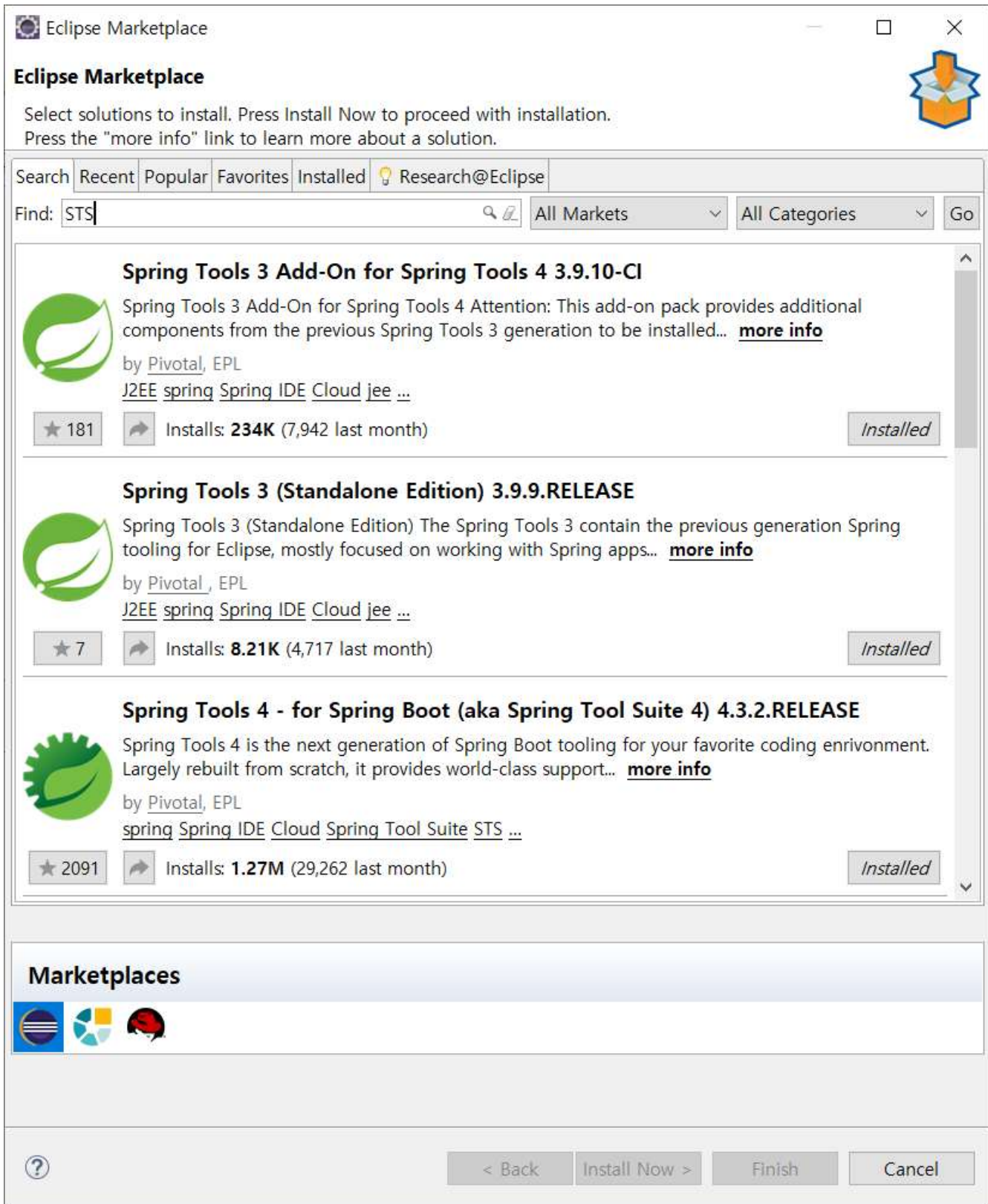


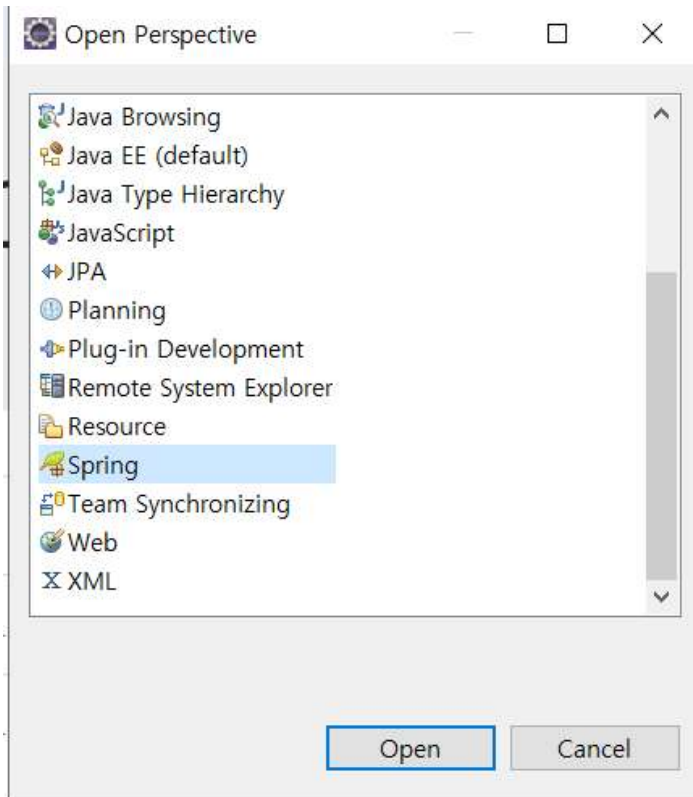
작업 1. 스프링 개발환경 구축

방법1.

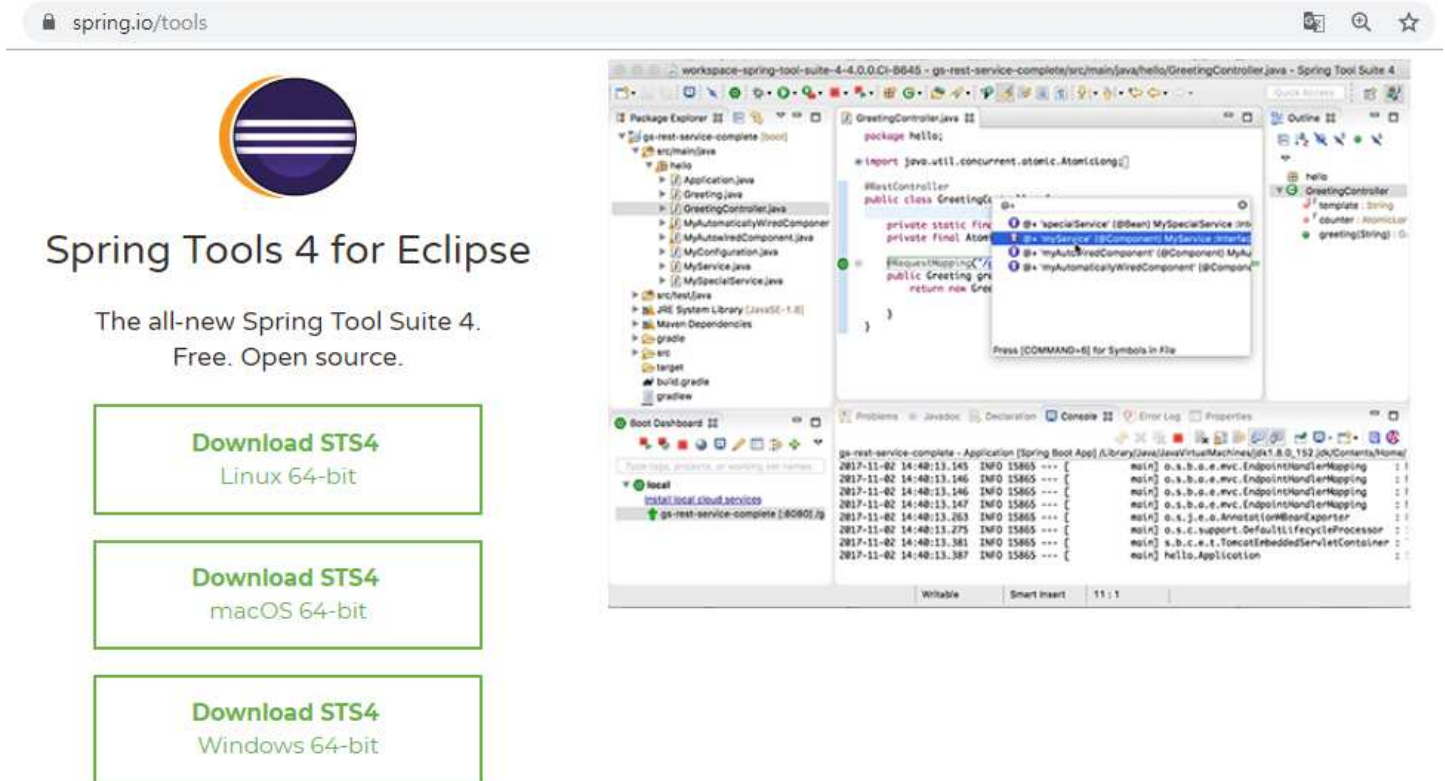
eclipse의 Marketplace에서 STS를 입력 검색하여 Spring Tools를 설치한다. 이미 설치된 경우에는 installed라고 표시된다.



Spring Tools가 설치되면 perspective를 추가한다.



방법 2. spring.io/tools 사이트를 방문하여 Eclipse용 STS를 다운로드하여 사용할 수 있다.



The screenshot shows the Spring Tools 4 for Eclipse download page on the left and a screenshot of the Spring Tool Suite 4 IDE on the right. The IDE shows a project named 'workspace-spring-tool-suite-4-4.0.0.CI-8645' with a package 'hello' containing a 'GreetingController.java' file. The code in the IDE is as follows:

```
package hello;

import java.util.concurrent.atomic.AtomicLong;

@RestController
public class GreetingController {

    private static final AtomicLong counter = new AtomicLong(0);

    @RequestMapping("/hello")
    public Greeting getGreeting() {
        return new Greeting(counter.incrementAndGet(), "Hello, World!");
    }
}
```

The download page on the left has the following content:

Spring Tools 4 for Eclipse
The all-new Spring Tool Suite 4.
Free. Open source.

Download STS4
Linux 64-bit

Download STS4
macOS 64-bit

Download STS4
Windows 64-bit

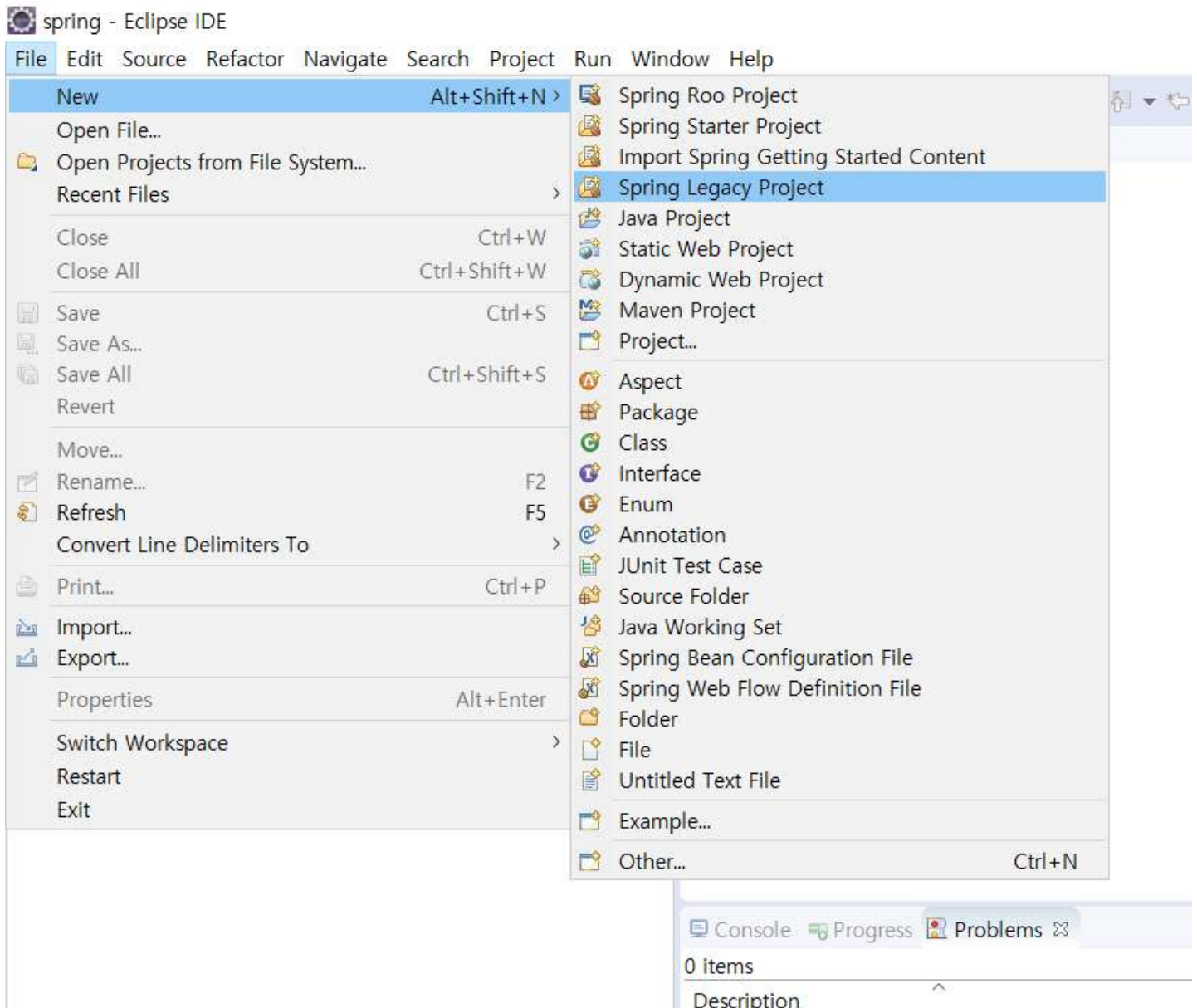
압축을 해제한 후 SpringToolSuite4.exe 파일을 실행한다. 에러가 발생하는 경우에는 eclipse.exe를 실행하여도 된다.

spring-tool-suite-4-4.3.2.RELEASE-e4.12.0-win32.win32.x86_64 > sts-4.3.2.RELEASE				
이름	수정한 날짜	유형	크기	
configuration	2019-09-01 오전 11:55	파일 폴더		
dropins	2019-08-12 오후 11:31	파일 폴더		
features	2019-09-01 오전 11:55	파일 폴더		
META-INF	2019-09-01 오전 11:55	파일 폴더		
p2	2019-09-01 오전 11:55	파일 폴더		
plugins	2019-09-01 오전 11:55	파일 폴더		
readme	2019-09-01 오전 11:55	파일 폴더		
.eclipseproduct	2019-03-08 오전 7:42	ECLIPSEPRODUCT...	1KB	
artifacts.xml	2019-08-12 오후 11:31	XML 문서	174KB	
eclipse.exe	2019-08-12 오후 11:29	응용 프로그램	120KB	
license.txt	2019-08-12 오후 11:27	텍스트 문서	12KB	
open-source-licenses.txt	2019-08-12 오후 11:31	텍스트 문서	934KB	
SpringToolSuite4.exe	2019-08-12 오후 11:29	응용 프로그램	408KB	
SpringToolSuite4.ini	2019-08-12 오후 11:31	구성 설정	1KB	

작업 2. 게시판 만들기 준비

2-1. 프로젝트 생성

File > New > Spring Legacy Project를 선택하고



프로젝트 이름을 입력하고 Templates에서 Spring MVC Project를 선택하고 Next를 클릭한다.

New Spring Legacy Project

Spring Legacy Project

Click 'Next' to load the template contents.

Project name:

☒ Use default location

Location:

Select Spring version:

Templates:

- ▼ Simple Projects
 - Simple Java
 - Simple Spring Maven
 - Simple Spring Web Maven
- > Batch
- > GemFire
- > Integration
- > Persistence
- Simple Spring Utility Project
- Spring MVC Project**

requires downloading [Configure templates...](#)

Description:
A new Spring MVC web application development project

URL:

Working sets

☐ Add project to working sets

Working sets:

top-level package 이름을 정의한다. 예는 com.swcodingschool.controller로 입력.

New Spring Legacy Project

Project Settings - Spring MVC Project

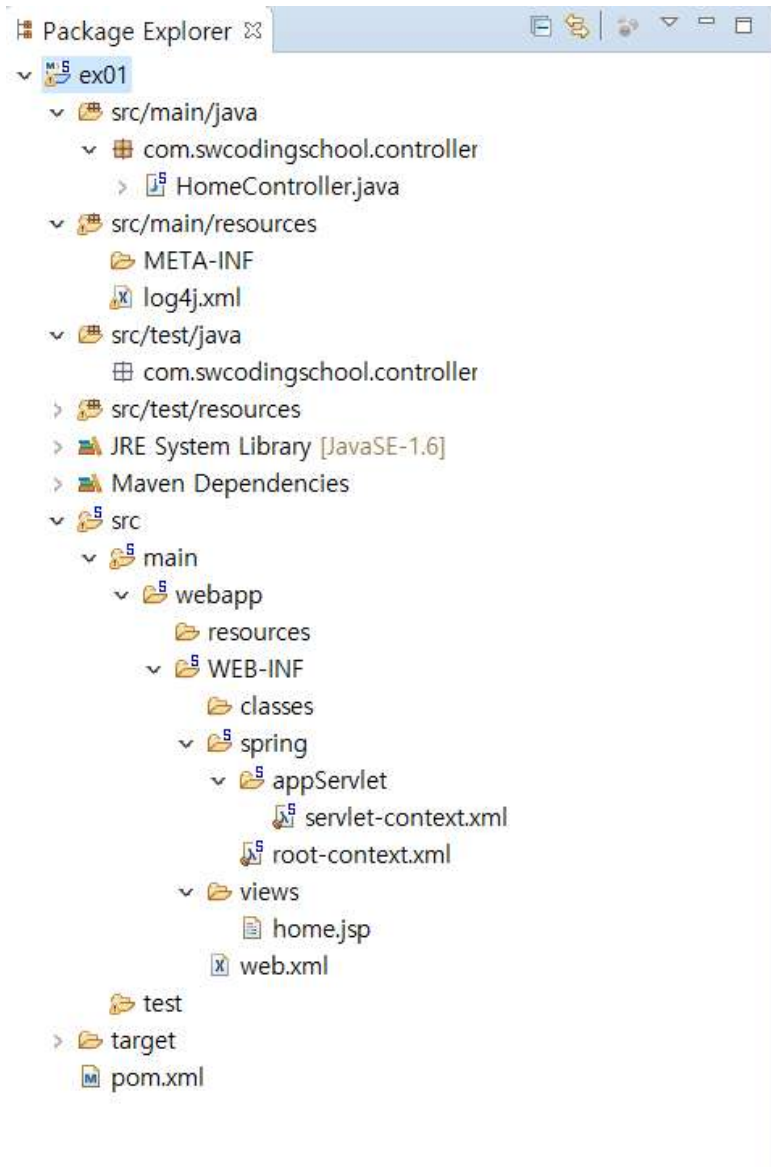
Define project specific settings. Required settings are denoted by "*".

Please specify the top-level package e.g. com.mycompany.myapp*

com.swcodingschool.controller

? < Back Next > Finish Cancel

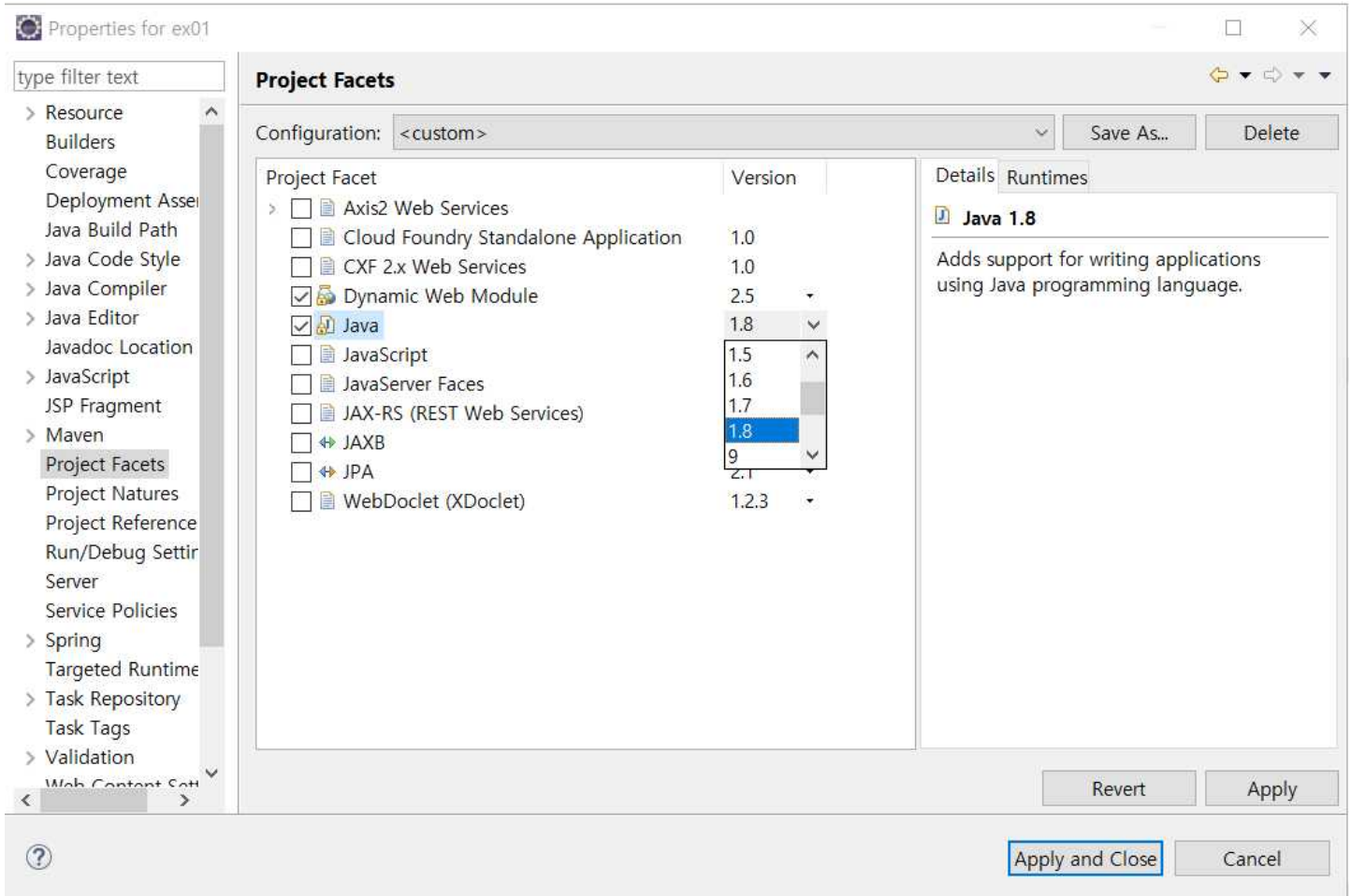
생성된 프로젝트의 전체 구조이다.



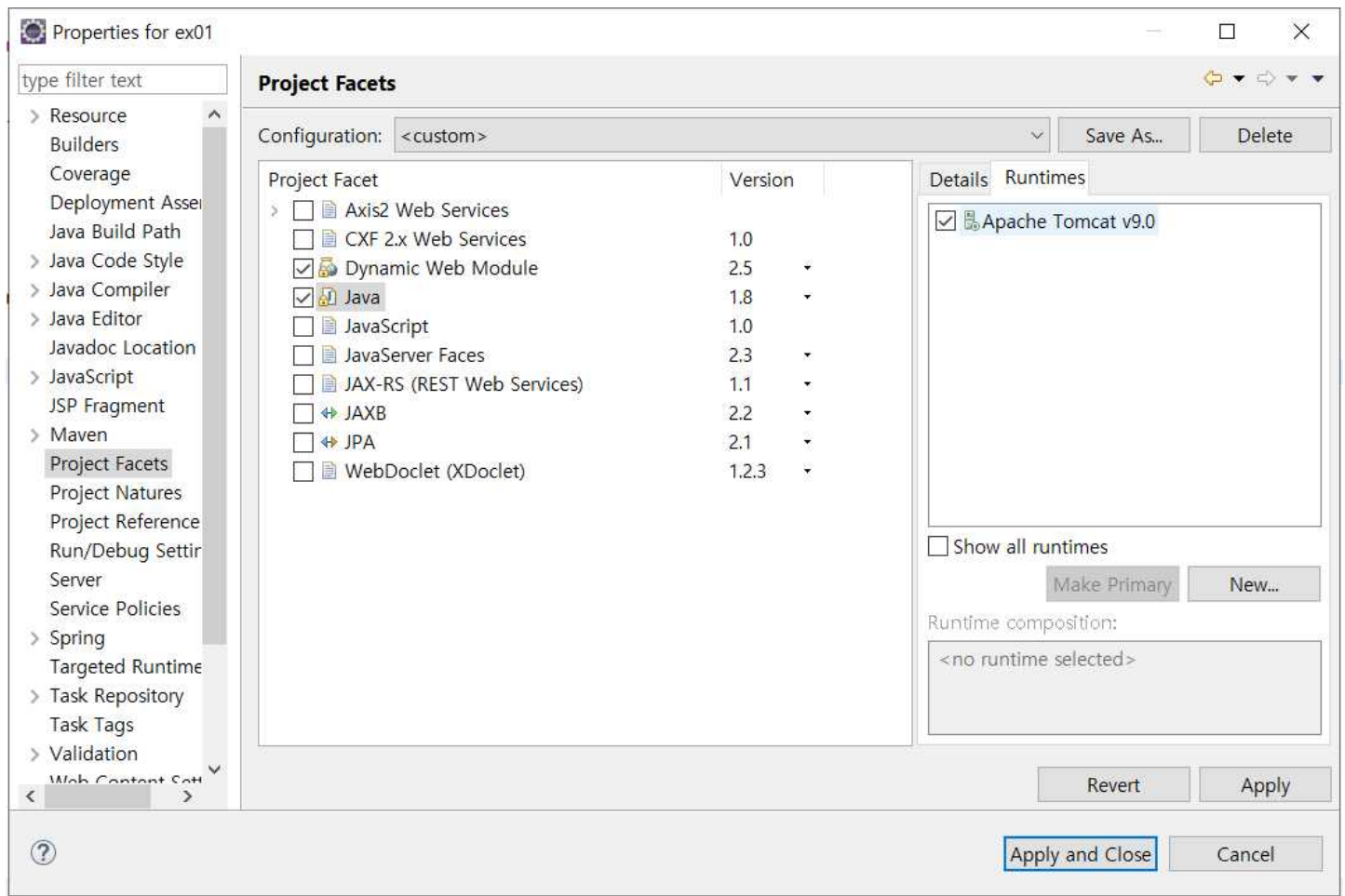
ex01/pom.xml 파일의 properties 태그에서 java-version과 springframework-version을 1.8과 5.1.4로 각각 수정한다.

```
<properties>
  <java-version>1.8</java-version>
  <org.springframework-version>5.1.4.RELEASE</org.springframework-version>
  <org.aspectj-version>1.6.10</org.aspectj-version>
  <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

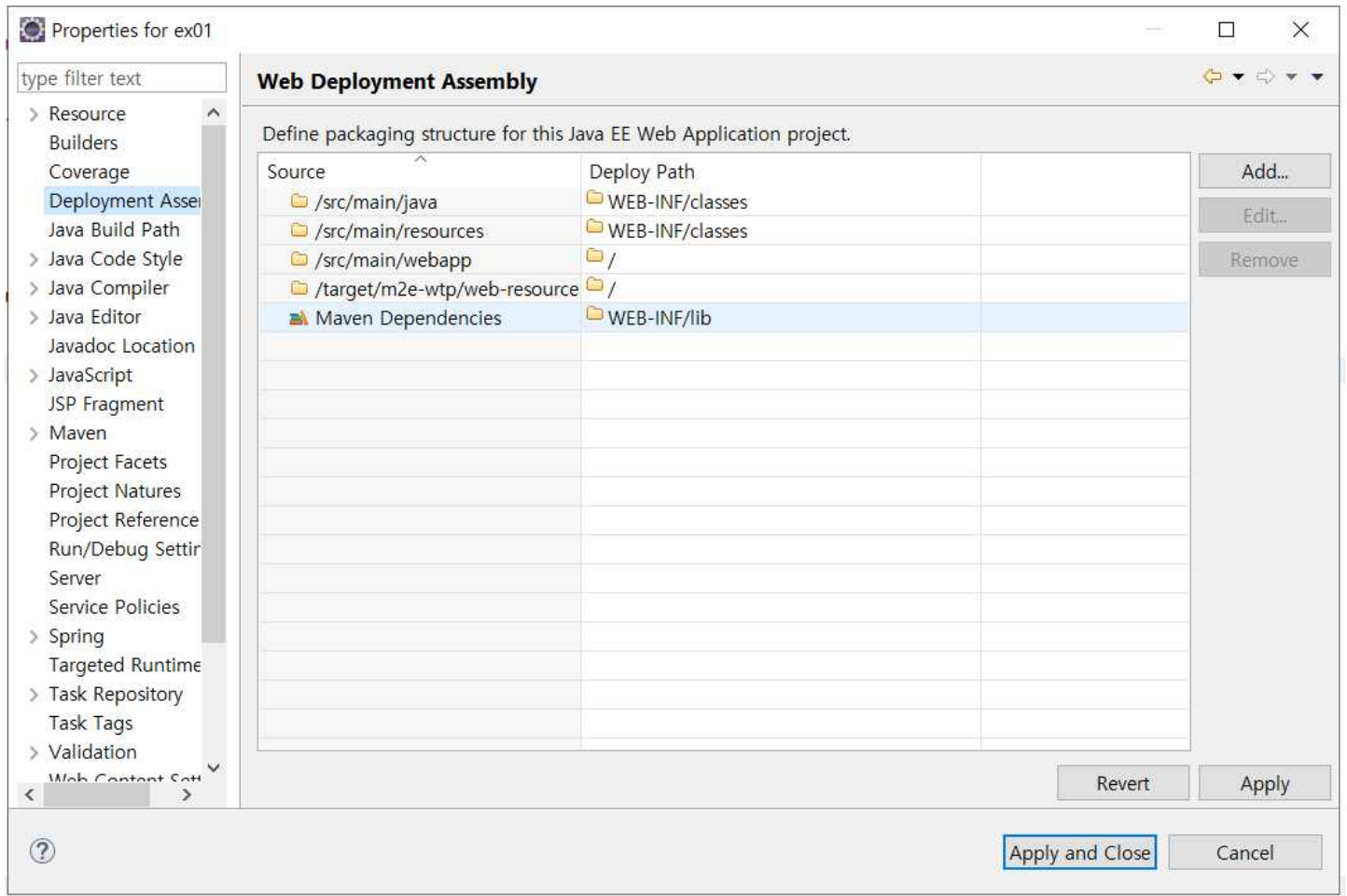
Project Properties 창을 열어 Project Facets에서 Java version을 1.8로 변경한다.



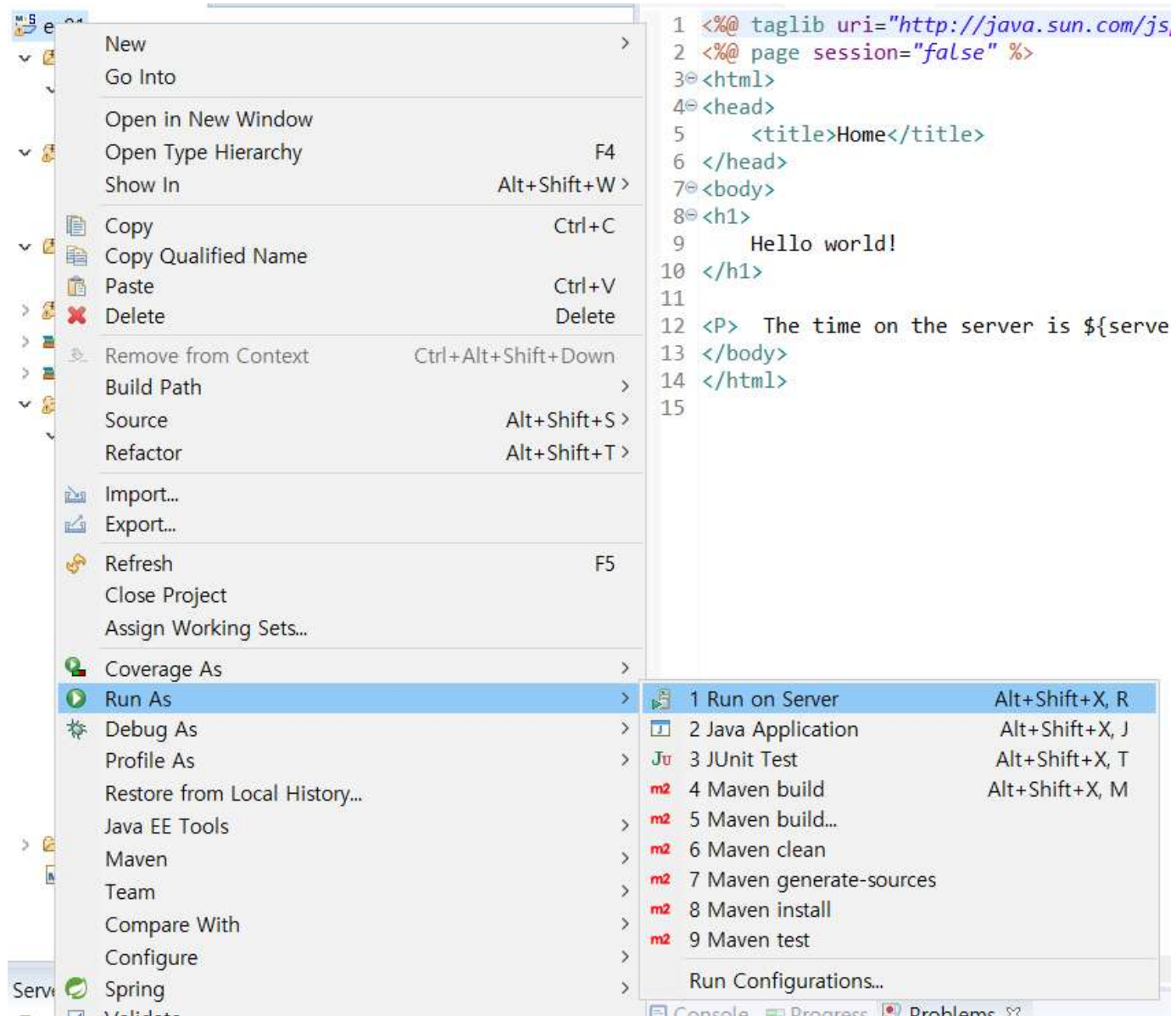
간혹 발생하는 오류를 예방하기 위하여 Runtimes탭을 클릭, 톰캣서버를 체크한다.



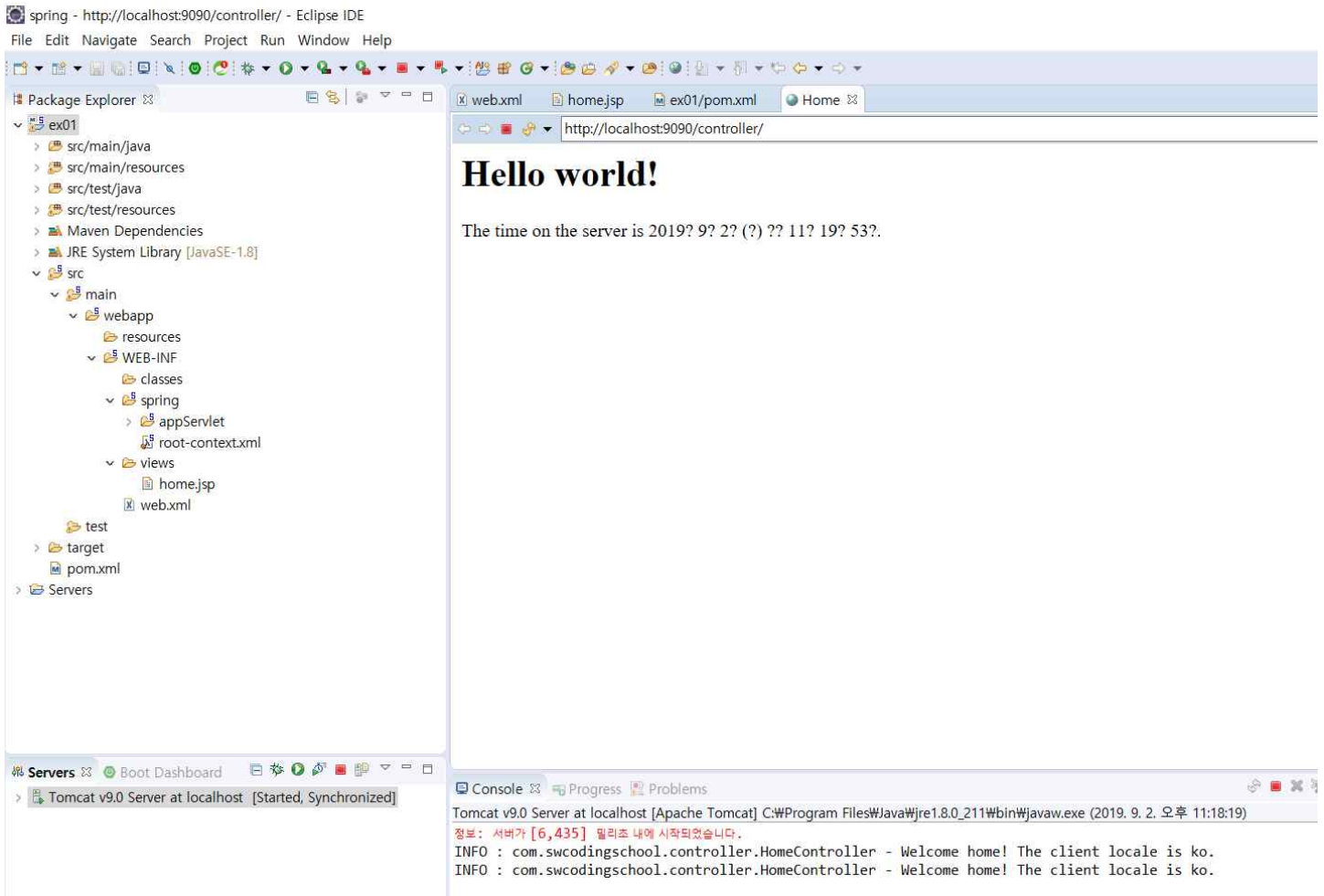
Deployment Assembly를 클릭, Maven Dependencies에 WEB-INF/lib이 추가되어 있음을 최종 확인



home.jsp 파일을 실행하기 위하여 File > Run As > Run on Server를 클릭한다.



아래와 같이 실행 결과를 확인할 수 있다.



실행결과 화면에서 한글이 깨지는 것은 Encoding을 UTF-8로 변경하여 해결한다.

web.xml 파일에서 </web-app> 앞에 아래와 같이 encoding filter를 추가한다.

```

31 </servlet-mapping>
32 k!-- 한글 처리를 위한 encoding filter 추가 -->
33 <filter>
34     <filter-name>encodingFilter</filter-name>
35     <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
36     <init-param>
37         <param-name>encoding</param-name>
38         <param-value>UTF-8</param-value>
39     </init-param>
40     <init-param>
41         <param-name>forceEncoding</param-name>
42         <param-value>true</param-value>
43     </init-param>
44 </filter>
45 <filter-mapping>
46     <filter-name>encodingFilter</filter-name>
47     <url-pattern>/*</url-pattern>
48 </filter-mapping>
49 </web-app>

```

encoding filter가 적용되어 실행된 결과는 다음과 같다.

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure for 'ex01', including 'src/main/java' with 'com.swcodingschool.controller' and 'HomeController.java', 'src/main/resources', 'src/test/java', 'src/test/resources', 'Maven Dependencies', 'Apache Tomcat v9.0', 'JRE System Library', and 'src/main/webapp' with 'resources', 'WEB-INF' (containing 'classes', 'spring', 'appServlet', 'servlet-context.xml', 'root-context.xml'), and 'views' (containing 'home.jsp', 'web.xml').

The main editor shows the browser view at 'http://localhost:9090/controller/'. The page content is:

Hello world!

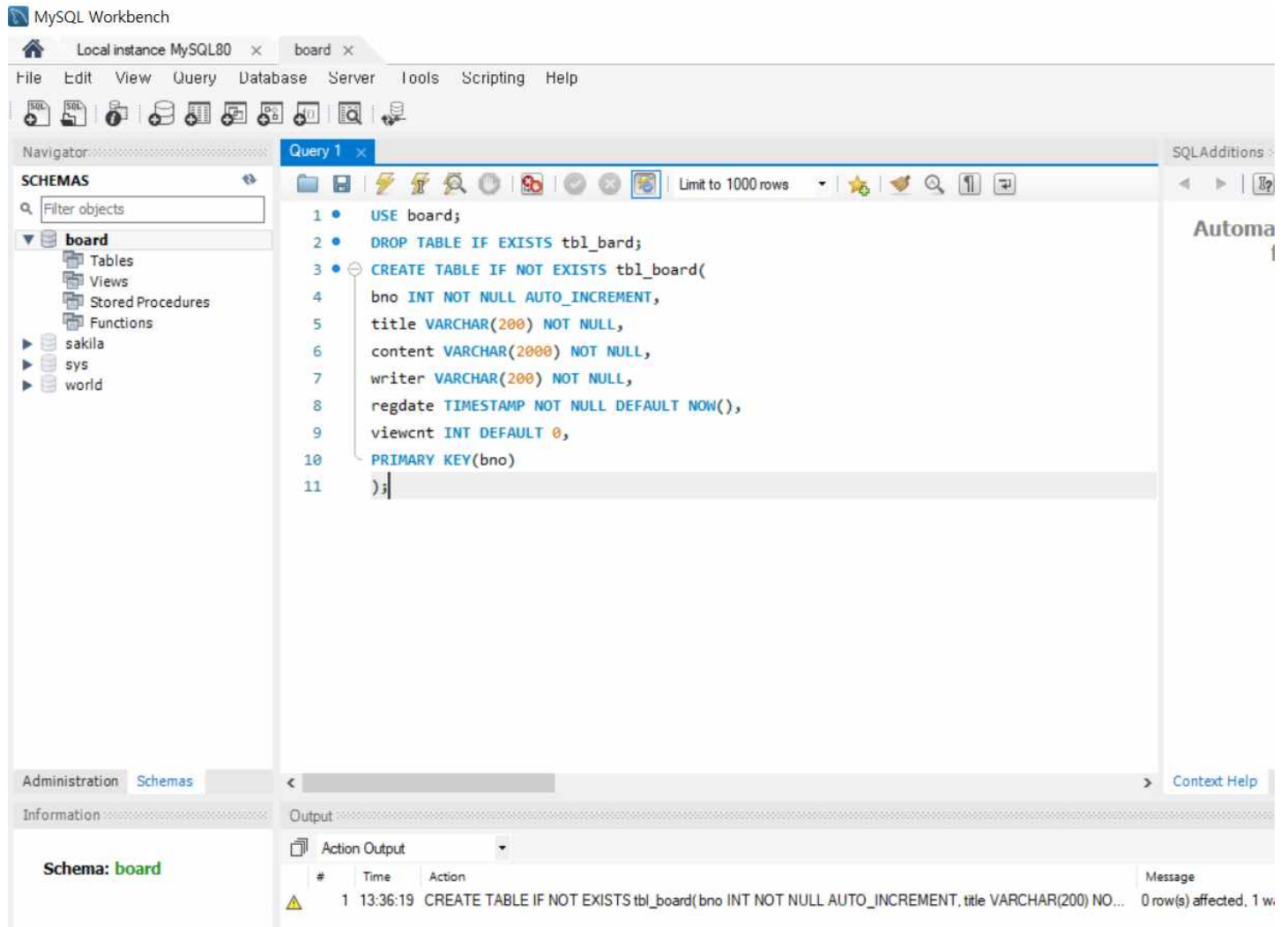
The time on the server is 2019년 9월 3일 (화) 오전 12시 20분 45초.

The bottom console shows the following output:

```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (2019. 9. 3. 오전 12:12:12)
정보: 서버가 [7, 328] 밀리초 내에 시작되었습니다.
INFO : com.swcodingschool.controller.HomeController - Welcome home! The client locale is ko.
INFO : com.swcodingschool.controller.HomeController - Welcome home! The client locale is ko.
```


2.2 MySQL 테이블 준비

MySQL Workbench를 이용하여 게시판용 테이블을 생성한다. 필요하다면 MySQL을 설치하고 게시판용 계정과 스키마를 생성한 후 테이블을 생성하도록 한다.



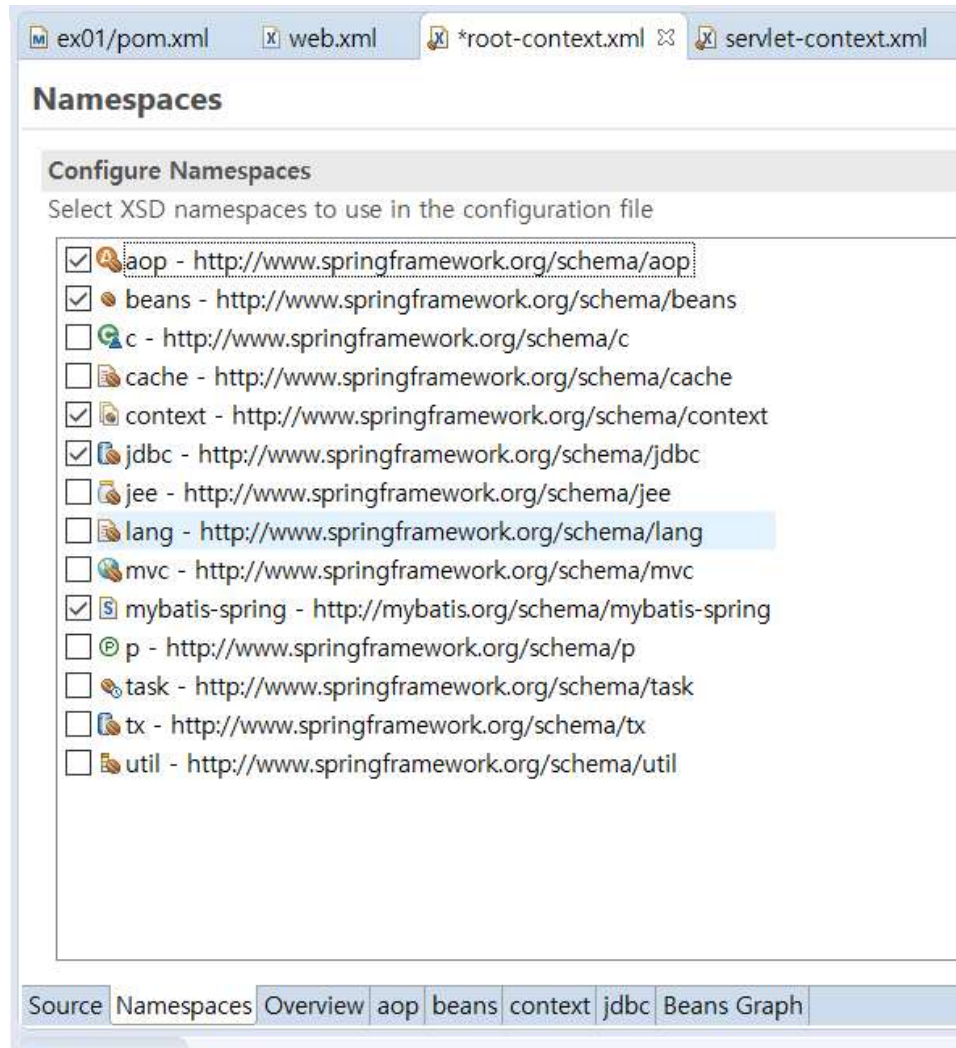
```

create table tbl_board(
  bno INT NOT NULL AUTO_INCREMENT,
  title VARCHAR(200) NOT NULL,
  content VARCHAR(2000) NOT NULL,
  writer VARCHAR(200) NOT NULL,
  regdate TIMESTAMP NOT NULL default NOW(),
  viewcnt INT DEFAULT 0,
  PRIMARY KEY(bno));
  
```

2.3 DB연결 설정

DB연결을 위한 설정하기위하여 root-context.xml 파일을 열고 Namespace 탭을 클릭 후 필요한 네임스페이스를 다음과 같이 선택한다.

aop, beans, context, jdbc, mybatis를 선택 또는 확인한다.



source 탭을 선택하고,

bean 태그를 이용하여 datasource와 sqlSessionFactory를 등록한다.

dataSource

```

14 <!-- Root Context: defines shared resources visible to all other web components -->
15 <!-- DB접속을 위한 설정정보 -->
16 <bean id="dataSource"
17     class="org.springframework.jdbc.datasource.DriverManagerDataSource">
18     <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
19     <property name="url"
20         value="jdbc:mysql://localhost:3306/board?characterEncoding=utf8&useSSL=false"></property>
21     <property name="username" value="board"></property>
22     <property name="password" value="board1234"></property>
23 </bean>

```

```

<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName"
        value="com.mysql.jdbc.Driver"></property>
    <property name="url"
value="jdbc:mysql://localhost:3306/board?characterEncoding=utf8&useSSL=false"></property>
    <property name="username" value="board"></property>
    <property name="password" value="board1234"></property>
</bean>

```

SqlSessionFactory 등록

```

25  <!-- Mybatis 사용을 위한 sqlSessionFactory 커넥션 추가 -->
26  <bean id="sqlSessionFactory"
27      class="org.mybatis.spring.SqlSessionFactoryBean">
28      <property name="dataSource" ref="dataSource" />
29      <property name="configLocation"
30          value="classpath:/mybatis-config.xml"></property>
31      <property name="mapperLocations"
32          value="classpath:mappers/**/*.xml"></property>
33  </bean>
34  <bean id="sqlSession"
35      class="org.mybatis.spring.SqlSessionTemplate"
36      destroy-method="clearCache">
37      <constructor-arg name="sqlSessionFactory"
38          ref="sqlSessionFactory"></constructor-arg>
39  </bean>
40

```

```

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="configLocation"
        value="classpath:/mybatis-config.xml"></property>
    <property name="mapperLocations"
        value="classpath:mappers/**/*.xml"></property>
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate"
    destroy-method="clearCache">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
</bean>

```

여기까지 작업 후 일단 테스트를 위한 실행을 해본다. 404오류가 나오면 정상이다.

오류의 원인은 root-context.xml 파일에 새롭게 추가한 sqlSessionFactory에서 지정한 설정위치 구성이 아직 안되어있기 때문인데 좀더 구체적으로 말하자면 Mybatis 설정파일 위치와 SQL연동 위치 설정을 위한 Mapper가 아직 설정되어 있지 않다.

2.4 Mybatis 환경설정

sqlSessionFactory에서 정의된 mybatis-config 파일을 만든다. src/main/resources 폴더 아래 xml 파일로 생성하고 다음과 같은 내용으로 구성한다.



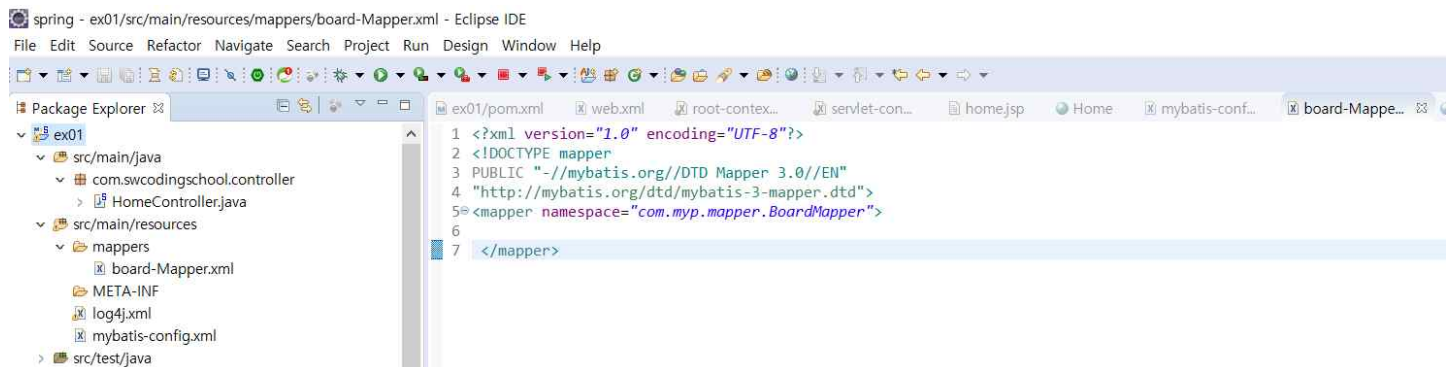
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration
3   PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6
7 </configuration>

```

2.5 Mapper 설정

다음으로 src/main/resources 폴더아래에 mappers 폴더를 생성하고 board-Mapper.xml 파일을 다음과 같은 내용으로 구성한다.



```

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.myp.mapper.BoardMapper">
6
7 </mapper>

```

Package Explorer view shows the project structure:

- ex01
 - src/main/java
 - com.swcodingschool.controller
 - HomeController.java
 - src/main/resources
 - mappers
 - board-Mapper.xml
 - META-INF
 - log4j.xml
 - mybatis-config.xml
 - src/test/java

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.swcodingschool.mapper.BoardMapper">

</mapper>

```

이제 다시 한번 프로젝트를 테스트 실행해보면 정상적으로 실행됨을 확인할 수 있다.

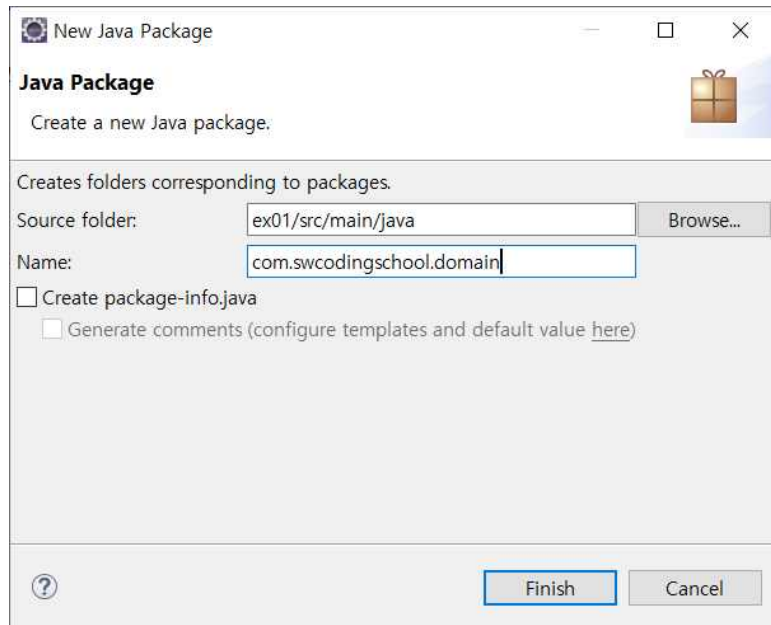
2.6 VO/DTO 생성

다음 단계는 VO(Value Object) 또는 DTO(Data Transfer Object)를 생성하는 것이다.

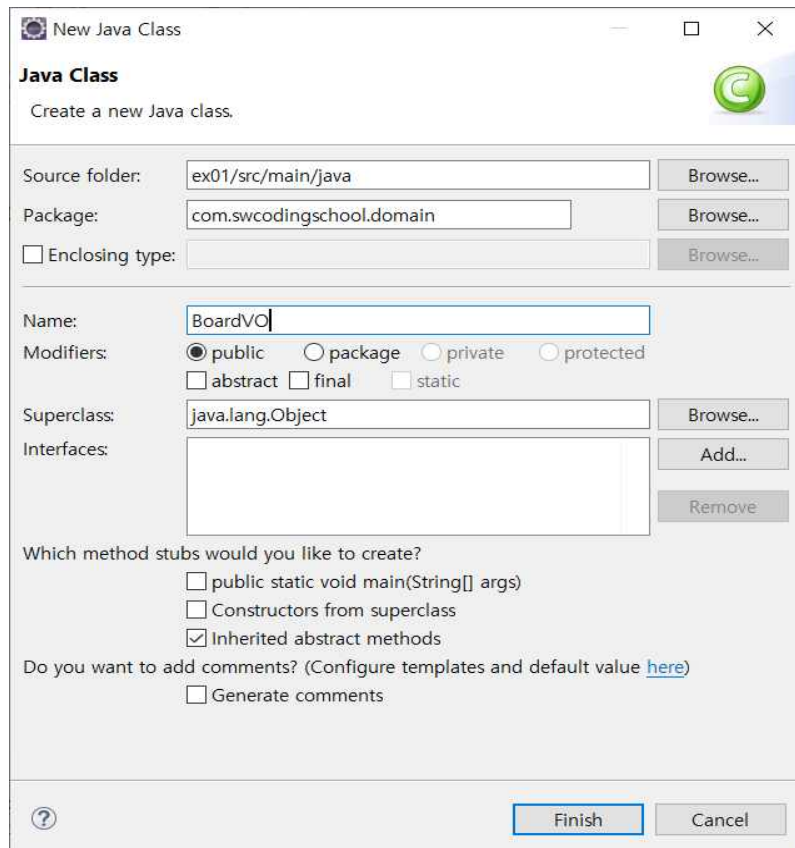
VO는 DB의 정보를 객체화 시키는 용도로 사용하며, DTO는 외부 시스템과 데이터 통신을 하기 위한 용도로 사용한다.

여기에서는 VO를 생성하도록 하자.

src/main/java 폴더 아래에 com.swcodingschool.domain 패키지를 생성하고



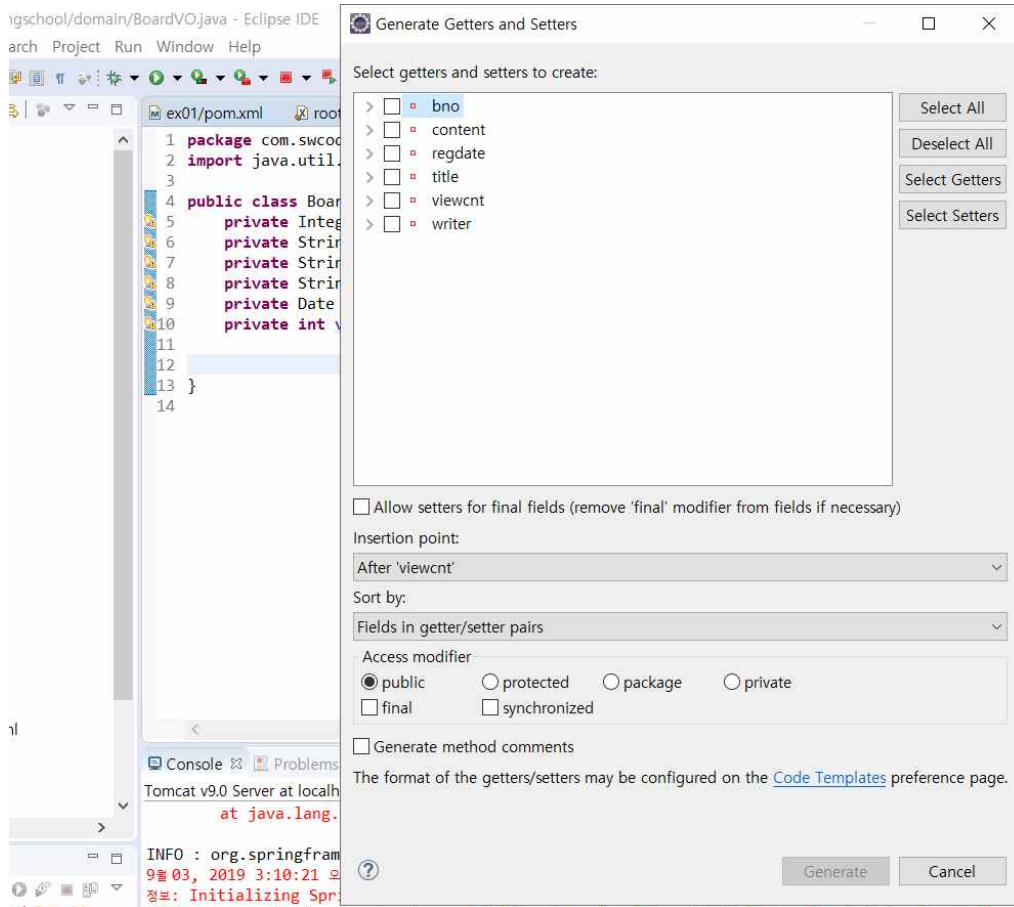
이 폴더에서 BoardVO 클래스를 다음과 같이 생성한다.



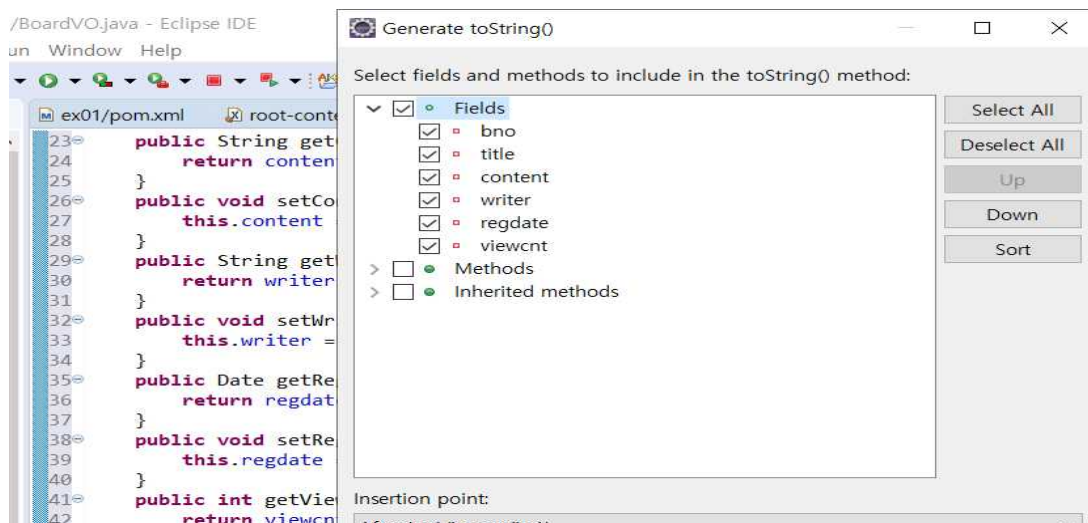
BoardVO 클래스의 멤버프로퍼티는 MySQL 테이블의 내용과 동일하게 다음과 같이 구성한다.

```
private Integer bno;
private String title;
private String content;
private String writer;
private Date regdate;
private int viewcnt;
```

메서드로는 getter와 setter를 만들어주고,



toString()을 생성한다.



최종 완성 코드는 다음과 같다.

```
import java.util.Date;

public class BoardVO {
    private Integer bno;
    private String title;
    private String content;
    private String writer;
    private Date regdate;
    private int viewcnt;

    public Integer getBno() {
        return bno;
    }
    public void setBno(Integer bno) {
        this.bno = bno;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getWriter() {
        return writer;
    }
    public void setWriter(String writer) {
        this.writer = writer;
    }
    public Date getRegdate() {
        return regdate;
    }
    public void setRegdate(Date regdate) {
        this.regdate = regdate;
    }
}
```

```
public int getViewcnt() {
    return viewcnt;
}
public void setViewcnt(int viewcnt) {
    this.viewcnt = viewcnt;
}
@Override
public String toString() {
    return "BoardVO [bno=" + bno + ", title=" + title + ", content=" + content + ",
writer=" + writer + ", regdate="
+ regdate + ", viewcnt=" + viewcnt + "]\n";
}
}
```

2.7 DAO 연동을 위한 mapper 구성

다음 단계는 DAO와 연동하기 위한 SQL작성 단계이다.

board-Mapper.xml 파일의 내용을 다음과 같이 구성한다.

```
<insert id="create"> <!-- DAO와 매핑을 위한 ID -->
INSERT INTO tbl_board(title, content, writer)
VALUES("#{title}", #{content}, #{writer})
</insert>

<!-- Read 게시물 읽기 -->
<select id="read" resultType="com.swcodingschool.domain.BoardVO"> <!-- 데이터를 받아오기 위
한 resultType 명시 -->
SELECT bno, title, content, writer, regdate, viewcnt
FROM tbl_board
WHERE bno = #{bno}
</select>

<!-- Update 게시글 수정 -->
<update id="update">
UPDATE tbl_board SET title=#{title}, content=#{content}
WHERE bno = #{bno}
</update>

<!-- Delete 삭제 -->
<delete id="delete">
DELETE FROM tbl_board WHERE bno = #{bno}
</delete>

<!-- ListAll 전체 글 목록 -->
<select id="listAll" resultType="com.swcodingschool.domain.BoardVO"> <!-- SQL내부에 연산자가
있을 경우 ![CDATA[ ]] 사용 -->
<![CDATA[
SELECT bno, title, content, writer, regdate, viewcnt
FROM tbl_board
WHERE bno > 0
ORDER BY bno DESC, regdate DESC
]]>
</select>
```

위의 코드에서 각 id는 DAO와 매핑을 하기 위한 ID이며 역할에 따라 입력으로 받는 패러미터와 리턴 결과의 타입이 명시되어 있다.

2.8 DAO 인터페이스 및 클래스 구현

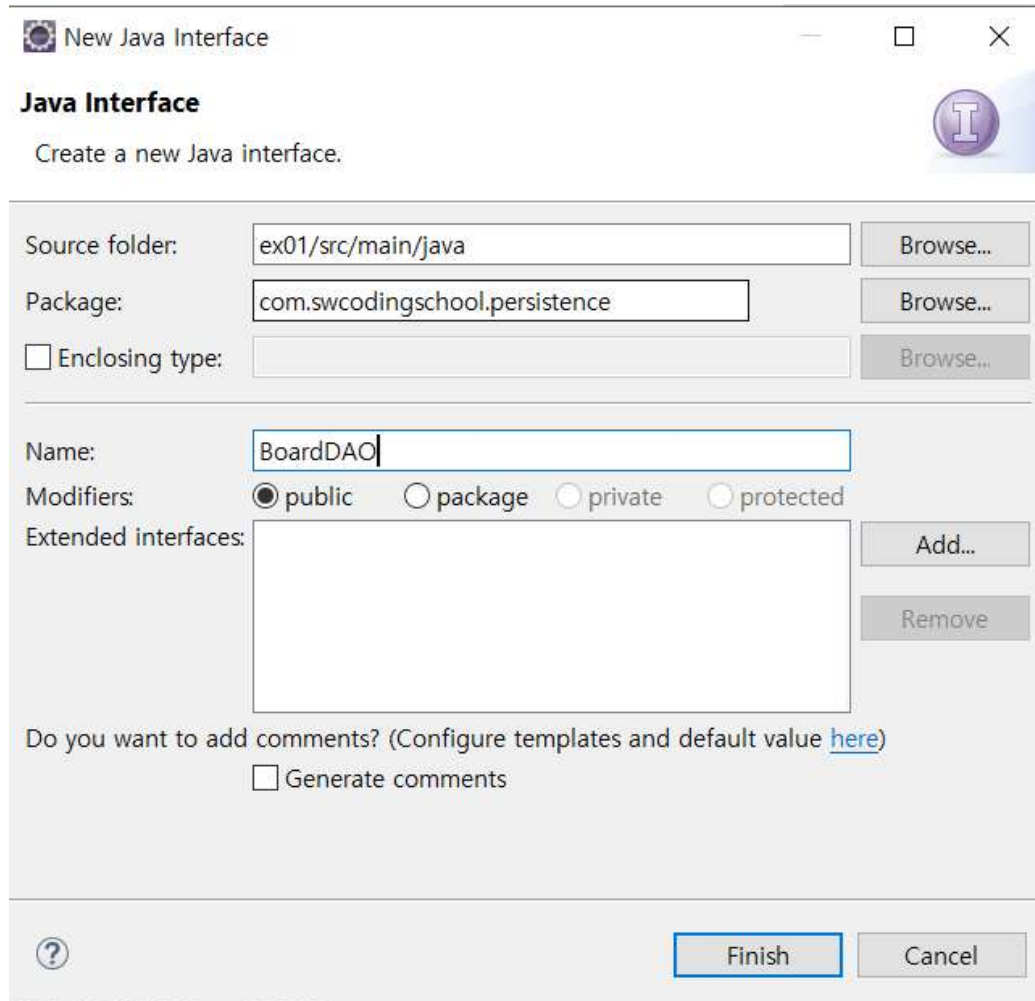
다음 단계는 DAO(Data Access Object) 데이터 접근 객체를 작성하는 단계이다.

서비스 <-> DAO <-> Mapper

서비스는 컨트롤러(서블릿)의 요청사항을 DAO에 전달한다.

BoardDAO를 작성하기 위하여 src/main/java 폴더 아래에

com.swcodingschool.persistence 패키지를 만들고, BoardDAO 인터페이스를 생성한다.



BoardDAO 인터페이스의 내용은 다음과 같이 구성한다.

```
package com.swcodingschool.persistence;
import java.util.List;
import com.swcodingschool.domain.BoardVO;

public interface BoardDAO {
    public void create(BoardVO vo) throws Exception;
    public BoardVO read(Integer bno) throws Exception;
    public void update(BoardVO vo) throws Exception;
    public void delete(Integer bno) throws Exception;
    public List<BoardVO> listAll() throws Exception;
}
```


그리고 DAO 인터페이스를 구현한 BoardDAOImpl 클래스를 작성하고

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces: ☒ com.swcodingschool.persistence.BoardDAO

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

클래스의 내용을 다음과 같이 구성한다.

```
package com.swcodingschool.persistence;

import java.util.List;
import javax.inject.Inject;
import org.apache.ibatis.session.SqlSession;
import org.springframework.stereotype.Repository;

import com.swcodingschool.domain.BoardVO;

@Repository
public class BoardDAOImpl implements BoardDAO {
```

```
@Inject
private SqlSession session;

private static String namespace = "com.swcodingschool.mapper.BoardMapper";

@Override
public void create(BoardVO vo) throws Exception {
    session.insert(namespace+".create", vo);
}

@Override
public BoardVO read(Integer bno) throws Exception {
    return session.selectOne(namespace + ".read", bno);
}

@Override
public void update(BoardVO vo) throws Exception {
    session.update(namespace+".update", vo);
}

@Override
public void delete(Integer bno) throws Exception {
    session.delete(namespace+".delete", bno);
}

@Override
public List<BoardVO> listAll() throws Exception {
    return session.selectList(namespace + ".listAll");
}
}
```

최종적으로 servlet-context.xml 파일의 아랫부분의 주소값
com.swcodingschool.controller을
com.swcodingschool로 수정하여 다음과 같도록 한다.

```
<context:component-scan base-package="com.swcodingschool" />
```

2.9 서비스 인터페이스 및 클래스 구현

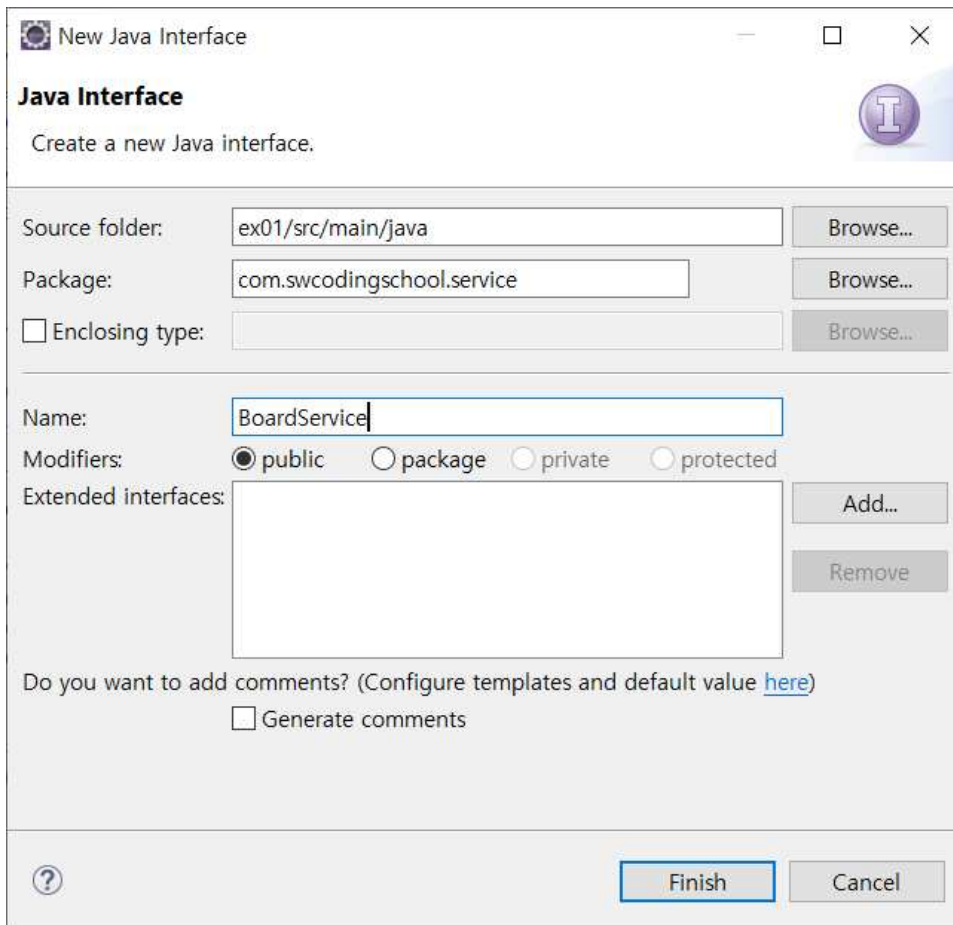
다음 단계는 Service 작성단계이다.

서비스는 유지보수와 로직프로세스를 유연하게 처리하기 위한 중간 단계로써 프레임워크 디자인 패턴에 들어가며, 프로젝트 규모가 커질수록 관리하기 용이하다.

controller > service(interface) -> ServiceImpl -> DAO

서비스 인터페이스 구성을 위해서는 src/main/java 폴더 아래에

com.swcodingschool.service 패키지를 생성하고 BoardService 인터페이스를 생성한 후



컨트롤러에서 요청될 service 메소드를 다음과 같이 구성한다.

```
package com.swcodingschool.service;

import java.util.List;
import com.swcodingschool.domain.BoardVO;

public interface BoardService {
    public void regist(BoardVO board) throws Exception;
    public BoardVO read(Integer bno) throws Exception;
    public void modify(BoardVO board) throws Exception;
    public void remove(Integer bno) throws Exception;
    public List<BoardVO> listAll() throws Exception;
}
```

서비스 클래스 구현을 위해 BoardServiceImpl 클래스를 생성하고

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces: ☒ com.swcodingschool.service.BoardService

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

그 내용을 다음과 같이 구성한다.

```
package com.swcodingschool.service;

import java.util.List;
import javax.inject.Inject;
import org.springframework.stereotype.Service;

import com.swcodingschool.domain.BoardVO;
import com.swcodingschool.persistence.BoardDAO;

@Service
public class BoardServiceImpl implements BoardService {
    @Inject
```

```
private BoardDAO dao;

@Override
public void regist(BoardVO board) throws Exception {
    dao.create(board);
}

@Override
public BoardVO read(Integer bno) throws Exception {
    return dao.read(bno);
}

@Override
public void modify(BoardVO board) throws Exception {
    dao.update(board);
}

@Override
public void remove(Integer bno) throws Exception {
    dao.delete(bno);
}

@Override
public List<BoardVO> listAll() throws Exception {
    return dao.listAll();
}

}
```


2.10 간단한 게시판 구동용 링크 구성

게시판 구성을 위해 WAS 구동시 처음으로 기동하는 home.jsp에 게시판 목록에 대한 링크를 구성하자. 다음과 같이 home.jsp를 수정한다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
    <title>Home</title>
    <link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<!-- BoardController.java의 RequestMapping을 통하여 board-Mapper.xml까지 전달된다. -->
<form action="listAll" method="get">
<body>
    <div class="jumbotron">
        <div class="container">
            <button type="submit">게시판</button>
        </div>
    </div>
    <h1>Hello world!</h1>
    <P>The time on the server is ${serverTime}.</P>
</body>
</form>
</html>
```

WAS테스트 구동 후 화면을 확인하면 게시판 링크가 활성화됨을 확인할 수 있다.



3. 게시판 기본 기능 구현

3.1 게시판 전체 목록 보기

게시판 전체 목록을 나타내기 위해서는 listAll.jsp를 작성한다.

/src/main/webapp/WEB-INF/views 폴더 아래에 listAll.jsp 파일을 생성하고, 다음과 같이 코드를 구성한다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ page session="false" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet"
<title>Board List</title>
</head>
<body>
<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">게시판 목록보기</h1>
    </div>
</div>

<form action = "regist" method = "get"> <!-- 글쓰기 페이지 호출 -->
<table class="table table-hover">
    <tr>
        <th>글번호</th>
        <th>제목</th>
        <th>작성자</th>
        <th>작성일</th>
        <th>조회수</th>
    </tr>

    <c:forEach items="${list}" var="boardVO">
        <tr>
            <td>${boardVO.bno}</td>
            <td><a href='/read?bno=${boardVO.bno}'>${boardVO.title}</a></td>
            <td>${boardVO.writer}</td>
            <td><fmt:formatDate pattern="yyyy-MM-dd HH:mm"
```

```

                                value="${boardV0.regdate}" /></td>
                        <td><span class="badge bg-red">${boardV0.viewcnt}</span></td>
                    </tr>
                </c:forEach>
            </table>
            <button type="submit">글쓰기</button>
        </form>

    </body>
</html>

```

다음은 Controller 클래스를 작성하는 단계이다.

이를 위해 com.swcodingschool.controller 폴더아래에 BoardController 클래스를 생성한다.

```

package com.swcodingschool.controller;

import javax.inject.Inject;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

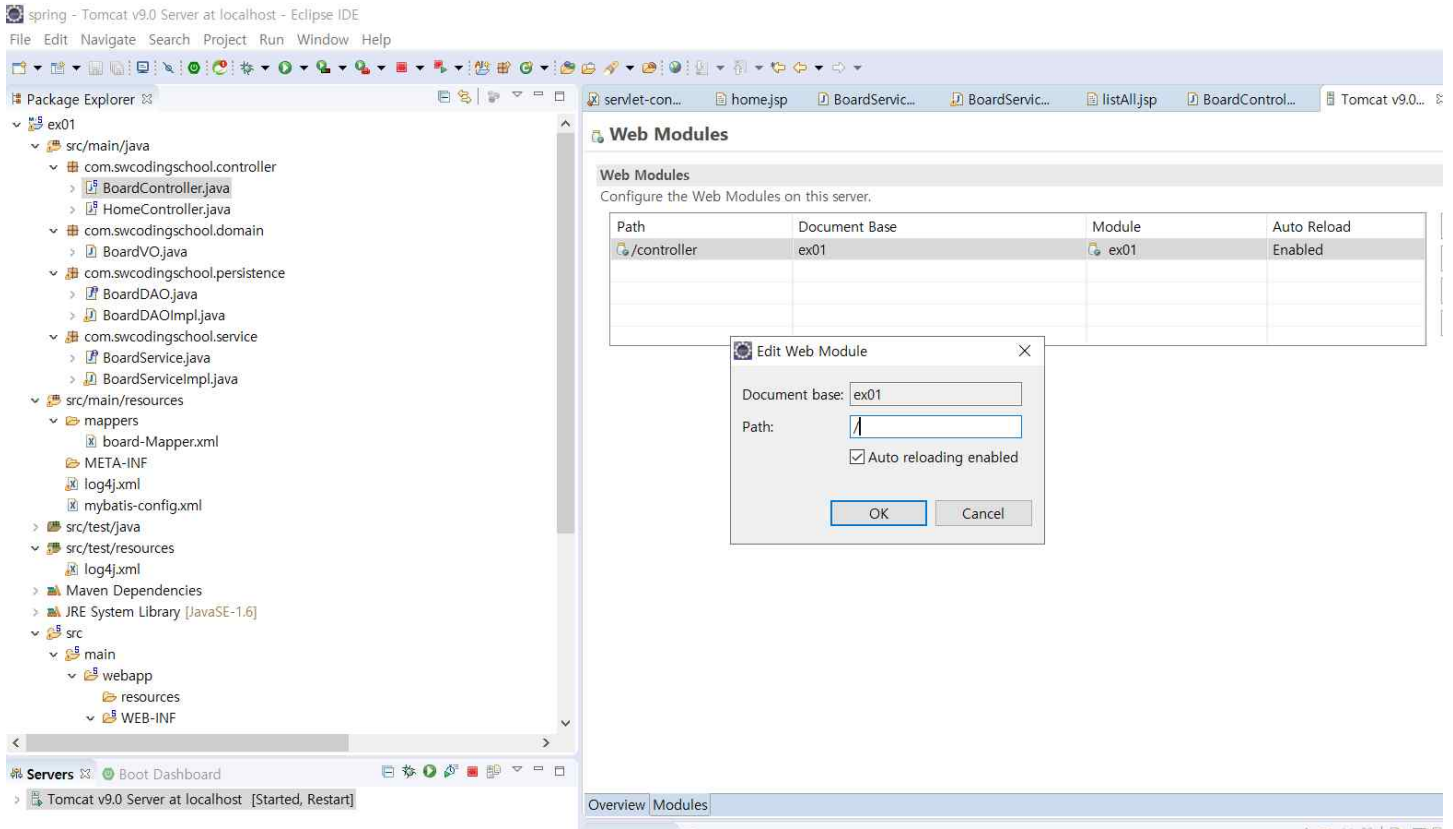
import com.swcodingschool.service.BoardService;

@Controller // 컨트롤러임을 명시
@RequestMapping(value = "/") // 주소 패턴
public class BoardController {
    @Inject // 주입(심부름꾼) 명시
    private BoardService service; // Service 호출을 위한 객체생성

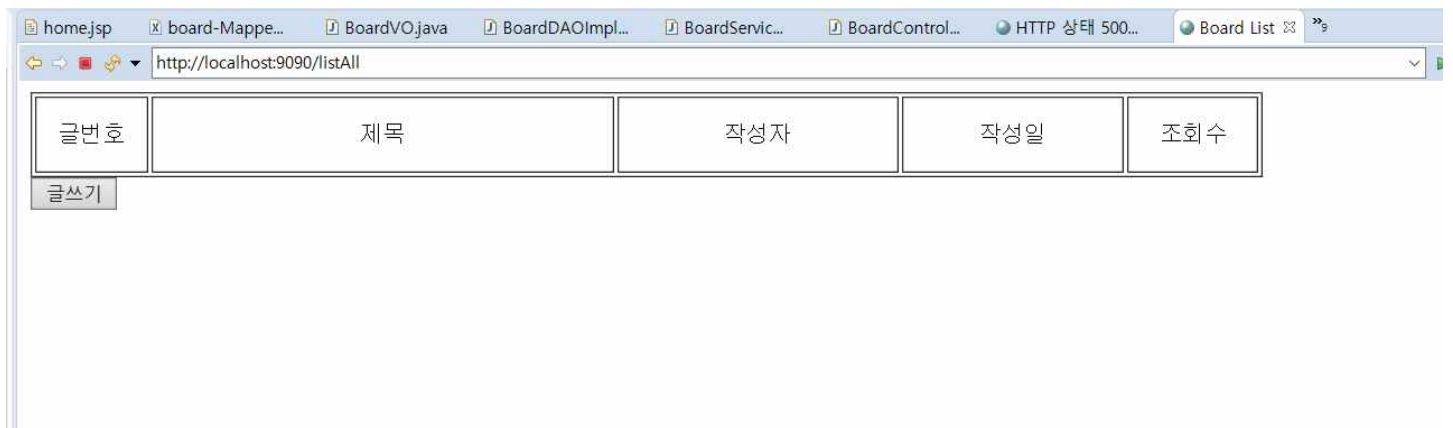
    // 게시판 목록보기
    // 주소 호출 명시 . 호출하려는 주소 와 REST 방식설정 (GET)
    @RequestMapping(value = "/listAll", method = RequestMethod.GET)
    public void listAll(Model model) throws Exception {
        // 메소드 인자값은 model 인터페이스(jsp전달 심부름꾼)
        // jsp에 심부름할 내역(서비스 호출)
        model.addAttribute("list", service.listAll());
    }
}

```

게시판 목록을 보기위한 컨트롤러가 완성되었으면 서버의 Module을 수정하여 base 디렉토리를 /로 변경한다.



테스트 구동을 하면 게시판 목록이 정상적으로 나타난다.



3.2 글쓰기

다음은 글쓰기 버튼을 클릭할 때 동작하기 위한 regist.jsp 파일을 생성하고

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
<meta charset="UTF-8">
<title>글쓰기</title>
</head>
<body>
<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">게시판 글쓰기</h1>
    </div>
</div>

<div class="container">
    <form method = "post" class="form-horizontal">

        <div class="form-group row">
            <label class="col-sm-2 control-label">제목</label>
            <div class="col-sm-8">
                <input type="text" name ="title" value ="${boardV0.title}">
            </div>
        </div>

        <div class="form-group row">
            <label class="col-sm-2 control-label">작성자</label>
            <div class="col-sm-8">
                <input type="text" name="writer" size="15" value = "${boardV0.writer}">
            </div>
        </div>

        <div class="form-group row">
            <label class="col-sm-2 control-label">내용</label>
            <div class="col-sm-8">
                <textarea                                name=content                                rows                                = "15"
cols="70">${boardV0.content}</textarea><br>
```

```

        </div>
    </div>

    <div class="form-group row">
        <div class="col-sm-offset-2 col-sm-10">
            <button type = "submit">등록</button>
        </div>
    </div>
</form>
</div>
</body>
</html>

```

controller 구현을 한다.

```

import org.springframework.web.servlet.mvc.support.RedirectAttributes;
...
import com.swcodingschool.domain.BoardVO;
...
// 게시판 글쓰기
@RequestMapping(value = "/regist", method = RequestMethod.GET) // GET 방식으로 페이지 호출
public void registerGET(BoardVO board, Model model) throws Exception {
}
@RequestMapping(value = "/regist", method = RequestMethod.POST) // POST방식으로 내용 전송
public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
    // 인자값으로 REDIRECT 사용
    service.regist(board); // 글작성 서비스 호출
    return "redirect:/listAll"; // 작성이 완료된 후, 목록페이지로 리턴
}

```

테스트 구동을 통해 쓰기 기능을 구현한 것을 확인할 수 있다.

http://localhost:9090/regist

제목

작성자

내용

3.3 읽기

다음은 읽기 기능을 추가하기 위하여 read.jsp파일을 생성하고,

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page session="false" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
<title>Read</title>
</head>

<body>

<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">글읽기</h1>
    </div>
</div>

<div class="container">
    <form class="form-horizontal">
        <div class="form-group row">
            <label class="col-sm-2 control-label">글번호</label>
            <div class="col-sm-8">
                <input      type="text"      name      ="bno"      value      ="${boardV0.bno}"
readonly="readonly">
            </div>
        </div>

        <div class="form-group row">
            <label class="col-sm-2 control-label">제목</label>
            <div class="col-sm-8">
                <input      type="text"      name      ="title"      value      ="${boardV0.title}"
readonly="readonly">
            </div>
        </div>
    </form>
</div>
```



```

<div class="form-group row">
    <label class="col-sm-2 control-label">작성자</label>
    <div class="col-sm-8">
        <input type="text" name="writer" size="15" value =
"${boardV0.writer}" readonly="readonly">
    </div>
</div>

<div class="form-group row">
    <label class="col-sm-2 control-label">내용</label>
    <div class="col-sm-8">
        <textarea name="content" rows="15" cols="70"
readonly="readonly">${boardV0.content}</textarea><br>
    </div>
</div>

<div class="form-group row">
    <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" formaction="modify" formmethod="get">수정</button>
        <button type="submit" formaction="remove" formmethod="post">삭제
</button>
        <button type="submit" formaction="listAll" formmethod="get">목록
</button>
    </div>
</div>
</form>
</div>
</body>

</html>

```

컨트롤러에도 read를 위한 메서드 추가...

```

import org.springframework.web.bind.annotation.RequestParam;
...
// 게시판 읽기
@RequestMapping(value = "/read", method = RequestMethod.GET) // GET 방식으로 페이지 호출
public void read(@RequestParam("bno")int bno, Model model) throws Exception{
    // 인자값은 파라미터 값으로 기본키인 글번호를 기준으로 Model을 사용하여 불러옴
    model.addAttribute(service.read(bno)); // read 서비스 호출
}

```

테스트 실행을 통해 실행결과를 확인한다.

3.4 수정 / 삭제

수정과 삭제는 읽기 화면에서 작동한다.

modify.jsp를 다음과 같이 구성한다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet"
<meta charset="UTF-8">
<title>Modify</title>
</head>
<body>
<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">글수정</h1>
    </div>
</div>

<div class="container">
    <form action="modify" method = "post" class="form-horizontal">
        <div class="form-group row">
```

```

        <label class="col-sm-2 control-label">글번호</label>
        <div class="col-sm-8">
            <input type="text" name="bno" value="${boardV0.bno}"
readonly="readonly">
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 control-label">제목</label>
        <div class="col-sm-8">
            <input type="text" name="title" value="${boardV0.title}"
readonly="readonly">
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 control-label">작성자</label>
        <div class="col-sm-5">
            <input type="text" name="writer" size="15" value =
"${boardV0.writer}"readonly="readonly">
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 control-label">내용</label>
        <div class="col-sm-8">
            <textarea name="content" rows="15" cols="70"
readonly="readonly">${boardV0.content}</textarea><br>
        </div>
    </div>

    <div class="form-group row">
        <div class="col-sm-offset-2 col-sm-10">
            <button type="submit">완료</button>
        </div>
    </div>
</form>
</div>

</body>
</html>

```

controller에 수정메서드를 추가한다.

```
// 글수정
@RequestMapping(value = "/modify", method = RequestMethod.GET) // GET 방식으로 페이지 호출
public void modifyGET(int bno, Model model) throws Exception {
    model.addAttribute(service.read(bno)); // 수정을 위한 글읽기 서비스 호출
}
@RequestMapping(value = "/modify", method = RequestMethod.POST)// POST방식으로 데이터 전송
public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
    service.modify(board); // 글수정 서비스 호출
    return "redirect:/listAll"; // 수정이 완료된 후, 목록페이지로 리턴
}
```

삭제메서드 또한 controller에 추가한다.

```
// 글삭제
@RequestMapping(value = "/remove", method = RequestMethod.POST)// POST방식으로 데이터 전송
public String removePOST(@RequestParam("bno") int bno, RedirectAttributes rttr) throws
Exception{
    service.remove(bno); // 글삭제 서비스 호출
    return "redirect:/listAll"; // 삭제가 완료된 후, 목록페이지로 리턴
}
```

3.5 뷰 카운트 증가

뷰 카운트 증가 기능을 구현하기 위해서는 ① mapper에 sql문을 추가 구성하고, ②DAO 인터페이스에 메서드 추가 및 DAO 구현체에 Override 구성한 후, ③service 구현체 read 메서드 내부에 끼워넣기 순서로 작업한다.

① 질의 추가 구성

board-Mapepr.xml 파일에

```
<!-- 뷰카운트 증가 -->
<update id="updateViewCnt">
    update tbl_board
    set viewcnt = viewcnt + 1
    where bno = #{bno}
</update>
```

② BoardDAO.java 에 인터페이스 메서드 추가

```
public interface BoardDAO {
    public void create(BoardVO vo) throws Exception;
    public BoardVO read(Integer bno) throws Exception;
    public void update(BoardVO vo) throws Exception;
    void updateViewCnt(Integer bno) throws Exception;
    public void delete(Integer bno) throws Exception;
    public List<BoardVO> listAll() throws Exception;
}
```

및 BoardDAOImpl.java파일에 구현

```
2
3 @Override
4 public void updateViewCnt(Integer bno) throws Exception {
5     session.update(namespace + ".updateViewCnt", bno);
6 }
7
```

③ BoardServiceImpl.java 파일, 즉 service 구현체 read 메서드에 끼워넣기

```
21 @Override
22 public BoardVO read(Integer bno) throws Exception {
23     dao.updateViewCnt(bno); // 뷰카운트 업데이트 추가
24     return dao.read(bno);
25 }
26
```

테스트 수행하면 정상적으로 뷰 카운트가 증가하는 것을 확인할 수 있다.

글번호	제목	작성자	작성일	조회수
3	글쓰기테스트2	홍길숙	2019-09-03 20:01	3
2	안녕하세요	홍길동	2019-09-03 19:25	1

글쓰기

3.6 pagination 기능 추가

3.6.1 TypeAlias 설정

src/main/resources 폴더의 mybatis-config.xml 파일에 다음과 같이 typeAliases를 추가한다.

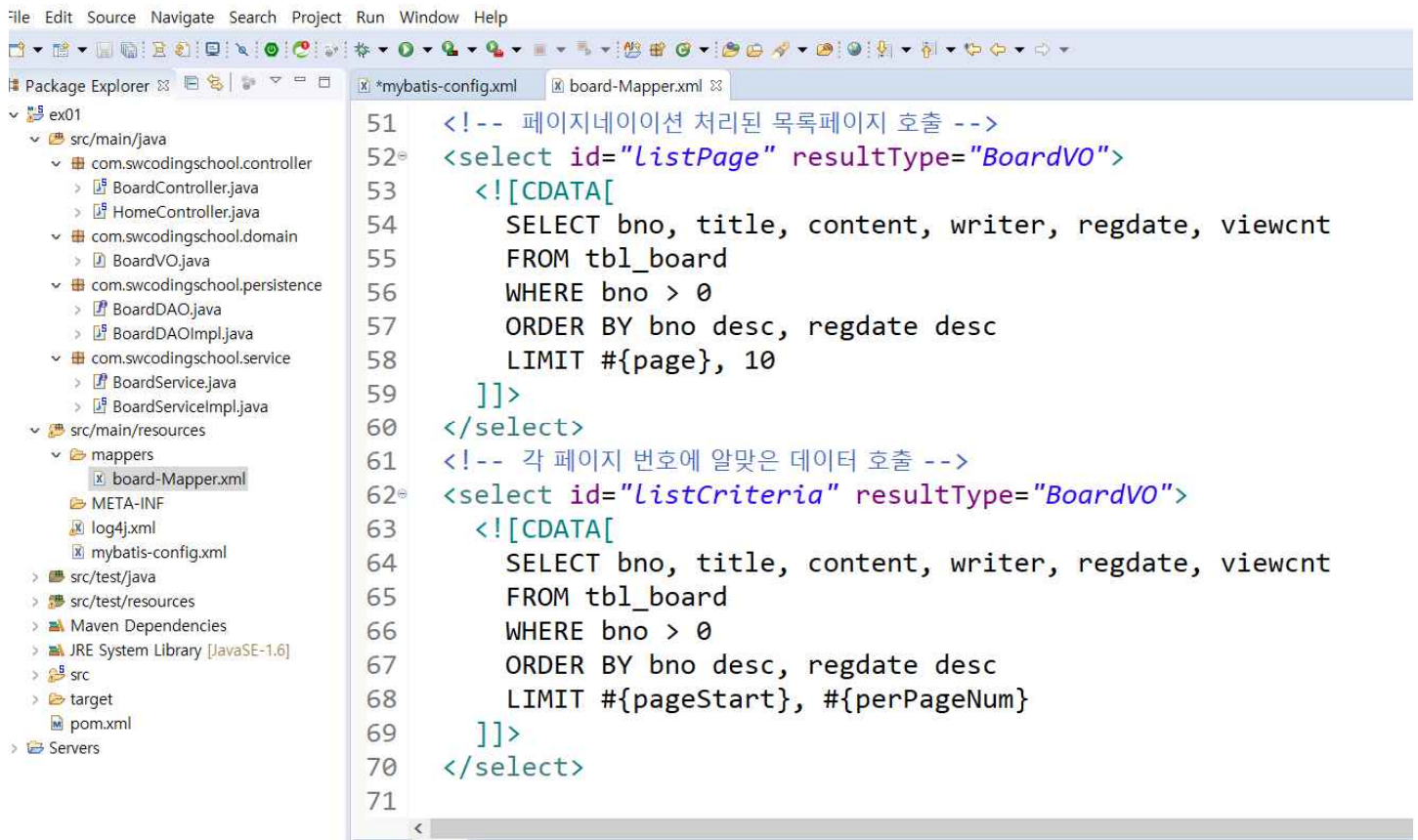
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration
3 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4 "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6
7 <typeAliases>
8   <package name="com.swcodingschool.domain"/>
9 </typeAliases>
10
11 </configuration>

```

3.6.2 페이지네이션 처리를 위한 질의 구성

src/main/resources/mappers/board-mapper.xml 파일에 페이지네이션 처리를 위한 질의를 추가로 구성한다.



추가로 구성된 질의는 다음 코드와 같다.

```

<!-- 페이지네이션 처리된 목록페이지 호출 -->
<select id="listPage" resultType="BoardVO">
    <![CDATA[
        SELECT bno, title, content, writer, regdate, viewcnt
        FROM tbl_board
        WHERE bno > 0
        ORDER BY bno desc, regdate desc
        LIMIT #{page}, 10
    ]]>
</select>
<!-- 각 페이지 번호에 알맞은 데이터 호출 -->
<select id="listCriteria" resultType="BoardVO">
    <![CDATA[
        SELECT bno, title, content, writer, regdate, viewcnt
        FROM tbl_board
        WHERE bno > 0
        ORDER BY bno desc, regdate desc
        LIMIT #{pageStart}, #{perPageNum}
    ]]>
</select>

```



```

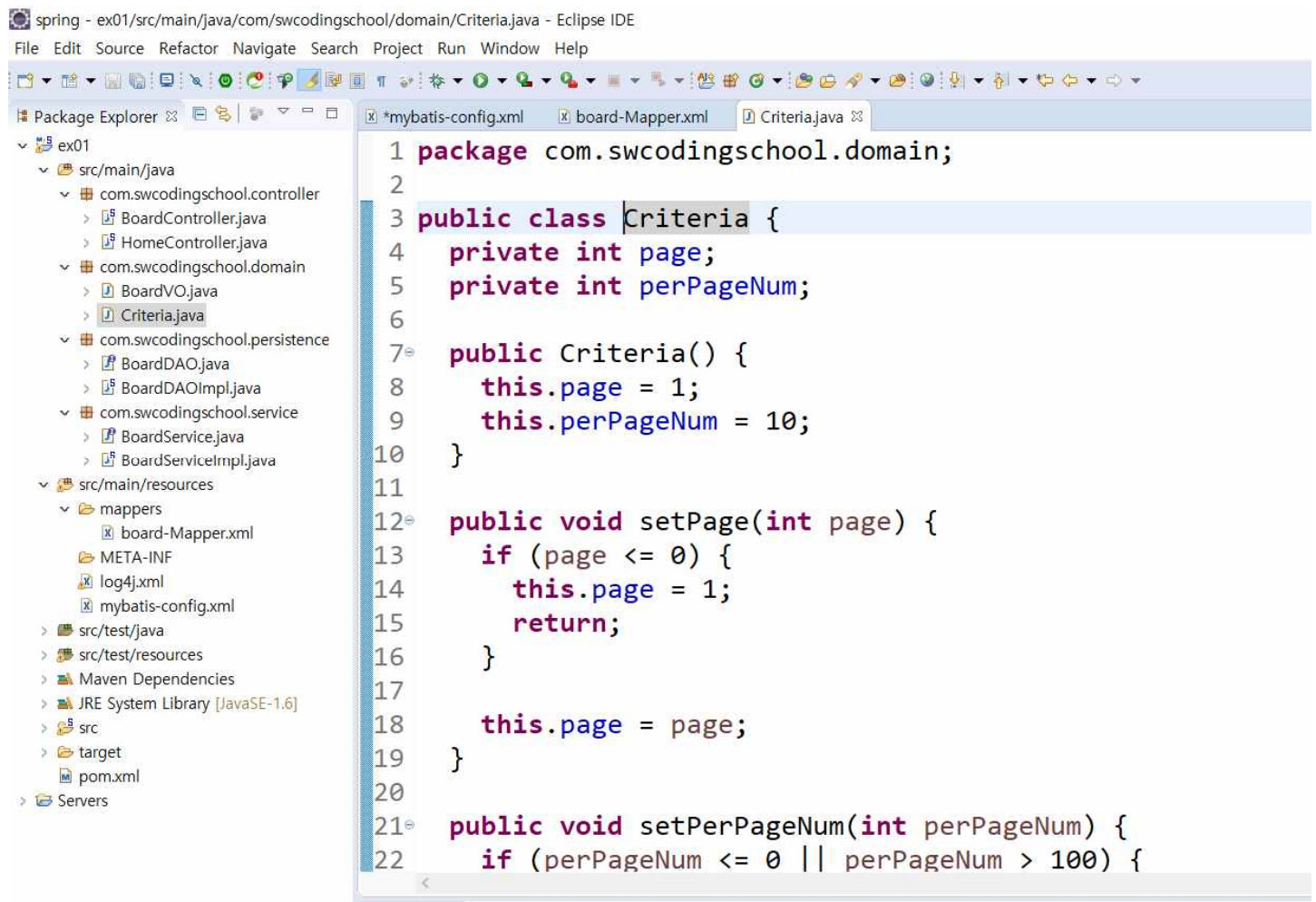
    ]]>
</select>

<!-- 전체 게시글 숫자 파악 -->
<select id="countPaging" resultType="int">
    <![CDATA[
        SELECT COUNT(bno)
        FROM tbl_board
        WHERE bno > 0
    ]]>
</select>

```

3.6.3 추가된 질의 처리를 위한 클래스 추가 - 페이지 기준 클래스

페이징 처리를 위한 기준 클래스를 com.swcodingschool.domain 패키지아래에 Criteria 클래스를 작성하여 추가한다.



```
package com.swcodingschool.domain;
```

```
public class Criteria {
    private int page;
```

```
private int perPageNum;

public Criteria() {
    this.page = 1;
    this.perPageNum = 10;
}

public void setPage(int page) {
    if (page <= 0) {
        this.page = 1;
        return;
    }

    this.page = page;
}

public void setPerPageNum(int perPageNum) {
    if (perPageNum <= 0 || perPageNum > 100) {
        this.perPageNum = 10;
        return;
    }

    this.perPageNum = perPageNum;
}

public int getPage() {
    return page;
}

// method for MyBatis SQL Mapper -
public int getPageStart() {
    return (this.page - 1) * perPageNum;
}

// method for MyBatis SQL Mapper
public int getPerPageNum() {
    return this.perPageNum;
}

@Override
public String toString() {
    return "Criteria [page=" + page + ", "
```

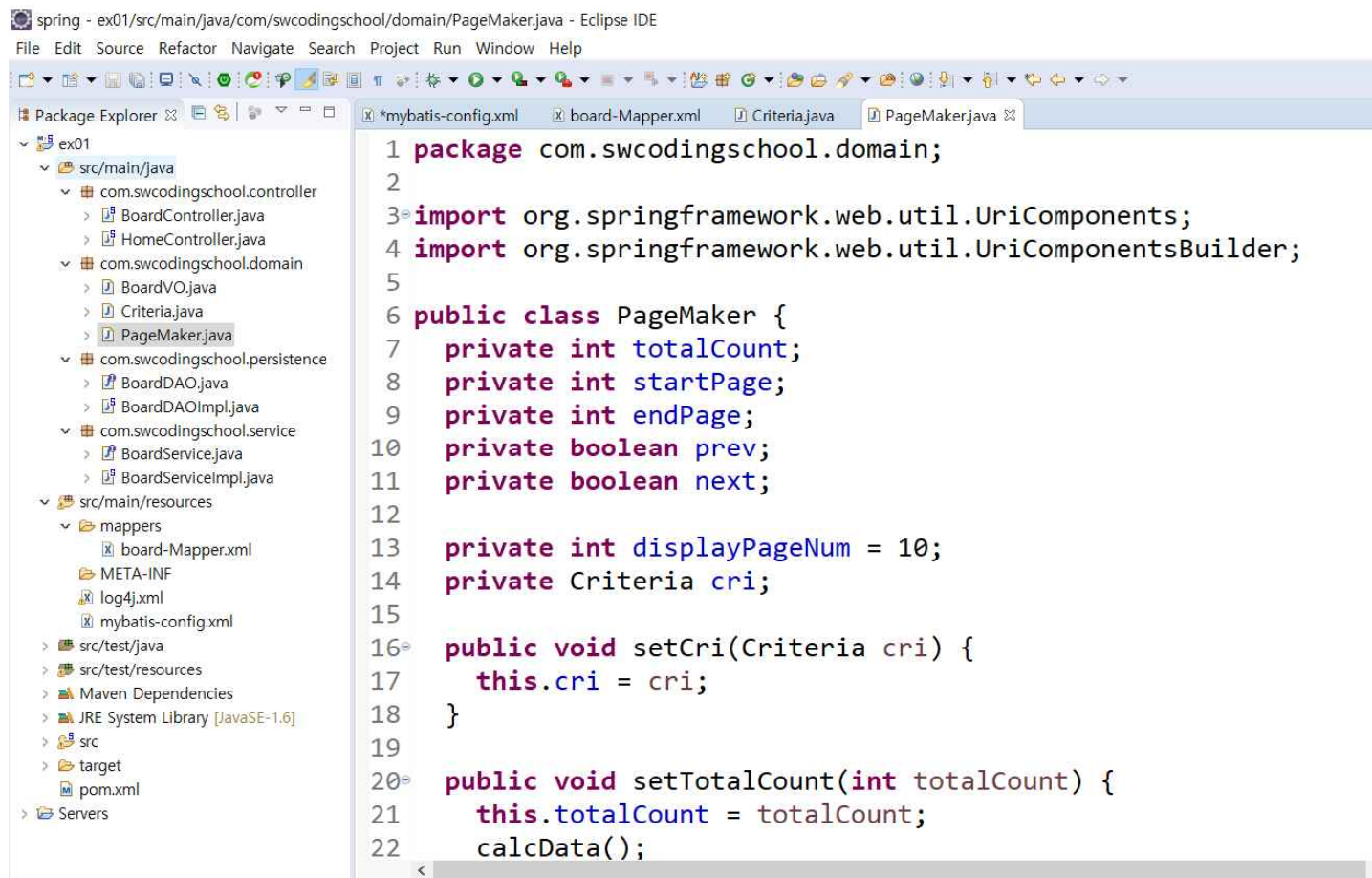
```

        + "perPageNum=" + perPageNum + "]];
    }
}

```

3.6.4 페이지 계산 클래스 추가

페이지 계산을 위해 com.swcodingschool.domain 패키지아래에 PageMaker 클래스를 추가한다.



```

package com.swcodingschool.domain;

import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

public class PageMaker {
    private int totalCount;
    private int startPage;
    private int endPage;
    private boolean prev;
    private boolean next;

    private int displayPageNum = 10;
    private Criteria cri;

    public void setCri(Criteria cri) {

```

```
this.cri = cri;
}

public void setTotalCount(int totalCount) {
    this.totalCount = totalCount;
    calcData();
}

private void calcData() {
    endPage = (int) (Math.ceil(cri.getPage() / (double) displayPageNum) * displayPageNum);
    startPage = (endPage - displayPageNum) + 1;

    int tempEndPage = (int) (Math.ceil(totalCount / (double) cri.getPerPageNum()));

    if (endPage > tempEndPage) {
        endPage = tempEndPage;
    }

    prev = startPage == 1 ? false : true;
    next = endPage * cri.getPerPageNum() >= totalCount ? false : true;
}

public int getTotalCount() {
    return totalCount;
}

public int getStartPage() {
    return startPage;
}

public int getEndPage() {
    return endPage;
}

public boolean isPrev() {
    return prev;
}

public boolean isNext() {
    return next;
}
```

```

public int getDisplayPageNum() {
    return displayPageNum;
}

public Criteria getCri() {
    return cri;
}

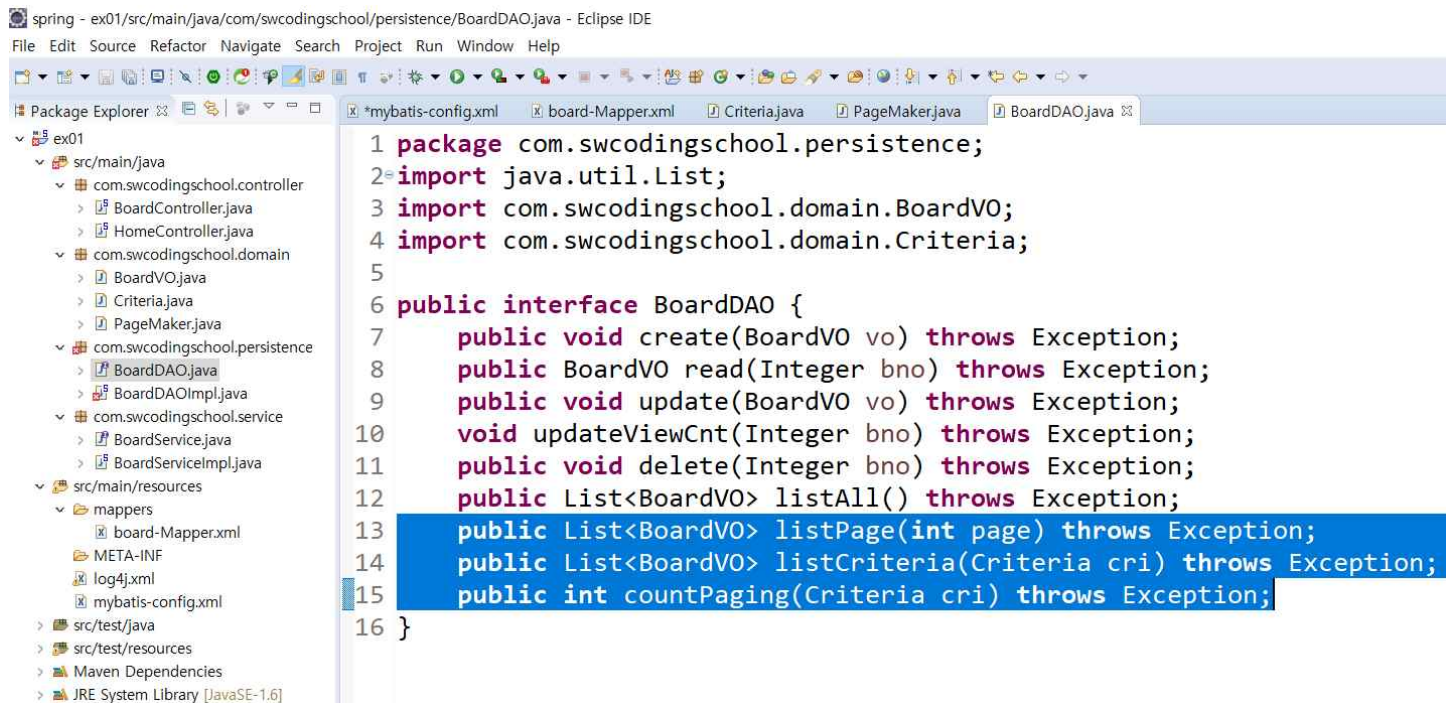
public String makeQuery(int page) {
    UriComponents uriComponents =
        UriComponentsBuilder.newInstance().queryParam("page", page).queryParam("perPageNum",
cri.getPerPageNum()).build();

    return uriComponents.toUriString();
}
}

```

3.6.5 DAO 인터페이스 추가

com.swcodingschool.persistence 패키지의 BoardDAO 인터페이스에 페이지네이션 처리를 위한 메서드를 추가한다.



```

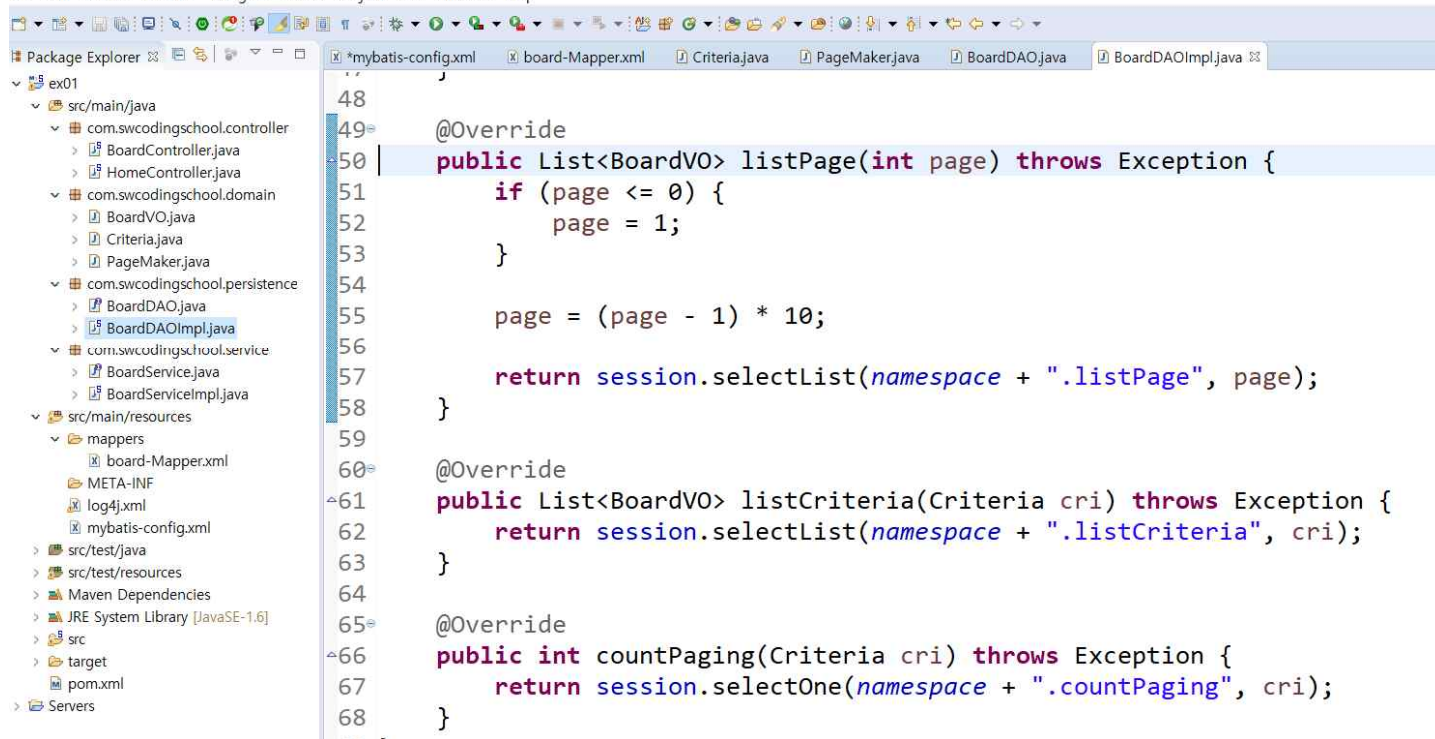
public List<BoardVO> listPage(int page) throws Exception;
public List<BoardVO> listCriteria(Criteria cri) throws Exception;
public int countPaging(Criteria cri) throws Exception;

```

3.6.6 DAO 구현 클래스 오버라이딩

spring - ex01/src/main/java/com/swcodingschool/persistence/BoardDAOImpl.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



```

@Override
public List<BoardVO> listPage(int page) throws Exception {
    if (page <= 0) {
        page = 1;
    }

    page = (page - 1) * 10;

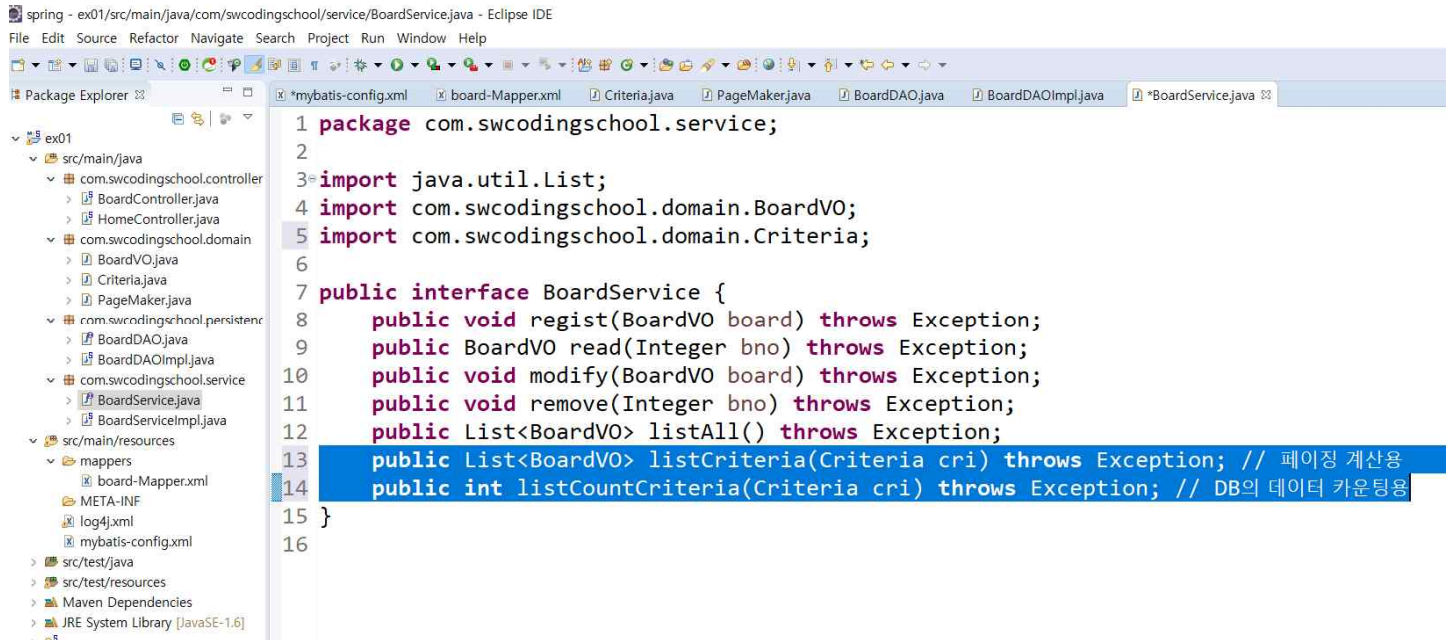
    return session.selectList(namespace + ".listPage", page);
}

@Override
public List<BoardVO> listCriteria(Criteria cri) throws Exception {
    return session.selectList(namespace + ".listCriteria", cri);
}

@Override
public int countPaging(Criteria cri) throws Exception {
    return session.selectOne(namespace + ".countPaging", cri);
}

```


3.6.7 서비스 인터페이스 추가

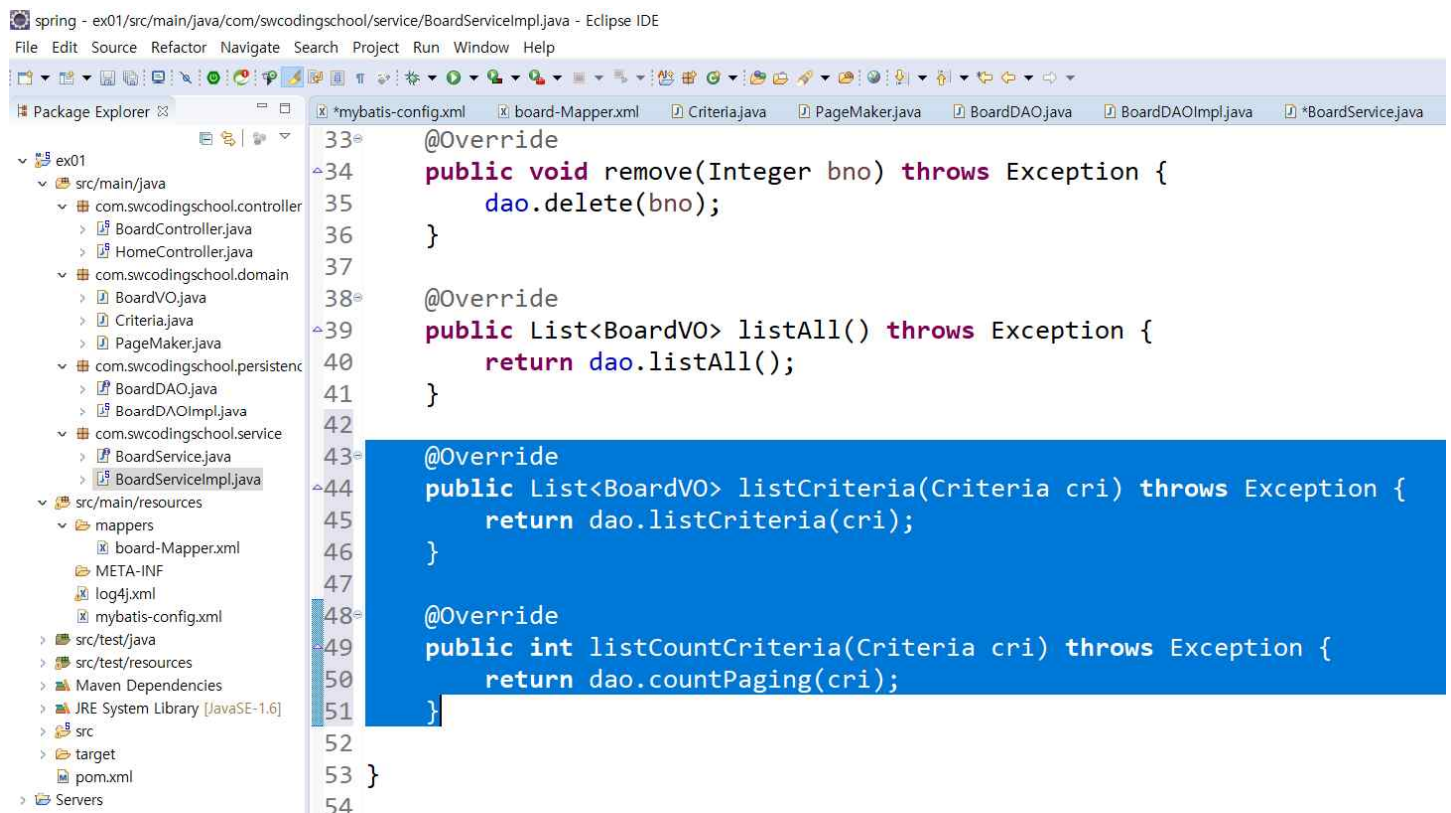


```

public List<BoardVO> listCriteria(Criteria cri) throws Exception; // 페이징 계산용
public int listCountCriteria(Criteria cri) throws Exception; // DB의 데이터 카운팅용

```

3.6.8 서비스구현 오버라이딩



```

@Override
public List<BoardVO> listCriteria(Criteria cri) throws Exception {
    return dao.listCriteria(cri);
}

```



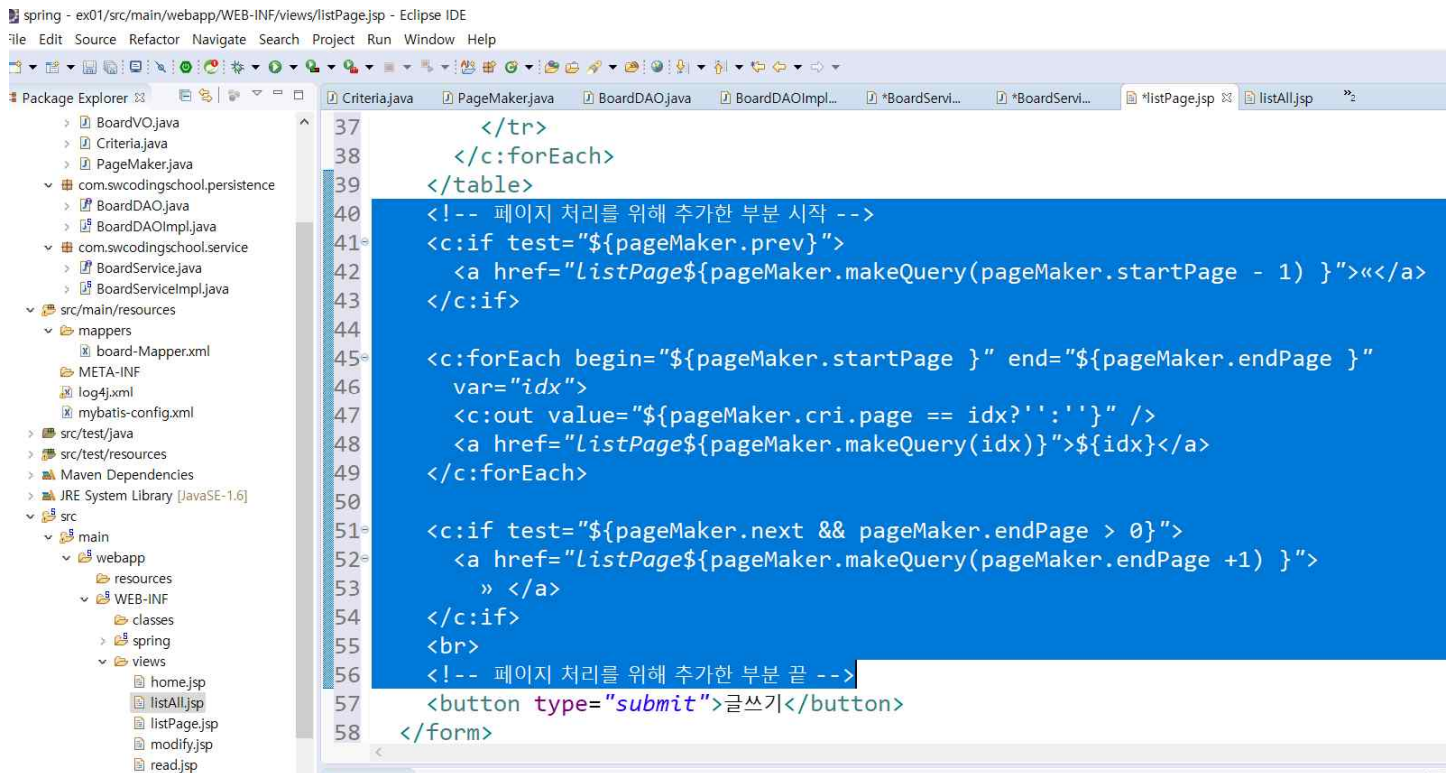
```

@Override
public int listCountCriteria(Criteria cri) throws Exception {
    return dao.countPaging(cri);
}

```

3.6.9 페이지네이션 처리를 위한 JSP생성

페이지처리를 위한 목록보기 페이지 listPage.jsp는 게시글 전체보기 페이지에 페이지처리를 위한 부분을 추가하여 구성할 수 있다.



```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ page session="false"%>
<!DOCTYPE html>
<html>
<head>
<link
                                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
<meta charset="UTF-8">
<title>게시판 목록</title>
</head>
<body>
    <div class="jumbotron">

```

```

<div class="container">
  <h1 class="display-3">게시판 목록보기</h1>
</div>
</div>

<form action="regist" method="get">
  <table class="table table-hover">
    <tr>
      <th>글번호</th>
      <th>제목</th>
      <th>작성자</th>
      <th>작성일</th>
      <th>조회수</th>
    </tr>

    <c:forEach items="${list}" var="boardV0">
      <tr>
        <td>${boardV0.bno}</td>
        <td><a href='/read?bno=${boardV0.bno}'>${boardV0.title}</a></td>
        <td>${boardV0.writer}</td>
        <td><fmt:formatDate pattern="yyyy-MM-dd HH:mm"
          value="${boardV0.regdate}" /></td>
        <td><span class="badge bg-red">${boardV0.viewcnt}</span></td>
      </tr>
    </c:forEach>
  </table>
  <!-- 페이지 처리를 위해 추가한 부분 시작 -->
  <ul class="pagination justify-content-center" style="margin:20px 0">
    <c:if test="${pageMaker.prev}">
      <li class="page-item">
        <a class="page-link" href="listPage${pageMaker.makeQuery(pageMaker.startPage - 1)
}"><</a>
      </li>
    </c:if>

    <c:forEach begin="${pageMaker.startPage }" end="${pageMaker.endPage }"
      var="idx">
      <li class="page-item">
        <c:out value="${pageMaker.cri.page == idx?'':''}" />
        <a class="page-link" href="listPage${pageMaker.makeQuery(idx)}">${idx}</a>
      </li>
    </c:forEach>
  </ul>

```

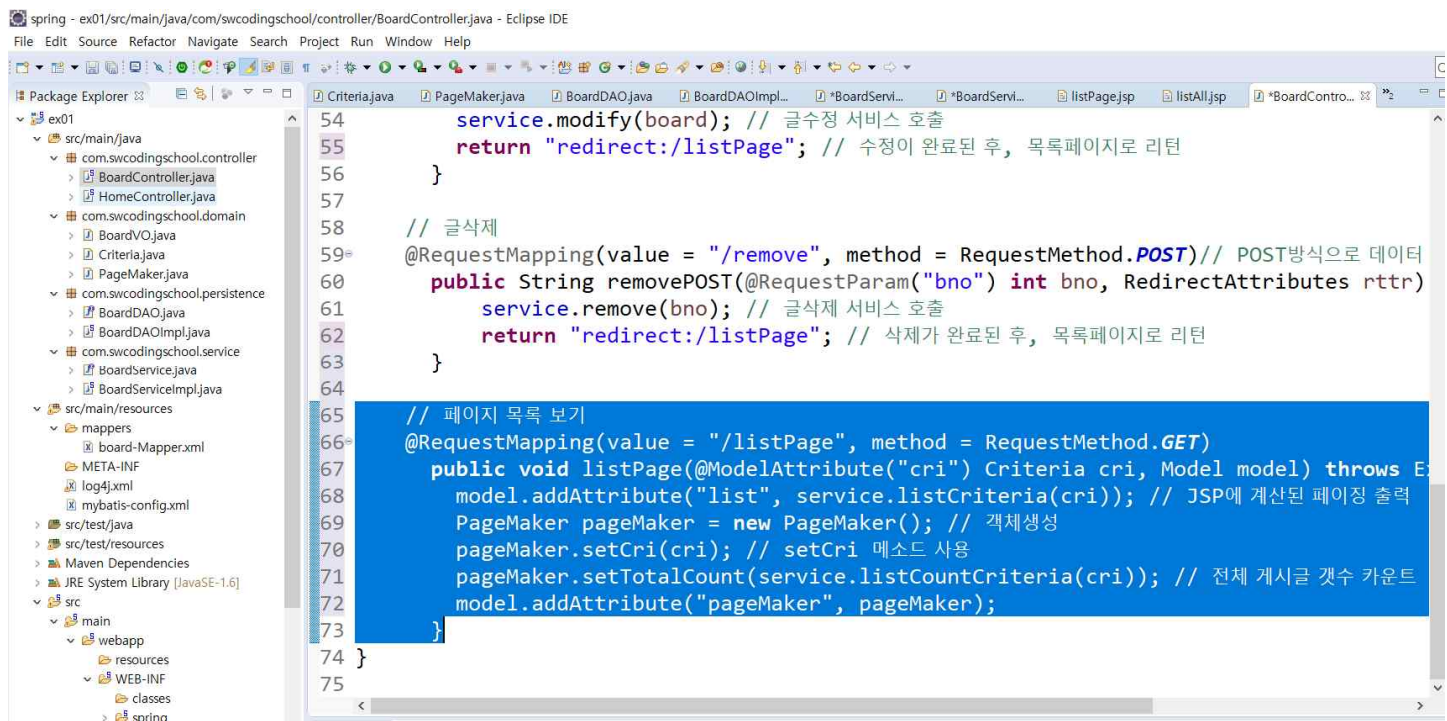
```

<c:if test="${pageMaker.next && pageMaker.endPage > 0}">
    <li class="page-item">
        <a class="page-link" href="listPage${pageMaker.makeQuery(pageMaker.endPage +1) }">
            » </a>
        </li>
    </c:if>
</ul>
<br>
<!-- 페이지 처리를 위해 추가한 부분 끝 -->
<button type="submit">글쓰기</button>
</form>

</body>
</html>

```

3.6.10 컨트롤러 메서드 추가



컨트롤러에 listPage를 추가하고 remove, modify, regist메서드의 redirect 값을 모두 listPage로 변경한다. 그리고 home.jsp의 form action값도 listAll에서 listPage로 반드시 변경한다.

```

package com.swcodingschool.controller;

import javax.inject.Inject;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;

```

```

import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.swcodingschool.service.BoardService;
import com.swcodingschool.domain.BoardVO;
import com.swcodingschool.domain.Criteria;
import com.swcodingschool.domain.PagerMaker;

@Controller // 컨트롤러임을 명시
@RequestMapping(value = "/") // 주소 패턴
public class BoardController {
    @Inject // 주입(심부름꾼) 명시
    private BoardService service; // Service 호출을 위한 객체생성

    // 게시판 목록보기
    @RequestMapping(value= "/listAll", method = RequestMethod.GET) // 주소 호출 명시 . 호출
    하려는 주소 와 REST 방식설정 (GET)
    public void listAll(Model model)throws Exception { // 메소드 인자값은 model 인터페이스
    (jsp전달 심부름꾼)
        model.addAttribute("list",service.listAll()); // jsp에 심부름할 내역(서비스 호
출)
    }

    // 게시판 글쓰기
    @RequestMapping(value = "/regist", method = RequestMethod.GET) // GET 방식으로 페이지
호출
    public void registerGET(BoardVO board, Model model) throws Exception {
    }
    @RequestMapping(value = "/regist", method = RequestMethod.POST) // POST방식으로 내용 전
송
    public String registPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
    // 인자값으로 REDIRECT 사용
        service.regist(board); // 글작성 서비스 호출
        return "redirect:/listPage"; // 작성이 완료된 후, 목록페이지로 리턴
    }

    // 게시판 읽기
    @RequestMapping(value = "/read", method = RequestMethod.GET) // GET 방식으로 페이지 호
출

```

```

        public void read(@RequestParam("bno")int bno, Model model) throws Exception{
            // 인자값은 파라미터 값으로 기본키인 글번호를 기준으로 Model을 사용하여 불러옴
            model.addAttribute(service.read(bno)); // read 서비스 호출
        }

// 글수정
@RequestMapping(value = "/modify", method = RequestMethod.GET) // GET 방식으로 페이지
호출
        public void modifyGET(int bno, Model model) throws Exception {
            model.addAttribute(service.read(bno)); // 수정을 위한 글읽기 서비스 호출
        }
@RequestMapping(value = "/modify", method = RequestMethod.POST)// POST방식으로 데이터
전송
        public String modifyPOST(BoardVO board, RedirectAttributes rttr) throws Exception {
            service.modify(board); // 글수정 서비스 호출
            return "redirect:/listPage"; // 수정이 완료된 후, 목록페이지로 리턴
        }

// 글삭제
@RequestMapping(value = "/remove", method = RequestMethod.POST)// POST방식으로 데이터
전송
        public String removePOST(@RequestParam("bno") int bno, RedirectAttributes rttr)
throws Exception{
            service.remove(bno); // 글삭제 서비스 호출
            return "redirect:/listPage"; // 삭제가 완료된 후, 목록페이지로 리턴
        }

// 페이지 목록 보기
@RequestMapping(value = "/listPage", method = RequestMethod.GET)
        public void listPage(@ModelAttribute("cri") Criteria cri, Model model) throws
Exception {
            model.addAttribute("list", service.listCriteria(cri)); // JSP에 계산된 페이지 출력
            PageMaker pageMaker = new PageMaker(); // 객체생성
            pageMaker.setCri(cri); // setCri 메소드 사용
            pageMaker.setTotalCount(service.listCountCriteria(cri)); // 전체 게시글 갯수 카운트
            model.addAttribute("pageMaker", pageMaker);
        }
    }

```