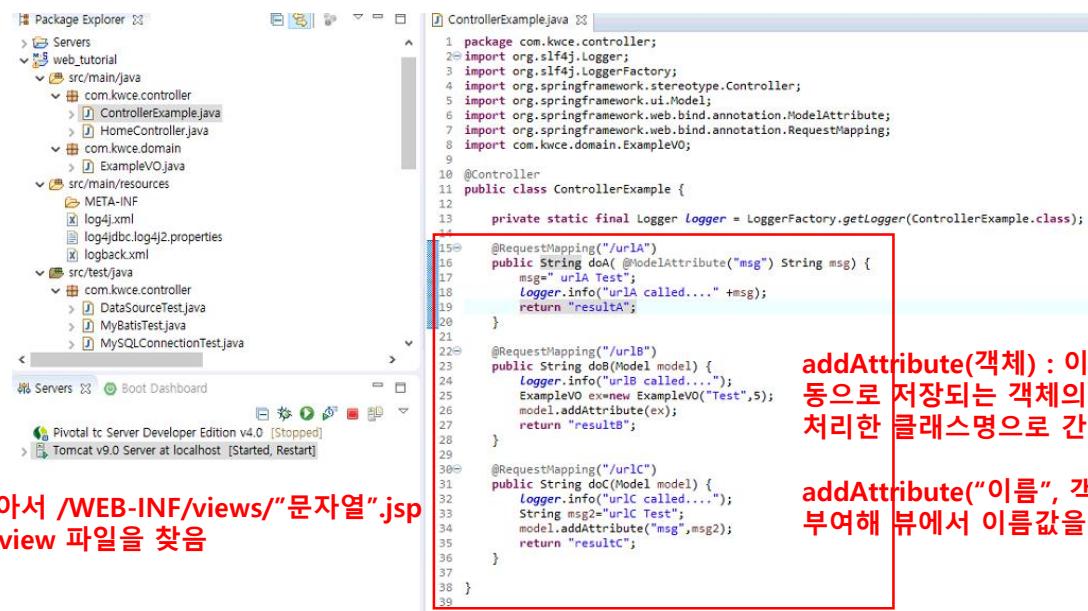


새로 생성되는 JSP 파일마다 UTF-8로
자동으로 생성되게 하는 설정

스프링 MVC

■ Controller 처리의 Example

- @RequestMapping("요청URL명") : 요청URL에서 해당 메소드를 처리
- @ModelAttribute("이름") 파라미터 : 요청 시 파라미터에 이름을 붙여서 처리해서 view에 전달
- Model 객체의 addAttribute 메서드로 객체를 view로 전달 가능



```

1 package com.kwce.controller;
2 import org.slf4j.Logger;
3 import org.slf4j.LoggerFactory;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.ModelAttribute;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import com.kwce.domain.ExampleVO;
9
10 @Controller
11 public class ControllerExample {
12
13     private static final Logger logger = LoggerFactory.getLogger(ControllerExample.class);
14
15     @RequestMapping("/urlA")
16     public String doA( @ModelAttribute("msg") String msg) {
17         msg=" urlA Test";
18         logger.info("urlA called..." +msg);
19         return "resultA";
20     }
21
22     @RequestMapping("/urlB")
23     public String doB(Model model) {
24         logger.info("urlB called...");
25         ExampleVO ex=new ExampleVO("Test",5);
26         model.addAttribute(ex);
27         return "resultB";
28     }
29
30     @RequestMapping("/urlC")
31     public String doC(Model model) {
32         logger.info("urlC called...");
33         String msg2="urlC Test";
34         model.addAttribute("msg",msg2);
35         return "resultC";
36     }
37
38 }
39

```

addAttribute(객체) : 이름을 지정하지 않는 경우 자동으로 저장되는 객체의 클래스 첫 글자를 소문자로 처리한 클래스명으로 간주

addAttribute("이름", 객체) : 객체에 특별한 이름을 부여해 뷰에서 이름값을 이용하여 객체 처리

String을 리턴 받아서 /WEB-INF/views/"문자열".jsp
이라는 경로에서 view 파일을 찾음

스프링 MVC

■ Controller 처리 Example

- src/main/webapp/WEB-INF/views 폴더에 resultA.jsp, resultB.jsp, resultC.jsp 작성
- Tomcat server restart 후 open된 URL 확인

컨트롤러에 딱 붙여짐



The screenshot shows the Tomcat logs with several INFO messages from org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping. These messages show mappings for various URLs like '/urlC', '/urlB', and '/urlA' to specific controller methods. A red box highlights the first message mapping '/urlC'.

Views Folder Structure:

```

views
  home.jsp
  resultA.jsp
  resultB.jsp
  resultC.jsp
  
```

Code Examples:

- resultA.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>
<span>ResultA ${msg}</span>
</body>
</html>
  
```

- resultB.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>
<span>${exampleVO.a}</span>
<span>${exampleVO.b}</span>
</body>
</html>
  
```

- resultC.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>
<span>${msg}</span>
</body>
</html>
  
```

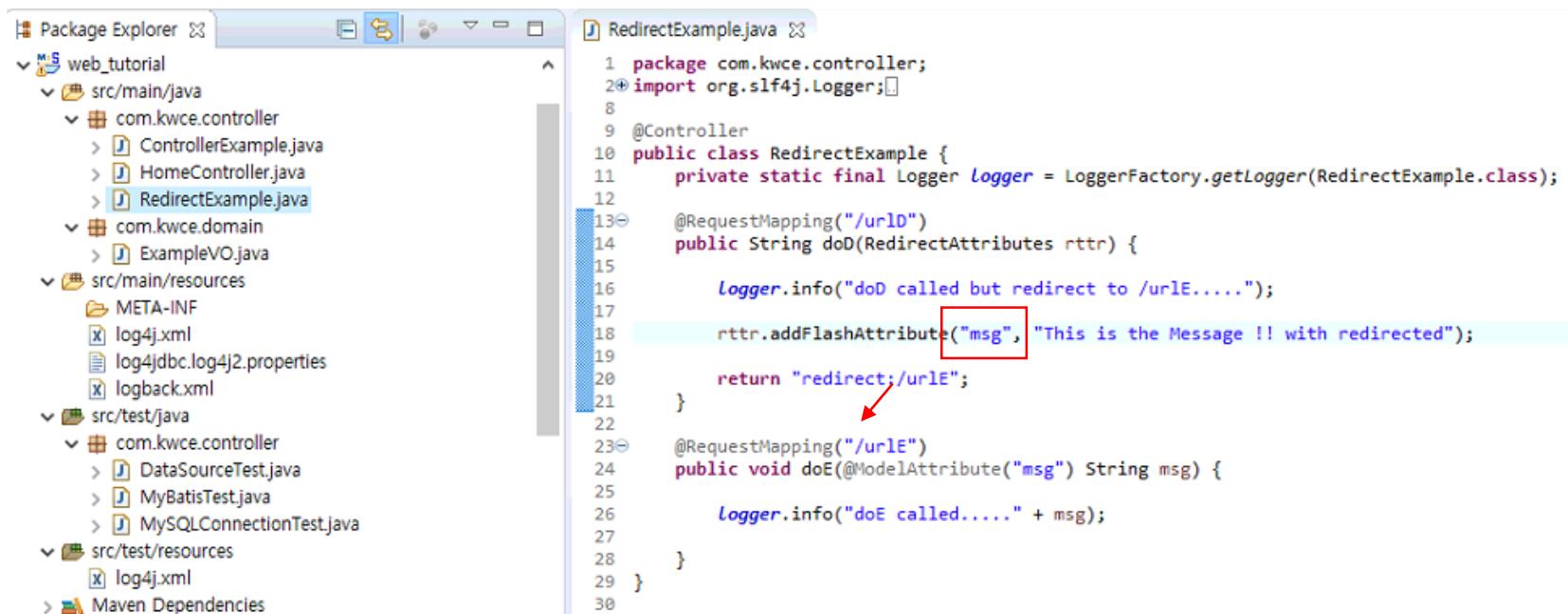
Browsing Results:

 - ResultA:** Shows the URL www.localhost:8080/controller/urlA in the browser address bar.
 - Test 5:** Shows the URL www.localhost:8080/controller/urlB in the browser address bar.
 - urlC Test:** Shows the URL www.localhost:8080/controller/urlC in the browser address bar.

스프링 MVC

■ 리다이렉트 Example

- 특정 경로로 호출했으나 다른 경로로 돌려 보내는 것
- RedirectAttributes 객체를 사용하면 리다이렉트 시점에 원하는 데이터를 추가로 전달



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under 'web_tutorial'. It includes packages like 'com.kwce.controller' containing 'ControllerExample.java', 'HomeController.java', and 'RedirectExample.java'; 'com.kwce.domain' containing 'ExampleVO.java'; and various configuration files like 'log4j.xml', 'log4jdbc.log4j2.properties', and 'logback.xml'. On the right, the code editor window shows the 'RedirectExample.java' file. The code defines a controller class 'RedirectExample' with two methods: 'doD' and 'doE'. The 'doD' method uses 'RedirectAttributes' to add a flash attribute 'msg' with the value 'This is the Message !! with redirected' and then performs a redirect. The 'doE' method handles the received 'msg' attribute from the model. A red box highlights the line 'rttr.addFlashAttribute("msg", "This is the Message !! with redirected");', and a red arrow points to the 'msg' parameter in the 'doE' method's parameter list.

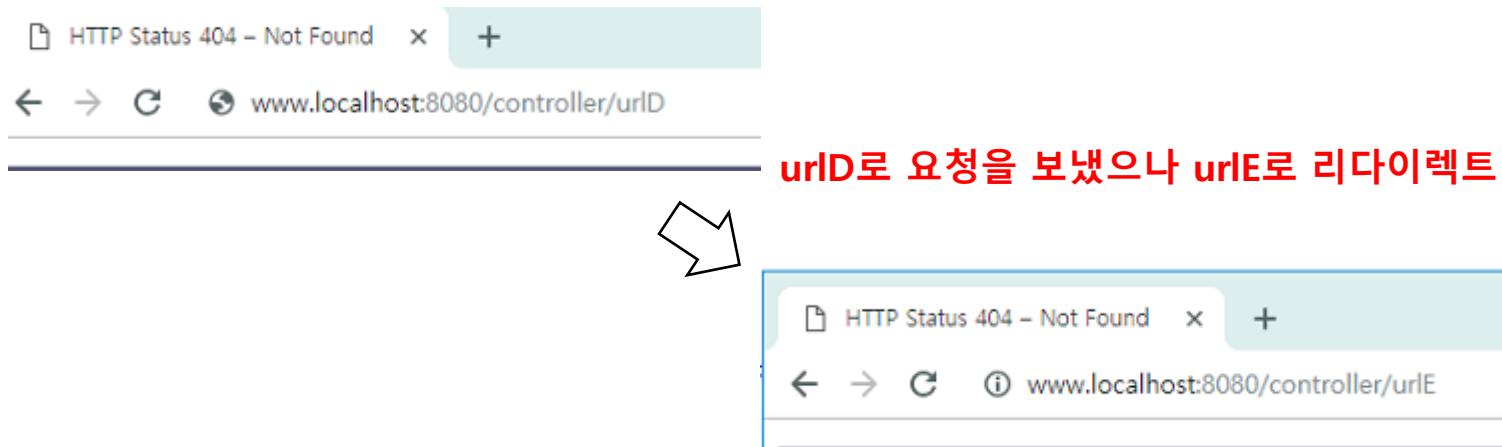
```

1 package com.kwce.controller;
2 import org.slf4j.Logger;
3
4 @Controller
5 public class RedirectExample {
6     private static final Logger logger = LoggerFactory.getLogger(RedirectExample.class);
7
8     @RequestMapping("/urlD")
9     public String doD(RedirectAttributes rttr) {
10
11         logger.info("doD called but redirect to /urlE.....");
12
13         rttr.addFlashAttribute("msg", "This is the Message !! with redirected");
14
15         return "redirect:/urlE";
16     }
17
18     @RequestMapping("/urlE")
19     public void doE(@ModelAttribute("msg") String msg) {
20
21         logger.info("doE called....." + msg);
22
23     }
24 }
25
26
27
28
29
30

```

스프링 MVC

■ 리다이렉트 Example 결과



```

INFO : org.springframework.web.servlet.method.annotation.RequestMappingHandlerAdapter - Looking for
INFO : org.springframework.web.servlet.handler.SimpleUrlHandlerMapping - Mapped URL path [/resources/**]
INFO : org.springframework.web.servlet.DispatcherServlet - FrameworkServlet 'appServlet': initialization
9월 22, 2018 5:21:36 오후 org.apache.catalina.core.StandardContext reload
정보: Reloading Context with name [/controller] is completed
INFO : com.kwce.controller.RedirectExample - doD called but redirect to /urlE.....
INFO : com.kwce.controller.RedirectExample - doE called.....This is the Message !! with redirected
  
```

doD에서의 msg가
doE에서 출력

스프링 MVC

■ GET 방식과 POST 방식

- @RequestMapping에 method 인자를 추가해서 사용가능
- ControllerExample.java 파일에 "/urlF"를 GET과 POST로 처리하는 함수 추가
- views 폴더 아래 GetandPost.jsp 파일을 다음과 같이 작성하여 추가

ControllerExample.java

```

27     logger.info("urlB called....");
28     ExampleVO ex=new ExampleVO("Test",5);
29     model.addAttribute(ex);
30     return "resultB";
31
32
33@RequestMapping("/urlC")
34 public String doc(Model model) {
35     logger.info("urlC called....");
36     String msg2="urlC Test";
37     model.addAttribute("msg",msg2);
38     return "resultC";
39 }
40
41@RequestMapping(value = "/urlF", method = RequestMethod.GET)
42 public String get(Model model) {
43     logger.info("urlF get called....");
44     return "GetandPost";
45 }
46
47@RequestMapping(value = "/urlF", method = RequestMethod.POST)
48 public String post(String StudentID, String name) {
49     logger.info("urlF post called....");
50     System.out.println(StudentID);
51     System.out.println(name);
52     return "redirect:/";
53 }
54
55 }
```

GetandPost.jsp

```

1 %@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>GET AND POST</title>
8 </head>
9 <body>
10    <h1>GET and POST TEST</h1>
11
12    <form method="POST" action= ./urlF>
13        <table>
14            <tr>
15                <td><label>학번</label></td>
16                <td><input type="text" name="StudentID" /></td>
17            </tr>
18            <tr>
19                <td><label>이름</label></td>
20                <td><input type="text" name="name" /></td>
21            </tr>
22        </table>
23        <input type="submit" value="전송" />
24    </form>
25
26 </body>
27 </html>
```

스프링 MVC

■ GET 방식과 POST 방식

- 결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from ServletContext resource [/WEB-INF/spring/root-context.xml]
INFO : org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor - JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlF},methods=[GET]}" onto public java.lang.String com.kwce.controller.ControllerExample.urlF(javax.servlet.http.HttpServletRequest)
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlB}}" onto public java.lang.String com.kwce.controller.HomeController.home()
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlC}}" onto public java.lang.String com.kwce.controller.HomeController.home()
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlA}}" onto public java.lang.String com.kwce.controller.HomeController.home()
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlF},methods=[POST]}" onto public void com.kwce.controller.ControllerExample.urlF(javax.servlet.http.HttpServletRequest)
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{/},methods=[GET]}" onto public java.lang.String com.kwce.controller.HomeController.home()
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlD}}" onto public java.lang.String com.kwce.controller.HomeController.home()
INFO : org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping - Mapped "{{urlE}}" onto public void com.kwce.controller.HomeController.home()
...

```

- GET으로 등록한 urlF 페이지의 입력내용이 POST로 등록한 urlF Controller로 전송

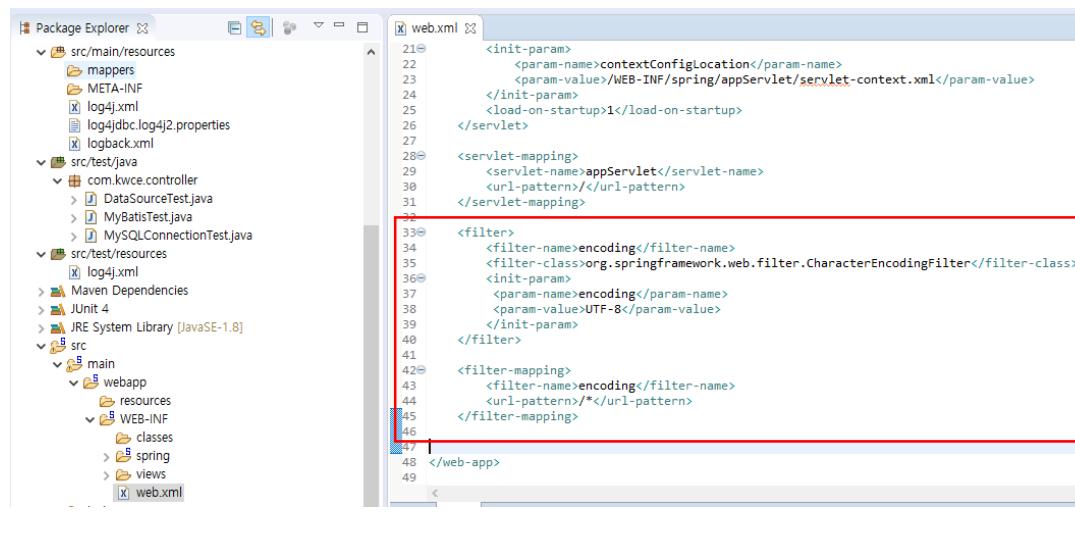


Encoding type이 맞지 않아 글자가
깨지는 현상 발생

스프링 MVC

■ GET 방식과 POST 방식

- src/main/webapp/WEB-INF/web.xml 파일에 필터 추가
- 서버 재실행 후 적용된 결과 확인



```

<init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
</init-param>
</servlet>
<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
<filter>
    <filter-name>encoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encoding</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

```

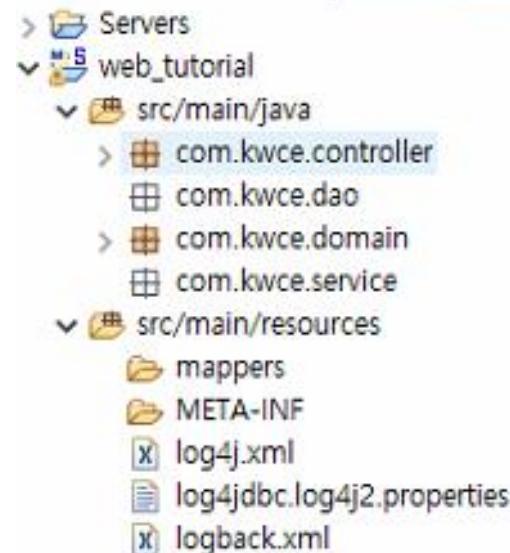
↑

INFO : com.kwce.controller.ControllerExample - urlF get called....
 INFO : com.kwce.controller.ControllerExample - urlF post called....
 201777777
 훌길동
 INFO : com.kwce.controller.HomeController - Welcome home! The client locale is ko_KR.

3. 스프링 + MyBatis

개발 패키지 구성

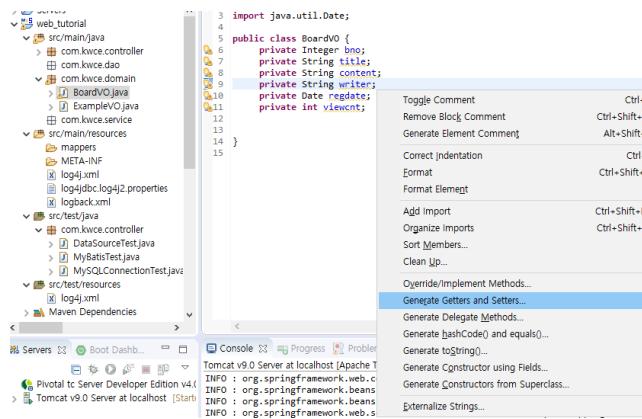
- com.kwce.dao, com.kwce.service, mappers 패키지 생성
 - com.kwce.dao
 - ✓ Dao를 저장하는 패키지
 - ✓ Dao는 Data Access Object의 약자로 DB를 사용해 데이터를 조회하거나 조작
 - mappers
 - ✓ DAO가 수행하기 위한 SQL문들이 xml 파일로 저장
 - com.kwce.service
 - ✓ 비즈니스 로직들을 저장하는 패키지



Domain 패키지의 BoardVO 작성

■ Domain 패키지 안에 BoardVO 작성

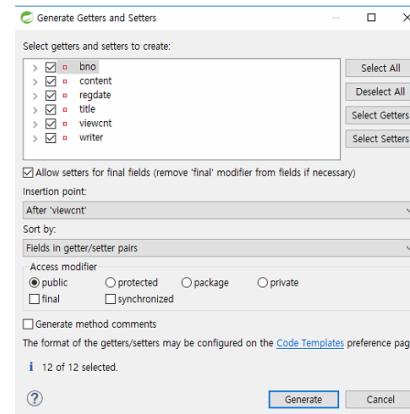
- 필드 부분 먼저 작성하고 alt + shift + s 누르고 Generate Getters and Setters 생성하고 `toString()` 생성



```

3 import java.util.Date;
4
5 public class BoardVO {
6     private Integer bno;
7     private String title;
8     private String content;
9     private String writer;
10    private Date regdate;
11    private int viewcnt;
12
13}
14
15}
16
17}
18
19}
20
21}
22
23}
24
25}
26
27}
28
29}
30
31}
32
33}
34
35}
36
37}
38
39}
40
41}
42
43}
44
45}
46
47}
48
49}
50
51}
52
53}
54
55}
56
57}
58
59}
60

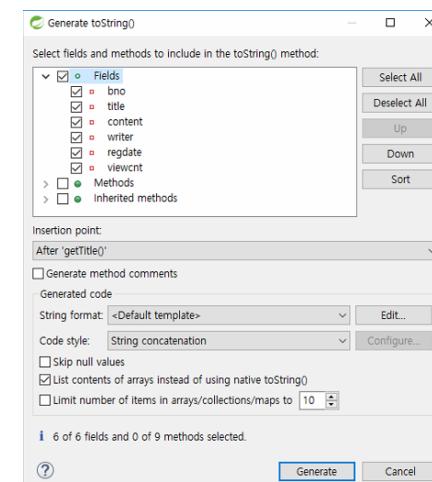
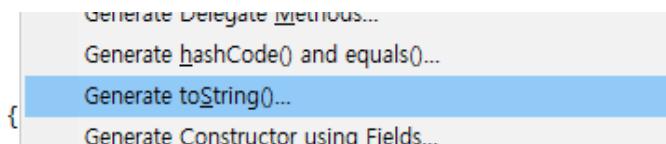
```



```

10 private Date regdate;
11 private int viewcnt;
12
13 public Integer getBno() {
14     return bno;
15 }
16
17 public void setBno(Integer bno) {
18     this.bno = bno;
19 }
20
21
22 public String getTitle() {
23     return title;
24 }
25 public void setTitle(String title) {
26     this.title = title;
27 }
28
29 public String getContent() {
30     return content;
31 }
32
33 public void setContent(String content) {
34     this.content = content;
35 }
36
37 public String getWriter() {
38     return writer;
39 }
40
41 public void setWriter(String writer) {
42     this.writer = writer;
43 }
44
45 public Date getRegdate() {
46     return regdate;
47 }
48
49 public void setRegdate(Date regdate) {
50     this.regdate = regdate;
51 }
52
53 public int getViewcnt() {
54     return viewcnt;
55 }
56
57 public void setViewcnt(int viewcnt) {
58     this.viewcnt = viewcnt;
59 }
60

```



```

@Override
public String toString() {
    return "BoardVO [bno=" + bno + ", title=" + title + ", content=" + content + ", writer=" + writer + ", regdate=" +
        + regdate + ", viewcnt=" + viewcnt + "]";
}

```

DAO 패키지의 BoardDAO 작성

- DAO 패키지 안에 BoardDAO (interface) 작성
- BoardDAO를 구현하는 BoardDAOImpl 클래스도 작성

The screenshot shows the Eclipse IDE interface with two code editors and a package explorer.

Package Explorer:

- Servers
- webTutorial
- src/main/java
 - com.kwce.controller
 - ControllerExample.java
 - HomeController.java
 - RedirectExample.java
 - com.kwce.dao
 - BoardDAO.java
 - BoardDAOImpl.java
 - com.kwce.domain
 - BoardVO.java
 - ExampleVO.java
 - com.kwce.service
- src/main/resources

Code Editors:

BoardDAO.java (Top Editor):

```

1 package com.kwce.dao;
2 import java.util.List;
3 import com.kwce.domain.BoardVO;
4
5 public interface BoardDAO {
6     public void create(BoardVO vo) throws Exception;
7
8     public BoardVO read(Integer bno) throws Exception;
9
10    public void update(BoardVO vo) throws Exception;
11
12    public void delete(Integer bno) throws Exception;
13
14    public List<BoardVO> listAll() throws Exception;
15
16 }
17

```

BoardDAOImpl.java (Bottom Editor):

```

1 package com.kwce.dao;
2
3 public class BoardDAOImpl implements BoardDAO{
4
5 }
6

```

A red circle highlights the word "BoardDAOImpl" in the implementation class. A tooltip message appears over the error icon in the status bar:

The type BoardDAOImpl must implement the inherited abstract method BoardDAO.read(Integer)

2 quick fixes available:

- Add unimplemented methods (highlighted with a red border)
- Make type 'BoardDAOImpl' abstract

Handwritten Notes:

빨간 줄이 쳐진 부분에 마우스를 대고 F2 누르고 Add unimplemented Methods 선택 시 Method 를 자동 생성

DAO 패키지의 BoardDAO 작성

- DAO를 구현하는 클래스 위에는 반드시 @Repository을 작성

BoardDAO.java

```

1 package com.kwce.dao;
2
3 import java.util.List;
4
5 import org.apache.ibatis.session.SqlSession;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Repository;
8 import com.kwce.domain.BoardVO;
9
10 @Repository
11 public class BoardDAOImpl implements BoardDAO{
12
13     @Autowired
14     private SqlSession session;
15     private static String namespace="com.kwce.mapper.BoardMapper";
16
17     @Override
18     public void create(BoardVO vo) throws Exception {
19         session.insert(namespace+".create",vo);
20     }
21     @Override
22     public BoardVO read(Integer bno) throws Exception {
23         return session.selectOne(namespace+".read",bno);
24     }
25     @Override
26     public void update(BoardVO vo) throws Exception {
27         session.update(namespace + ".update",vo);
28     }
29     @Override
30     public void delete(Integer bno) throws Exception {
31         session.delete(namespace+".delete",bno);
32     }
33     @Override
34     public List<BoardVO> listAll() throws Exception {
35         return session.selectList(namespace+".listAll");
36     }
37 }

```

BoardDAOImpl.java

```

public interface SqlSession extends Closeable {
    /**
     * Retrieve a single row mapped from the statement key
     * @param <T> the returned object type
     * @param statement
     * @return Mapped object
     */
    <T> T selectOne(String statement);

    /**
     * Retrieve a single row mapped from the statement key and parameter.
     * @param <T> the returned object type
     * @param statement Unique identifier matching the statement to use.
     * @param parameter A parameter object to pass to the statement.
     * @return Mapped object
     */
    <T> T selectOne(String statement, Object parameter);

    /**
     * Retrieve a list of mapped objects from the statement key and parameter.
     * @param <E> the returned list element type
     * @param statement Unique identifier matching the statement to use.
     * @return List of mapped object
     */
    <E> List<E> selectList(String statement);

    /**
     * Retrieve a list of mapped objects from the statement key and parameter.
     * @param <E> the returned list element type
     * @param statement Unique identifier matching the statement to use.
     * @param parameter A parameter object to pass to the statement.
     * @return List of mapped object
     */
    <E> List<E> selectList(String statement, Object parameter);

    /**
     * Retrieve a list of mapped objects from the statement key and parameter,
     * within the specified row bounds.
     * @param <E> the returned list element type
     * @param statement Unique identifier matching the statement to use.
     * @param parameter A parameter object to pass to the statement.
     * @param rowBounds Bounds to limit object retrieval
     * @return List of mapped object
     */
    <E> List<E> selectList(String statement, Object parameter, RowBounds rowBounds);
}

```

F3을 눌러서 해당 클래스의 구현부로 진입

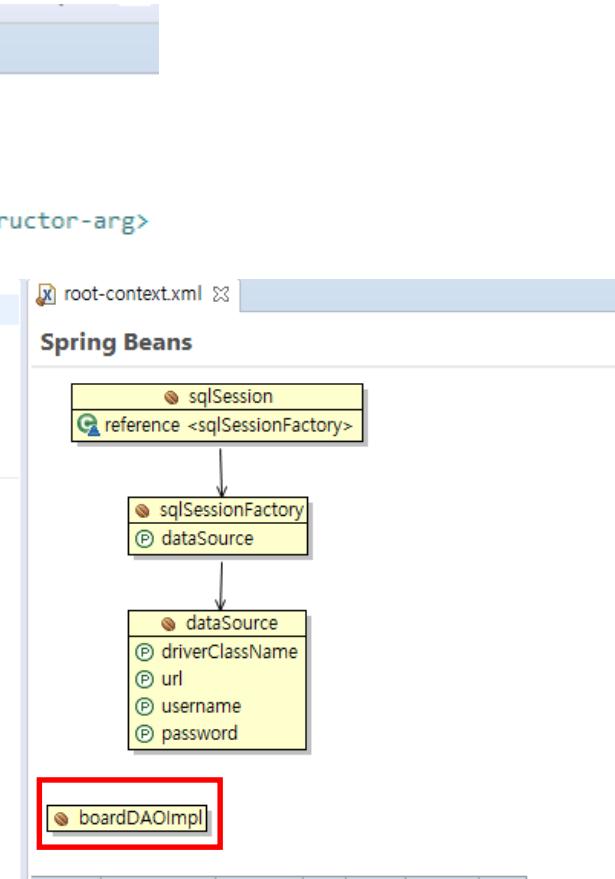
Method 사용에 대한 것은 해당하는 주석부분을 참고할 것

SqlSession을 주입하여 사용

DAO를 작성하는 이유?
향후 데이터베이스 관련 기술이
변경되더라도 DAO만 변경해서
처리할 수 있게 한다.

DAO 패키지의 BoardDAO 작성

- root-context.xml 파일안에 com.kwce.dao 패키지를 scan 할 수 있도록 설정



```

<?xml version="1.0" encoding="UTF-8"?>
<beans>
    <!-- Bean Definitions -->
    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
    <!-- Component Scan Configuration -->
    <context:component-scan base-package="com.kwce.dao" />
</beans>

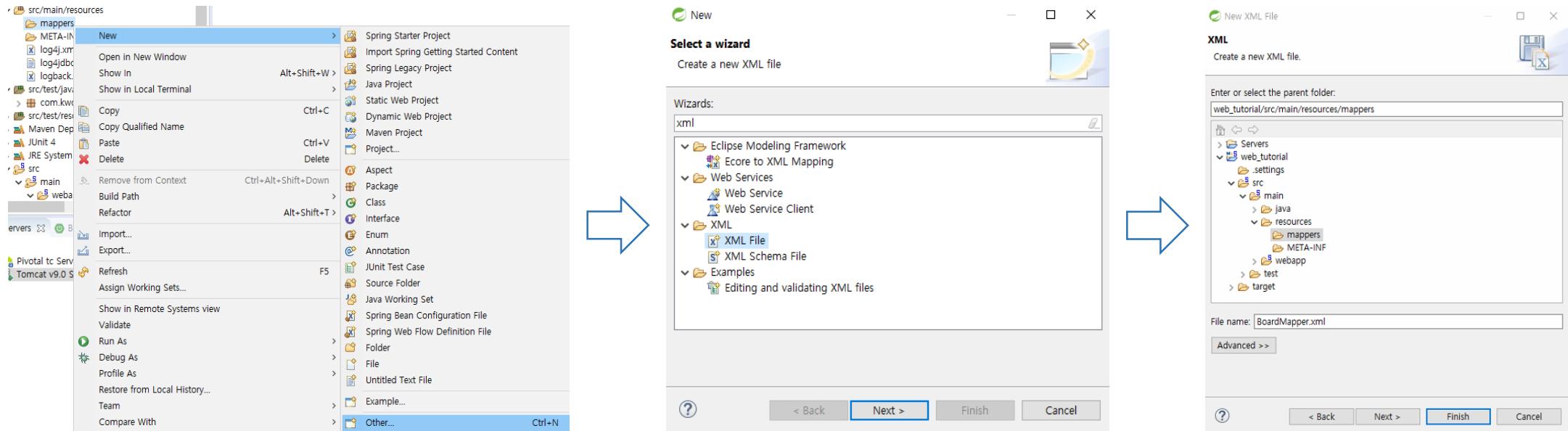
```

The code editor shows the configuration of the root context XML file. A red box highlights the component scan configuration at line 39:

- boardDAOImpl이 추가된 것 확인
 - 이 때 클래스명 첫 글자가 소문자로 바뀌어서 들어감

Mapper 작성

- src/main/resources의 mappers안에 boardMapper.xml 파일 생성



Mapper 작성

■ boardMapper.xml 파일 작성

- namespace를 “com.kwce.mapper.BoardMapper”로 지정
- 원하는 쿼리문을 가져올 때 id명을 가지고 가져오고 결과로 resultType 지정

```

boardMapper.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper
3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5 <mapper namespace="com.kwce.mapper.BoardMapper">
6   <insert id="create">
7     INSERT
8       INTO tbl_board
9         ( title,
10           content,
11           writer )
12      VALUES
13        ( #{title},
14          #{content},
15          #{writer} )
16    </insert>
17
18   <select id="read" resultType="com.kwce.domain.BoardVO">
19     SELECT
20       bno,
21       title,
22       content,
23       writer,
24       regdate,
25       viewcnt
26     FROM tbl_board
27     WHERE bno = #{bno}
28   </select>
29
30   <update id="update">
31     UPDATE tbl_board
32       SET title =#{title},
33           content =#{content}
34     WHERE bno =#{bno}
35   </update>
36
37   <delete id="delete">
38     DELETE
39       FROM tbl_board
40     WHERE bno = #{bno}
41   </delete>
42
43
44   <select id="listAll" resultType="com.kwce.domain.BoardVO">
45     <![CDATA[
46       SELECT
47         bno,
48         title,
49         content,
50         writer,
51         regdate,
52         viewcnt
53       FROM tbl_board
54       WHERE bno > 0
55       ORDER BY bno DESC, regdate DESC
56     ]]>
57   </select>
58
59
60 </mapper>
61

```

> 를 String으로 인식하게 하기 위해
![CDATA[]]] 사용

DAO 패키지의 BoardDAO 작성

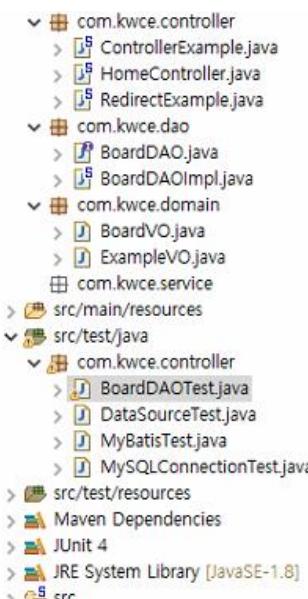
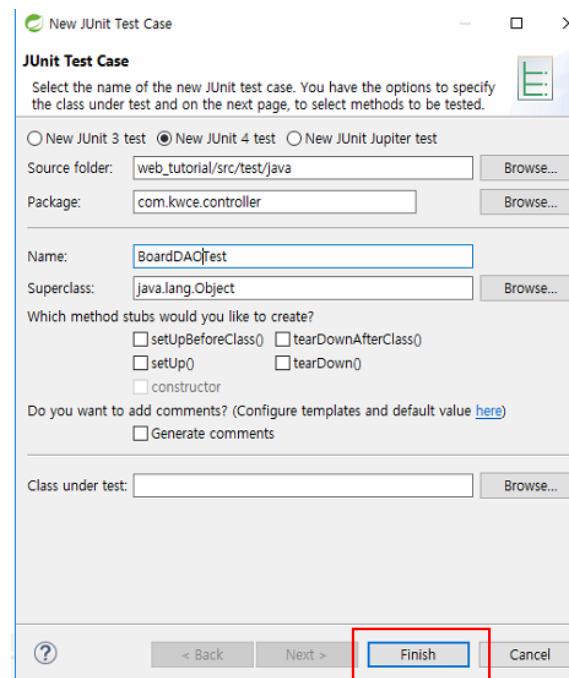
■ root-context.xml에 property 값 추가

```
root-context.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2<beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:aop="http://www.springframework.org/schema/aop"
5   xmlns:jdbc="http://www.springframework.org/schema/jdbc"
6   xmlns:context="http://www.springframework.org/schema/context"
7   xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
8   xsi:schemaLocation="http://www.springframework.org/schema/jdbc http://www.springframework.org/
9     http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
10    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
11    http://www.springframework.org/schema/context http://www.springframework.org/schema/context
12    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
13
14  <!-- Root Context: defines shared resources visible to all other web components -->
15<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
16  <property name="driverClassName" value="net.sf.log4jbc.sql.jdbcapi.DriverSpy"></property>
17  <property name="url" value="jdbc:log4jdbc:mysql://127.0.0.1:3306/web_tutorial?useSSL=false"/>
18  <property name="username" value="root"></property>
19  <property name="password" value="123456"></property>
20</bean>
21
22<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
23  <property name="dataSource" ref="dataSource" />
24
25<property name="mapperLocations"
26   value="classpath:mappers/**/*Mapper.xml"></property>
27
28</bean>
29
30<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate"
31   destroy-method="clearCache">
32   <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
33</bean>
34
35
36<context:component-scan base-package="com.kwce.dao">
37</context:component-scan>
38
39</beans>
40
```

Mapper파일을 참조할 수 있도록 추가

JUnit을 이용한 BoardDAO Test

- BoardDAOTest 클래스 작성
 - 글 생성 테스트



```

5 import org.slf4j.LoggerFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.test.context.ContextConfiguration;
8 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
9 import com.kwce.dao.BoardDAO;
10 import com.kwce.domain.BoardVO;
11
12 @RunWith(SpringJUnit4ClassRunner.class)
13 @ContextConfiguration(locations= {"file:src/main/webapp/WEB-INF/spring/**/root-context.xml"})
14 public class BoardDAOTest {
15@    @Autowired
16    private BoardDAO dao;
17
18    private static final Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);
19
20@    @Test
21    public void testCreate() throws Exception {
22        BoardVO board = new BoardVO();
23        board.setTitle("새로운 글을 넣습니다.");
24        board.setContent("새로운 글을 넣습니다.");
25        board.setWriter("201700000");
26        dao.create(board);
27    }
28
29
30}
31

```

JUnit을 이용한 BoardDAO Test

■ 글 생성 테스트 결과

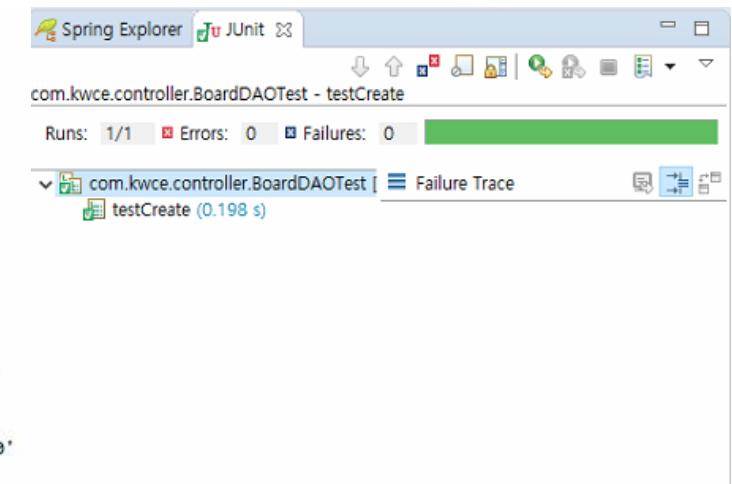
- Alt + Shift + X 누르고 T 선택 => jUnit Test
- 로그를 통하여 어떤 쿼리문이 들어갔는지 확인 가능

```

INFO : jdbc.connection - 1. Connection opened
INFO : jdbc.audit - 1. Connection.new Connection returned
INFO : jdbc.audit - 1. Connection.getAutoCommit() returned true
INFO : jdbc.audit - 1. PreparedStatement.new PreparedStatement returned
INFO : jdbc.audit - 1. Connection.prepareStatement(INSERT
    INTO tbl_board
        ( title,
          content,
          writer )
    VALUES
        (
            ?,
            ?,
            ?
        )) returned net.sf.log4j.jdbc.sql.jdbcapi.PreparedStatementSpy@78e16155
INFO : jdbc.audit - 1. PreparedStatement.setString(1, "새로운 글을 넣습니다.") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(2, "새로운 글을 넣습니다.") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(3, "201700000") returned
INFO : jdbc.sqlonly - INSERT INTO tbl_board ( title, content, writer ) VALUES ( '새로운 글을 넣습니다.', '새로운 글을 넣습니다.', '201700000'
)

INFO : jdbc.sqltiming - INSERT INTO tbl_board ( title, content, writer ) VALUES ( '새로운 글을 넣습니다.', '새로운 글을 넣습니다.', '201700000'
)
{executed in 67 msec}

```

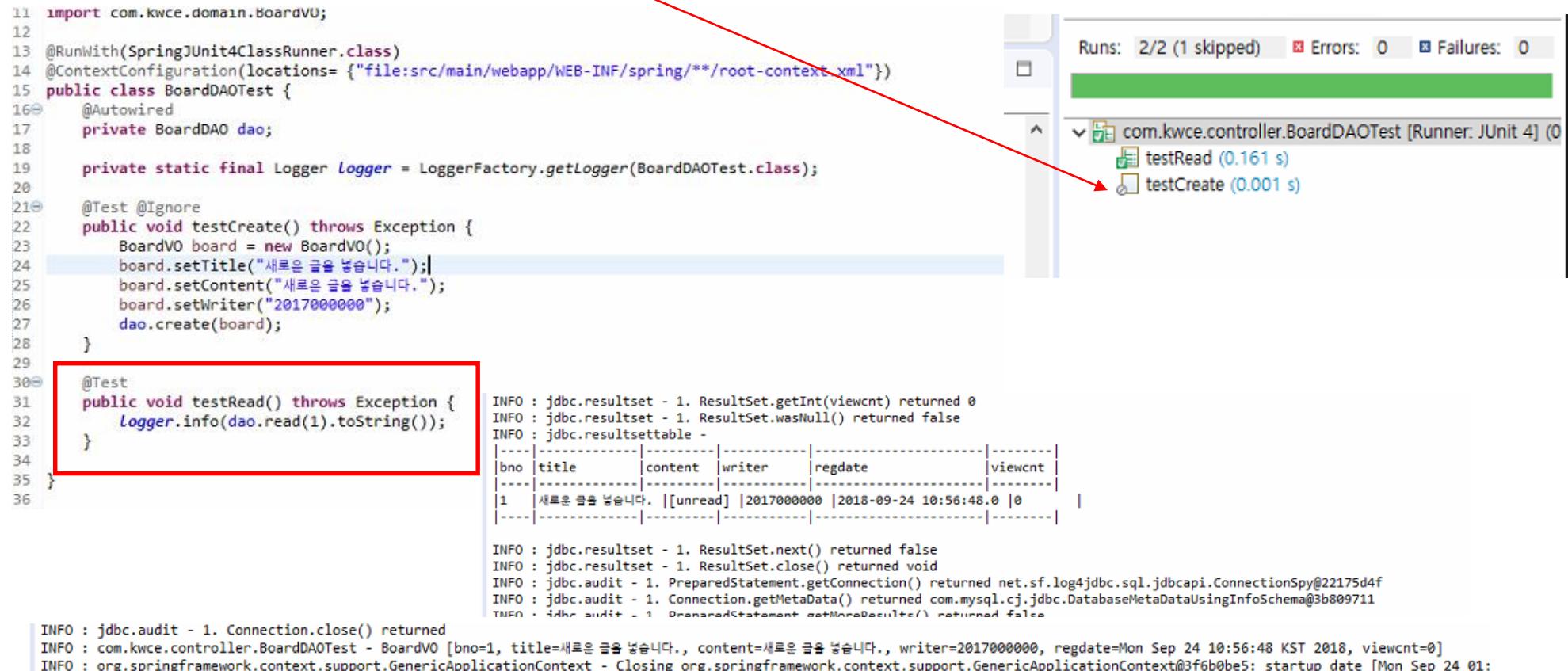


bno	title	content	writer	regdate	viewcnt
1	새로운 글을 넣습니다.	새로운 글을 넣습니다.	201700000	2018-09-24 01:56:48	0
NULL	NULL	NULL	NULL	NULL	NULL

JUnit을 이용한 BoardDAO Test

■ 글 조회 결과

- 1번 글을 조회하고 @Ignore 애노테이션으로 testCreate는 무시



The screenshot shows a Java code editor on the left and a test results window on the right.

Java Code (BoardDAOTest.java):

```

11 import com.kwce.domain.BoardVO;
12
13 @RunWith(SpringJUnit4ClassRunner.class)
14 @ContextConfiguration(locations= {"file:src/main/webapp/WEB-INF/spring/**/root-context.xml"})
15 public class BoardDAOTest {
16     @Autowired
17     private BoardDAO dao;
18
19     private static final Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);
20
21     @Test @Ignore
22     public void testCreate() throws Exception {
23         BoardVO board = new BoardVO();
24         board.setTitle("새로운 글을 넣습니다.");
25         board.setContent("새로운 글을 넣습니다.");
26         board.setWriter("2017000000");
27         dao.create(board);
28     }
29
30     @Test
31     public void testRead() throws Exception {
32         logger.info(dao.read(1).toString());
33     }
34 }
35
36

```

A red box highlights the `testRead()` method, and a red arrow points from this box to the test results window.

Test Results:

Runs: 2/2 (1 skipped) Errors: 0 Failures: 0

- com.kwce.controller.BoardDAOTest [Runner: JUnit 4] (0)
 - testRead (0.161 s)
 - testCreate (0.001 s)

Console Output:

```

INFO : jdbc.resultset - 1. ResultSet.getInt(viewcnt) returned 0
INFO : jdbc.resultset - 1. ResultSet.wasNull() returned false
INFO : jdbc.resultsettable -
|---|-----|-----|-----|-----|-----|
|bno |title |content |writer |regdate |viewcnt |
|---|-----|-----|-----|-----|-----|
|1  |새로운 글을 넣습니다.|[unread] |2017000000 |2018-09-24 10:56:48.0 |0 |
|---|-----|-----|-----|-----|-----|
INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
INFO : jdbc.audit - 1. PreparedStatement.getConnection() returned net.sf.log4jdbc.sql.jdbcapi.ConnectionSpy@22175d4f
INFO : jdbc.audit - 1. Connection.getMetaData() returned com.mysql.cj.jdbc.DatabaseMetaDataUsingInfoSchema@3b809711
INFO : jdbc.audit - 1. PreparedStatement.getMoreResults() returned false
INFO : jdbc.audit - 1. Connection.close() returned
INFO : com.kwce.controller.BoardDAOTest - BoardVO [bno=1, title=새로운 글을 넣습니다., content=새로운 글을 넣습니다., writer=2017000000, regdate=Mon Sep 24 10:56:48 KST 2018, viewcnt=0]
INFO : org.springframework.context.support.GenericApplicationContext - Closing org.springframework.context.support.GenericApplicationContext@3f6b0be5: startup date [Mon Sep 24 01:

```

JUnit을 이용한 BoardDAO Test

- 글 수정 테스트
 - 1번 글을 수정

```

    @RunWith(SpringJUnit4ClassRunner.class)
    @ContextConfiguration(locations= {"file:src/main/webapp/WEB-INF/spring/**/root-context.xml"})
    public class BoardDAOTest {
        @Autowired
        private BoardDAO dao;

        private static final Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);

        @Test @Ignore
        public void testCreate() throws Exception {
            BoardVO board = new BoardVO();
            board.setTitle("새로운 글을 남습니다.");
            board.setContent("새로운 글을 남습니다.");
            board.setWriter("2017000000");
            dao.create(board);
        }

        @Test @Ignore
        public void testRead() throws Exception {
            logger.info(dao.read(1).toString());
        }

        @Test
        public void testUpdate() throws Exception {
            BoardVO board = new BoardVO();
            board.setBno(1);
            board.setTitle("수정된 글입니다.");
            board.setContent("수정 테스트");
            dao.update(board);
        }
    }
  
```



Spring Explorer JUnit

Finished after 0.568 seconds

Runs: 3/3 (2 skipped) Errors: 0 Failures: 0

com.kwce.controller.BoardDAOTest [Runner: JUnit 4] (

- testRead (0.000 s)
- testCreate (0.400 s)
- testUpdate (0.155 s)

INFO : jdbc.audit - 1. PreparedStatement.setInt(3, 1) returned
 INFO : jdbc.sqlonly - UPDATE tbl_board SET title ='수정된 글입니다.', content ='수정 테스트' WHERE bno =1
 INFO : jdbc.sqltiming - UPDATE tbl_board SET title ='수정된 글입니다.', content ='수정 테스트' WHERE bno =1 {executed in 18 msec}
 INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
 INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
 INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
 INFO : jdbc.audit - 1. PreparedStatement.close() returned
 INFO : jdbc.connection - 1. Connection closed
 INFO : jdbc.audit - 1. Connection.close() returned
 INFO : org.springframework.context.support.GenericApplicationContext - Closing one springframework.context.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

bno	title	content	writer	regdate	viewcnt
1	수정된 글입니다.	수정 테스트	2017000000	2018-09-24 01:56:48	0
*	NULL	NULL	NULL	NULL	NULL

JUnit을 이용한 BoardDAO Test

- 글 삭제 테스트
 - 1번 글 삭제

```

L3  @RunWith(SpringJUnit4ClassRunner.class)
L4  @ContextConfiguration(locations= {"file:src/main/webapp/WEB-INF/spring/**/root-context.xml"})
L5  public class BoardDAOTest {
L6    @Autowired
L7    private BoardDAO dao;
L8
L9    private static final Logger logger = LoggerFactory.getLogger(BoardDAOTest.class);
L10
L11  @Test @Ignore
L12  public void testCreate() throws Exception {
L13    BoardVO board = new BoardVO();
L14    board.setTitle("새로운 글을 남습니다.");
L15    board.setContent("새로운 글을 남습니다.");
L16    board.setWriter("2017000000");
L17    dao.create(board);
L18  }
L19
L20  @Test @Ignore
L21  public void testRead() throws Exception {
L22    logger.info(dao.read(1).toString());
L23  }
L24
L25  @Test @Ignore
L26  public void testUpdate() throws Exception {
L27    BoardVO board = new BoardVO();
L28    board.setBno(1);
L29    board.setTitle("수정된 글입니다.");
L30    board.setContent("수정 테스트");
L31    dao.update(board);
L32  }
L33
L34  @Test
L35  public void testDelete() throws Exception{
L36    dao.delete(1);
L37  }
L38
L39
L40
L41
L42
L43
L44
L45
L46
L47
L48
L49
L50
  
```



```

      FROM tbl_board
      WHERE bno = ?) returned net.sf.log4jdbc.sql.jdbcapi.PreparedStatementSpy@3ef41c6
INFO : jdbc.audit - 1. PreparedStatement.setInt(1, 1) returned
INFO : jdbc.sqlonly - DELETE FROM tbl_board WHERE bno = 1

INFO : jdbc.sqltiming - DELETE FROM tbl_board WHERE bno = 1
{executed in 87 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
INFO : idbc.audit - 1. PreparedStatement.isClosed() returned false
  
```

Finished after 0.642 seconds

Runs: 4/4 (3 skipped) Errors: 0 Failures: 0

com.kwce.controller.BoardDAOTest [Runner: JUnit 4]

- testRead (0.000 s)
- testCreate (0.403 s)
- testDelete (0.225 s)**
- testUpdate (0.001 s)

Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	
no	title	content	writer	regdate	viewcnt
1	NULL	NULL	NULL	NULL	NULL

JUnit을 이용한 BoardDAO Test

■ 글 리스트 조회 테스트

- @Before는 @Test로 지정한 함수가 시행되기 전에 수행 (@After는 반대)

```
@Before
public void testTenCreate() throws Exception {
    BoardVO board = new BoardVO();
    for(int i=0; i<10; i++) {
        board.setTitle(i+"번째 글을 남았습니다.");
        board.setContent("새로운 글을 남습니다.");
        board.setWriter("2017000000");
        dao.create(board);
    }
}

@Test
public void testListAll() throws Exception{
    List<BoardVO> list = dao.listAll();
    for (BoardVO boardVO : list) { //항상 for문
        logger.info(boardVO.getBno() + ":" + boardVO.getTitle());
    }
}
```



Spring Explorer JUnit

Finished after 0.728 seconds

Runs: 5/5 (4 skipped) Errors: 0 Failures: 0

com.kwce.controller.BoardDAOTest [Runner: JUnit 4] (0.000 s)

- testRead (0.383 s)
- testListAll (0.332 s)
- testCreate (0.000 s)
- testDelete (0.000 s)
- testUpdate (0.000 s)

INFO : jdbc.audit - 11. jdbc.getConnection() returned
 INFO : jdbc.connection - 11. Connection closed
 INFO : jdbc.audit - 11. Connection.close() returned
 INFO : com.kwce.controller.BoardDAOTest - 10:9번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 9:8번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 8:7번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 7:6번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 6:5번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 5:4번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 4:3번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 3:2번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 2:1번째 글을 남았습니다.
 INFO : com.kwce.controller.BoardDAOTest - 1:0번째 글을 남았습니다.
 INFO : org.springframework.context.support.GenericApplicationContext - Clos

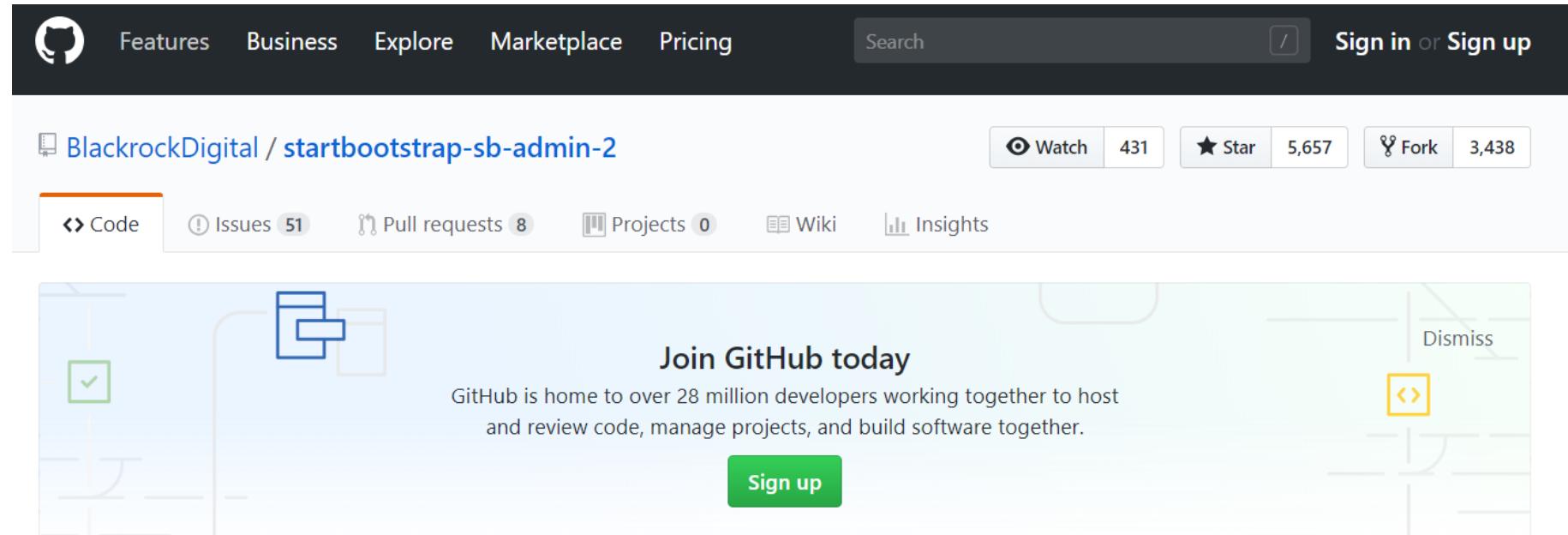
testTenCreate method 실행 후에 testListAll method 실행

4. 게시판 기능 구현

페이지 디자인

■ 부트스트랩 사용

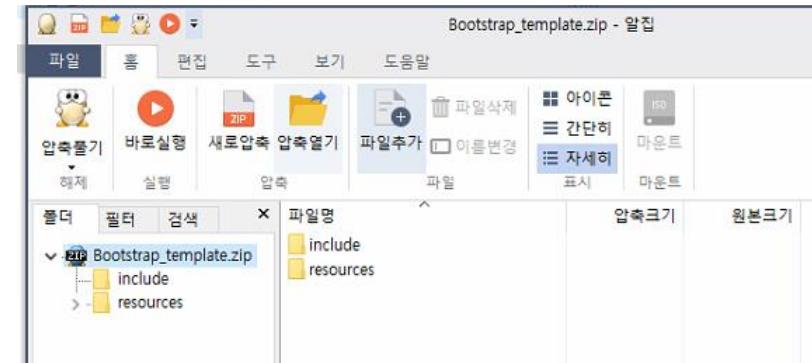
- <https://startbootstrap.com/template-overviews/sb-admin-2/> 사이트에서 제공하는 오픈소스로 부트스트랩을 이용한 무료 템플릿 이용



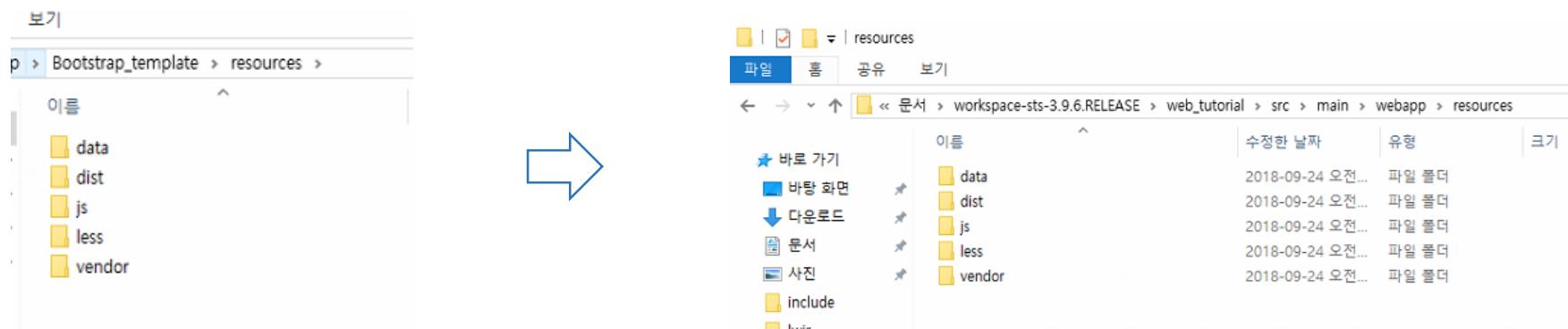
A free, open source, Bootstrap admin theme created by Start Bootstrap <http://startbootstrap.com/template-overviews/sb-admin-2/>

페이지 디자인

- 유캠퍼스에서 Bootstrap_template.zip 파일 다운로드

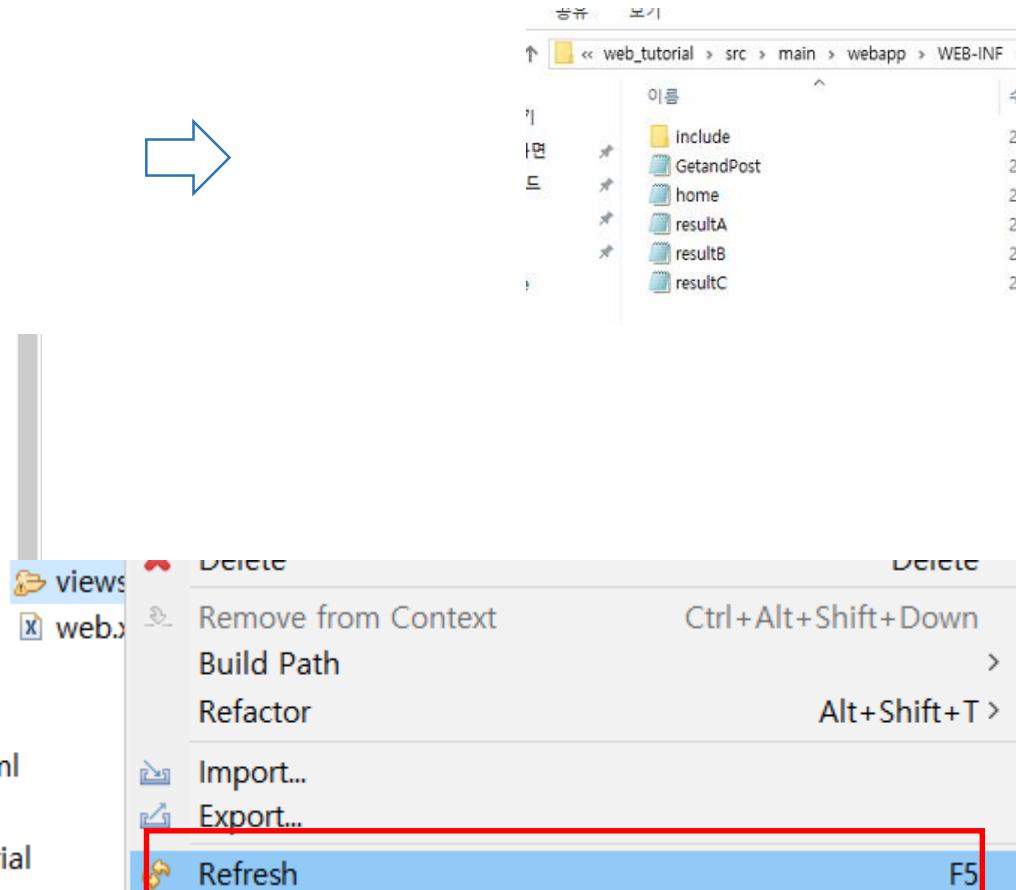
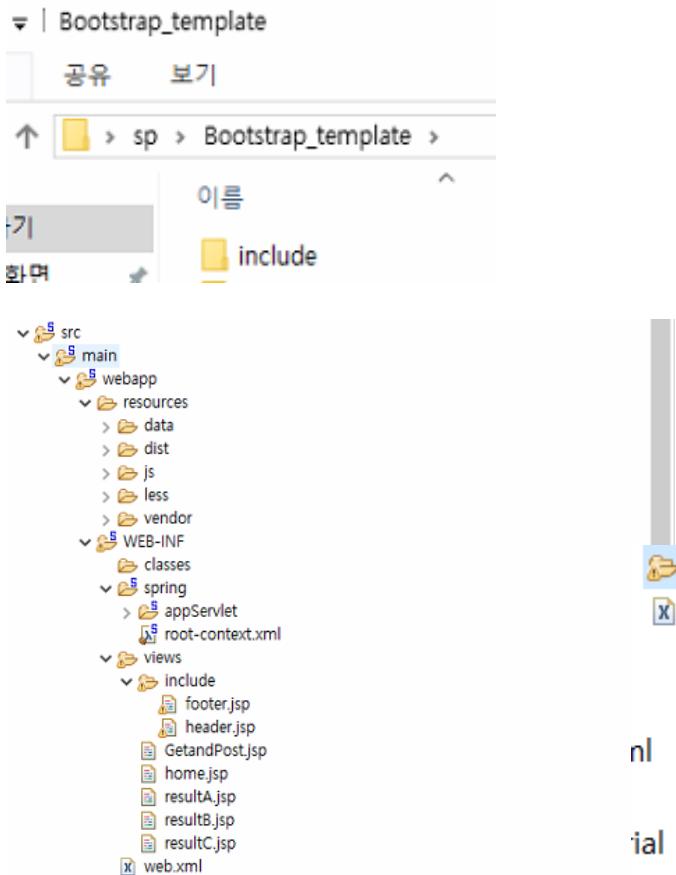


- resources 폴더안의 파일들을 src/main/webapp/resources 폴더에 복사&붙여넣기



페이지 디자인

- include 폴더는 src/main/webapp/WEB-INF/views 폴더로 복사&붙여넣기



폴더 우 클릭
-> [Refresh] 클릭

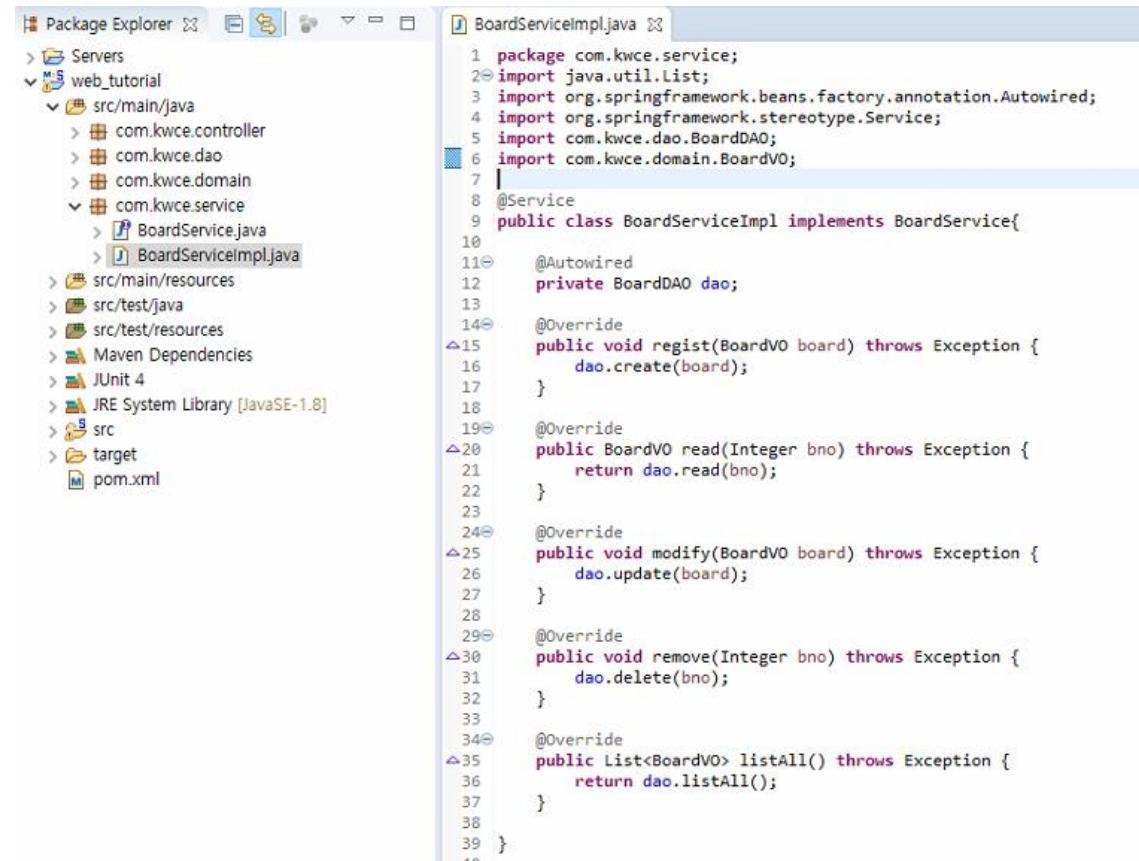
Service 패키지의 BoardService 구현

- Service 패키지 부분은 고객의 요구사항이 반영되는 영역으로 비즈니스 계층으로 불림
- BoardService를 구현 받는 BoardServiceImpl 클래스 추가

```
1 package com.kwce.service;
2
3 import java.util.List;
4
5 public interface BoardService {
6
7     public void regist(BoardVO board) throws Exception;
8
9     public BoardVO read(Integer bno) throws Exception;
10
11    public void modify(BoardVO board) throws Exception;
12
13    public void remove(Integer bno) throws Exception;
14
15    public List<BoardVO>listAll() throws Exception;
16
17 }
18
19 }
```

Service 패키지의 BoardService 구현

- BoardDAO를 주입하여 사용
- BoardService를 구현하는 클래스 위에 반드시 @Service 표시

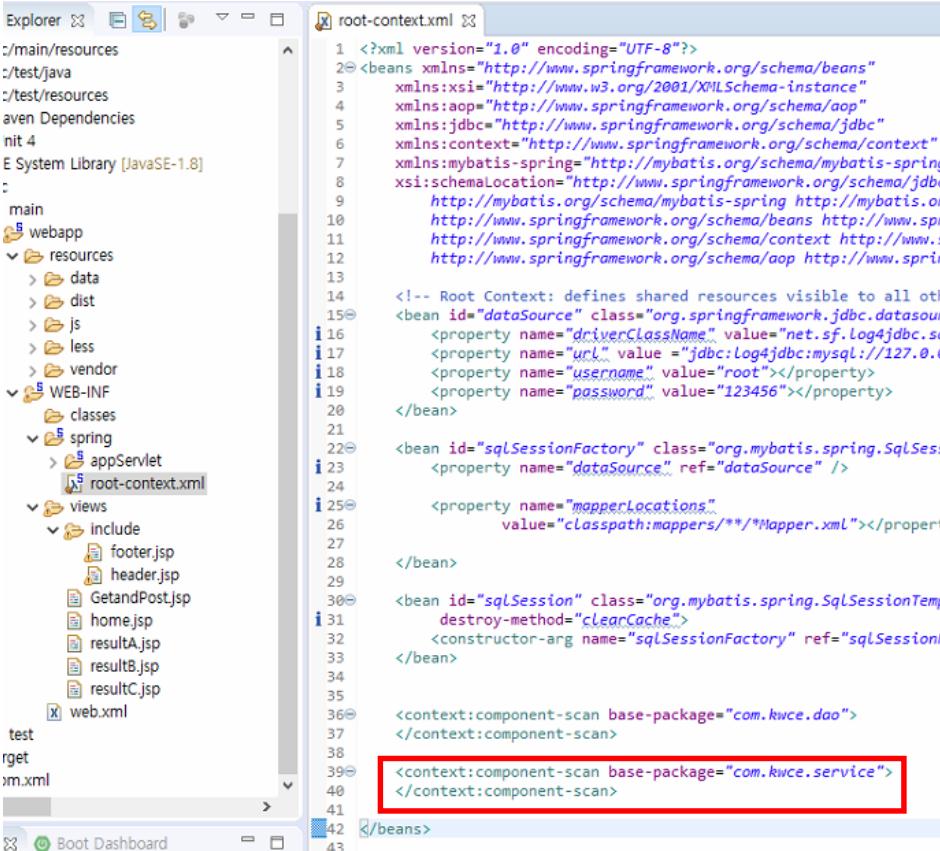


The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under 'webTutorial'. It includes packages like 'com.kwce.controller', 'com.kwce.dao', 'com.kwce.domain', and 'com.kwce.service'. Within 'com.kwce.service', there are files 'BoardService.java' and 'BoardServiceImpl.java'. The right side of the interface shows the code editor with the file 'BoardServiceImpl.java' open. The code implements the 'BoardService' interface, utilizing a 'BoardDAO' dependency injected via the '@Autowired' annotation.

```
1 package com.kwce.service;
2 import java.util.List;
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Service;
5 import com.kwce.dao.BoardDAO;
6 import com.kwce.domain.BoardVO;
7
8 @Service
9 public class BoardServiceImpl implements BoardService{
10
11     @Autowired
12     private BoardDAO dao;
13
14     @Override
15     public void regist(BoardVO board) throws Exception {
16         dao.create(board);
17     }
18
19     @Override
20     public BoardVO read(Integer bno) throws Exception {
21         return dao.read(bno);
22     }
23
24     @Override
25     public void modify(BoardVO board) throws Exception {
26         dao.update(board);
27     }
28
29     @Override
30     public void remove(Integer bno) throws Exception {
31         dao.delete(bno);
32     }
33
34     @Override
35     public List<BoardVO> listAll() throws Exception {
36         return dao.listAll();
37     }
38
39 }
```

Service 패키지의 BoardService 구현

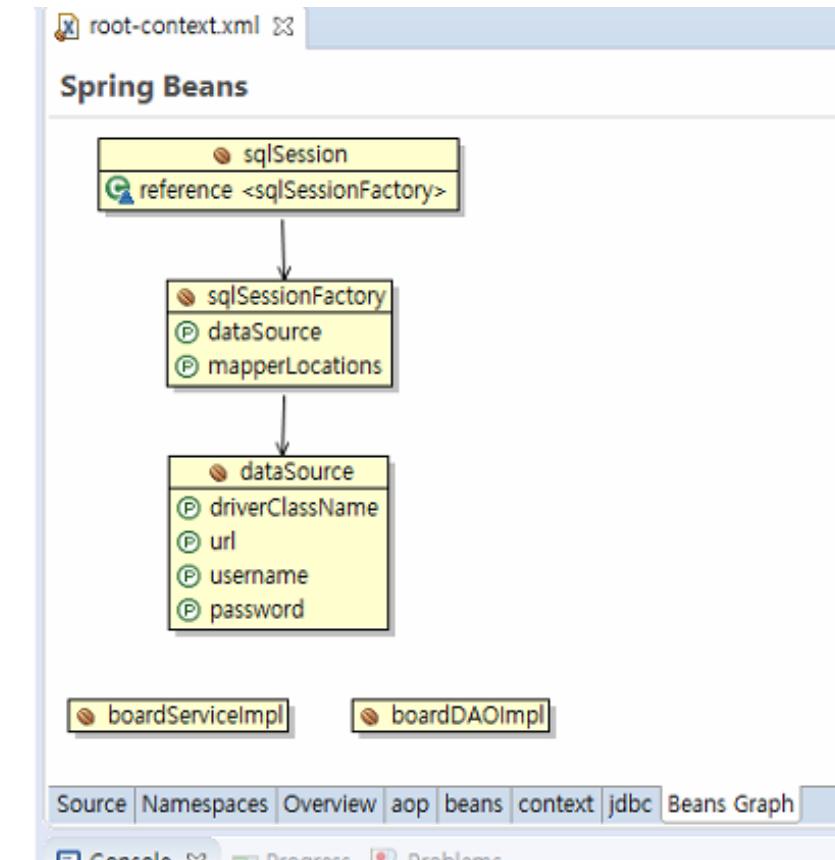
- root-context.xml에서 “com.kwce.service” 패키지를 scan할 수 있도록 <context:component-scan> 사용



```

<?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:aop="http://www.springframework.org/schema/aop"
       xmlns:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
       xsi:schemaLocation="http://www.springframework.org/schema/jdbc
                           http://mybatis.org/schema/mybatis-spring
                           http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/context
                           http://www.mybatis.org/schema/aop
                           http://www.springframework.org/schema/aop"
                           http://www.springframework.org/schema/context"/>
<!-- Root Context: defines shared resources visible to all other beans -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="net.sf.log4j.db.sql.Driver"/>
    <property name="url" value="jdbc:log4jdb:mysql://127.0.0.1:3306/test"/>
    <property name="username" value="root"/>
    <property name="password" value="123456"/>
</bean>
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="mapperLocations" value="classpath:mappers/**/*Mapper.xml" />
</bean>
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory" />
</bean>
<context:component-scan base-package="com.kwce.dao" />
<context:component-scan base-package="com.kwce.service" />
</beans>

```



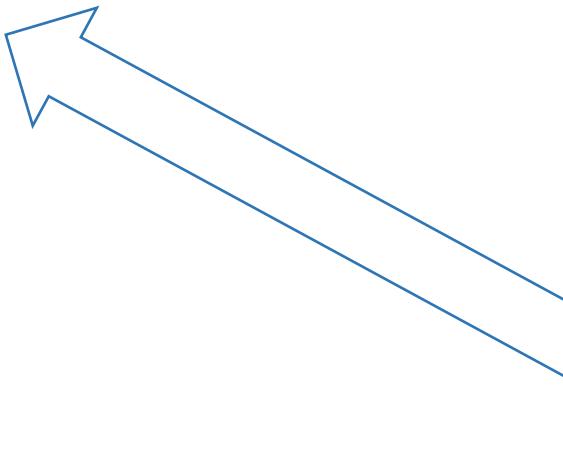
Controller 구현

- 컨트롤러 구현에 앞서서 URI 설계
 - 해당 URL을 어떤 방식(GET, POST)으로 지정할지 어떤 기능이 있는지 구상

방식	URL	설명
GET	/board/register	게시물의 등록 페이지
POST	/board/register	게시물을 실제로 등록
GET	/board/read?bno=xxx	특정 번호의 게시물을 조회
GET	/board/modify?bno=xxx	게시물의 수정화면으로 이동
POST	/board/modify	게시물을 수정
POST	/board/remove	게시물을 삭제
GET	/board/list	게시물 목록을 확인

Controller 구현

- URI에 관한 Method 하나씩
추가 예정



```
BoardController.java ✘
1 package com.kwce.controller;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.RequestMapping;
8
9 import com.kwce.service.BoardService;
10
11 @Controller
12 @RequestMapping("/board/**")
13 public class BoardController {
14
15     private static final Logger logger = LoggerFactory.getLogger(BoardController.class);
16
17     @Autowired
18     private BoardService service;
19
20
21
22
23
24
25
26
27
28
29 }
```

글 리스트

- Src/main/webapp/WEB-INF/views에 board라는 폴더를 만들고 list.jsp를 다음과 같이 구현

The image shows a software development environment with a sidebar containing navigation links like Dashboard, Charts, Tables, Forms, UI Elements, Multi-Level Dropdown, and Sample Pages. A large blue rectangular box highlights the main workspace area.

Project Structure:

```

    - src
      - main
        - webapp
          - resources
          - WEB-INF
            - classes
          - spring
          - views
            - board
              - list.jsp
  
```

list.jsp Content:

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"%>
2
3 <%@include file="../include/header.jsp"%>
4
5 <div class="row">
6   <div class="col-lg-12">
7     <h1 class="page-header">학번</h1>
8   </div><!-- /.col-lg-12 -->
9 </div><!-- /.row -->
10
11 <div class="row">
12   <div class="col-lg-12">
13     <div class="panel panel-default">
14       <div class="panel-heading">
15         Board List Page
16         <button id="regBtn" type="button" onclick="location.href='#'" class="btn btn-xs pull-right">Register
17           New Board</button>
18         </div><!-- /.panel-heading -->
19
20       <div class="panel-body">
21         <table class="table table-striped table-bordered table-hover">
22           <thead>
23             <tr>
24               <th>#번호</th>
25               <th>제목</th>
26               <th>작성자</th>
27               <th>작성일</th>
28               <th>수정일</th>
29               <th> 조회수</th>
30             </tr>
31           </thead>
32           <tbody>
33             <tr>
34               <td>##</td>
35               <td>##</td>
36               <td>##</td>
37               <td>##</td>
38               <td>##</td>
39               <td>##</td>
40             </tr>
41           </tbody>
42         </table>
43       </div> <!-- end panel-body -->
44     </div><!-- /.col-lg-12 -->
45   </div><!-- /.row -->
46
47 <%@include file="../include/footer.jsp"%>
48
49
50
51
  
```

Annotations:

- A red box highlights the header section: `<h1 class="page-header">학번</h1>`. An arrow points from this box to the text "본인 학번 기입".
- A red box highlights the registration button: `<button id="regBtn" type="button" onclick="location.href='#'" class="btn btn-xs pull-right">Register New Board</button>`. An arrow points from this box to the text "글 등록 링크처리를 위한 버튼".
- A red box highlights the first row of the table: `<tr>` to `</tr>`. An arrow points from this box to the text "이 부분에 List를 하나씩 보여주기 위한 for문 구현 예정".
- A red box highlights the footer inclusion: `<%@include file="../include/footer.jsp"%>`.

글 리스트 컨트롤러 및 페이지 구현

- 전체글을 보여주는 글 리스트 페이지의 Controller 구현
- “컨텍스트 루트/board/list” URI로 페이지 위치가 부여됨

The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer view, displaying the project structure under 'webTutorial'. On the right is the code editor view showing 'BoardController.java'.

```

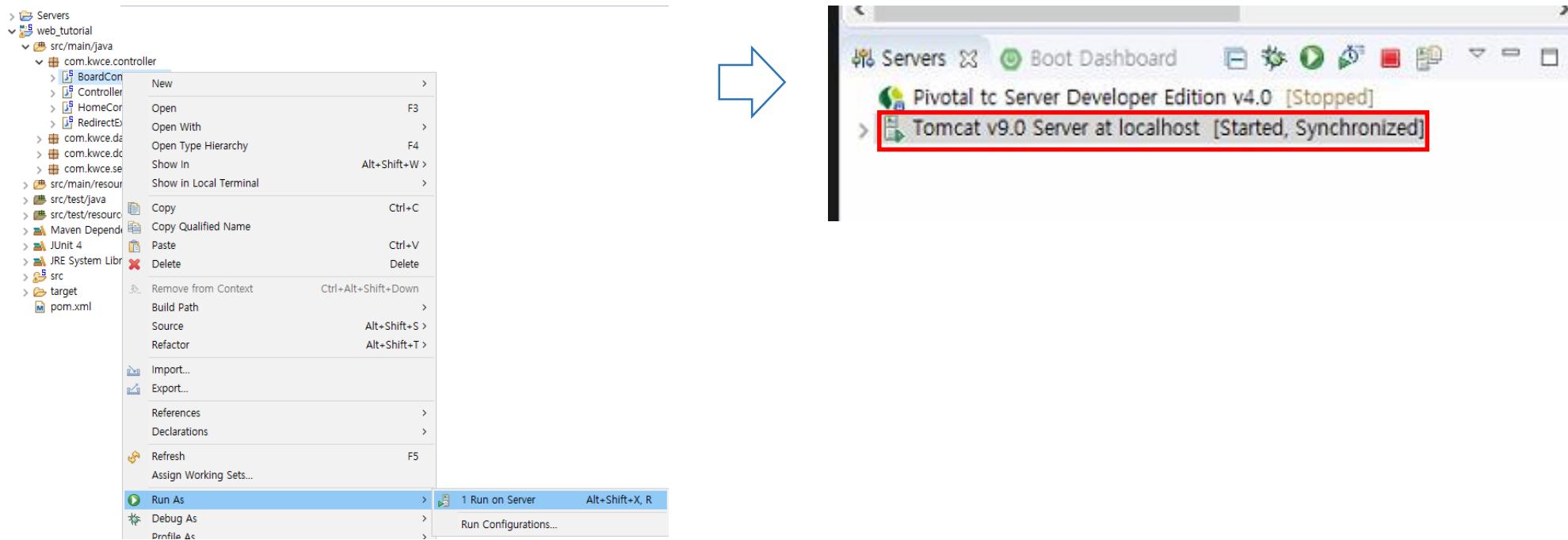
1 package com.kwce.controller;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10
11 import com.kwce.service.BoardService;
12
13 @Controller
14 @RequestMapping("/board/*")
15 public class BoardController {
16
17     private static final Logger logger = LoggerFactory.getLogger(BoardController.class);
18
19     @Autowired
20     private BoardService service;
21
22     @RequestMapping(value = "/list", method = RequestMethod.GET)
23     public String list(Model model) throws Exception{
24         logger.info("list page get....");
25         model.addAttribute("list", service.listAll());
26         return "board/list";
27     }
28
29 }

```

A red box highlights the line '@RequestMapping("/list", method = RequestMethod.GET)' in the code. To the right of the code, the text '/board 하위에 URI 추가' is written in red, indicating that the URI is added under the /board context root.

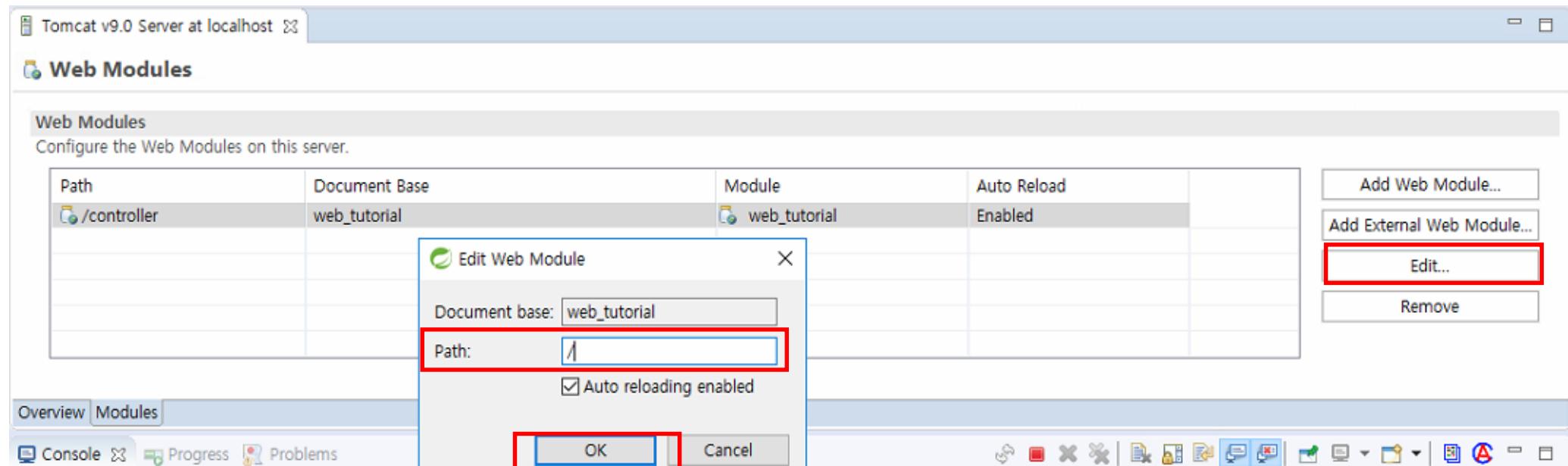
글 리스트

- BoardController -> [Run As] -> [Run On Server]
- 좌측 하단에 Servers 탭에서 Tomcat 클릭

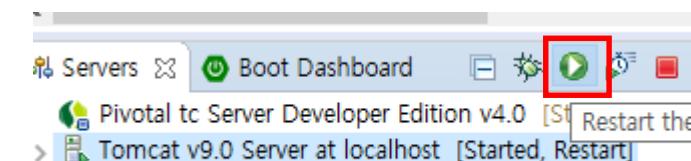


글 리스트

- Modules 탭에서 현재 올라온 webTutorial 모듈을 Edit 하여 Path를 /controller -> / 로 수정 (컨텍스트루트 변경)

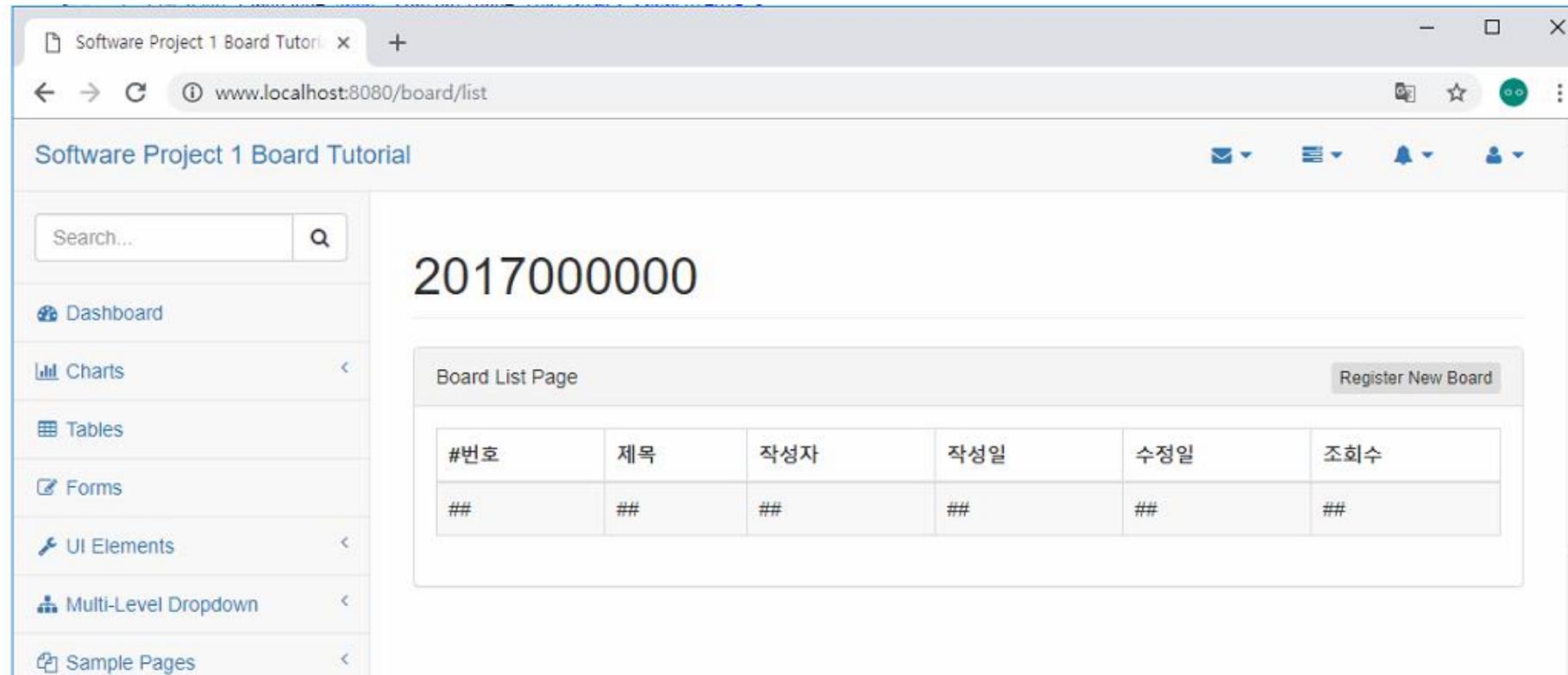


- 반드시 변경 후 Tomcat server restart
 - 변경하지 않으면 Bootstrap이 제대로 적용되지 않음



글 리스트

- www.localhost:8080/board/list 로 접속



- 부트스트랩이 적용된 것 확인

JSTL(JSP Standard Tag Library)

- JSTL(JSP Standard Tag Library) 이용

JSTL은 표준화된 태그 라이브러리를 제공하며 보다 편리하게 웹 응용프로그램을 개발 할 수 있도록 지원

```
<% @taglib prefix="접두어" uri="http://java.sun.com/jsp/jstl/라이브러리이름"%>
```

라이브러리	기능	접두어
core	언어의 반복문, 제어, 변수 선언 등의 기능 제공	c
fmt	숫자, 날짜, 시간의 포맷팅 기능	fmt
sql	데이터베이스의 데이터를 입력/수정/삭제/조회 기능 제공	sql
xml	Xml 문서를 처리할 때 필요한 기능 제공	x
Functions	문자열을 처리하는 함수 제공	fn

JSTL(JSP Standard Tag Library)

- ex) core 라이브러리 사용 예제

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

- 변수 선언 및 반복문

```
<c:set var="a" value="10"/>
<c:forEach var="i" begin="1" end="${a}">
    ${i} <br>
</c:forEach>
```

1
2
3
4
5
6
7
8
9
10

- 조건문

```
<c:set var="b" value="8"/>
<c:if test="${b le 10}">
    B는 10보다 작다.<br>
</c:if>
```

B는 10보다 작다.

- 리스트 반복

```
<%
    java.util.List list= new java.util.ArrayList();
    list.add("a");
    list.add("b");
    list.add("c");
    pageContext.setAttribute("array",list);
%>

<c:forEach var="list" items="${array}">
    ${list}<br>
</c:forEach>
```

a
b
c

Items 속성은 배열, Collection, iterator 등을 담을 수 있음

JSTL(JSP Standard Tag Library)

- 변수 출력

```
<c:set var="test" value="abcdefg" />  
변수 값 출력: <c:out value="${test}" />
```



변수 값 출력: abcdefg

- ex) format 라이브러리 사용 예제

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
  
<c:set var="now" value="<%new java.util.Date()%>" />  
<fmt:formatDate value="${now}" pattern="yy-MM-dd hh:mm" type="both"/><br>  
<fmt:formatDate value="${now}" pattern="yy-MM-dd" type="date"/><br>  
<fmt:formatDate value="${now}" pattern="yy-MM-dd E요일 a" type="date"/><br>  
<fmt:formatDate value="${now}" pattern="HH:mm" type="time"/><br>
```

18-09-24 06:38

18-09-24

18-09-24 월요일 오후

18:38

글 리스트

- 다음과 같이 list.jsp 파일 수정

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4<%@include file="../include/header.jsp"%>
5<div class="row">
6    <div class="col-lg-12">
7        <h1 class="page-header">201700000</h1>
8    </div><!-- /.col-lg-12 -->
9</div><!-- /.row -->
10<br/>
11<div class="row">
12    <div class="col-lg-12">
13        <div class="panel panel-default" onclick="location.href='/board/register'">
14            <div class="panel-heading">
15                Board List Page
16                <button id='regBtn' type="button" onclick="location.href='##'" class="btn btn-xs pull-right">Register
17                New Board</button>
18            </div><!-- .panel-heading -->
19
20<br/>
21        <div class="panel-body">
22            <table class="table table-striped table-bordered table-hover">
23                <thead>
24                    <tr>
25                        <th>#번호</th>
26                        <th>제목</th>
27                        <th>작성자</th>
28                        <th>작성일</th>
29                        <th>조회수</th>
30                    </tr>
31                </thead>
32            </table>
33        <c:forEach var="boardVO" items="${list}">
34            <tr>
35                <td>${boardVO.bno}</td>
36                <td><a href="/board/read?bno=${boardVO.bno}'>${boardVO.title}</a></td>
37                <td>${boardVO.writer}</td>
38                <td><fmt:formatDate pattern="yyyy-MM-dd HH:mm"
39                                value="${boardVO.regdate}" /></td>
40                <td><span class="badge bg-red">${boardVO.viewcnt}</span></td>
41            </tr>
42        </c:forEach>
43
44    </div>
45    </div> <!-- end panel-body -->
46</div><!-- end panel -->
47</div><!-- /.col-lg-12 -->
48</div><!-- /.row -->
49<br/>
50
51 <%@include file="../include/footer.jsp"%>
```

글 등록 페이지 링크 추가

글 제목을 누르면 해당 글로
조회, 글을 구별하기 위해 URI에
?bno=xxx 를 붙임

글 리스트

- 글 조회 결과
 - BoardDAOTest 때 넣은 글들을 확인할 수 있음

Software Project 1 Board Tutorial

Search...

Dashboard

Charts

Tables

Forms

UI Elements

Multi-Level Dropdown

Sample Pages

2017000000

Board List Page

Register New Board

#번호	제목	작성자	작성일	조회수
10	9번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
9	8번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
8	7번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
7	6번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
6	5번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
5	4번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
4	3번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
3	2번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
2	1번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0
1	0번째 글을 넣었습니다.	2017000000	2018-09-24 11:34	0

글 등록 기능 구현

- 글 등록 페이지는 GET 방식과 POST 방식으로 구현
- 글 등록 후에 BoardVO 객체에 담긴 값을 Controller에서 받고 DB로 전달
- 글 등록 후에는 메시지를 전달하면서 /board/list 페이지로 리다이렉트

The screenshot shows the Eclipse IDE interface with two main panes. On the left is the package explorer showing the project structure under 'webTutorial'. On the right are two code editors.

Java Code (BoardController.java):

```

21  @Autowired
22  private BoardService service;
23
24  @RequestMapping(value= "/list",method = RequestMethod.GET)
25  public String list(Model model) throws Exception{
26      Logger.info("list page get.....");
27      model.addAttribute("list",service.listAll());
28      return "board/list";
29  }
30
31  @RequestMapping(value = "/register", method = RequestMethod.GET)
32  public String registerGET(BoardVO board, Model model) throws Exception {
33      Logger.info("register page get.....");
34      return "board/register";
35  }
36
37  @RequestMapping(value = "/register", method = RequestMethod.POST)
38  public String registPOST(BoardVO board, RedirectAttributes rttr) throws
39      Logger.info("register page post.....");
40      Logger.info(board.toString());
41      service.regist(board);
42      rttr.addFlashAttribute("msg", "SUCCESS");
43      return "redirect:/board/list";
44  }

```

JSP Code (list.jsp):

```

46 </div><!-- /.row -->
47 <script>
48     var result = '${msg}';
49
50     if(result == 'SUCCESS'){
51         alert("처리가 완료되었습니다.");
52     }
53
54 </script>
55 <%@include file="../include/footer.jsp"%>
56
57
58
59

```

A red box highlights the script section of the JSP code, specifically the alert message logic. A red annotation 'list.jsp에 추가' (Added to list.jsp) is placed next to the highlighted code.

글 등록 기능 구현

■ 글 등록 페이지 뷰 구현

- Input 태그로 입력 받은 내용은 Post 방식으로 /board/register 로 전송

```

register.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4 <%@include file="../include/header.jsp"%>
5 <div class="row">
6   <div class="col-lg-12">
7     <h1 class="page-header">2017000000</h1>
8   </div><!-- /.col-lg-12 -->
9 </div><!-- /.row -->
10 <div class="row">
11   <div class="col-lg-12">
12
13   <div class="panel panel-default">
14     <div class="panel-heading">Board Register</div><!-- /.panel-heading -->
15
16     <div class="panel-body">
17       <form role="form" action="/board/register" method="post">
18         <div class="form-group">
19           <label>Title</label> <input class="form-control" name='title'>
20         </div>
21         <div class="form-group">
22           <label>Text area</label>
23           <textarea class="form-control" rows="3" name='content'></textarea>
24         </div>
25         <div class="form-group">
26           <label>Writer</label> <input class="form-control" name='writer'>
27         </div>
28         <button id="SubmitBtn" type="submit" class="btn btn-default">Submit Button</button>
29
30         <button type="reset" class="btn btn-default">Reset Button</button>
31
32         <button id="ListBtn" class="btn btn-default">List</button>
33       </form>
34     </div> <!-- end panel-body -->
35   </div> <!-- end panel -->
36 </div> <!-- /.row -->
37
38 <%@include file="../include/footer.jsp"%>
```

```

39 <script>
40
41 $(document).ready(function(){
42   var Formobj= $("form[role='form']");
43
44   $("#ListBtn").on("click",function(){
45     Formobj.attr("action","/board/list");
46     Formobj.attr("method", "get");
47     Formobj.submit();
48   });
49
50   $("#SubmitBtn").on("click",function(){
51     if($("#input[name=title]").val()==''){
52       alert("제목을 입력하세요!");
53       $("#input[name='title']").focus();
54       return false;
55     }
56     if($("#input[name=content]").val()==''){
57       alert("내용을 입력하세요!");
58       $("#input[name='content']").focus();
59       return false;
60     }
61     if($("#input[name='writer']").val()==''){
62       alert("이름을 입력하세요!");
63       $("#input[name='writer']").focus();
64       return false;
65     }
66   });
67 });
68
69 </script>
70 <%@include file="../include/footer.jsp"%>
```

글 등록 기능 구현

■ 글 등록

The screenshot shows a software application interface for a "Software Project 1 Board Tutorial". On the left, there is a sidebar with various menu items: Dashboard, Charts, Tables, Forms, UI Elements, Multi-Level Dropdown, and Sample Pages. A search bar is at the top left. In the center, a modal window titled "Board Register" is open, showing fields for "Title" (with placeholder "글을 씁니다") and "Text area" (with placeholder "글을 씁니다"). Below the modal, the URL "localhost:8080/board/list" is visible. A message box in the foreground displays the text "localhost:8080 내용: 처리가 완료되었습니다." (localhost:8080 content: processing completed). At the bottom right, a table titled "Board List Page" shows a list of posts with columns: #번호 (Post ID), 제목 (Title), 작성자 (Writer), 작성일 (Created Date), and 조회수 (View Count). The table contains five entries, each with a small circular icon next to the view count.

#번호	제목	작성자	작성일	조회수
11	글을 씁니다	2017000000	2018-09-26 22:28	0
10	9번째 글을 넣었습니다.	2017000000	2018-09-26 22:26	0
9	8번째 글을 넣었습니다.	2017000000	2018-09-26 22:26	0
8	7번째 글을 넣었습니다.	2017000000	2018-09-26 22:26	0
7	6번째 글을 넣었습니다.	2017000000	2018-09-26 22:26	0

글 조회 기능 구현

■ 글 조회 컨트롤러

- URI 뒤쪽에 붙은 bno값을 받아서 DB에서 글을 조회



BoardController.java

```
37  @RequestMapping(value = "/register", method = RequestMethod.POST)
38  public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
39      Logger.info("register page post.....");
40      Logger.info(board.toString());
41      service.regist(board);
42      rttr.addFlashAttribute("msg", "SUCCESS");
43      return "redirect:/board/list";
44  }
45
46  @RequestMapping(value = "/read", method = RequestMethod.GET)
47  public String read(@RequestParam("bno") int bno,
48                      Model model) throws Exception{
49      Logger.info("read page get....");
50      model.addAttribute(service.read(bno));
51      return "board/read";
52  }
```

글 조회 기능 구현

■ 글 조회 페이지 뷰 구현

- Bno값을 input 태그로 가지고 있다가 수정페이지 혹은 삭제페이지 링크 시 같이 전달

```

read.jsp ✘
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4 <%@include file="../include/header.jsp"%>
5<div class="row">
6<div class="col-lg-12">
7<h1 class="page-header">2017000000</h1>
8</div><!-- /.col-lg-12 -->
9</div><!-- /.row -->
10<div class="row">
11<div class="col-lg-12">
12<div class="panel panel-default">
13<div class="panel-heading">Board Read Page</div>
14<!-- /.panel-heading -->
15<div class="panel-body">
16
17<div class="form-group">
18<label>Bno</label> <input class="form-control" name="bno"
19 value='<c:out value="${boardVO.bno }"/>' readonly="readonly">
20</div>
21
22<div class="form-group">
23<label>Title</label> <input class="form-control" name="title"
24 value='<c:out value="${boardVO.title }"/>' readonly="readonly">
25</div>
26
27<div class="form-group">
28<label>Text area</label>
29<textarea class="form-control" rows="3" name='content'
30 readonly="readonly"><c:out value="${boardVO.content}" /></textarea>
31</div>
32
33<div class="form-group">
34<label>Writer</label> <input class="form-control" name='writer'
35 value='<c:out value="${boardVO.writer}"/>' readonly="readonly">
36</div>
37<button id="ModiBtn" class="btn btn-default">Modify</button>
38<button id="ListBtn" class="btn btn-default" onclick="location.href='/board/list'">List</button>
39<button id="DelBtn" class="btn btn-default">Delete</button>
40</div><!-- end panel body -->
41<form id="Form" method="post">
42<input type="hidden" name="bno" value="${boardVO.bno}">
43</form>
44<!-- end panel-body -->
45</div>
46<!-- end panel -->
47</div>
48<!-- /.row -->
49<script>
50 $(document).ready(function(){
51 var Formobj= $("#Form");
52 $("#ModiBtn").on("click",function(){
53 Formobj.attr("action","/board/modify");
54 Formobj.attr("method", "get");
55 Formobj.submit();
56 });
57 $("#ListBtn").on("click", function(){
58 self.location = "/board/list";
59 });
60 $("#DelBtn").on("click", function(){
61 Formobj.attr("action","/board/remove");
62 Formobj.submit();
63 });
64 });
65 </script>
66 <%@include file="../include/footer.jsp"%>

```

글 조회 기능 구현

■ 글 조회 페이지

- bno 값에 해당되는 글 조회

The screenshot shows a web application interface for a 'Software Project 1 Board Tutorial'. The URL in the browser is `localhost:8080/board/read?bno=8`. On the left, there is a sidebar with navigation links: Dashboard, Charts, Tables, Forms, UI Elements, Multi-Level Dropdown, and Sample Pages. A search bar is also present in the sidebar. The main content area displays a large title '2017000000' and a section titled 'Board Read Page'. Below this, there are fields for 'Bno' (containing '8'), 'Title' (containing '7번째 글을 넣었습니다.'), and a 'Text area' (containing '새로운 글을 넣습니다.'). At the bottom, there is a 'Writer' field containing '2017000000' and three buttons: 'Modify', 'List', and 'Delete'.

글 수정 기능 구현

■ 글 수정 컨트롤러 구현

- modify 페이지로 접속을 위한 get 방식, 수정 후 데이터 변경을 위한 post 방식

```
53
54 @RequestMapping(value = "/modify", method = RequestMethod.GET)
55 public String modifyGET(@RequestParam("bno") int bno, Model model) throws Exception{
56     Logger.info("modify page get....");
57     model.addAttribute(service.read(bno));
58     return "board/modify";
59 }
60
61 @RequestMapping(value = "/modify", method = RequestMethod.POST)
62 public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception{
63     Logger.info("modify page post.....");
64     Logger.info(board.toString());
65     service.modify(board);
66     rttr.addFlashAttribute("msg", "SUCCESS");
67     return "redirect:/board/list";
68 }
69 }
```

글 수정 기능 구현

■ 글 수정 뷰 구현

- form 태그를 사용하여 Post 방식으로 /board/modify로 전송

```

modify.jsp ✘
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4 <%@include file="../include/header.jsp"%>
5<div class="row">
6<div class="col-lg-12">
7<h1 class="page-header">2017000000</h1>
8</div><!-- /.col-lg-12 -->
9</div><!-- /.row -->
10<div class="row">
11<div class="col-lg-12">
12<div class="panel panel-default">
13
14<div class="panel-heading">Board Modify Page</div>
15<!-- .panel-heading -->
16<div class="panel-body">
17
18<form role="form" method="post" action="/board/modify" >
19<div class="form-group">
20<label>Bno</label> <input class="form-control" name='bno'
21 value="${boardVO.bno}" readonly="readonly"/>
22</div>
23<div class="form-group">
24<label>Title</label> <input class="form-control" name='title'
25 value="${boardVO.title}" >
26</div>
27<div class="form-group">
28<label>Text area</label>
29<textarea class="form-control" rows="3" name='content'
30 ><c:out value="${boardVO.content}" /></textarea>
31</div>
32<div class="form-group">
33<label>Writer</label> <input class="form-control" name='writer'
34 value="${boardVO.writer}" readonly="readonly">
35</div>

```

title, content
readonly 제거

```

modify.jsp ✘
36
37<button id="ModiBtn" class="btn btn-default">Modify</button>
38<button id="ListBtn" class="btn btn-default">List</button>
39</form>
40</div>
41<!-- end panel-body -->
42
43</div>
44<!-- end panel-body -->
45</div>
46<!-- end panel -->
47</div>
48<!-- /.row -->
49<script>
50 $(document).ready(function(){
51 $("#ListBtn").on("click",function(){
52 var formobj=$("#form[role='form']");
53 formobj.attr("action","/board/list");
54 formobj.attr("method","get");
55 formobj.submit();
56 });
57 $("#ModiBtn").on("click",function(){
58 if($("#input[name=title]").val()==''){
59 alert("제목을 입력하세요!");
60 $("#input[name='title']").focus();
61 return false;
62 }
63 if($("#input[name=content]").val()==''){
64 alert("내용을 입력하세요!");
65 $("#input[name='content']").focus();
66 return false;
67 }
68 return true;
69 });
70 });
71</script>
72
73 <%@include file="../include/footer.jsp"%>

```

글 수정 기능 구현

■ 글 수정

The screenshot illustrates the process of modifying a board entry. It consists of two main parts: a left sidebar and a right content area.

Left Sidebar: Labeled "Software Project 1 Board Tutorial". It contains a search bar, a dashboard icon, and a navigation menu with items: Charts, Tables, Forms, UI Elements, Multi-Level Dropdown, and Sample Pages. Below the menu is a "Writer" section showing "2017000000" and buttons for "Modify" and "List".

Right Content Area: Labeled "Board Modify Page". It shows a form with fields: "Bno" (11), "Title" (글을 수정합니다), and "Text area" (글을 수정할게요). A message box displays the URL "localhost:8080/board/list".

Bottom Right: A separate window titled "Software Project 1 Board Tutorial" shows the "Board List Page". It lists one item: "#번호 11" and "제목 글을 수정합니다".

Message Box: A central message box contains the text "localhost:8080 내용: 처리가 완료되었습니다."

글 삭제 기능 구현

■ 글 삭제 컨트롤러

- read 페이지에서 삭제 버튼으로 바로 삭제
- 삭제 후 메시지 전송, 별도의 jsp 파일 구현 X

```
@RequestMapping(value = "/remove", method = RequestMethod.POST)
public String remove(@RequestParam("bno") int bno,
                     RedirectAttributes rttr) throws Exception{
    service.remove(bno);
    Logger.info("remove page post.....");
    rttr.addFlashAttribute("msg", "SUCCESS");
    return "redirect:/board/list";
}
```

글 삭제 기능 구현

■ 글 삭제

The image shows two screenshots of a web application interface. On the left, a 'Board Read Page' is displayed with a large title '2017000000'. The page includes fields for 'Bno' (11), 'Title' ('글을 수정합니다'), 'Text area' ('글을 수정할게요'), and 'Writer' ('2017000000'). Below these are buttons for 'Modify', 'List', and 'Delete'. A blue arrow points from this screen to the right. On the right, a 'Board List Page' is shown with a table of entries:

#번호	제목	작성자	작성일
10	9번째 글을 넣었습니다.	2017000000	2018-09-26 22:26
9	8번째 글을 넣었습니다.	2017000000	2018-09-26 22:26
8	7번째 글을 넣었습니다.	2017000000	2018-09-26 22:26
7	6번째 글을 넣었습니다.	2017000000	2018-09-26 22:26
6	5번째 글을 넣었습니다.	2017000000	2018-09-26 22:26

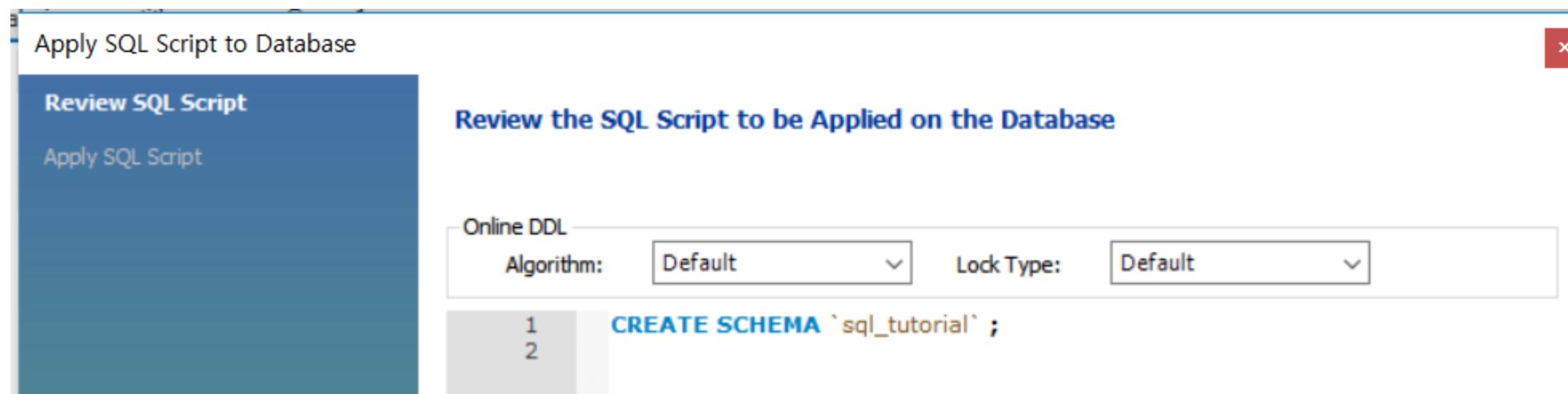
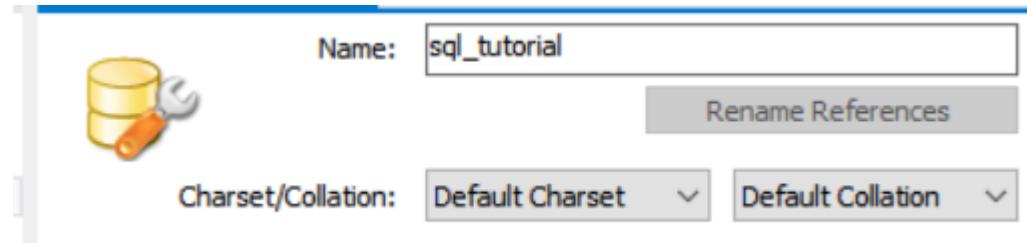
A modal window titled 'localhost:8080 내용:' is open, displaying the message '처리가 완료되었습니다.' (Processing completed). A blue '확인' (Confirm) button is visible in the bottom right of the modal.

5. MySQL 기본

USE 명령어

■ 스키마 생성

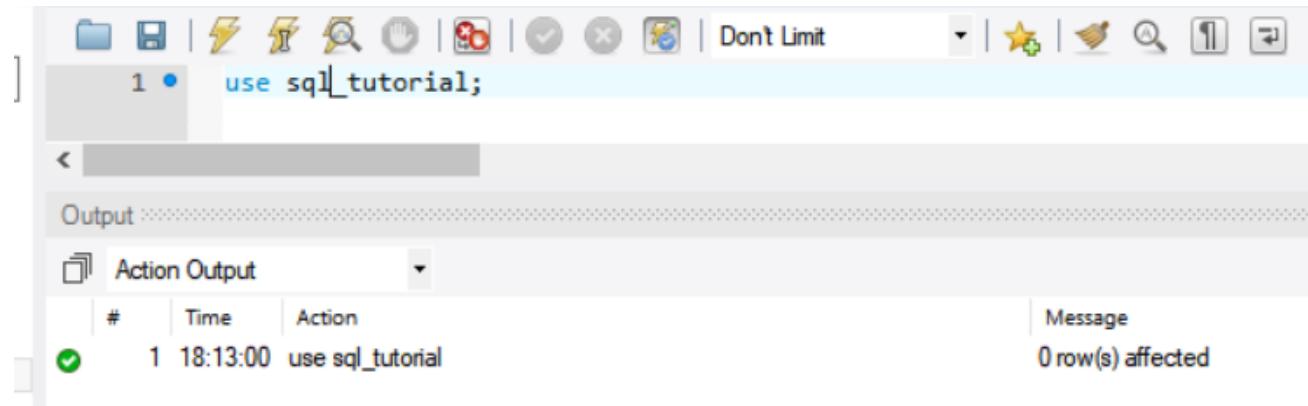
- "sqlTutorial"이라는 이름으로 생성



USE 명령어

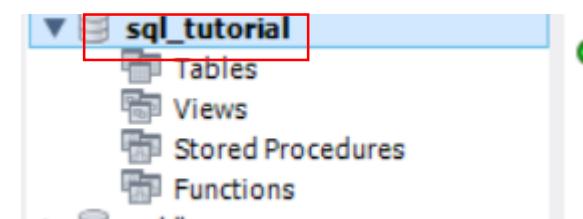
■ USE 구문

- 여러 개의 DB가 MySQL에 존재 가능한 상황에서 어느 DB에 쿼리를 날릴지 지정



The screenshot shows the MySQL Workbench interface. In the top query editor window, the command `use sqlTutorial;` is entered. Below it, the 'Output' pane displays the result of the command: a single row in the 'Action Output' table with a green checkmark, the time `18:13:00`, and the action `use sqlTutorial`. To the right, a message box shows `0 row(s) affected`. The database tree on the left has a node for `sqlTutorial`, which is highlighted with a red box.

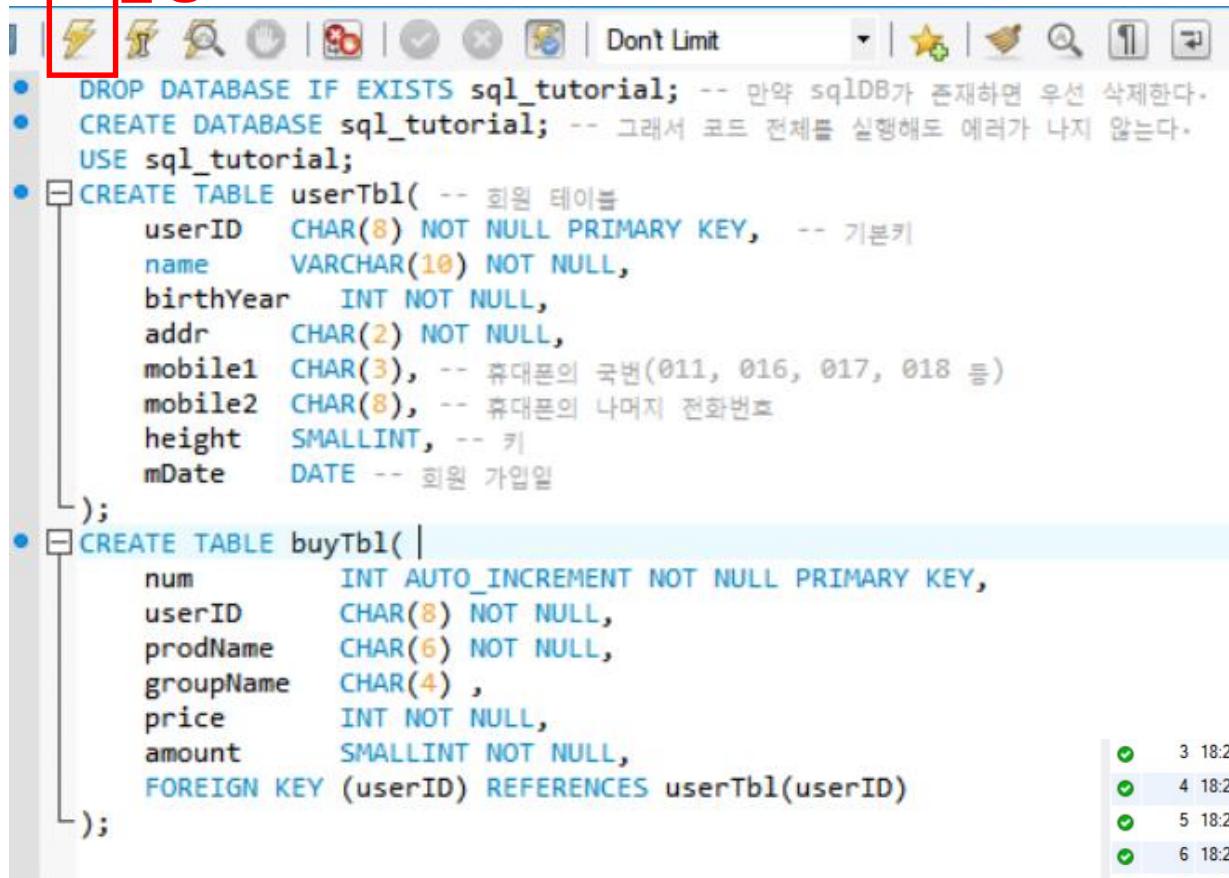
- USE 명령어로 지정된 DB는 진한 글씨로 변환



테이블 생성

■ CREATE 명령으로 userTbl 생성

실행



```

• DROP DATABASE IF EXISTS sqlTutorial; -- 만약 sqlDB가 존재하면 우선 삭제한다.
• CREATE DATABASE sqlTutorial; -- 그래서 코드 전체를 실행해도 에러가 나지 않는다.
USE sqlTutorial;
• CREATE TABLE userTbl( -- 회원 테이블
    userID CHAR(8) NOT NULL PRIMARY KEY, -- 기본키
    name VARCHAR(10) NOT NULL,
    birthYear INT NOT NULL,
    addr CHAR(2) NOT NULL,
    mobile1 CHAR(3), -- 휴대폰의 국번(011, 016, 017, 018 등)
    mobile2 CHAR(8), -- 휴대폰의 나머지 전화번호
    height SMALLINT, -- 키
    mDate DATE -- 회원 가입일
);
• CREATE TABLE buyTbl(
    num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    userID CHAR(8) NOT NULL,
    prodName CHAR(6) NOT NULL,
    groupName CHAR(4),
    price INT NOT NULL,
    amount SMALLINT NOT NULL,
    FOREIGN KEY (userID) REFERENCES userTbl(userID)
);

```

-- : 주석처리

PRIMARY KEY : 기본 키, 구별을 위한 값으로
중복을 허용하지 않음

AUTO_INCREMENT : 자동으로 숫자를 증가
NOT NULL : NULL을 허용하지 않음

3	18:29:54	DROP DATABASE IF EXISTS sqlTutorial	1 row(s) affected
4	18:29:54	CREATE DATABASE sqlTutorial	1 row(s) affected
5	18:29:54	USE sqlTutorial	0 row(s) affected
6	18:29:54	CREATE TABLE userTbl(-- 회원 테이블 userID CHAR(8) NOT NULL PRIMARY KEY, -- 기본키 name VAR... 0 row(s) affected	
7	18:29:54	CREATE TABLE buyTbl(num INT AUTO_INCREMENT NOT NULL PRIMARY KEY, userID CHAR(8)... 0 row(s) affected	

INSERT 문

■ INSERT 문법

- 1. INSERT INTO 테이블이름 (필드이름1, 필드이름2, 필드이름3, ...) VALUES (데이터값1, 데이터값2, 데이터값3, ...)
- 2. INSERT INTO 테이블이름
VALUES (데이터값1, 데이터값2, 데이터값3, ...)

```
• INSERT INTO userTbl  (userID, name, birthYear, addr, mobile1, mobile2, height, mDate)
  VALUES('LSG','이승기', 1987, '서울', '011','1111111', 182,'2008-08-08');

• INSERT INTO userTbl VALUES('KBS','김범수', 1979, '경남', '011','2222222', 173,'2012-04-04');
• INSERT INTO userTbl VALUES('KHH','김경호', 1971, '전남', '019','3333333', 177,'2007-07-07');
• INSERT INTO userTbl VALUES('JYP','조웅필', 1950, '경기', '011','4444444', 166,'2009-04-04');
• INSERT INTO userTbl VALUES('SSK','성시경', 1979, '서울', NULL, NULL, 186,'2013-12-12');
• INSERT INTO userTbl VALUES('LJB','임재범', 1963, '서울', '016','6666666', 182,'2009-09-09');
• INSERT INTO userTbl VALUES('YJS','윤중선', 1969, '경남', NULL, NULL, 170,'2005-05-05');
• INSERT INTO userTbl VALUES('EJW','은지원', 1978, '경북', '011','8888888', 174,'2014-03-03');
• INSERT INTO userTbl VALUES('JKW','조관우', 1965, '경기', '018','9999999', 172,'2010-10-10');
• INSERT INTO userTbl VALUES('BBK','바비킴', 1973, '서울', '011','0000000', 176,'2013-05-05');

• INSERT INTO buyTbl VALUES(NULL, 'KBS', '운동화', NULL, 30, 2);
• INSERT INTO buyTbl VALUES(NULL, 'KBS', '노트북', '전자', 1000, 1);
• INSERT INTO buyTbl VALUES(NULL, 'JYP', '모니터', '전자', 200, 1);
• INSERT INTO buyTbl VALUES(NULL, 'BBK', '모니터', '전자', 200, 5);
• INSERT INTO buyTbl VALUES(NULL, 'KBS', '정바지', '의류', 50, 3);
• INSERT INTO buyTbl VALUES(NULL, 'BBK', '메모리', '전자', 80, 10);
• INSERT INTO buyTbl VALUES(NULL, 'SSK', '책', '서적', 15, 5);
• INSERT INTO buyTbl VALUES(NULL, 'EJW', '책', '서적', 15, 2);
• INSERT INTO buyTbl VALUES(NULL, 'EJW', '청바지', '의류', 50, 1);
• INSERT INTO buyTbl VALUES(NULL, 'BBK', '운동화', NULL, 30, 2);
• INSERT INTO buyTbl VALUES(NULL, 'EJW', '책', '서적', 15, 1);
• INSERT INTO buyTbl VALUES(NULL, 'BBK', '운동화', NULL, 30, 2);
```

SELECT 문

- SELECT * FROM 테이블명 : FROM에 * 붙이면 전체 조회
- WHERE를 붙이면 조건을 불일 수 있음

The figure consists of three side-by-side screenshots of MySQL Workbench. Each screenshot shows a SQL query in the top pane and its results in the bottom pane.

Screenshot 1:

```

3
4 •  SELECT * FROM usertbl;

```

Result Grid:

	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
1	BBK	바비킴	1973	서울	011	0000000	176	2013-05-05
2	EJW	은지원	1978	경북	011	8888888	174	2014-03-03
3	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
4	JYP	조용풀	1950	경기	011	4444444	166	2009-04-04
5	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
6	KHH	김경호	1971	전남	019	3333333	177	2007-07-07
7	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
8	LSG	이승기	1987	서울	011	1111111	182	2008-08-08
9	SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
10	YJS	윤종신	1969	경남	NULL	NULL	170	2005-05-05
11	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Screenshot 2:

```

1
2 •  use sqlTutorial;
3 •  SELECT * FROM buytbl;
4
5

```

Result Grid:

	num	userID	prodName	groupName	price	amount
1	1	KBS	운동화	NULL	30	2
2	2	KBS	노트북	전자	1000	1
3	3	JYP	모니터	전자	200	1
4	4	BBK	모니터	전자	200	5
5	5	KBS	청바지	의류	50	3
6	6	BBK	메모리	전자	80	10
7	7	SSK	책	서적	15	5
8	8	EJW	책	서적	15	2
9	9	EJW	청바지	의류	50	1
10	10	BBK	운동화	NULL	30	2
11	11	EJW	책	서적	15	1
12	12	BBK	운동화	NULL	30	2
13	NULL	NULL	NULL	NULL	NULL	NULL

Screenshot 3:

```

2 •  use sqlTutorial;
3 •  SELECT * FROM buytbl WHERE userID='KBS';
4
5

```

Result Grid:

	num	userID	prodName	groupName	price	amount
1	1	KBS	운동화	NULL	30	2
2	2	KBS	노트북	전자	1000	1
3	5	KBS	청바지	의류	50	3
4	NULL	NULL	NULL	NULL	NULL	NULL

- `SELECT * FROM sqlTutorial.buytbl;` 이런 방식으로 스키마명을 붙여서도 사용가능

SELECT 문

■ AND, OR

```

3 •  SELECT userID, Name
4   FROM userTbl
5   WHERE birthYear >= 1970 OR height >= 182;
6

```

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: []

userID	Name
BBK	바비김
EJW	은지원
KBS	김범수
KHH	김경호
LJB	임재범
LSG	이승기
SSK	성시경
NUL	NUL

```

3 •  SELECT userID, Name
4   FROM userTbl
5   WHERE height >= 180 AND height >= 182;
6

```

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: []

userID	Name
LJB	임재범
LSG	이승기
SSK	성시경
NUL	NUL

■ BETWEEN

```

SELECT Name, height
  FROM userTbl
 WHERE height >= 180 AND height <= 183;

```

```

SELECT Name, height
  FROM userTbl
 WHERE height BETWEEN 180 AND 183;

```



→

name	height
임재범	182
이승기	182

SELECT 문

■ ORDER BY

- 해당 값에 대해서 오름차순 또는 내림차순으로 SELECT 가능

10
11 • `SELECT Name, mDate FROM userTbl ORDER BY mDate;`

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

Name	mDate
윤종신	2005-05-05
김경호	2007-07-07
이승기	2008-08-08
조용필	2009-04-04

17
18 • `SELECT name, mDate, height FROM userTbl ORDER BY height DESC, name ASC;`

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

name	mDate	height
성시경	2013-12-12	186
이승기	2008-08-08	182
임재범	2009-09-09	182
김경호	2007-07-07	177
바비킴	2013-05-05	176
은지원	2014-03-03	174

10
11 • `SELECT Name, mDate FROM userTbl ORDER BY mDate DESC;`

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

Name	mDate
은지원	2014-03-03
성시경	2013-12-12
바비킴	2013-05-05
김병수	2012-04-04

Height 기준 내림차순
으로 정렬하고 키가 동
일하다면 이름 순으로
오름차순 정렬

SELECT 문

■ GROUP BY

- 그룹을 묶어주는 구문
- 중복되는 ID 값이 있고 그것들의 합을 구할 때 사용

```
19  
20  
21 • USE sql_tutorial;  
22 • SELECT userID, amount FROM buyTbl ORDER BY userID;  
23
```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

userID	amount
BBK	5
BBK	10
BBK	2
BBK	2
EJW	2
EJW	1
EJW	1
JYP	1
KBS	2



```
3  
4 • SELECT userID, SUM(amount) FROM buyTbl GROUP BY userID;  
5 -- 집계 합수이다.
```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

userID	SUM(amount)
BBK	19
EJW	4
JYP	1
KBS	6
SSK	5

SELECT 문

■ LIMIT

- 출력되는 레코드 개수를 제한

1 • SELECT * FROM sqlTutorial.buytbl;

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	NULL	30	2	
2	KBS	노트북	전자	1000	1	
3	JYP	모니터	전자	200	1	
4	BBK	모니터	전자	200	5	
5	KBS	청바지	의류	50	3	
6	BBK	메모리	전자	80	10	
7	SSK	책	서적	15	5	
8	EJW	책	서적	15	2	
9	EJW	청바지	의류	50	1	
10	BBK	운동화	NULL	30	2	
11	EJW	책	서적	15	1	
12	BBK	운동화	NULL	30	2	
NULL	NULL	NULL	NULL	NULL	NULL	NULL



2 • SELECT COUNT(*) FROM sqlTutorial.buytbl;

count(*)
12



테이블 전체 행 개수

1 • SELECT * FROM sqlTutorial.buytbl LIMIT 5; -- 개수

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	NULL	30	2	
2	KBS	노트북	전자	1000	1	
3	JYP	모니터	전자	200	1	
4	BBK	모니터	전자	200	5	
5	KBS	청바지	의류	50	3	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

1 • SELECT * FROM sqlTutorial.buytbl LIMIT 3, 5; -- 시작 인덱스, 개수

	num	userID	prodName	groupName	price	amount
4	BBK	모니터	전자	200	5	
5	KBS	청바지	의류	50	3	
6	BBK	메모리	전자	80	10	
7	SSK	책	서적	15	5	
8	EJW	책	서적	15	2	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

UPDATE 문

■ 데이터를 수정하는 경우

- WHERE를 쓰지 않을 경우 전체가 다 바뀔 수도 있으니 주의

```

1 •  SELECT * FROM sqlTutorial.buyTbl;
2
3
4
5
6
7
8
9
10

```

Result Grid | Filter Rows: Edit:

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	HULL		30	2
2	KBS	노트북	전자		1000	1
3	JYP	모니터	전자		200	1
4	BBK	모니터	전자		200	5
5	KBS	청바지	의류		50	3
6	BBK	메모리	전자		80	10
7	SSK	책	서적		15	5
8	EJW	책	서적		15	2
9	EJW	청바지	의류		50	1
10	BBK	운동화	HULL		30	2
11	EJW	책	서적		15	1
12	BBK	운동화	HULL		30	2
	HULL	HULL	HULL	HULL	HULL	HULL



```

5
6 • UPDATE buyTbl
7     SET groupName = '의류'
8     WHERE prodName = '운동화';
9
10

```

Result Grid | Filter Rows: Edit:

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	의류		30	2
2	KBS	노트북	전자		1000	1
3	JYP	모니터	전자		200	1
4	BBK	모니터	전자		200	5
5	KBS	청바지	의류		50	3
6	BBK	메모리	전자		80	10
7	SSK	책	서적		15	5
8	EJW	책	서적		15	2
9	EJW	청바지	의류		50	1
10	BBK	운동화	의류		30	2
11	EJW	책	서적		15	1
12	BBK	운동화	의류		30	2
	HULL	HULL	HULL	HULL	HULL	HULL

DELETE 문

- 데이터를 삭제하는 경우
 - 반드시 WHERE를 생각할 것

The diagram illustrates the deletion of a specific row from a database table. On the left, a screenshot of MySQL Workbench shows a SELECT query and its result grid. On the right, another screenshot shows a DELETE query with a WHERE clause and its result grid, where the row matching the WHERE condition is missing.

Left Screenshot (Before Deletion):

```
1 •   SELECT * FROM sqlTutorial.buytbl;
```

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	의류	30	2	
2	KBS	노트북	전자	1000	1	
3	JYP	모니터	전자	200	1	
4	BBK	모니터	전자	200	5	
5	KBS	청바지	의류	50	3	
6	BBK	메모리	전자	80	10	
7	SSK	책	서적	15	5	
8	EJW	책	서적	15	2	
9	EJW	청바지	의류	50	1	
10	BBK	운동화	의류	30	2	
11	EJW	책	서적	15	1	
12	BBK	운동화	의류	30	2	
	NUL	NUL	NUL	NUL	NUL	NUL

Right Screenshot (After Deletion):

```
3  
4 •   DELETE FROM buytbl WHERE userID='JYP';  
5  
6 •   SELECT * FROM sqlTutorial.buytbl;
```

	num	userID	prodName	groupName	price	amount
1	KBS	운동화	의류	30	2	
2	KBS	노트북	전자	1000	1	
4	BBK	모니터	전자	200	5	
5	KBS	청바지	의류	50	3	
6	BBK	메모리	전자	80	10	
7	SSK	책	서적	15	5	
8	EJW	책	서적	15	2	
9	EJW	청바지	의류	50	1	
10	BBK	운동화	의류	30	2	
11	EJW	책	서적	15	1	
12	BBK	운동화	의류	30	2	
	NUL	NUL	NUL	NUL	NUL	NUL

6. 각종 TIPS

자주 쓰는 이클립스 단축키

■ 실행

- Ctrl + F11
- Alt + Shift + X : 누르고 나면 옆에 창이 뜨는데 선택하여 실행하려는 것 선택

■ 소스 네비게이션

- Ctrl + 마우스 커서(또는 F3)
- Alt + →, Alt + ← : 이전 화면 혹은 이후 화면
- Ctrl + O : 해당 소스의 메소드 리스트를 확인하려 할 때
- F4 : 클래스명을 선택하고 누르면 해당 클래스의 Hierarchy 를 조회

■ 문자열 찾기

- Ctrl + K : 찾을 문자열 블록으로 지정하고 검색
- Ctrl + Shift + K : 밑에서 문자열 검색
- Ctrl + F : 기본 검색

■ 소스편집

- Ctrl + Space : 입력 중에 완성 기능 호출
 - ✓ syso 입력한 후 Ctrl + Space : System.out.println() 자동완성,
 - ✓ for 입력한 후 Ctrl + Space : for문 자동 완성,
 - ✓ try 입력한 후 Ctrl + Space : try-catch 자동완성
- F2 : 컴파일 에러 줄에 커서를 놓고 이 키를 누르면 에러의 힌트 제공
 - ✓ Import, 예외 throw ,build path 등록, interface implement시 함수 자동완성

자주 쓰는 이클립스 단축키

- Ctrl + L : 지정한 소스 줄로 이동
- Ctrl + Shift + Space : 메소드 괄호에 커서를 놓으면 파라미터 힌트 보여줌
- Ctrl + D : 한 줄 삭제
- Ctrl + W : 파일 닫기
- Ctrl + i : 들여쓰기 자동 설정
- Ctrl + Shift + F : 줄 정렬
- Ctrl + Shift + / : 블록을 주석으로 처리 /* */
- Ctrl + Shift + W : 블록 주석을 해제
- Ctrl + / : 라인/블록 주석 처리
- Ctrl + T : 상속 계층 팝업 창
- Ctrl + O : 메소드나 필드 이동
- F3 : 선언된 변수나 메소드 정의로 이동
- Ctrl + Shift + O : 자동 import
- Alt + Shift + S : 소스 메뉴 출력(Import, getter setter, toString, Construct, hashCode, equal 자동완성 등)
- Ctrl + Shift + X : 대문자로 변환
- Ctrl + Shift + Y : 소문자로 변환

자주 쓰는 이클립스 단축키

■ 디버그

- F11 : 디버깅 시작
- F8 : 디버깅 계속
- F6 : 한 줄 씩 디버깅
- F5 : 한 줄 씩 디버깅 (함수 내부까지 디버깅)

스프링에서 주로 사용하는 애노테이션 정리

애노테이션	설명	사용
@Controller	스프링 MVC 컨트롤러 객체임을 명시하는 애노테이션	클래스
@RequestMapping	특정 URL에 매칭되는 클래스나 메소드임을 명시하는 애노테이션	클래스, 메소드
@RequestParam	요청에서 특정한 파라미터의 값을 찾아낼 때 사용하는 애노테이션	파라미터
@RequestHeader	요청에서 특정 HTTP 헤더 정보를 추출할 때 사용	파라미터
@PathVariable	현재 URI에서 원하는 정보를 추출할 때 사용	파라미터
@ModelAttribute	자동으로 해당 객체를 뷰까지 전달하도록 만드는 애노테이션	메소드, 파라미터
@ResponseBody	리턴 타입이 HTTP의 응답 메시지로 전송	메소드, 리턴타입
@RequestBody	요청 문자열이 그대로 파라미터로 전달	파라미터
@Repository	DAO 객체	클래스
@Service	서비스 객체	클래스
@InitBinder	파라미터를 수집해서 객체로 만들 경우에 커스터마이징	메소드
@CookieValue	현재 사용자의 쿠키가 존재하는 경우 쿠키의 이름을 이용해서 쿠키의 값을 추출	파라미터

netstat 사용법 및 프로세스 죽이기

- 특정 소프트웨어를 실행 시킬려고 하지만 접속하려는 포트가 이미 사용 중이어서 포트 충돌이 날 때
 - netstat -ano | findstr <port number> 로 포트번호를 이용하는 프로세스 조회
 - taskkill -f /pid <찾은 프로세스id> (-f옵션은 강제종료)

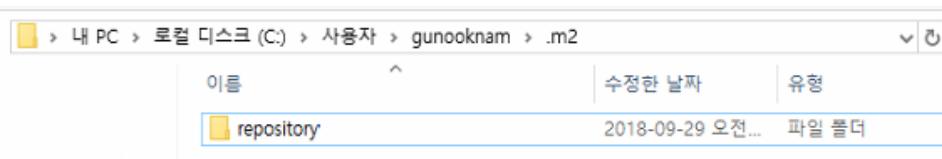
관리자: 명령 프롬프트

```
C:\WINDOWS\system32>netstat -ano | findstr 8080
TCP    0.0.0.0:8080          0.0.0.0:0              LISTENING
TCP    [::]:8080            [::]:0                  LISTENING
25276
25276
```

C:\WINDOWS\system32>taskkill -f /pid 25276
성공: 프로세스(PID 25276)가 종료되었습니다.

Maven Build 에러 해결방법

- 메이븐 업데이트
 - 해당 프로젝트에 오른쪽 마우스 버튼 클릭 -> [maven] -> [update project]
-> Force update of snapshot/release 체크 후 OK 클릭
- Clean Project
 - 프레임워크(스프링 혹은 이클립스) 상단 작업표시줄의 [project] -> [clean]
- .m2폴더의 repository 삭제 후 이클립스 재시작
 - pom.xml의 내용 재빌드



웹 개발 IDE

■ Sublime text

- <https://www.sublimetext.com/>
- 참고 동영상 - <https://opentutorials.org/module/406/3595>

■ Atom

- <https://atom.io/>
- 참고 동영상 - <https://opentutorials.org/module/1579>

■ Brackets

- <http://brackets.io/>
- 참고 동영상 - <https://opentutorials.org/course/2418/13455>

■ Visual Studio Code

- <https://code.visualstudio.com/>
- 참고 동영상 - <https://www.youtube.com/watch?v=jR2zWjCT2XI&list=PLG7te9eYUi7vnribGociCy0Z-yD9Q8hwT>

References

- 코드로 배우는 스프링 웹 프로젝트
- 생활코딩 - Opentutorial
- 개발자를 위한 스프링 프레임워크 원리부터 실전까지
- 이것이 MySQL이다.
- do it 자바스크립트 + 제이쿼리 입문

END OF PRESENTATION

해당 TUTORIAL 관련 질문사항은 ngotic@kw.ac.kr 로 ...