



박 경 태  
[comsi.java@gmail.com](mailto:comsi.java@gmail.com)

# 고급 자바 프로그래밍 : STS를 이용한 Spring 프로그래밍

# 강의 내용

순서	내 용
1	<ul style="list-style-type: none"><li>Spring IoC를 이용한 비즈니스 컴포넌트 만들기</li></ul>
2	<ul style="list-style-type: none"><li>Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기</li><li>Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리<ul style="list-style-type: none"><li>JdbcTemplate 클래스를 이용한 JDBC의 반복적 코드 제거와 SQL분리</li></ul></li></ul>
3	<ul style="list-style-type: none"><li>Spring MVC를 이용한 MVC 아키텍쳐 적용하기</li></ul>
4	<ul style="list-style-type: none"><li>Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등)</li></ul>
5	<ul style="list-style-type: none"><li>Spring과 MyBatis 연동하기</li><li>Spring과 JPA 연동하기<ul style="list-style-type: none"><li>표준 ORM(Object Relational Mapping)을 이용한 RDB와의 객체 지향적 연결과 관리</li></ul></li></ul>

# http://github.com/hopypark

hopypark Advanced Java - Dropbx

GitHub, Inc. [US] | https://github.com/hopypark

나의 북마크 헨진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 깃모닝팝스 고급두뇌를 위한 학습 Time Series Forecast Calculation of Information [Linear Algebra] Basic

Search or jump to... Pull requests Issues Marketplace Explore

Overview Repositories 6 Stars 0 Followers 0 Following 0

Pinned repositories

- Lecture2018
- Intro\_ML

Customize your pinned repositories

Jupyter Notebook

hopypark

Add a bio Edit profile

135 contributions in the last year

Contribution settings

Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep

Mon Wed Fri

Learn how we count contributions.

Less More

Contribution activity

Jump to ▾ 2018

September 2018

Created 22 commits in 2 repositories

Jump to ▾ 2017

Jump to ▾ 2016

Lecture2018/JavaSpring x Advanced Java - Dropbo x

GitHub, Inc. [US] | https://github.com/hopypark/Lecture2018/tree/master/JavaSpring

나의 북마크 텐진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 학습 Time Series Forecasting Calculation of Information [Linear Algebra] Back

Search or jump to... / Pull requests Issues Marketplace Explore

hopypark / Lecture2018 Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master Lecture2018 / JavaSpring / Create new file Upload files Find file History

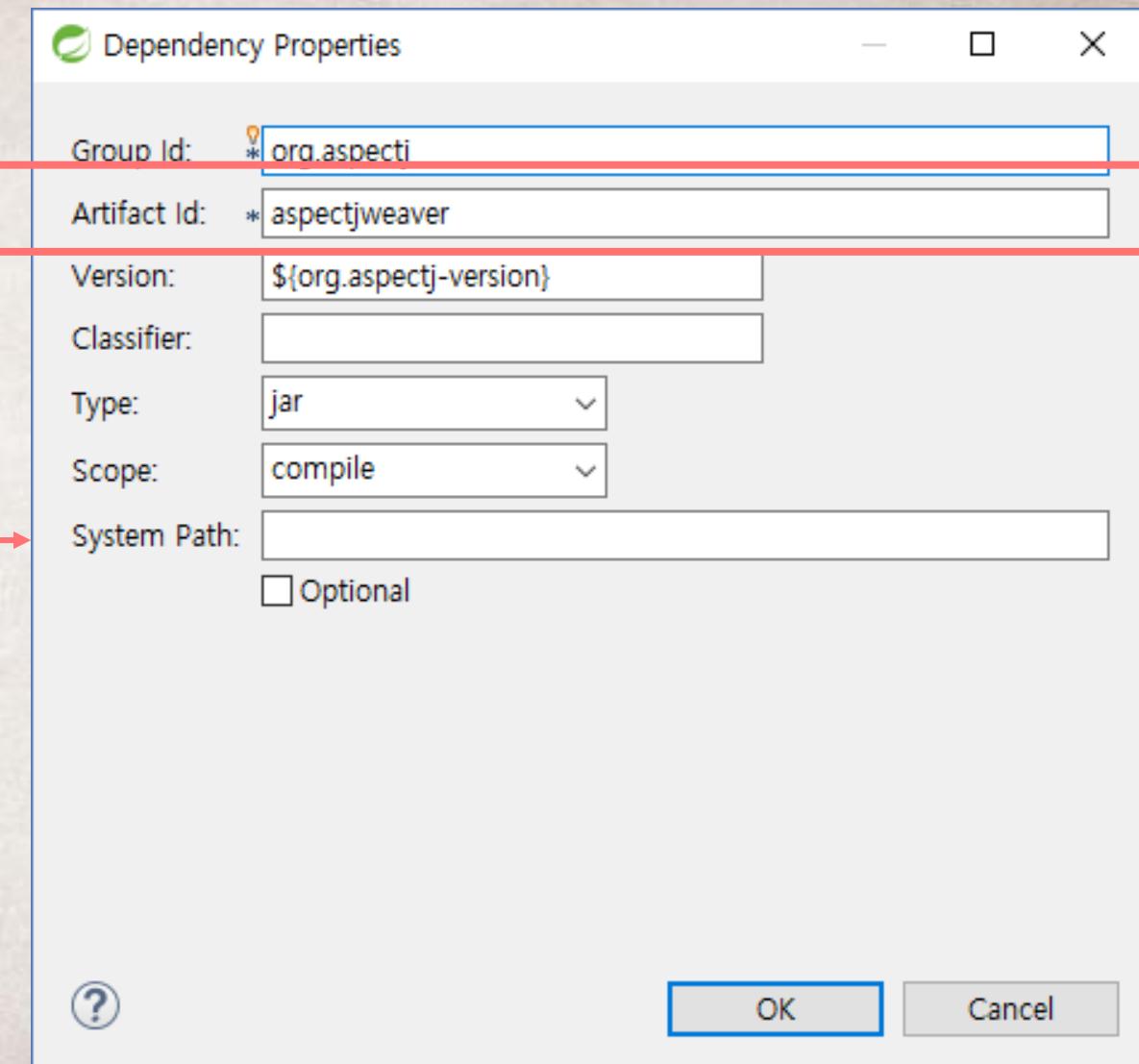
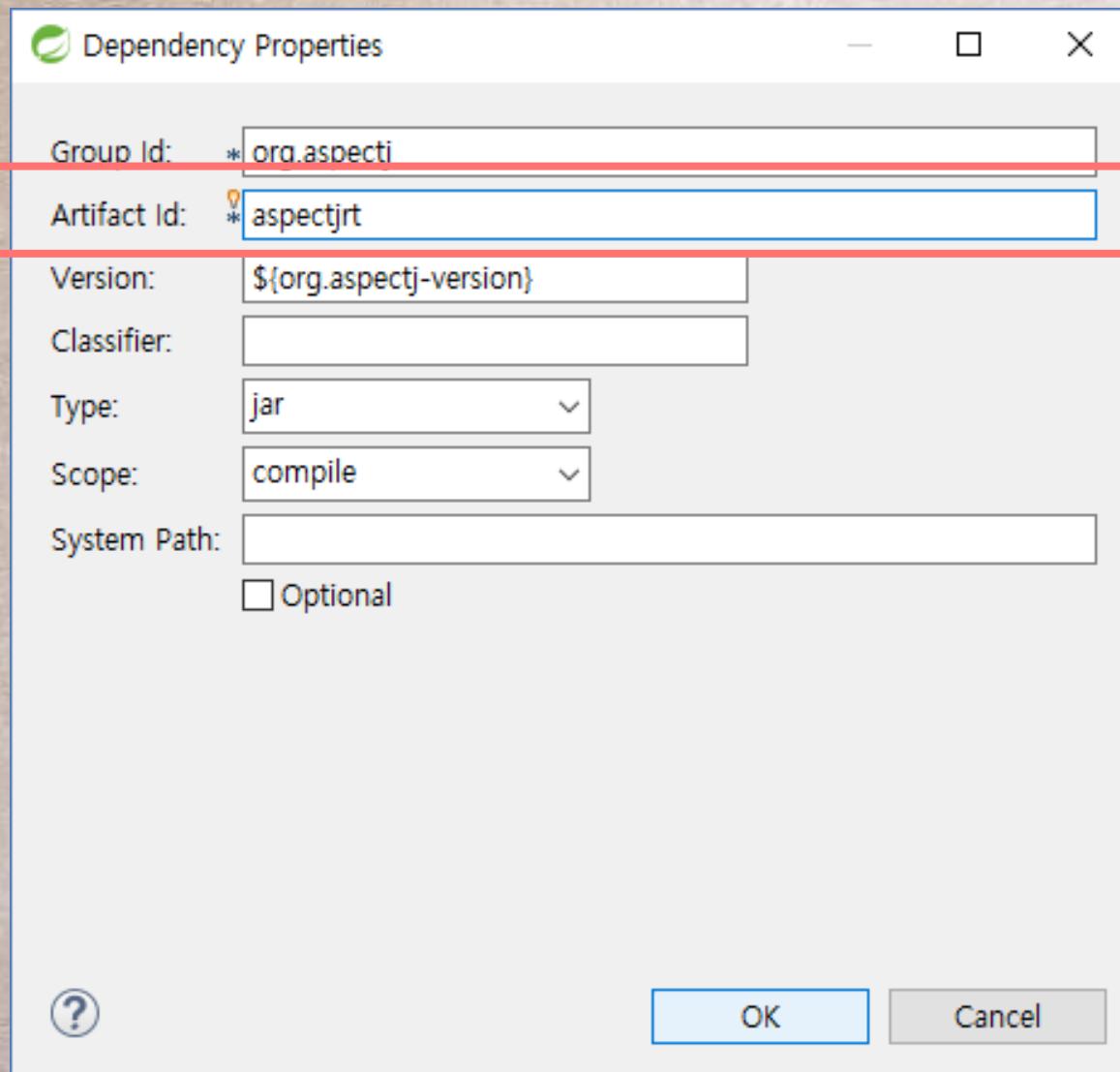
hopypark Add files via upload Latest commit 7c4ddc7 a minute ago

..

2018_01_spring.zip	Add files via upload	3 days ago
BoardWeb.zip	Add files via upload	16 days ago
Week00.H2Database와 STS설치.pdf	Add files via upload	9 days ago
Week02.스프링과 JDBC.pdf	Add files via upload	a minute ago
Week03.스프링과 MVC.pdf	Add files via upload	3 days ago
h2database_conn.txt	Add files via upload	16 days ago
readme.md	Create readme.md	16 days ago
sql.txt	Add files via upload	16 days ago
user.zip	Add files via upload	9 days ago
webapp_jsp.zip	Add files via upload	3 days ago
readme.md		

오류처리

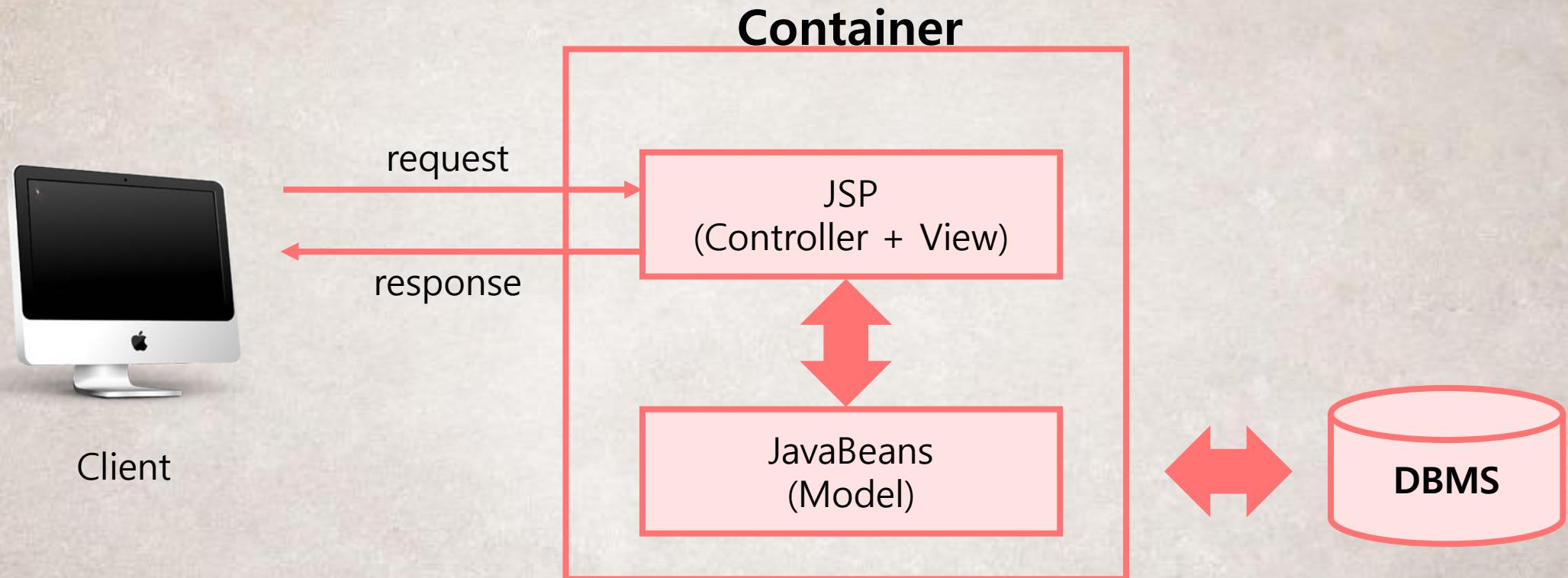
# AOP 설정 오류(aspectj 처리 API 변경)



# Model 1 아키텍처로 게시판 개발

# Model 1 아키텍처 구조

- Model 1 아키텍처는 JSP와 JavaBean만 사용하여 웹 개발(2000년대 초~)



→ 자바에서 Bean은 객체를 의미함.

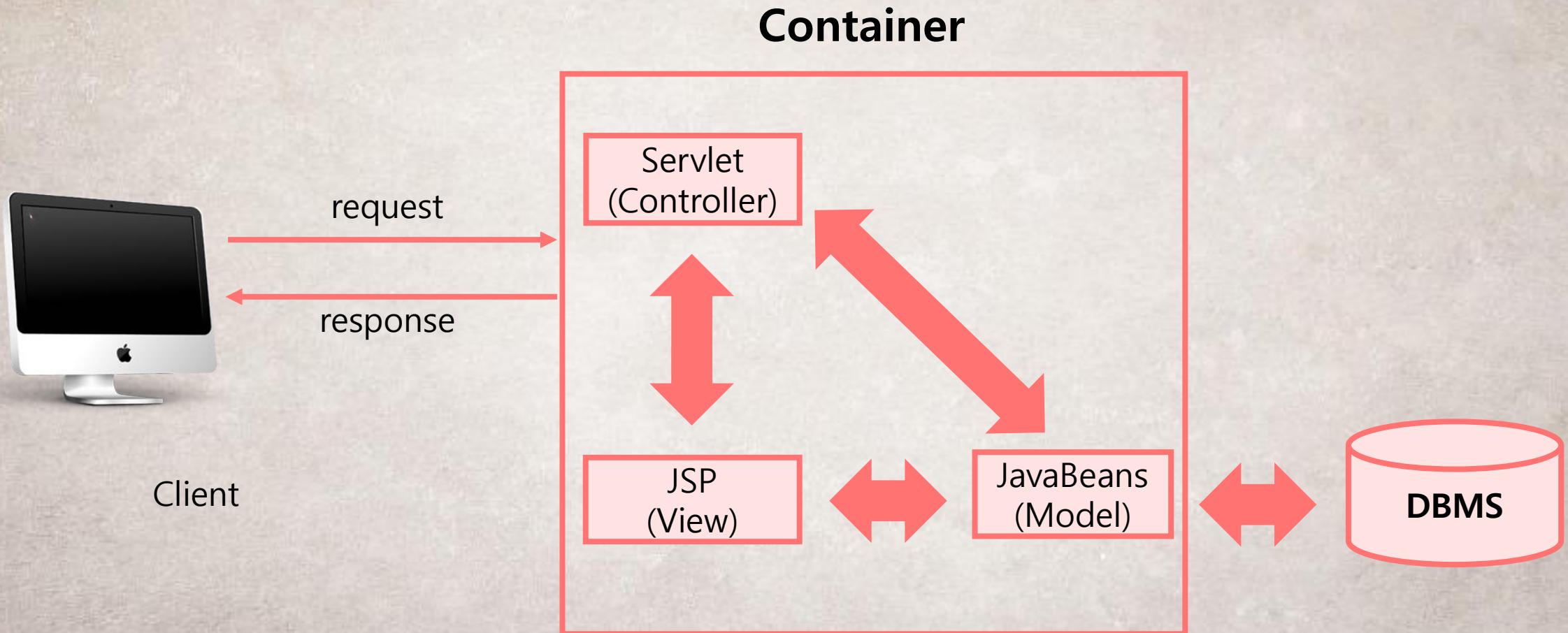
→ 자바 서버 페이지(Java Server Pages, JSP)는 HTML내에 자바 코드를 삽입하여 웹 서버에서 동적으로 웹 페이지를 생성하여 웹 브라우저에 돌려주는 언어

# Model 1 아키텍처 구조

- **Model** : 데이터베이스 연동 로직을 제공하면서 DB에서 검색한 데이터가 저장되는 자바 객체
  - 스프링 IoC와 AOP 실습을 하면서 VO와 DAO 클래스를 사용했으며, 이 두 클래스가 바로 Model 기능의 자바 객체
  - 모델1에서는 JSP 파일이 가장 중요한 역할을 수행 – Controller와 View
    - Controller는 사용자의 요청 처리와 관련된 자바 코드
    - View는 사용자와 상호작용하는 영역
- **단점**
  - Controller와 View 기능을 모두 처리하는 특징으로 자바코드와 마크업 언어가 섞임
  - JSP 파일에 대한 디버깅이나 유지보수가 어려워 짐
  - 적인 개발인력과 간단한 프로젝트에 적합

# Model 2 아키텍처(MVC) 등장

- Model 1 구조의 단점을 보완하기 위해 만들어진 구조



# Model 2 아키텍처 구조

- 자바 로직과 화면 디자인이 통합되어 유지보수가 어렵다는 Model1의 문제를 해결하기 위해 고안된 웹 개발 모델이 Model2(MVC)
- Model 2는 가장 큰 특징은 **Controller**의 등장이며 이는 서블릿 클래스를 중심으로 구현됨
  - Model 1의 JSP에서 자바 코드만 Controller로 이동하면 Model 2가 됨
  - **MVC아키텍처에서 가장 중요한 부분이 Controller인데, 이 Controller를 성능과 유지보수의 편의성을 고려하여 잘 만드는 것이 중요함**
- MVC 아키텍처에서 각요소의 기능과 개발 주체

기능	구성요소	개발주체
Model	VO, DAO 클래스	자바 개발자
View	JSP 페이지	웹디자이너
Controller	Servlet	자바 개발자 또는 MVC 프레임워크

→자바 서블릿(Java Servlet)은 자바를 사용하여 웹페이지를 동적으로 생성하는 서버측 프로그램

# 모델2를 이용한 게시판 구현

- JSP를 이용한 View 페이지

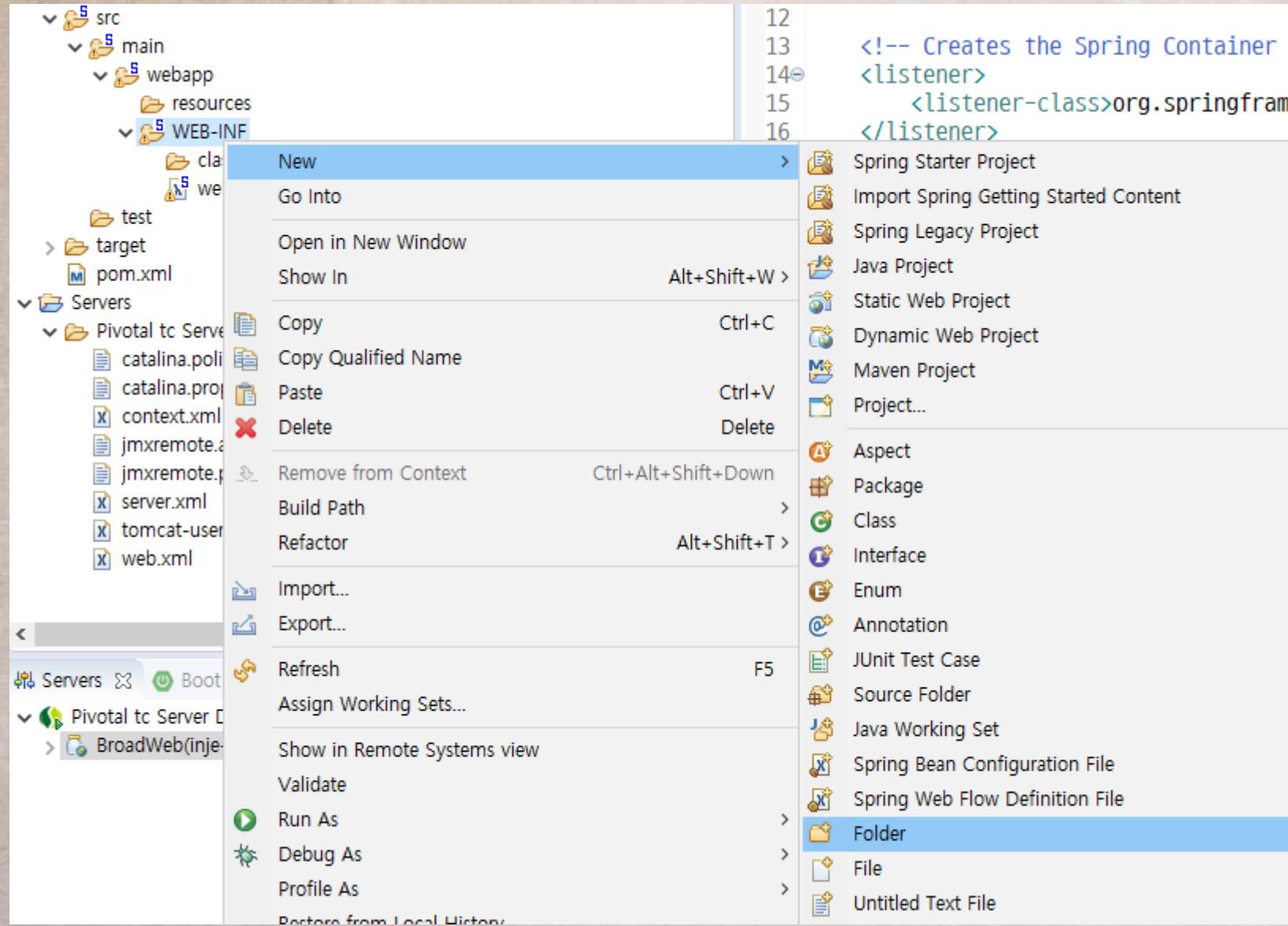
# Project folder structure

- **src/main/java** : java 소스 폴드(controller, model)
- **src/main/resources**: 자바코드에서 사용할 리소스(mybatis의 Mapper, sqlMapConfig.xml 등)
- 웹 디렉토리 → 직접 생성
  - src/main/webapp: jsp 파일과 web application content.
  - src/main/webapp/resource : js, css, image등
  - Src/main/webapp/WEB-INF/web.xml: 웹프로젝트 배치 기술서
  - src/main/webapp/WEB-INF/classes : 컴파일된 클래스
  - src/main/webapp/WEB-INF/spring: 스프링 환경 설정파일
    - src/main/webapp/WEB-INF/spring/root-context.xml:
    - src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml: 클라이언트 요청 처리
  - src/main/webapp/WEB-INF/views: jsp 파일

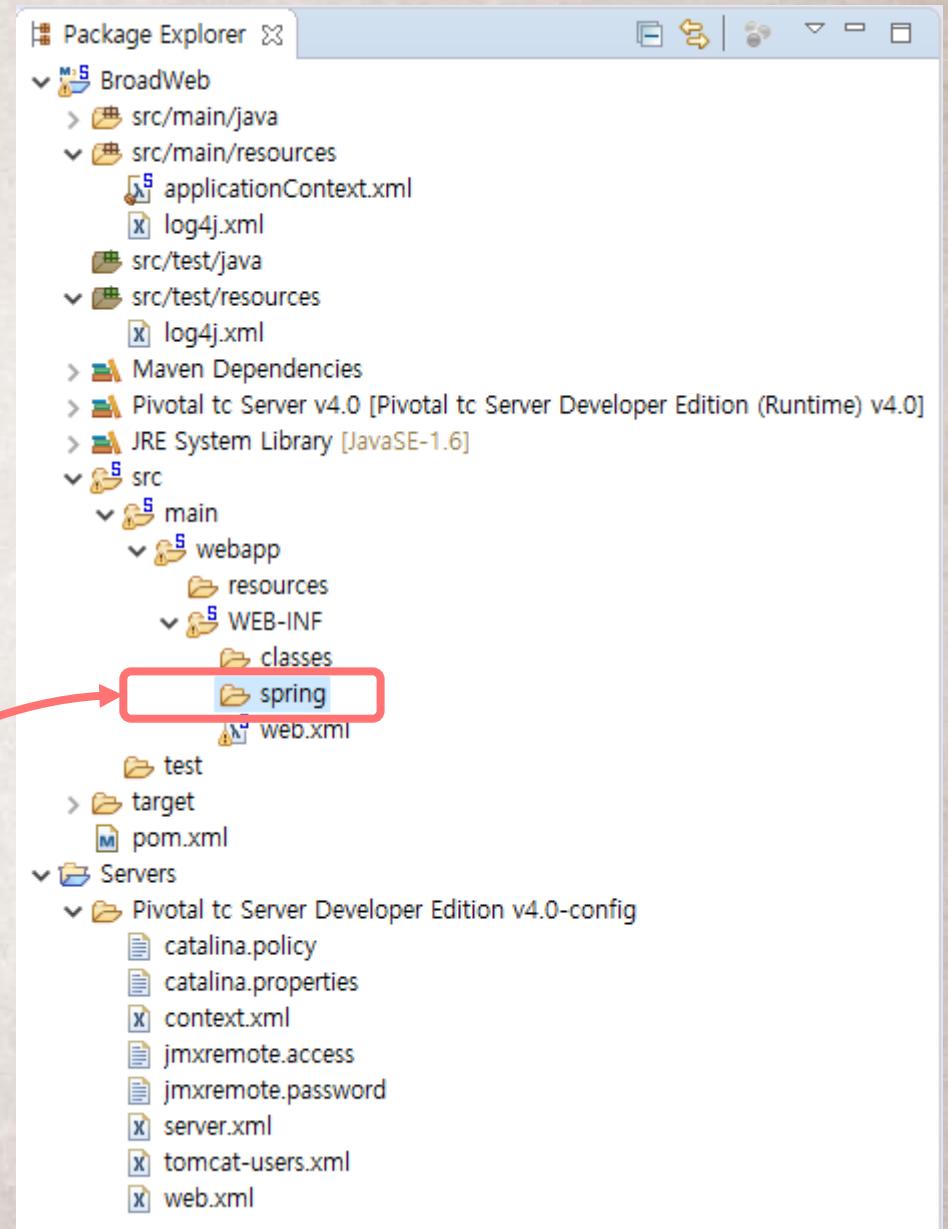
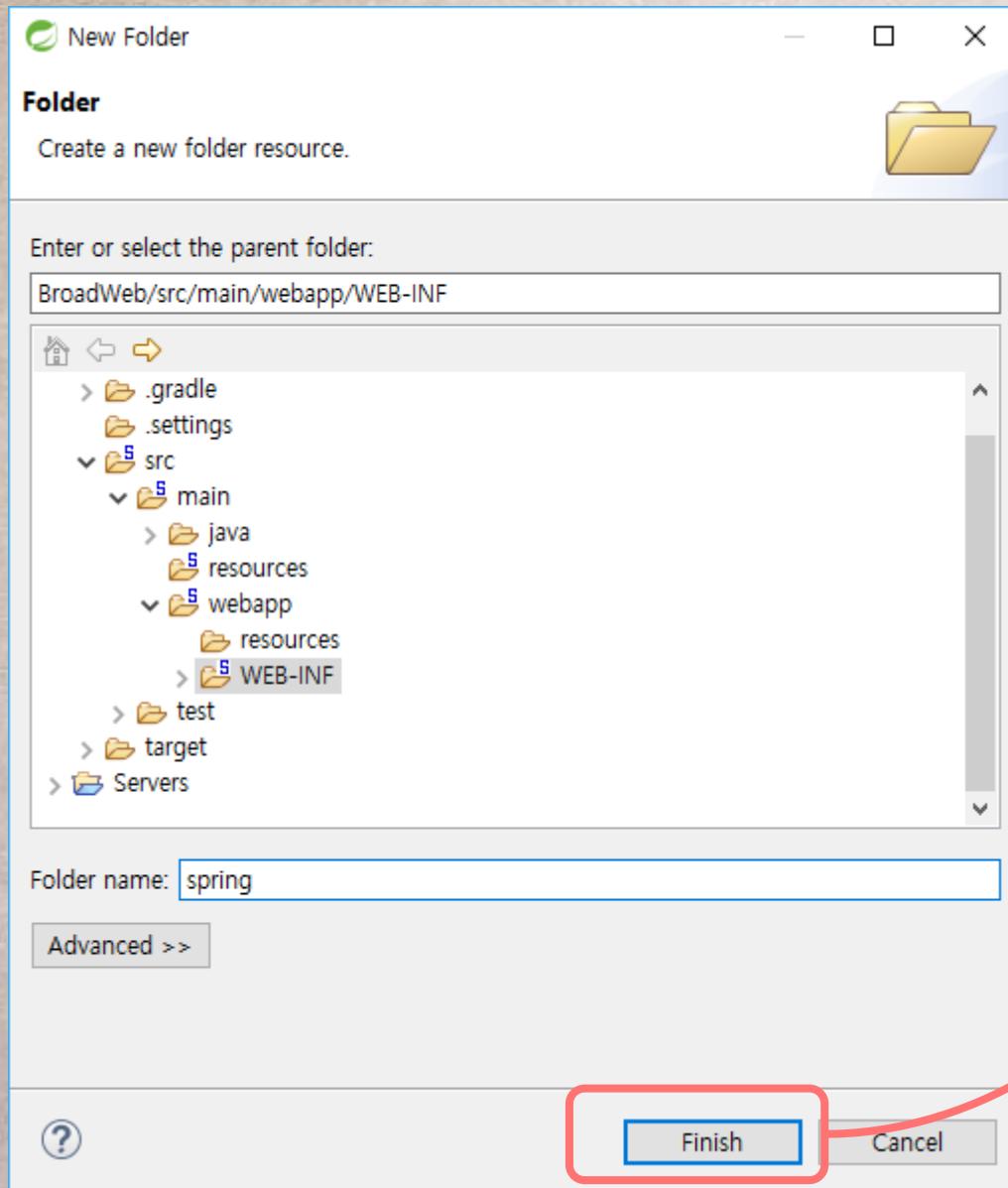
# web.xml 확인 및 파일 추가

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
5
6   <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
7   <context-param>
8     <param-name>contextConfigLocation</param-name>
9     <!-- 스프링의 환경 설정 파일인 root-context.xml 파일 참조 -->
10    <param-value>/WEB-INF/spring/root-context.xml</param-value> ← 파일 추가
11  </context-param>
12
13  <!-- Creates the Spring Container shared by all Servlets and Filters -->
14  <listener>
15    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
16  </listener>
17
18  <!-- Processes application requests -->
19  <servlet>
20    <servlet-name>appServlet</servlet-name>
21    <!-- 스프링에 내장된 서블릿 클래스 -->
22    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
23  <init-param>
24    <param-name>contextConfigLocation</param-name>
25    <!-- xml 파일에 정의된 객체를 로딩 -->
26    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value> ← 파일 추가
27  </init-param>
28  <!-- 가장 첫번째 순위를 뜻함 -->
29  <load-on-startup>1</load-on-startup>
30 </servlet>
31  <servlet-mapping>
32    <servlet-name>appServlet</servlet-name>
33    <url-pattern>/</url-pattern>
34    <!-- DispatcherServlet이 모든 요청 가로챔 -->
35    <!-- 특정 url로 변경하여 사용가능 ex) *.do -->
36  </servlet-mapping>
37 </web-app>
```

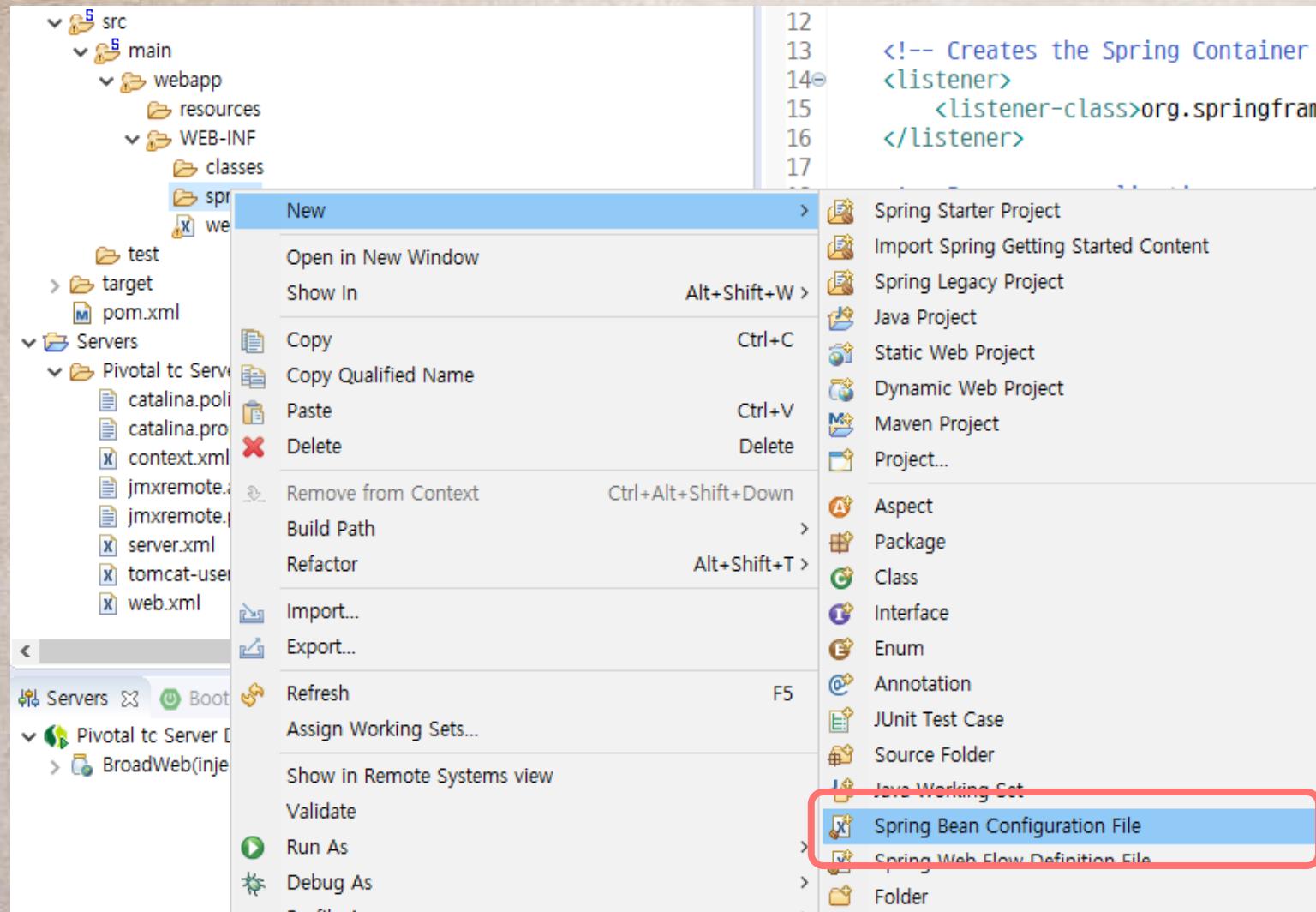
# /WEB-INF/spring/root-context.xml[스프링 설정파일]



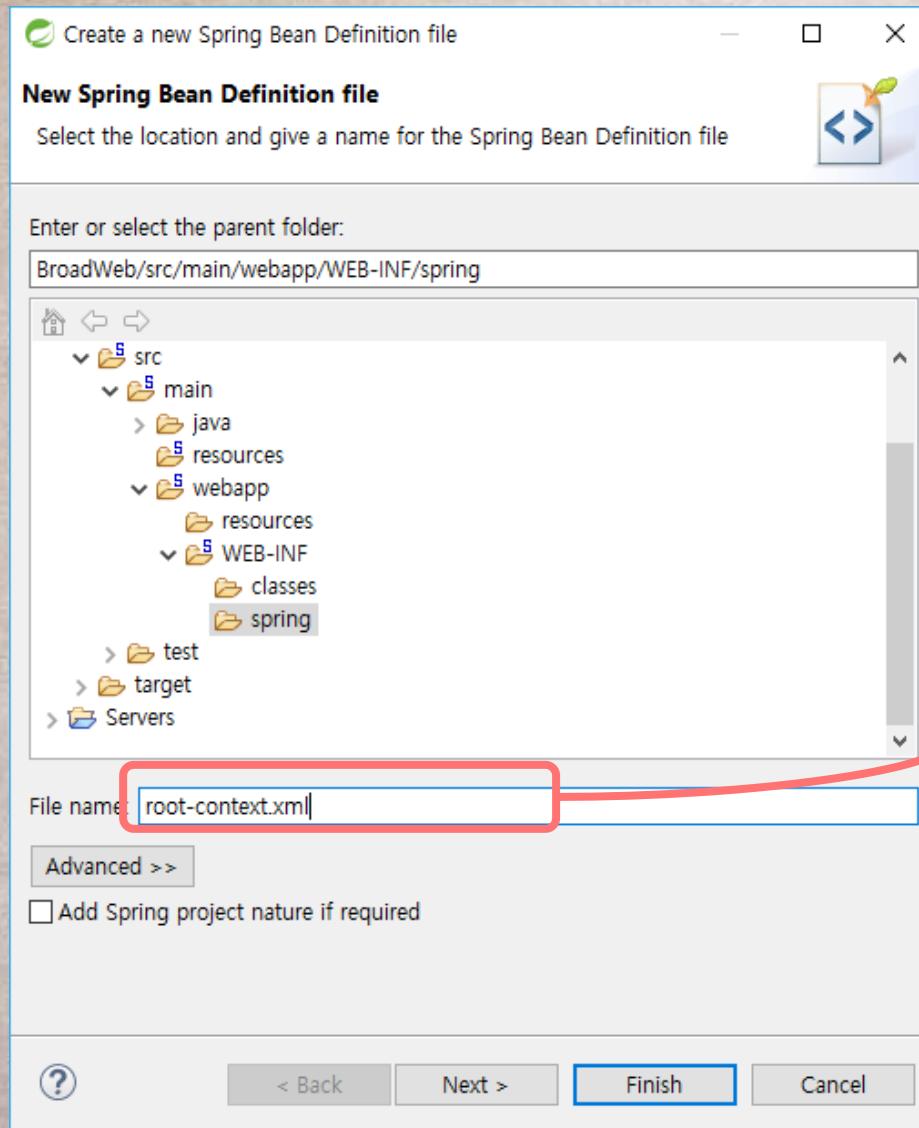
# /WEB-INF/spring/root-context.xml[스프링 설정파일]



# /WEB-INF/spring/root-context.xml[스프링 설정파일]

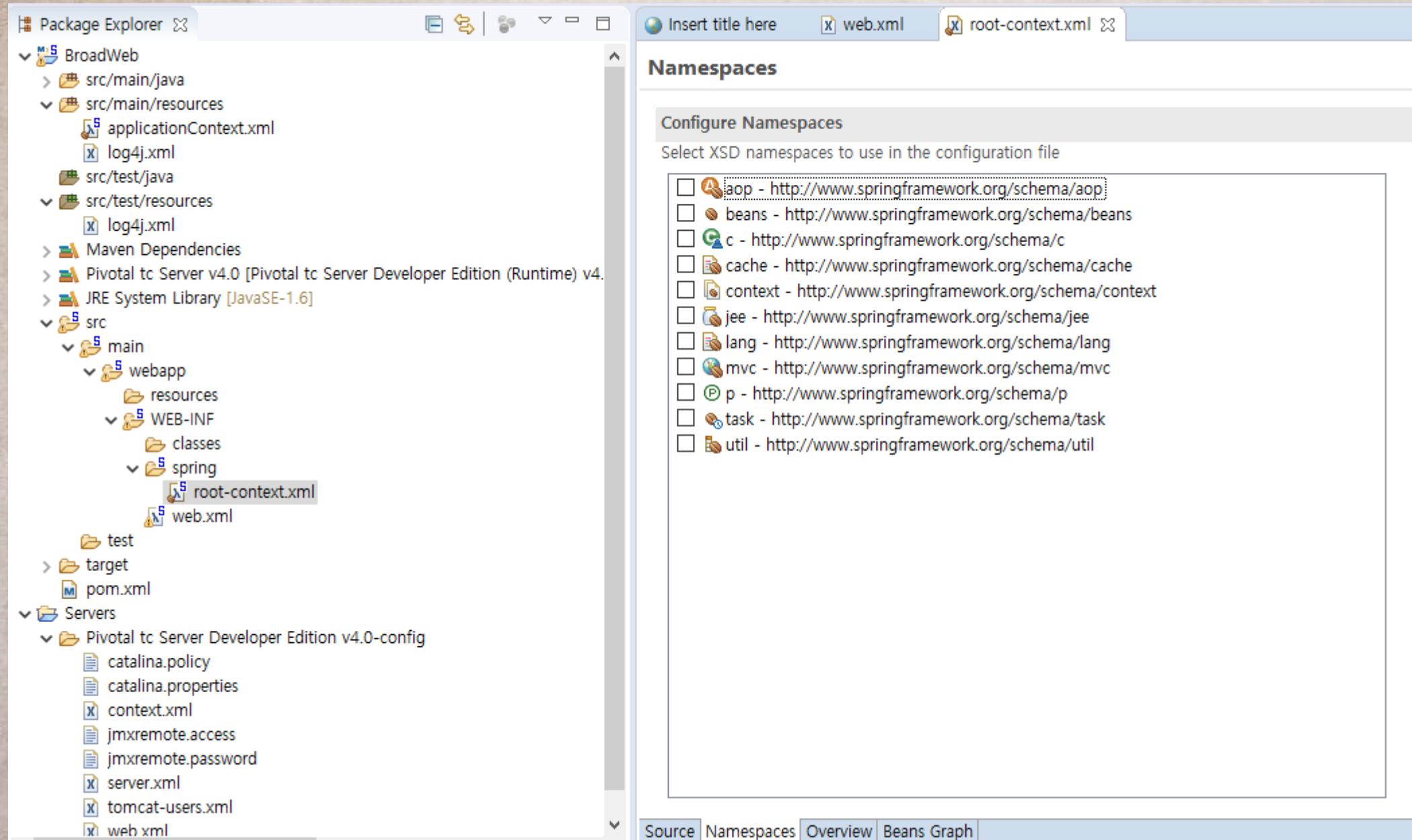


# /WEB-INF/spring/root-context.xml[스프링 설정파일]

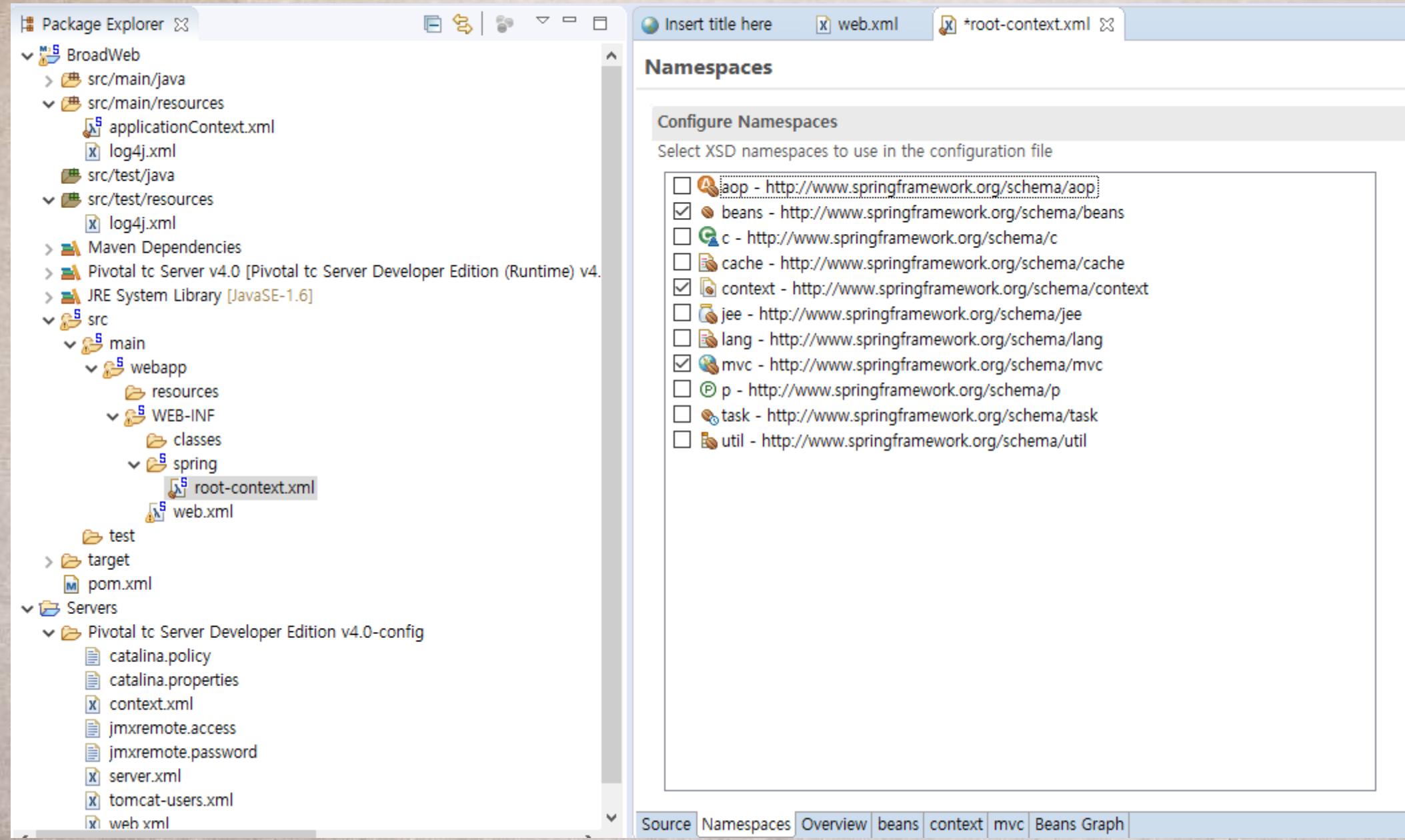


root-context.xml

# root-context.xml 파일 수정(namespace 설정)



# root-context.xml 파일 수정(namespace 설정)

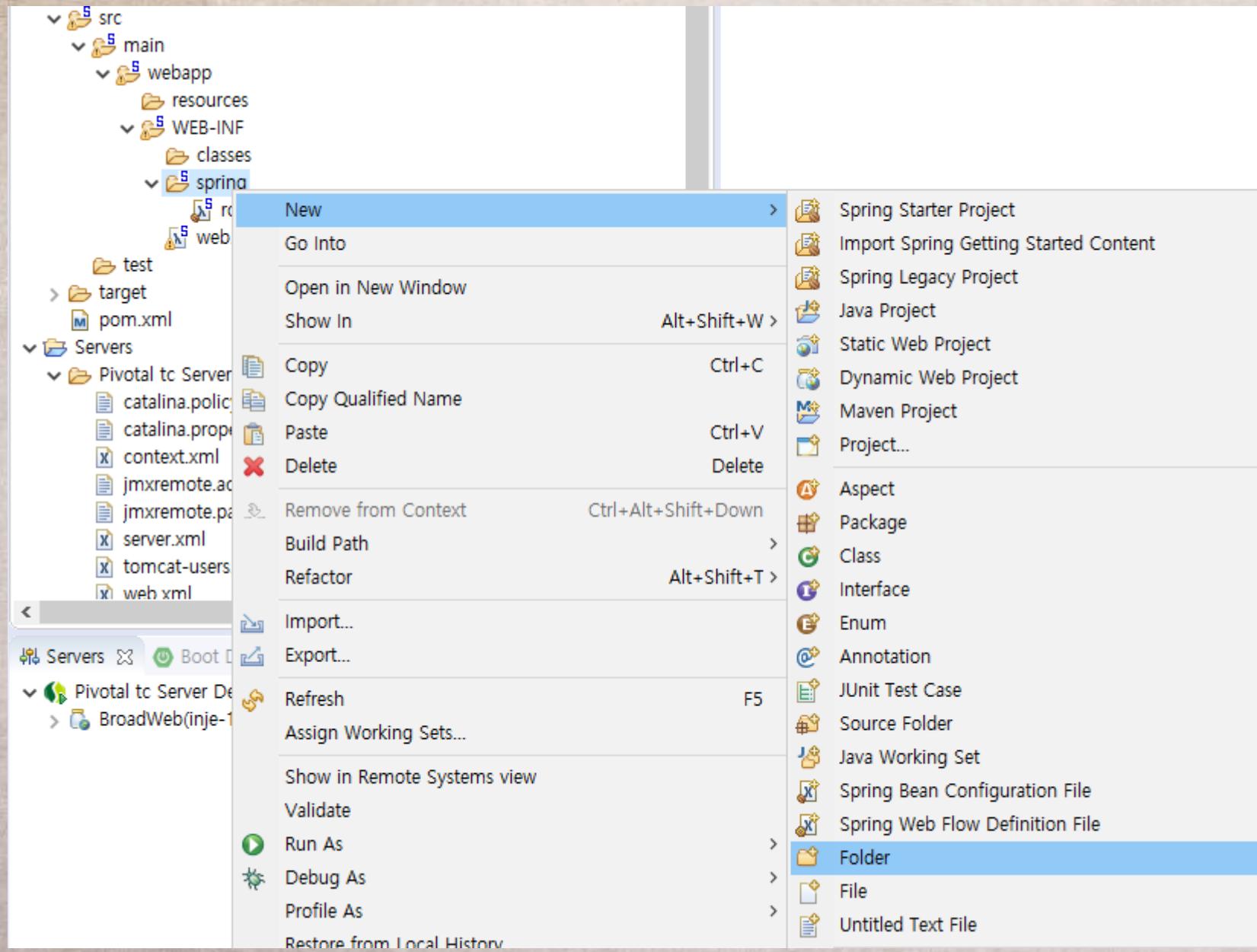


# root-context.xml 파일 설정

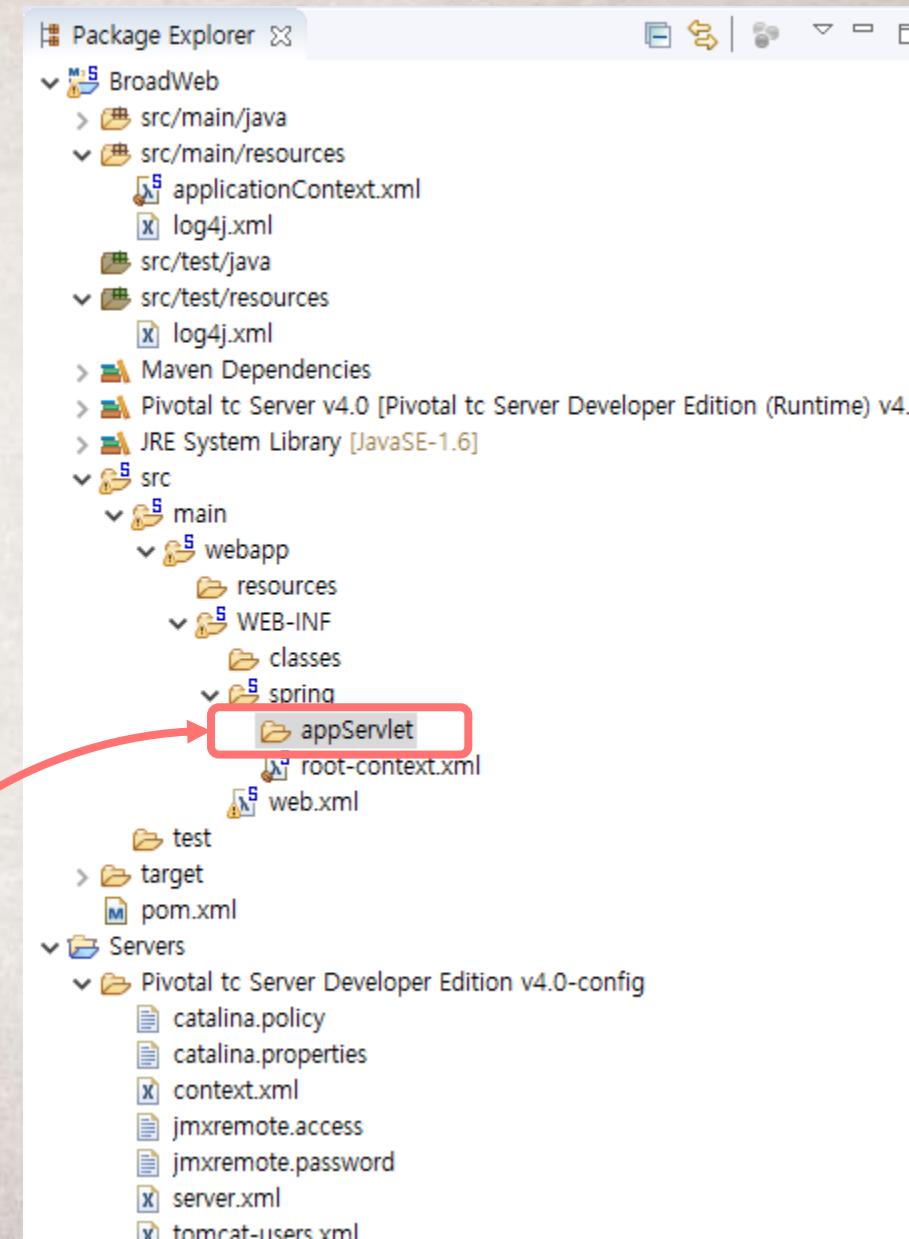
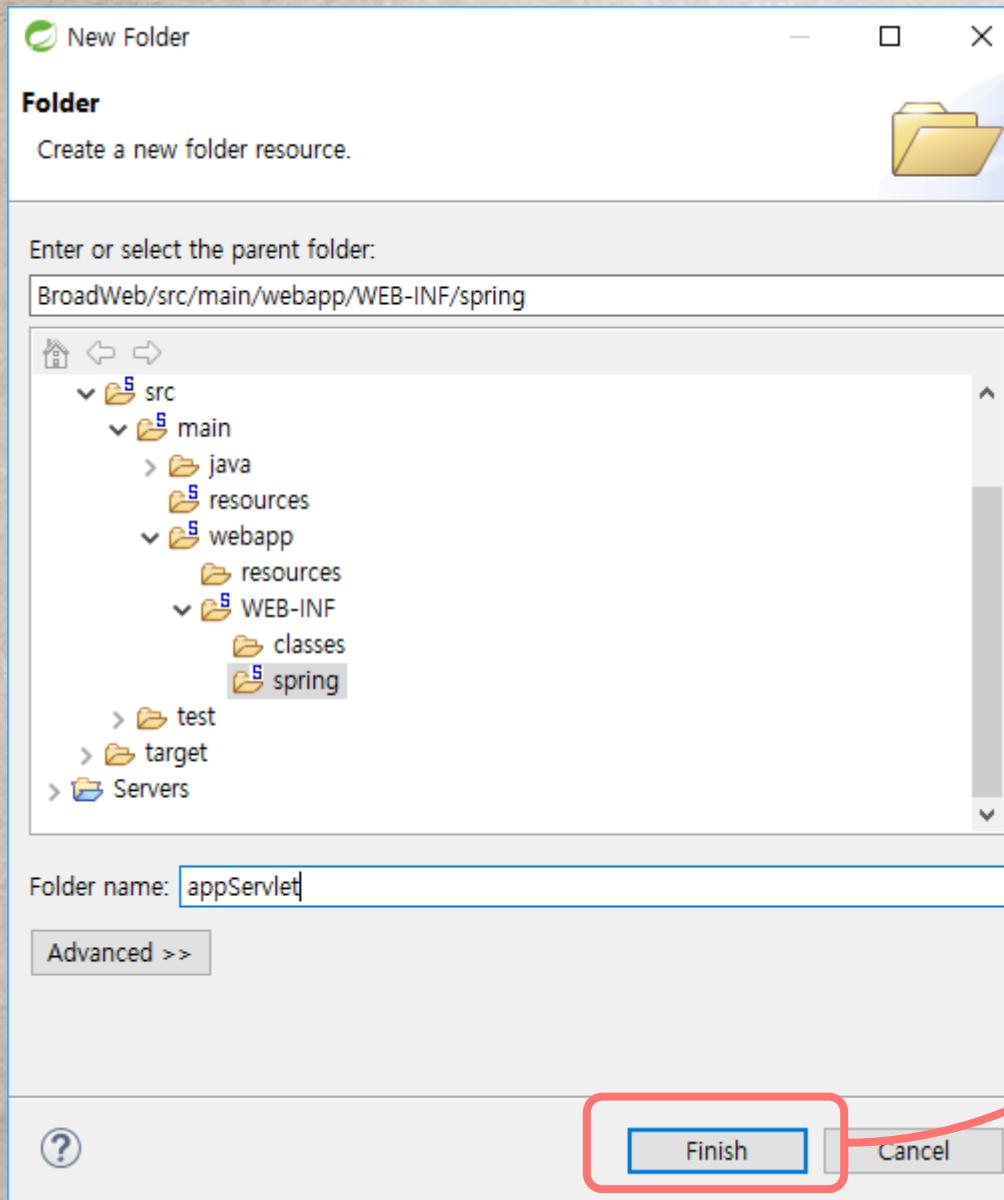
The screenshot shows a Java IDE interface with three tabs at the top: 'web.xml', 'servlet-context.xml', and 'root-context.xml'. The 'root-context.xml' tab is currently selected and active. The code editor displays the XML configuration for the root context of a Spring application.

```
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:context="http://www.springframework.org/schema/context"
5   xmlns:mvc="http://www.springframework.org/schema/mvc"
6   xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
7     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd">
9
10
11 </beans>
12
13
14
```

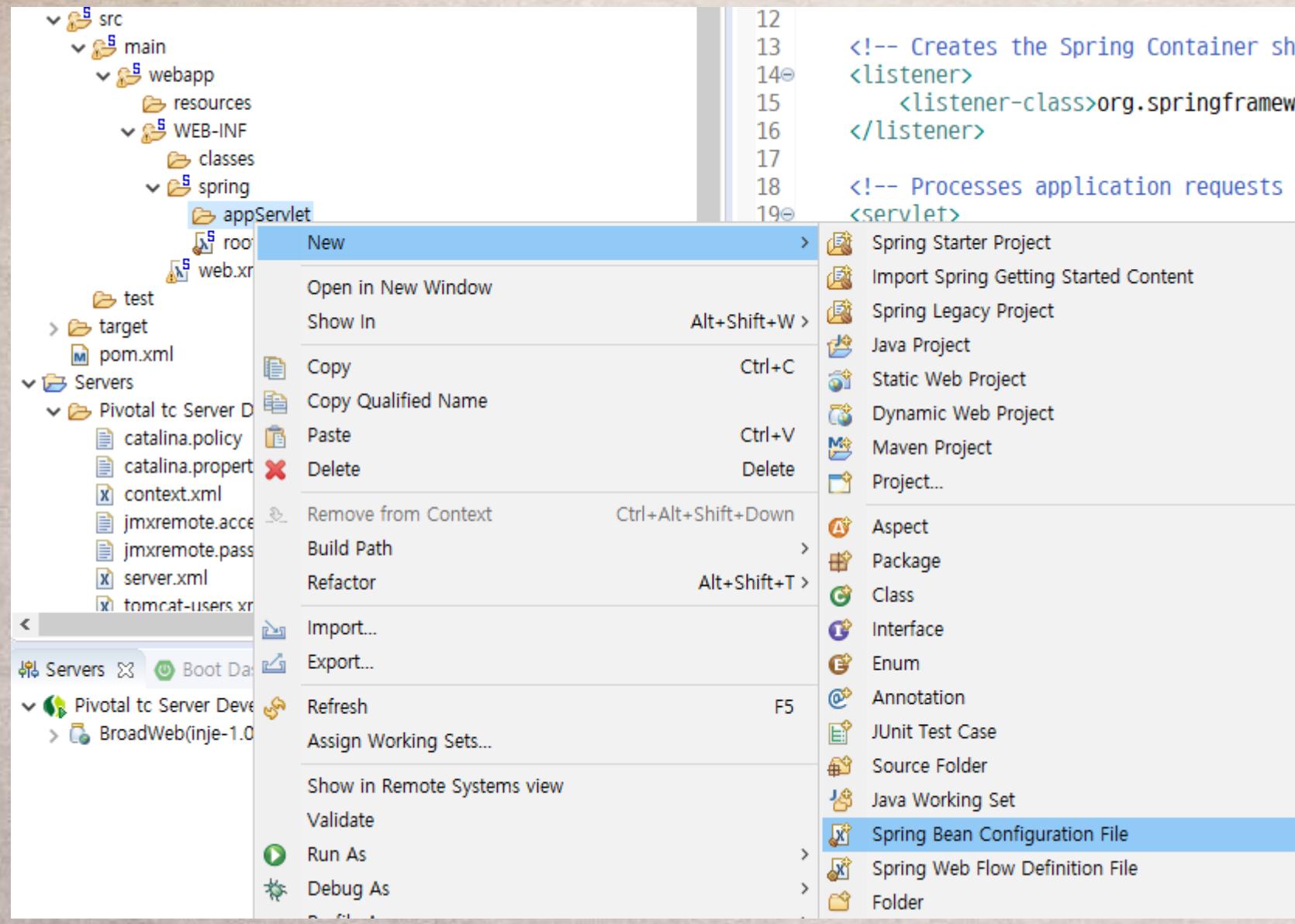
# /WEB-INF/spring/appServlet/servlet-context.xml



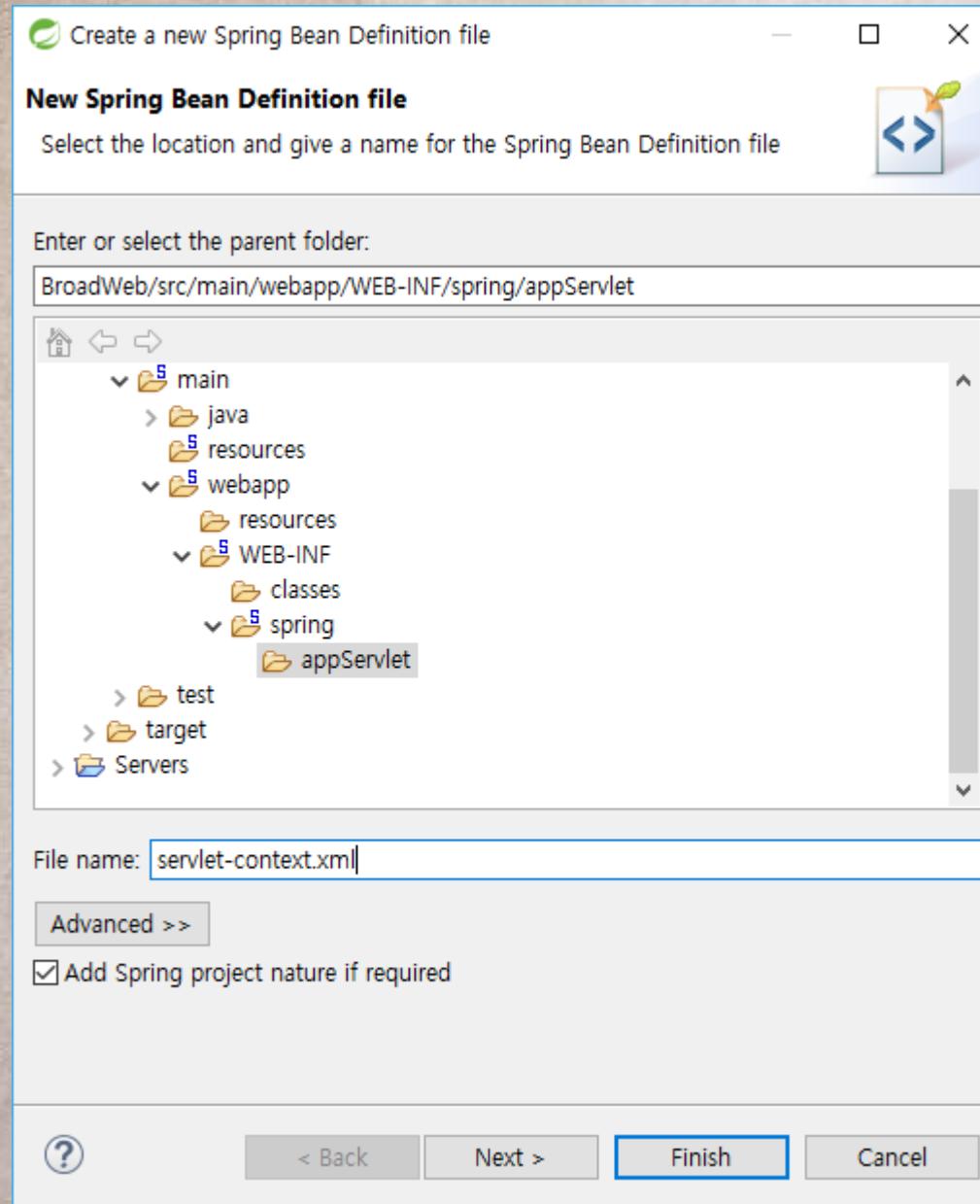
# /WEB-INF/spring/appServlet/servlet-context.xml



# /WEB-INF/spring/appServlet/servlet-context.xml



# /WEB-INF/spring/appServlet/servlet-context.xml



# /WEB-INF/spring/appServlet/servlet-context.xml

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure for "BroadWeb". Key components include:
  - src/main/java**
  - src/main/resources**: Contains `applicationContext.xml` and `log4j.xml`.
  - src/test/resources**: Contains `log4j.xml`.
  - Maven Dependencies**
  - Pivotal tc Server v4.0 [Pivotal tc Server Developer Edition (Runtime) v4]**
  - JRE System Library [JavaSE-1.6]**
  - src**: Contains **main**, which has **webapp** and **WEB-INF**.
    - webapp** contains **resources**.
    - WEB-INF** contains **classes**, **spring**, and **appServlet**.
      - spring** contains `root-context.xml` and `web.xml`.
      - appServlet** contains `servlet-context.xml` and `root-context.xml`.
  - test**
  - target**: Contains **m2e-wtp**, which has **web-resources**, **META-INF**, and **maven**.
- Editors:** The `servlet-context.xml` file is open in the main editor area. The top tab bar shows `web.xml`, `servlet-context.xml` (selected), and `root-context.xml`.
- Configure Namespaces Dialog:** A modal dialog titled "Namespaces" is displayed. It contains a section "Configure Namespaces" with the sub-instruction "Select XSD namespaces to use in the configuration file". A list of Spring namespaces is provided with checkboxes:
  - `aop - http://www.springframework.org/schema/aop`
  - `beans - http://www.springframework.org/schema/beans`
  - `c - http://www.springframework.org/schema/c`
  - `cache - http://www.springframework.org/schema/cache`
  - `context - http://www.springframework.org/schema/context`
  - `jee - http://www.springframework.org/schema/jee`
  - `lang - http://www.springframework.org/schema/lang`
  - `mvc - http://www.springframework.org/schema/mvc`
  - `p - http://www.springframework.org/schema/p`
  - `task - http://www.springframework.org/schema/task`
  - `util - http://www.springframework.org/schema/util`
- Bottom Navigation Bar:** Shows tabs for "Source", "Namespaces" (selected), "Overview", "beans", "context", "mvc", and "Beans Graph".

# /WEB-INF/spring/appServlet/servlet-context.xml

```
web.xml servlet-context.xml 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:context="http://www.springframework.org/schema/context"
5   xmlns:mvc="http://www.springframework.org/schema/mvc"
6   xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
7     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd">
9
10  <context:component-scan base-package="kr.ac.inje.comsi" />
11
12  <!-- 어노테이션이 사용 가능하도록 설정 -->
13  <mvc:annotation-driven/>
14  <!-- 처리할 수 없는 요청은 컨테이너(톰캣)에게 위임하는 설정 -->
15  <!-- 정적인 요소들(html, js, css, gif)을 넘김 -->
16  <mvc:default-servlet-handler />
17
18</beans>
19
20
```

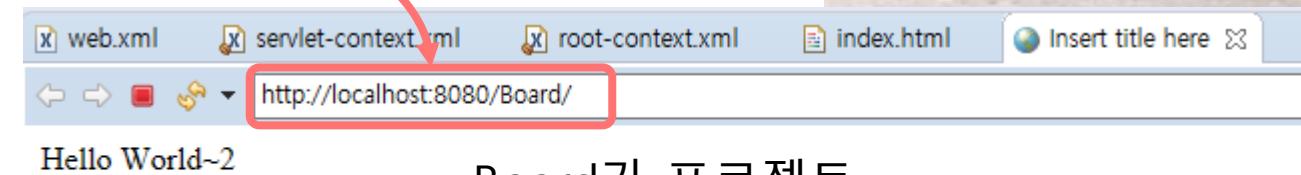
내용 추가

# pom.xml 파일 수정

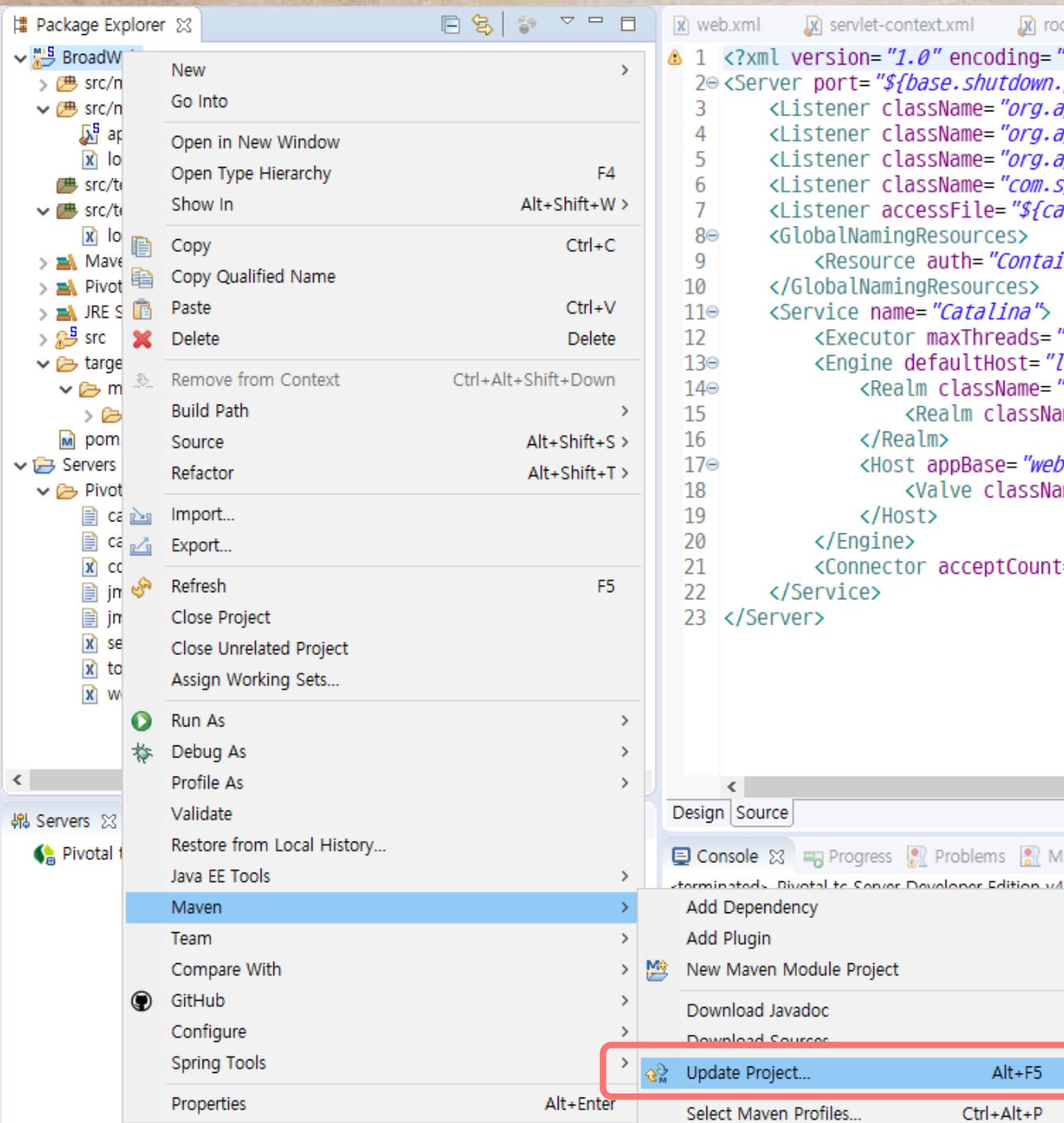
The screenshot shows a Java IDE interface with several tabs at the top: web.xml, servlet-context.xml, root-context.xml, index.html, Insert title here, and BroadWeb/pom.xml. The BroadWeb/pom.xml tab is active, displaying the XML code for the Maven project. A red box highlights the line <artifactId>Board</artifactId>. Below the code, a red arrow points from the highlighted text to a browser window at the bottom of the screen.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>kr.ac</groupId>
6     <artifactId>Board</artifactId> // Line 6 highlighted with a red box
7     <name>Broadweb</name>
8     <packaging>war</packaging>
9     <version>1.0.0-BUILD-SNAPSHOT</version>
10    <properties>
11      <java-version>1.8</java-version>
12      <org.springframework-version>4.3.15.RELEASE</org.springframework-version>
13      <org.aspectj-version>1.6.10</org.aspectj-version>
14      <org.slf4j-version>1.6.6</org.slf4j-version>
15      <com.h2database>1.4.197</com.h2database>
16    </properties>
17    <dependencies>
18      <!-- Spring -->
19      <dependency>
20        <groupId>org.springframework</groupId>
21        <artifactId>spring-context</artifactId>
22        <version>${org.springframework-version}</version>
23        <exclusions>
24          <!-- Exclude Commons Logging in favor of SLF4j -->
25          <exclusion>
26            <groupId>commons-logging</groupId>
27            <artifactId>commons-logging</artifactId>
28          </exclusion>
29        </exclusions>
```

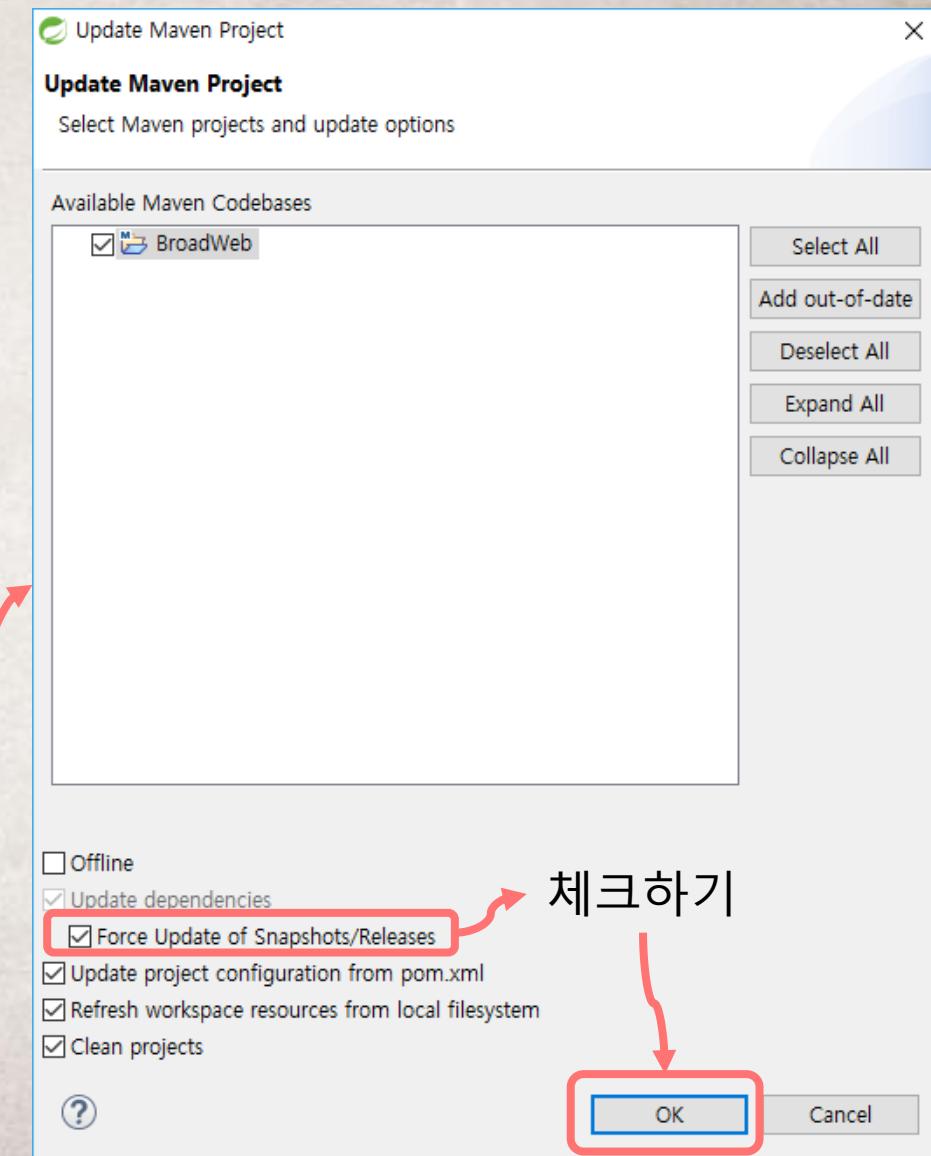
<artifactId> 태그 내의 값을  
"Board"로 변경



Board가 프로젝트  
폴더로 추가됨

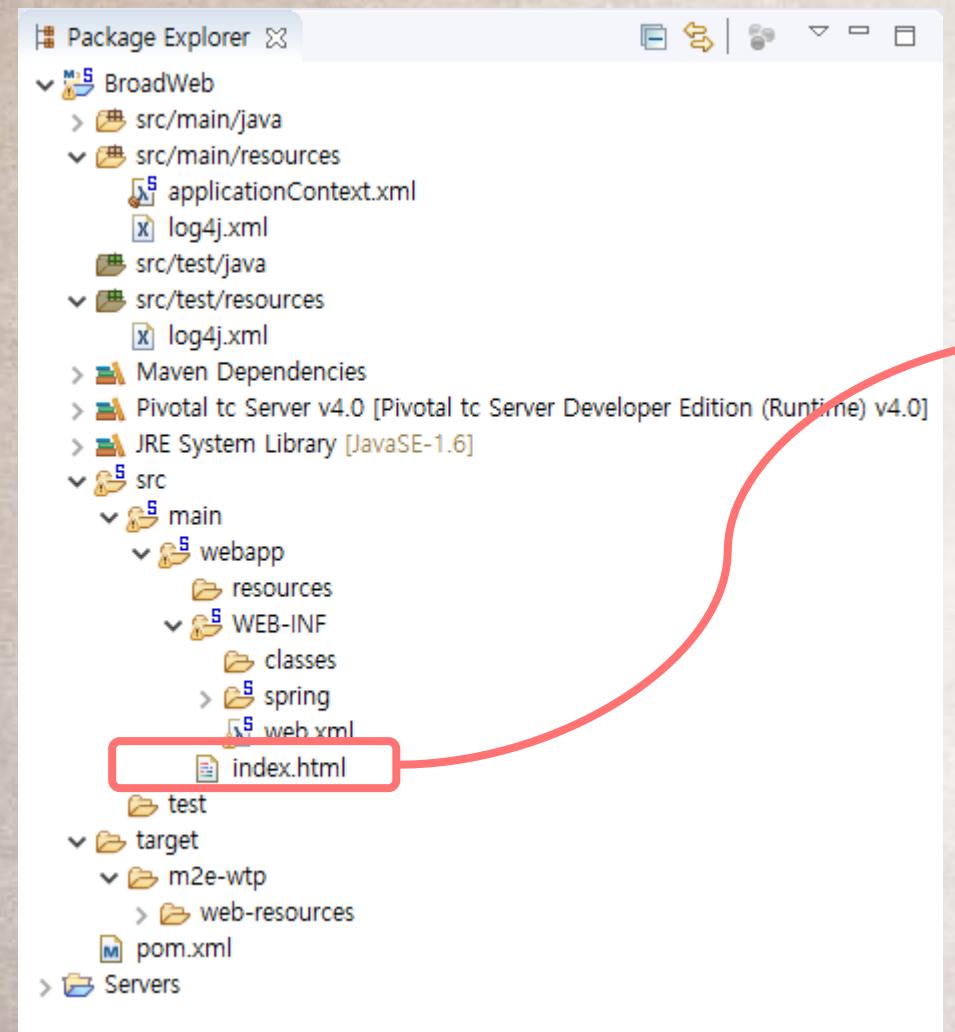


←수정된 프로젝트 내용 갱신하기



# html[JSP] 페이지 추가와 서버 구동

# webapp 폴더에 index.html 파일 추가



```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
Hello World~
</body>
</html>
```

# 웹 서버 설정하기

The screenshot shows the Pivotal tc Server Developer Edition interface. On the left, a server named "Pivotal tc Server Developer Edition v4.0 [Stopped]" is selected in the "Servers" list. A context menu is open over this server, with the "Add and Remove..." option highlighted in blue. A red arrow points from this highlighted option to the "Add and Remove..." button in the "Add and Remove..." dialog window on the right.

**Add and Remove...**

**Add and Remove**  
Modify the resources that are configured on the server

**Available:**

- BroadWeb(inje-1.0.0-BUILD)

**Configured:**

- (empty)

**Buttons:**

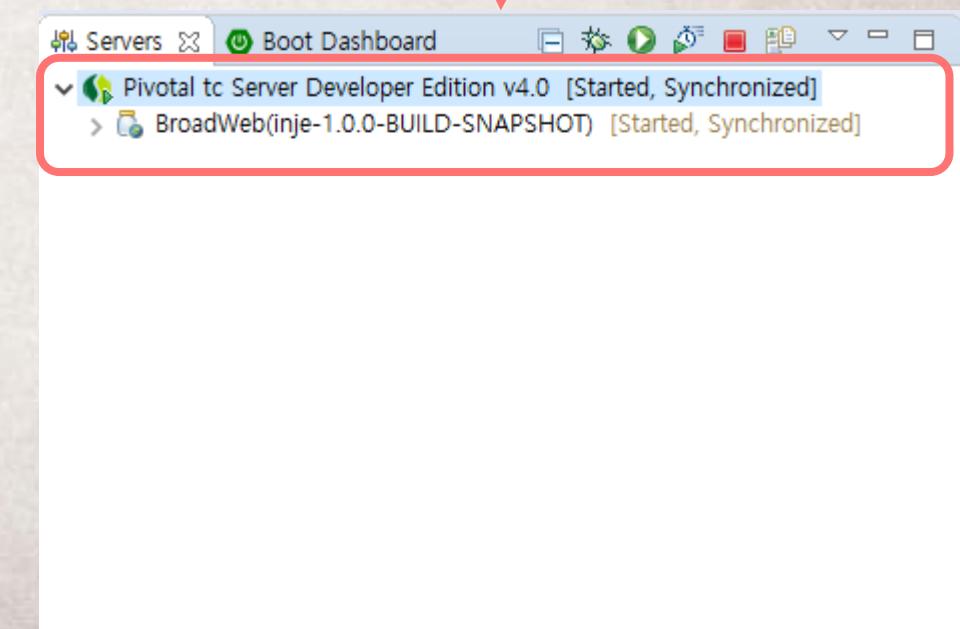
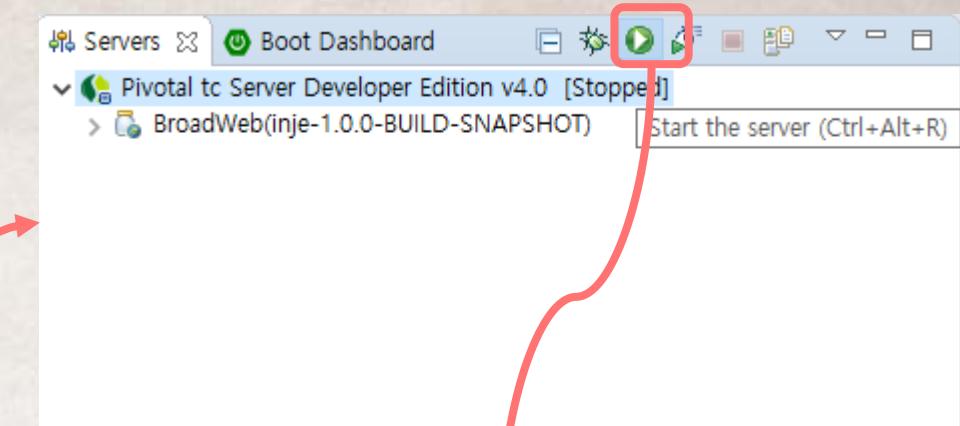
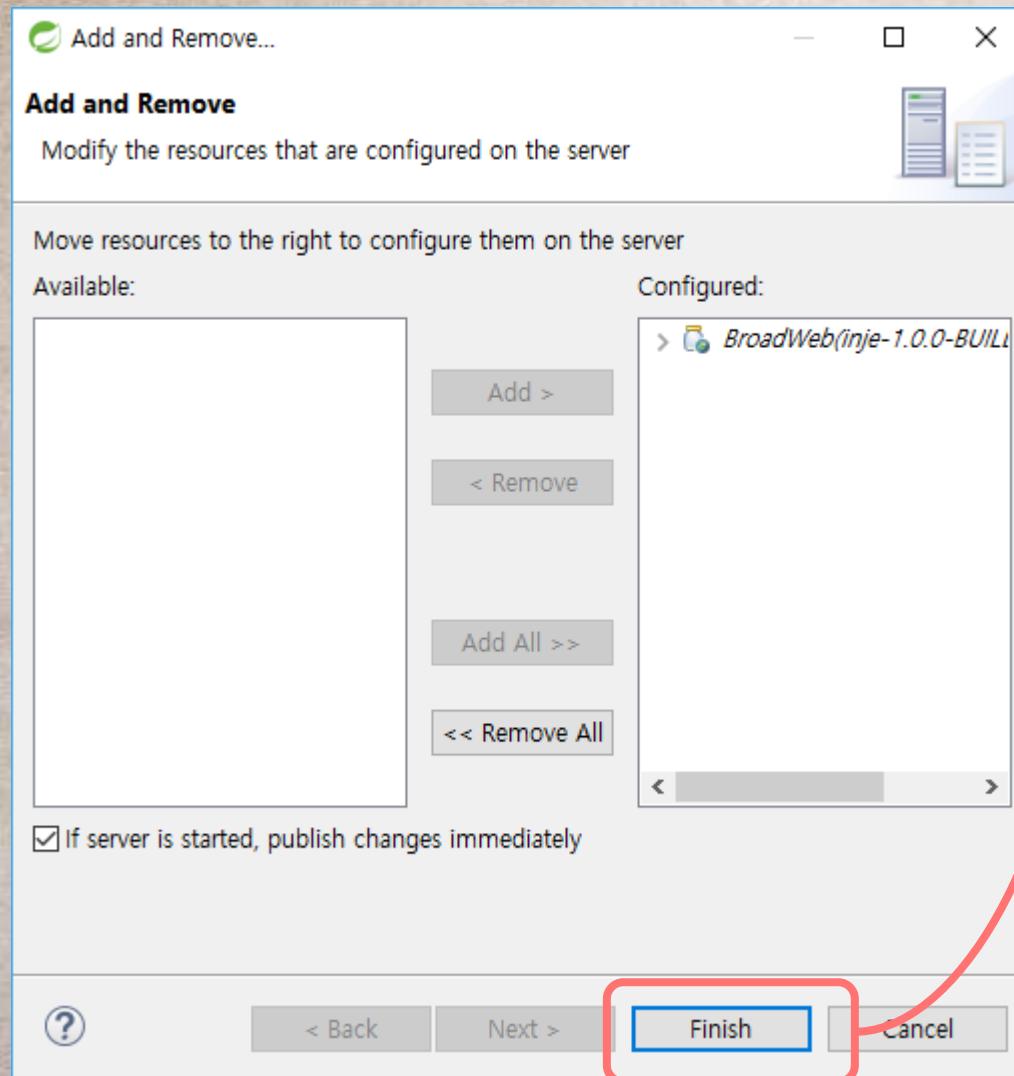
- Add > (highlighted with a red box)
- < Remove
- Add All >>
- << Remove All

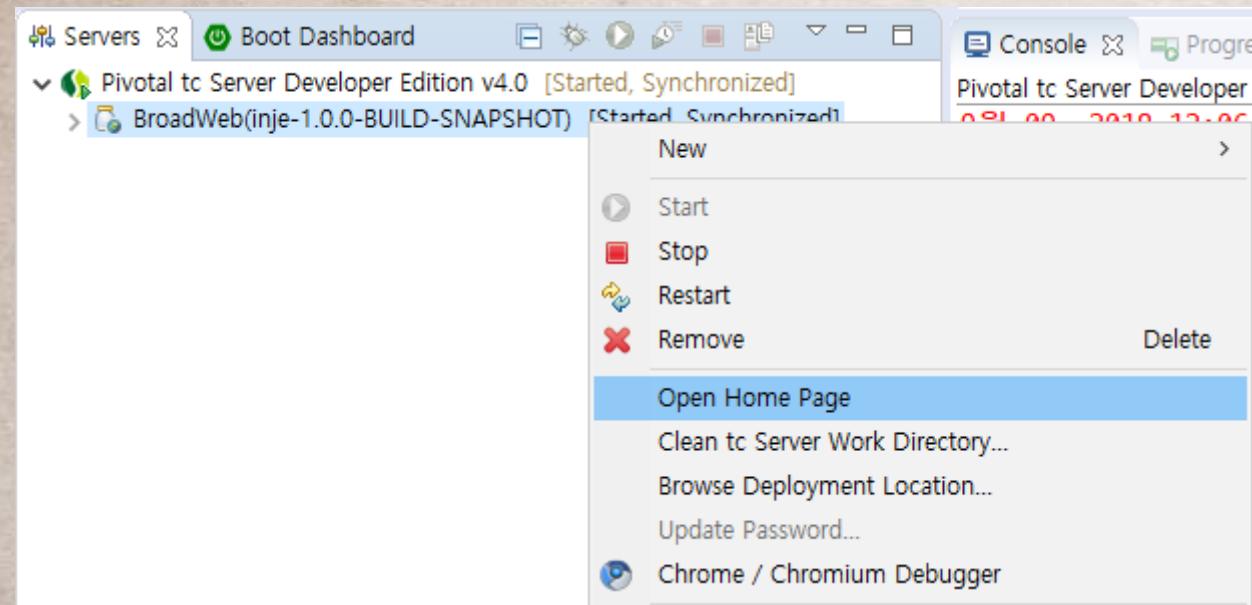
If server is started, publish changes immediately

**Dialog Buttons:**

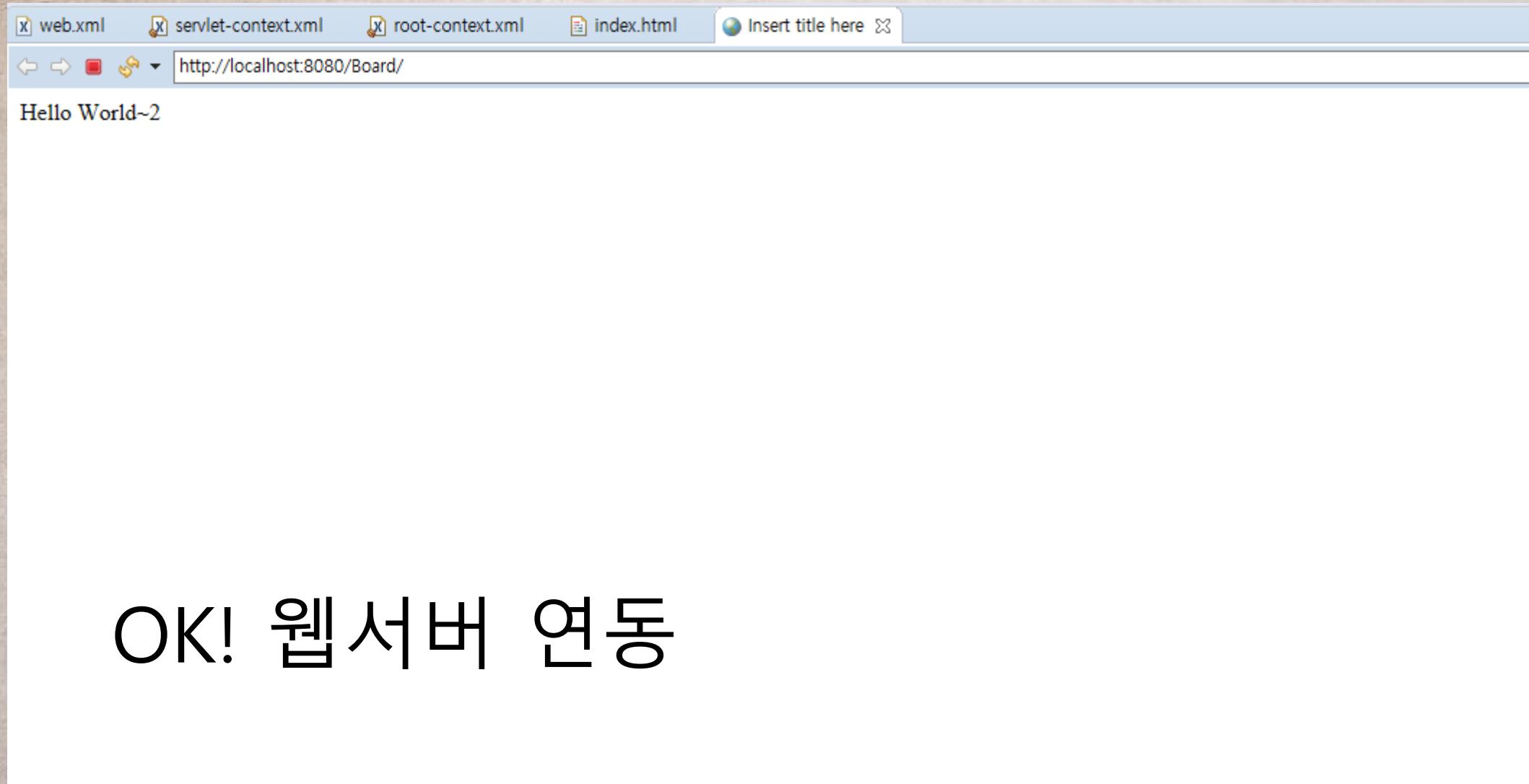
- ? (Help)
- < Back
- Next >
- Finish (highlighted with a blue border)
- Cancel

# 웹 서버 설정하기





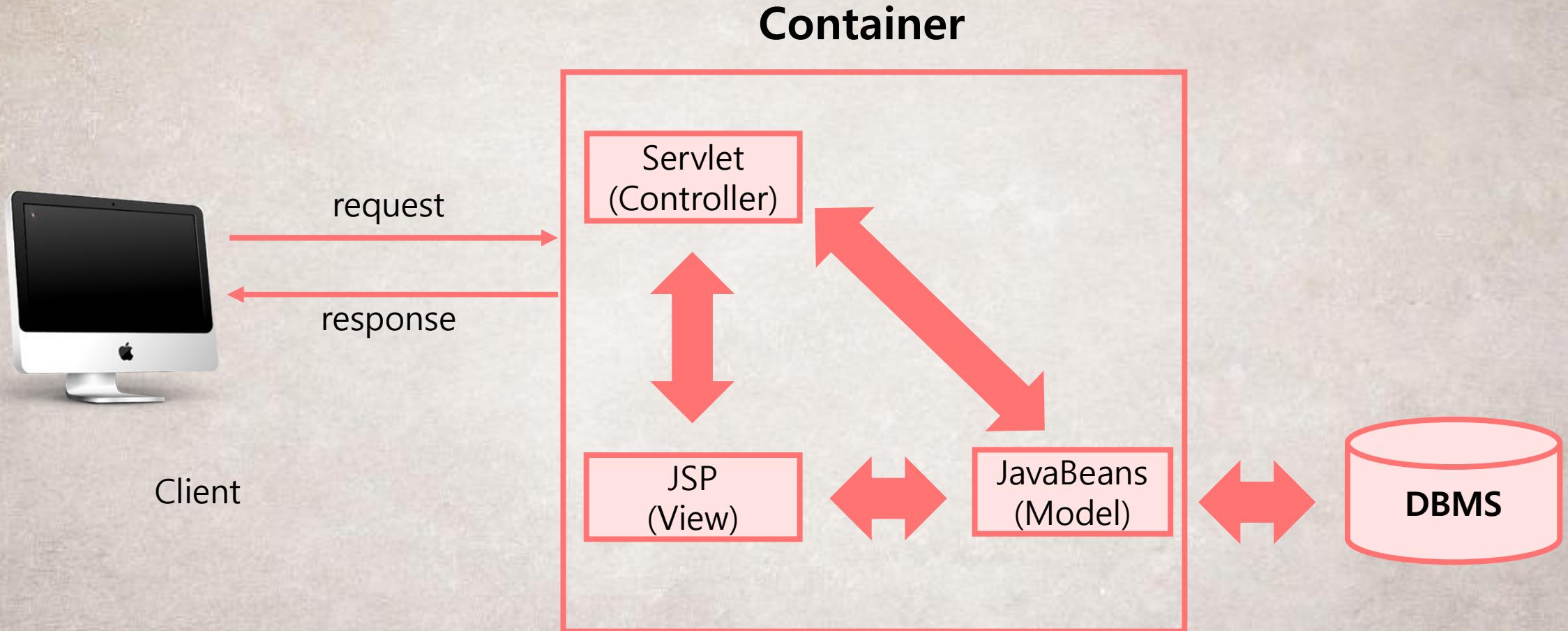
# 웹서버에서 전송된 기본 html 파일-index.html



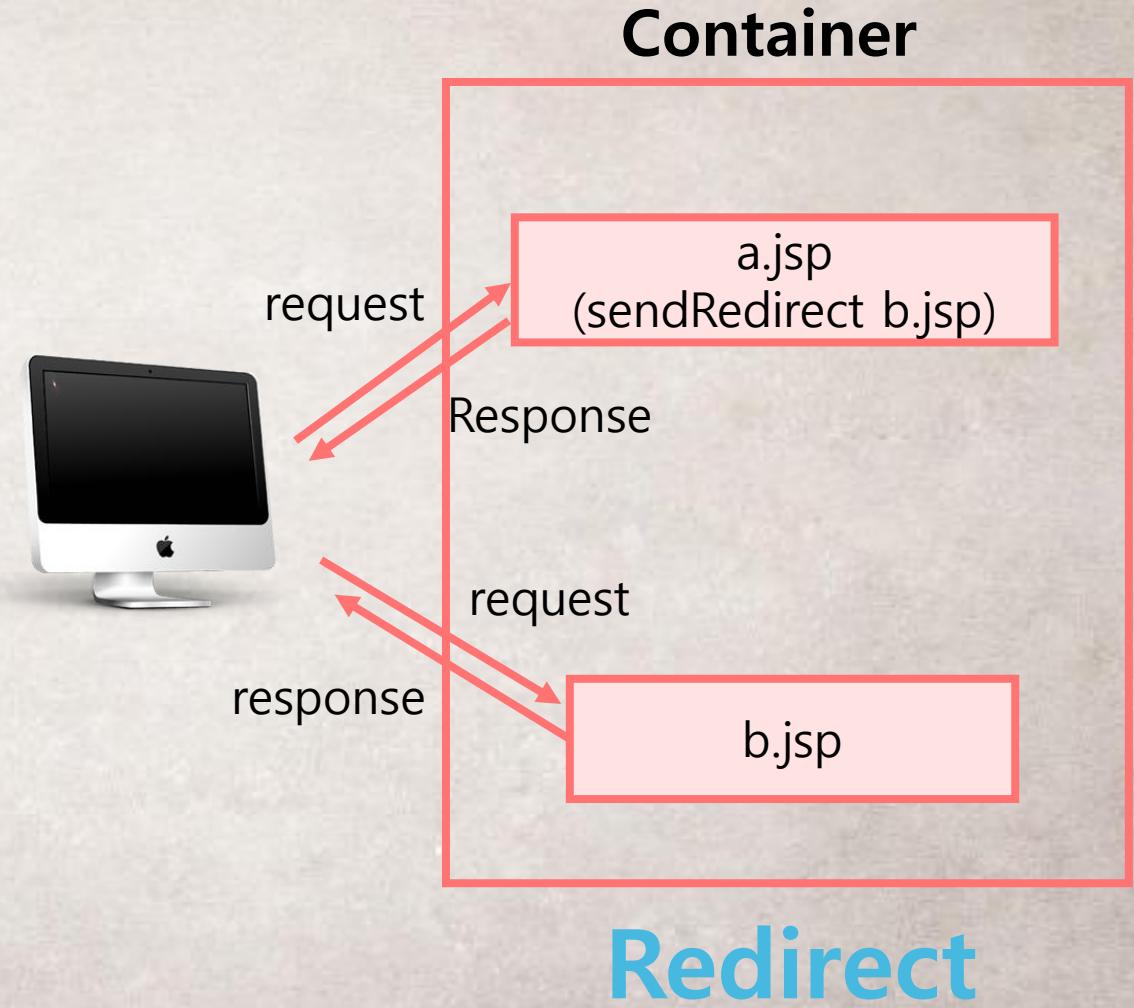
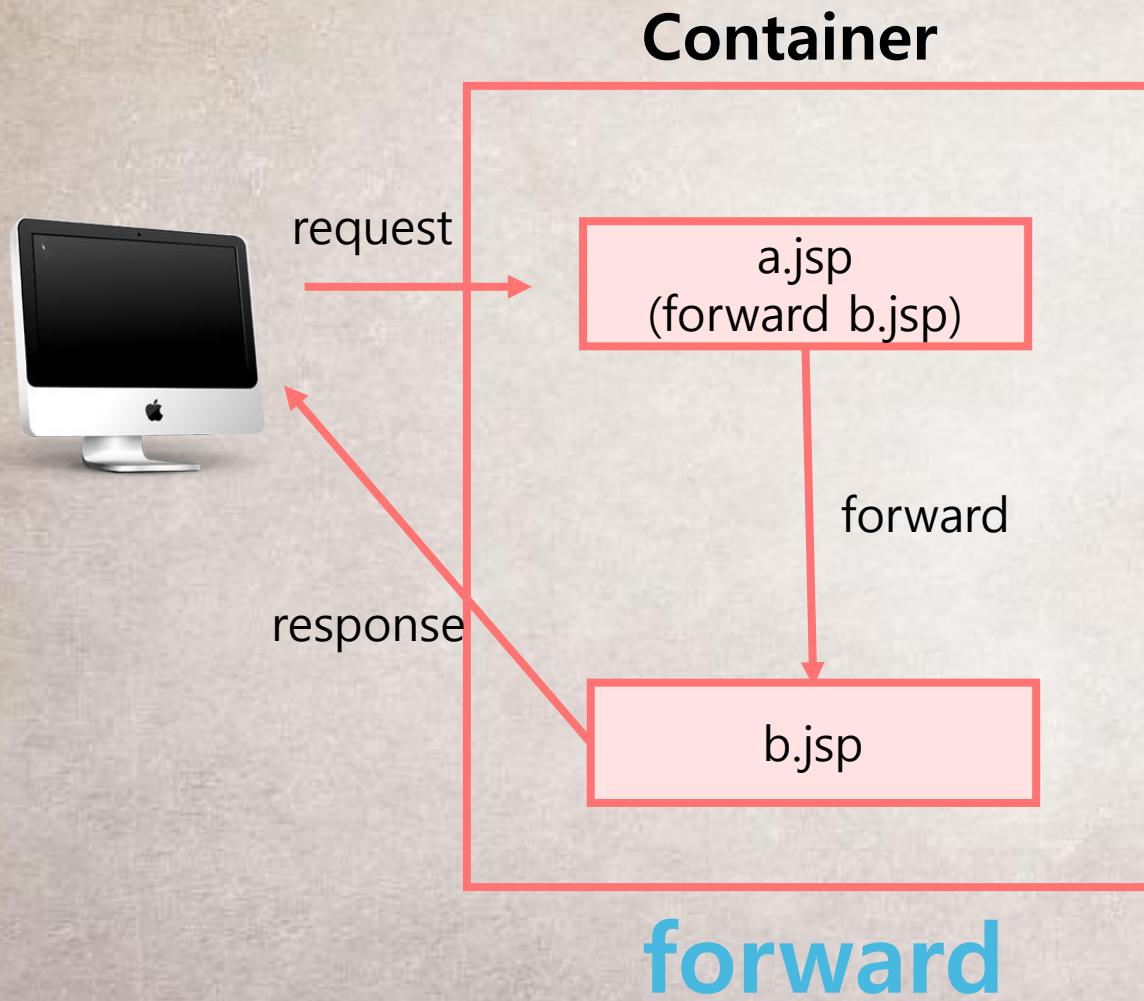
OK! 웹서버 연동

# 게시판 모델2 작업

# Model2 아키텍처



# 참고: 포워드(Forward)와 리다이렉트(Redirect) 차이



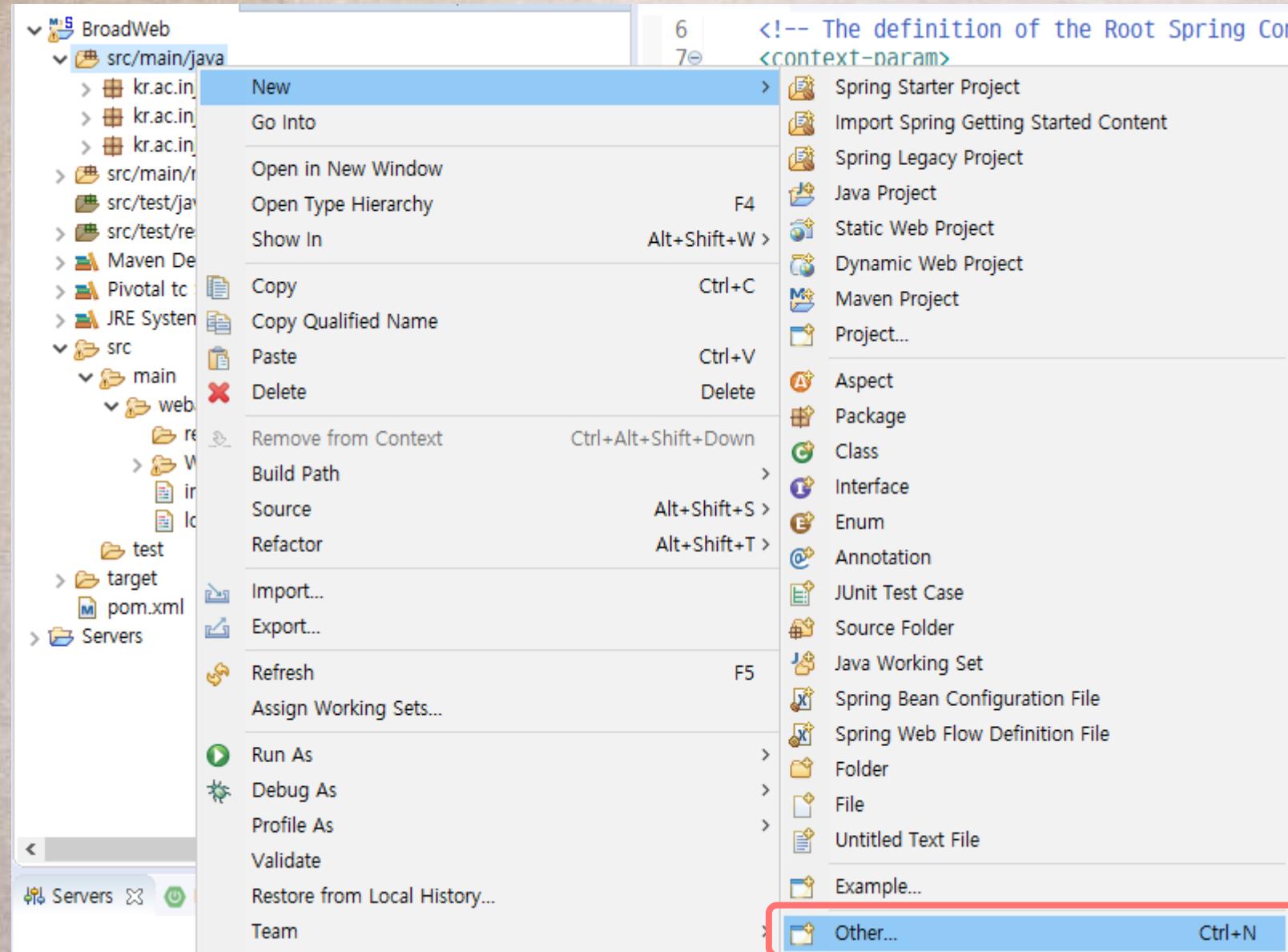
# web.xml 수정

The screenshot shows an IDE interface with two tabs: 'login.jsp' and 'web.xml'. The 'web.xml' tab is active and displays the XML configuration for a Spring-based web application. A red oval highlights the section from line 10 to line 25, which defines the DispatcherServlet and its mappings.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:
3   <context-param>
4     <param-name>contextConfigLocation</param-name>
5     <param-value>/WEB-INF/spring/root-context.xml</param-value>
6   </context-param>
7   <listener>
8     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
9   </listener>
10  <!--
11  <servlet>
12    <servlet-name>appServlet</servlet-name>
13    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
14    <init-param>
15      <param-name>contextConfigLocation</param-name>
16      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
17    </init-param>
18    <load-on-startup>1</load-on-startup>
19  </servlet>
20  <servlet-mapping>
21    <servlet-name>appServlet</servlet-name>
22    <url-pattern>/</url-pattern>
23  </servlet-mapping>
24  -->
25 </web-app>
```

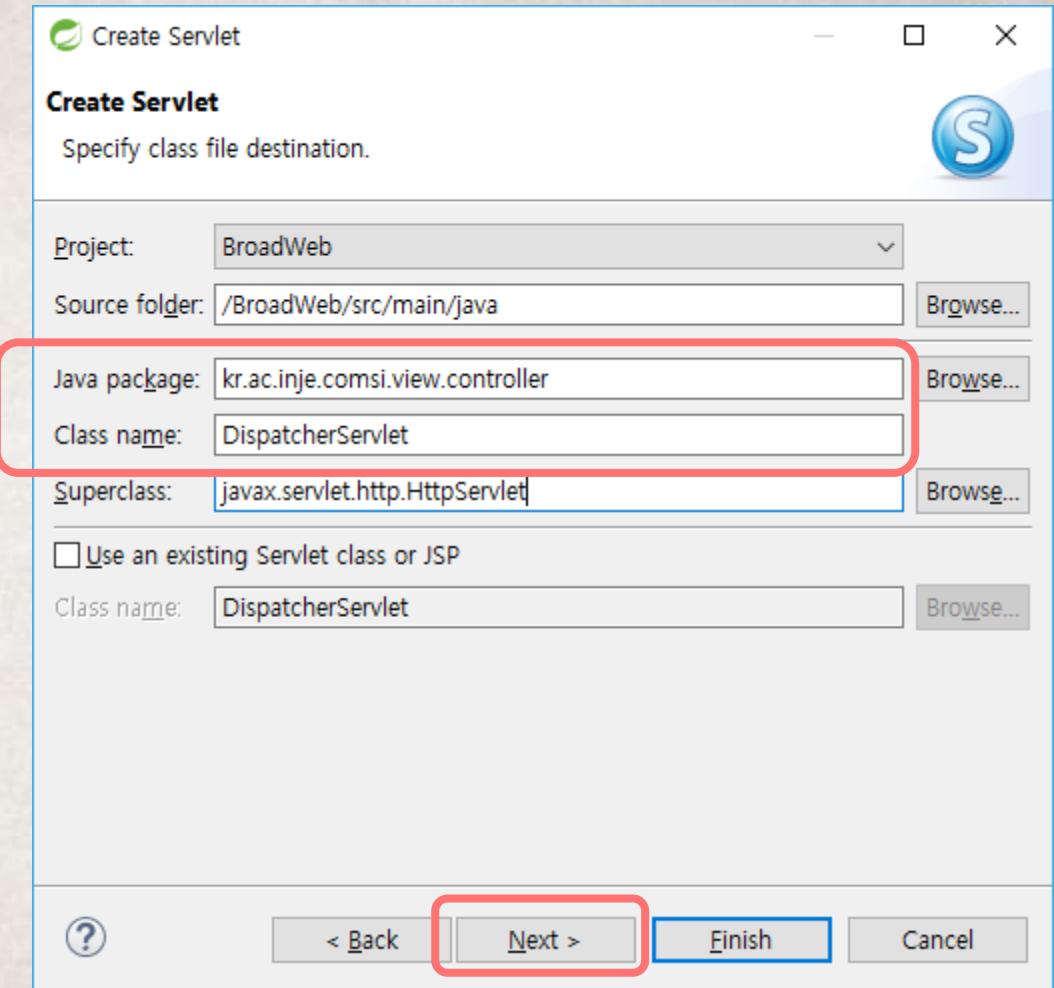
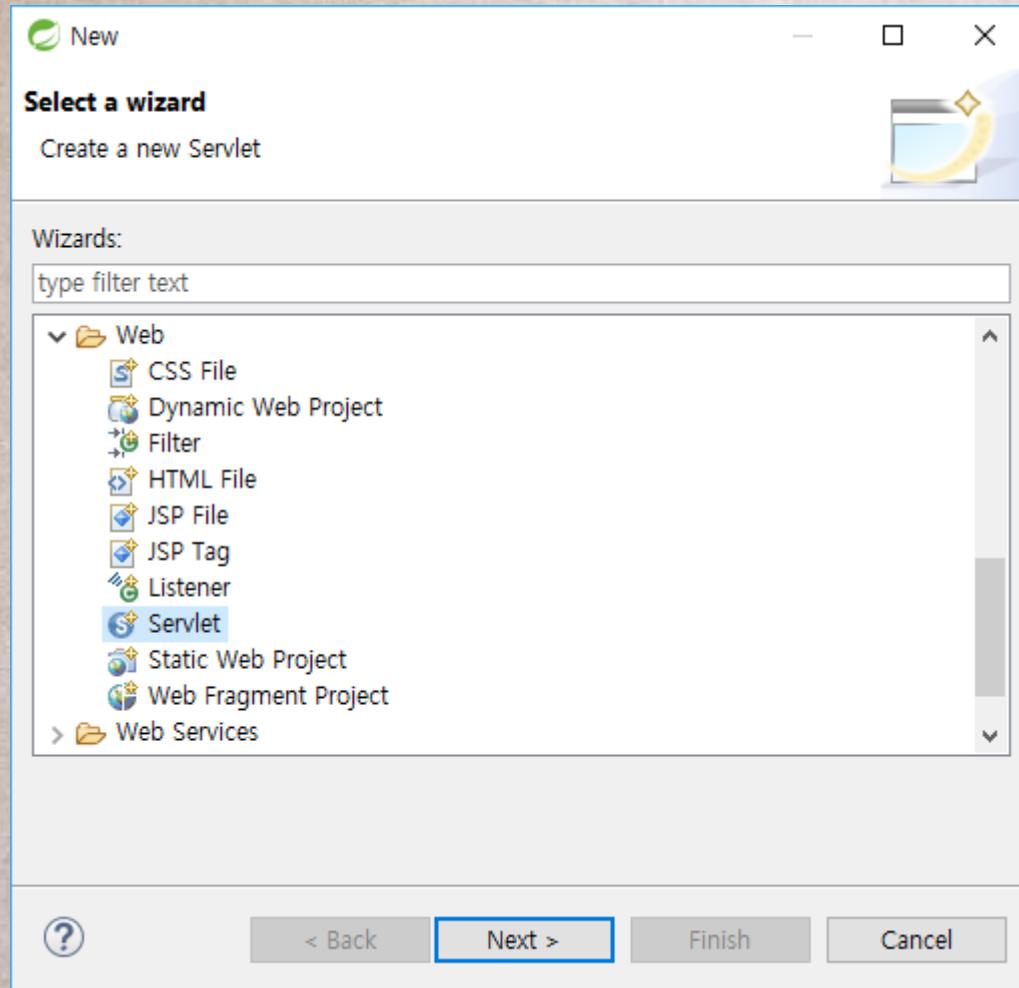
주석처리

# Controller 구현 – DispatcherServlet



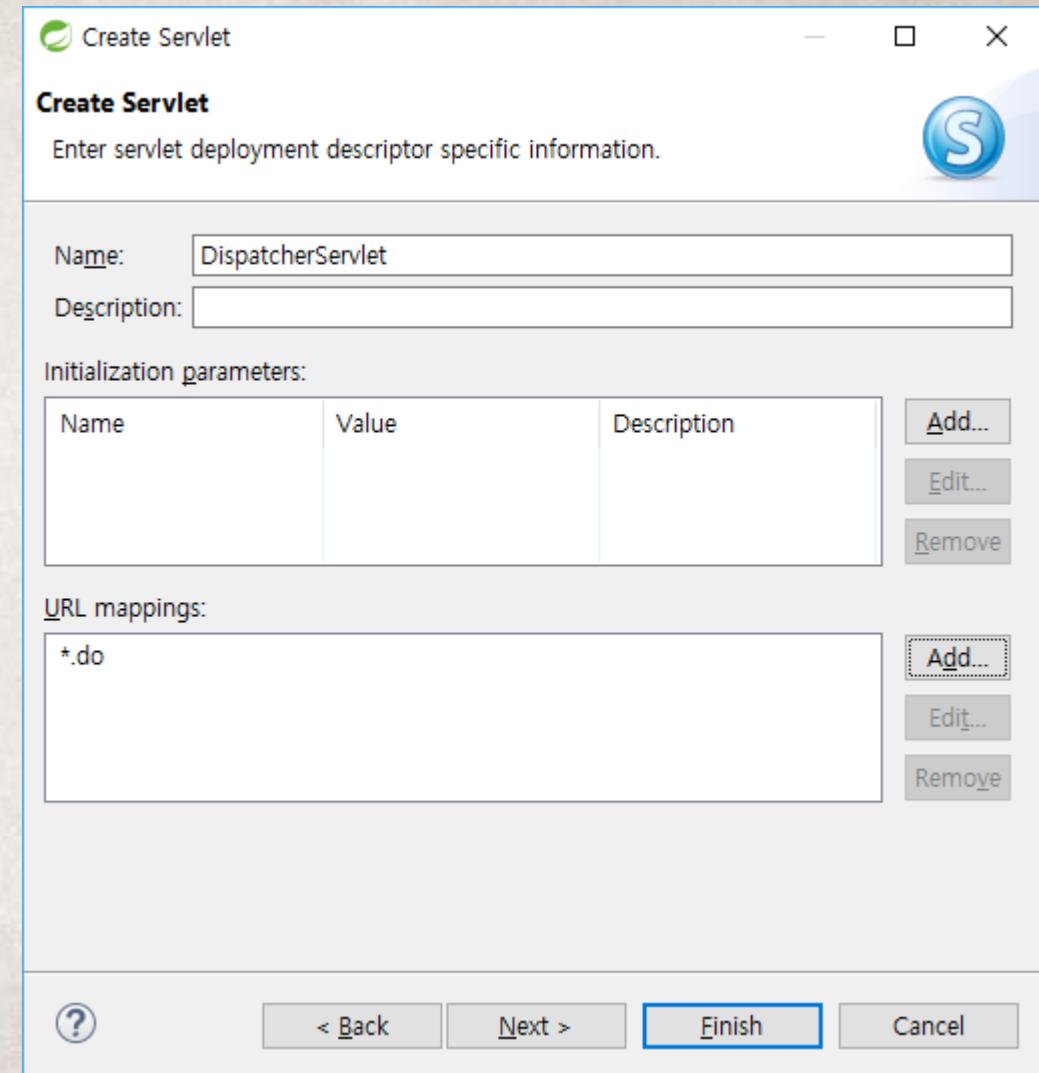
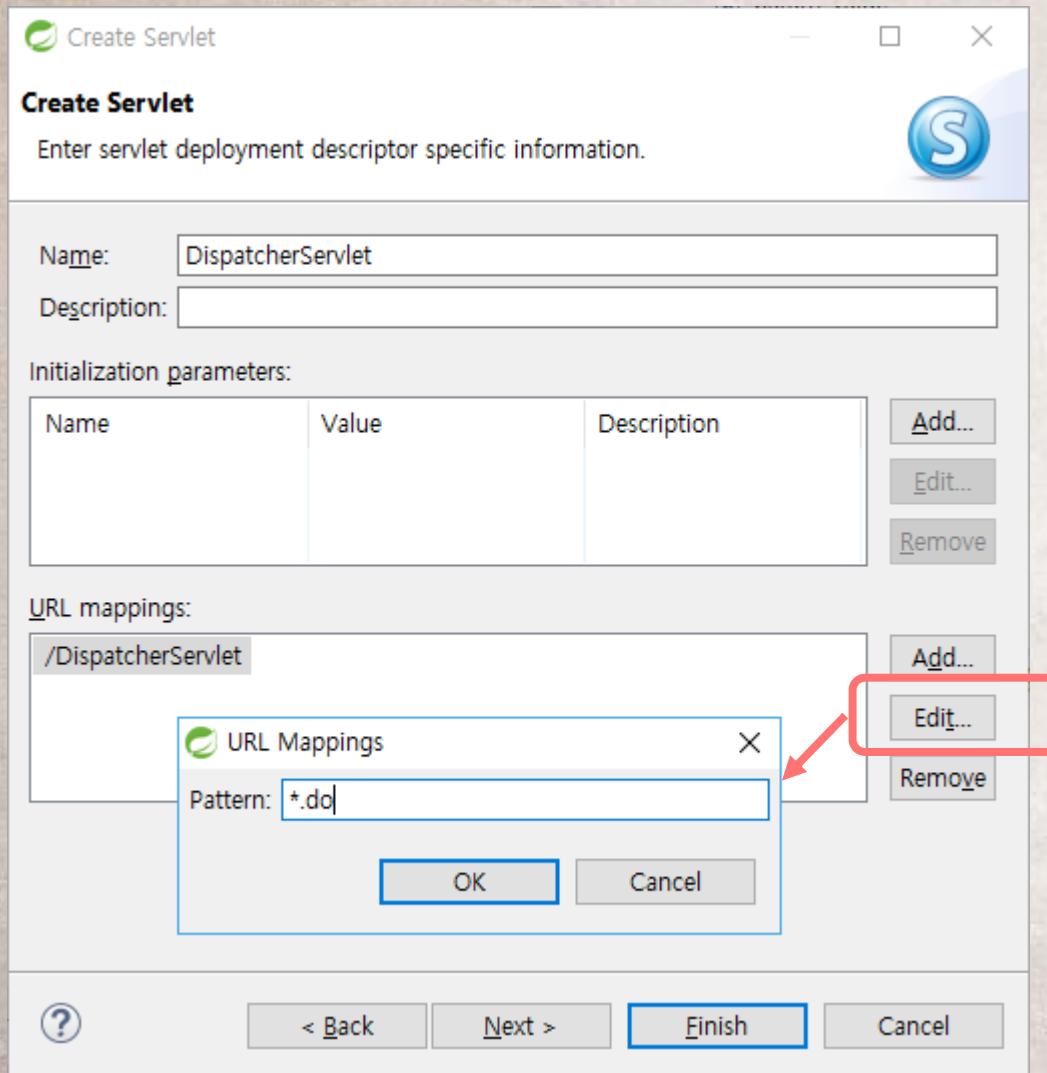
프로젝트 탐색창에서  
src/main/java 폴더에서

# Controller 구현 – DispatcherServlet 클래스



Java package – kr.ac.inje.comsi.view.controller  
Class name - DispatcherServlet

# DispatcherServlet 클래스 – URL mapping 처리



# 클라이언트의 \*.do 요청이 있을 때,

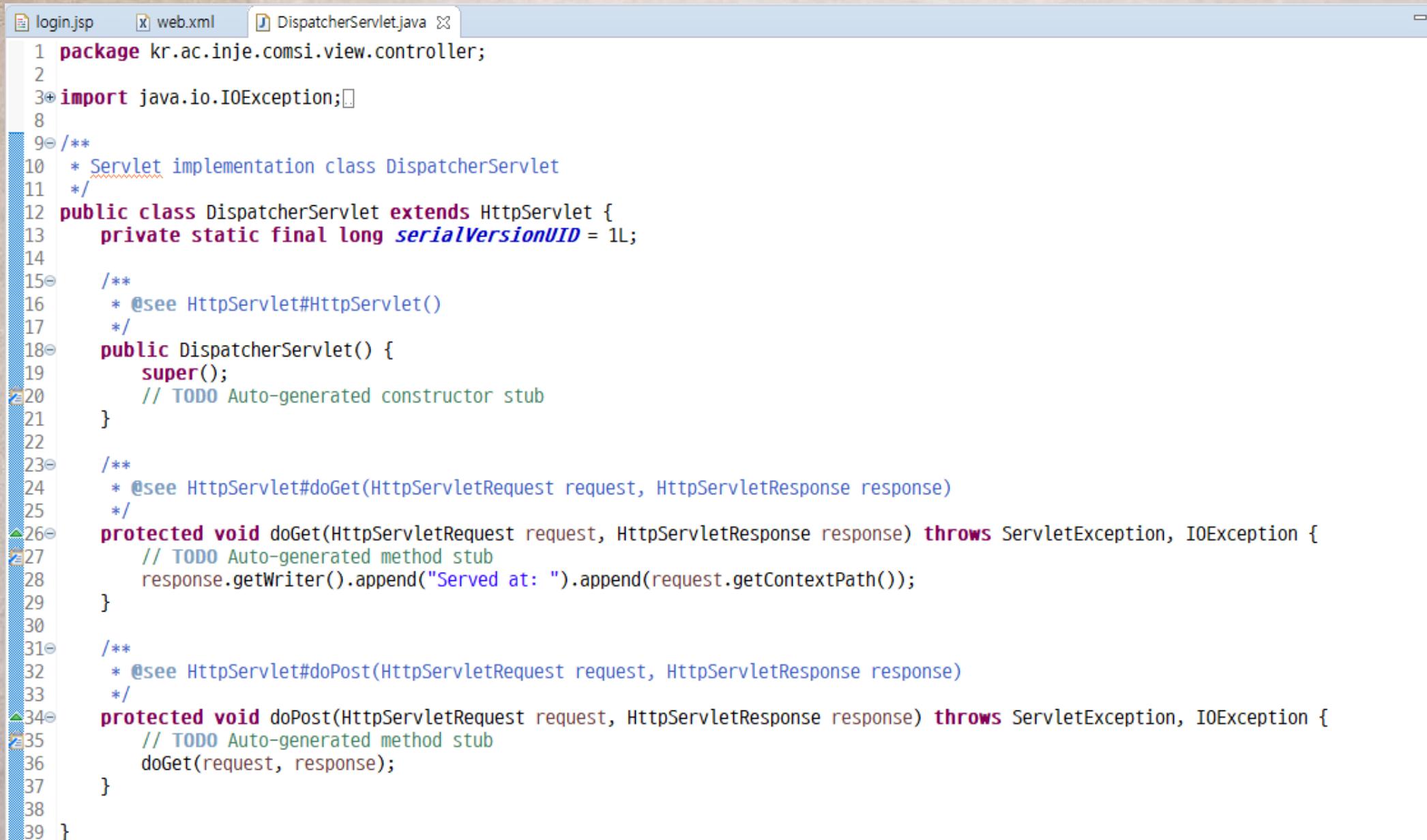


```
login.jsp  web.xml  DispatcherServlet.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
3   <context-param>
4     <param-name>contextConfigLocation</param-name>
5     <param-value>/WEB-INF/spring/root-context.xml</param-value>
6   </context-param>
7   <listener>
8     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
9   </listener>
10  <servlet>
11    <description></description>
12    <display-name>DispatcherServlet</display-name>
13    <servlet-name>DispatcherServlet</servlet-name>
14    <servlet-class>kr.ac.inje.comsi.view.controller.DispatcherServlet</servlet-class>
15  </servlet>
16  <servlet-mapping>
17    <servlet-name>DispatcherServlet</servlet-name>
18    <url-pattern>*.do</url-pattern>
19  </servlet-mapping>
20 </web-app>
```

추가됨

DispatcherServlet 객체 생성 및 활성화

# 생성된 DispatcherServlet 클래스



The screenshot shows a Java code editor with the tab bar at the top containing "login.jsp", "web.xml", and "DispatcherServlet.java". The code itself is a Java servlet implementation:

```
1 package kr.ac.inje.comsi.view.controller;
2
3+ import java.io.IOException;
4
5 /**
6  * Servlet implementation class DispatcherServlet
7  */
8
9 public class DispatcherServlet extends HttpServlet {
10    private static final long serialVersionUID = 1L;
11
12    /**
13     * @see HttpServlet#HttpServlet()
14     */
15    public DispatcherServlet() {
16        super();
17        // TODO Auto-generated constructor stub
18    }
19
20    /**
21     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
22     */
23    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
24        // TODO Auto-generated method stub
25        response.getWriter().append("Served at: ").append(request.getContextPath());
26    }
27
28    /**
29     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
30     */
31    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
32        // TODO Auto-generated method stub
33        doGet(request, response);
34    }
35
36
37
38
39 }
```

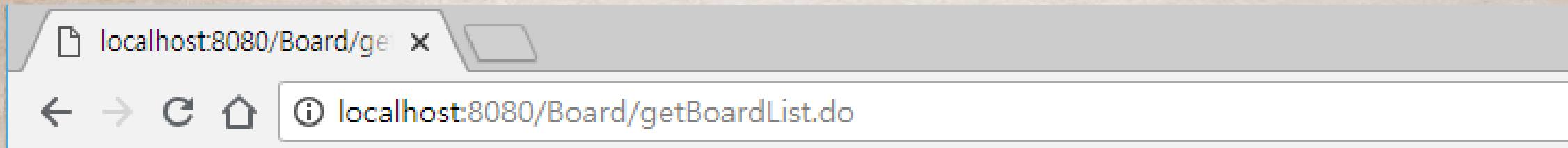
# process() 메소드 추가

```
42@ private void process(HttpServletRequest request, HttpServletResponse response) throws IOException {
43    // 1. 클라이언트의 요청 path 정보를 추출한다.
44    String uri = request.getRequestURI();
45    String path = uri.substring(uri.lastIndexOf("/")));
46    System.out.println(path);
47
48    // 2. 클라이언트의 요청 path에 따라 적절히 분기처리 한다.
49    if (path.equals("/login.do")) {
50        System.out.println("로그인 처리");
51    } else if (path.equals("/logout.do")) {
52        System.out.println("로그아웃 처리");
53    } else if (path.equals("/insertBoard.do")) {
54        System.out.println("글 등록 처리");
55    } else if (path.equals("/updateBoard.do")) {
56        System.out.println("글 수정 처리");
57    } else if (path.equals("/deleteBoard.do")) {
58        System.out.println("글 삭제 처리");
59    } else if (path.equals("/getBoard.do")) {
60        System.out.println("글 상세 조회 처리");
61    } else if (path.equals("/getBoardList.do")) {
62        System.out.println("글 목록 검색 처리");
63    }
64}
```

# doGet(), doPost()메소드 수정

```
▲27✉  protected void doGet(HttpServletRequest request, HttpServletResponse response)
28      throws ServletException, IOException {
29          process(request, response);
30      }
31
32✉  /**
33   * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
34   *      response)
35   */
▲36✉  protected void doPost(HttpServletRequest request, HttpServletResponse response)
37      throws ServletException, IOException {
38          request.setCharacterEncoding("UTF-8");
39          process(request, response);
40      }
41
```

# DispatcherServlet의 작동확인



- http://localhost:8080/Board/login.do
- http://localhost:8080/Board/logout.do
- http://localhost:8080/Board/insertBoard.do
- http://localhost:8080/Board/updateBoard.do
- http://localhost:8080/Board/deleteBoard.do
- http://localhost:8080/Board/getBoard.do
- http://localhost:8080/Board/getBoardList.do

# DispatcherServlet의 작동 결과

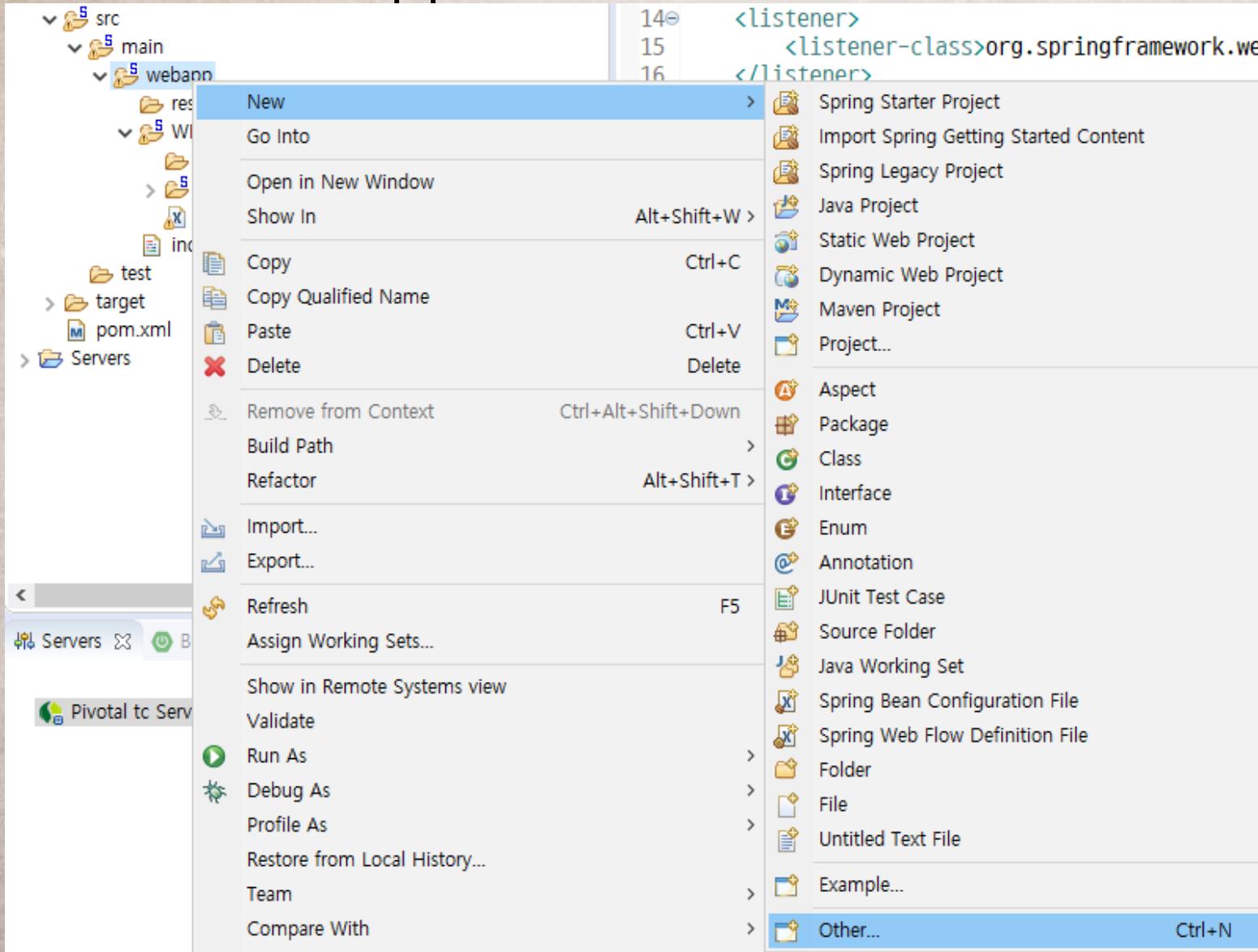
```
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Closing Root WebApplicationConte  
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization started  
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Refreshing Root WebApplicationCo  
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from S  
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization complete  
/login.do  
로그인 처리  
/logout.do  
로그아웃 처리  
/insertBoard.do  
글 등록 처리  
/updateBoard.do  
글 수정 처리  
/deleteBoard.do  
글 삭제 처리  
/getBoard.do  
글 상세 조회 처리  
/getBoardList.do  
글 목록 검색 처리
```

클라이언트의 요청에 따른 처리 호출

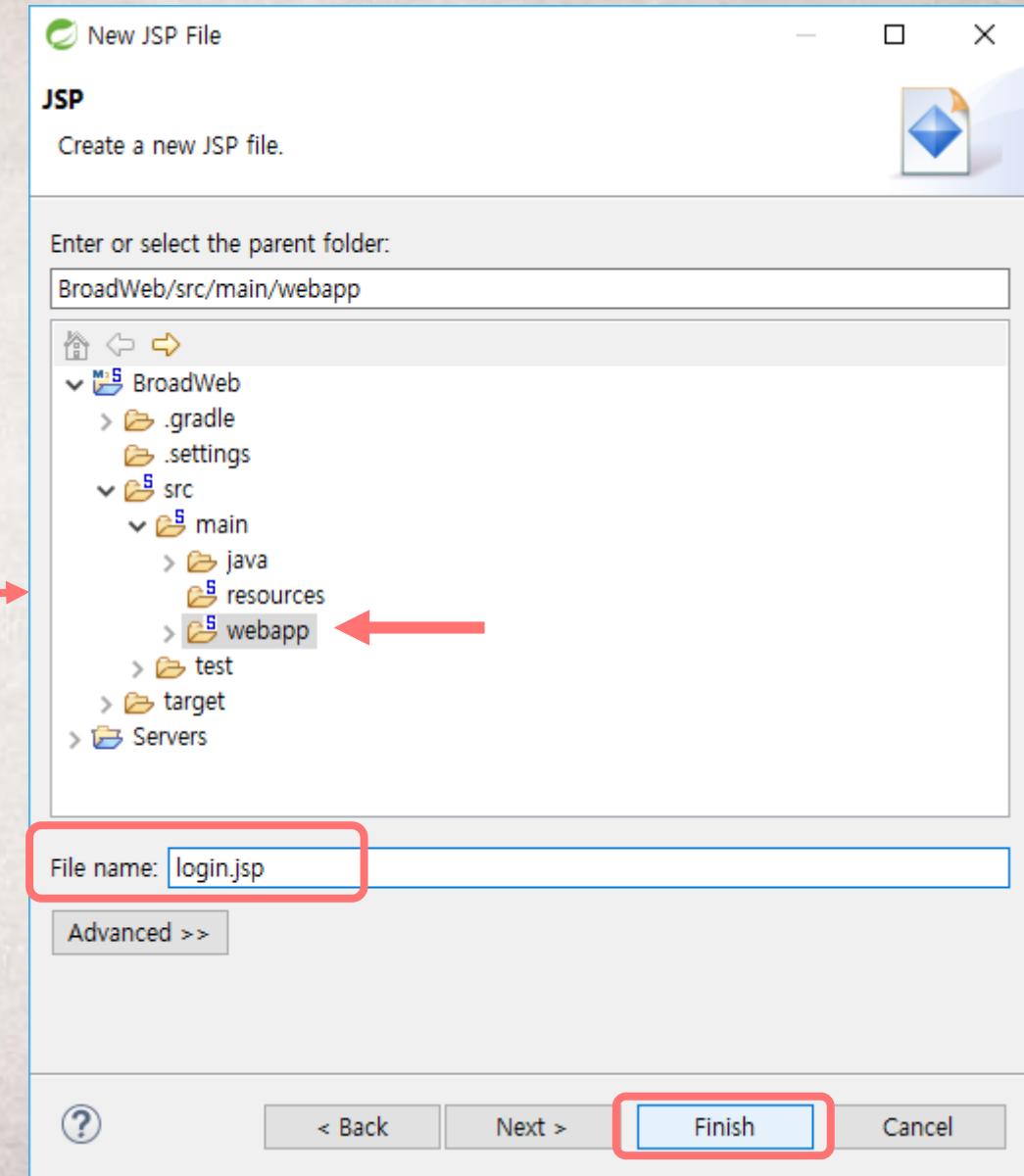
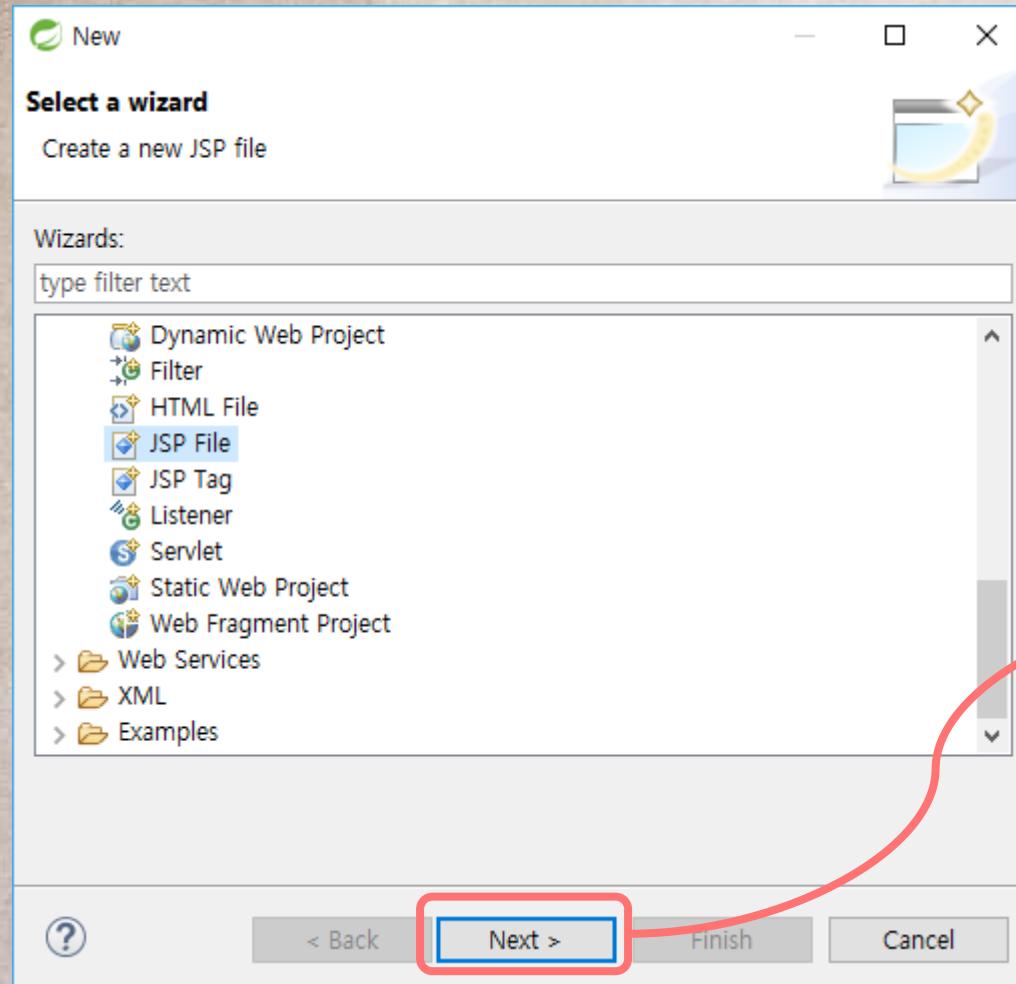
로그인 가능 구현

# login.jsp – JSP 파일(View 구현)

- src/main/webapp 폴더에 등록해야 함.



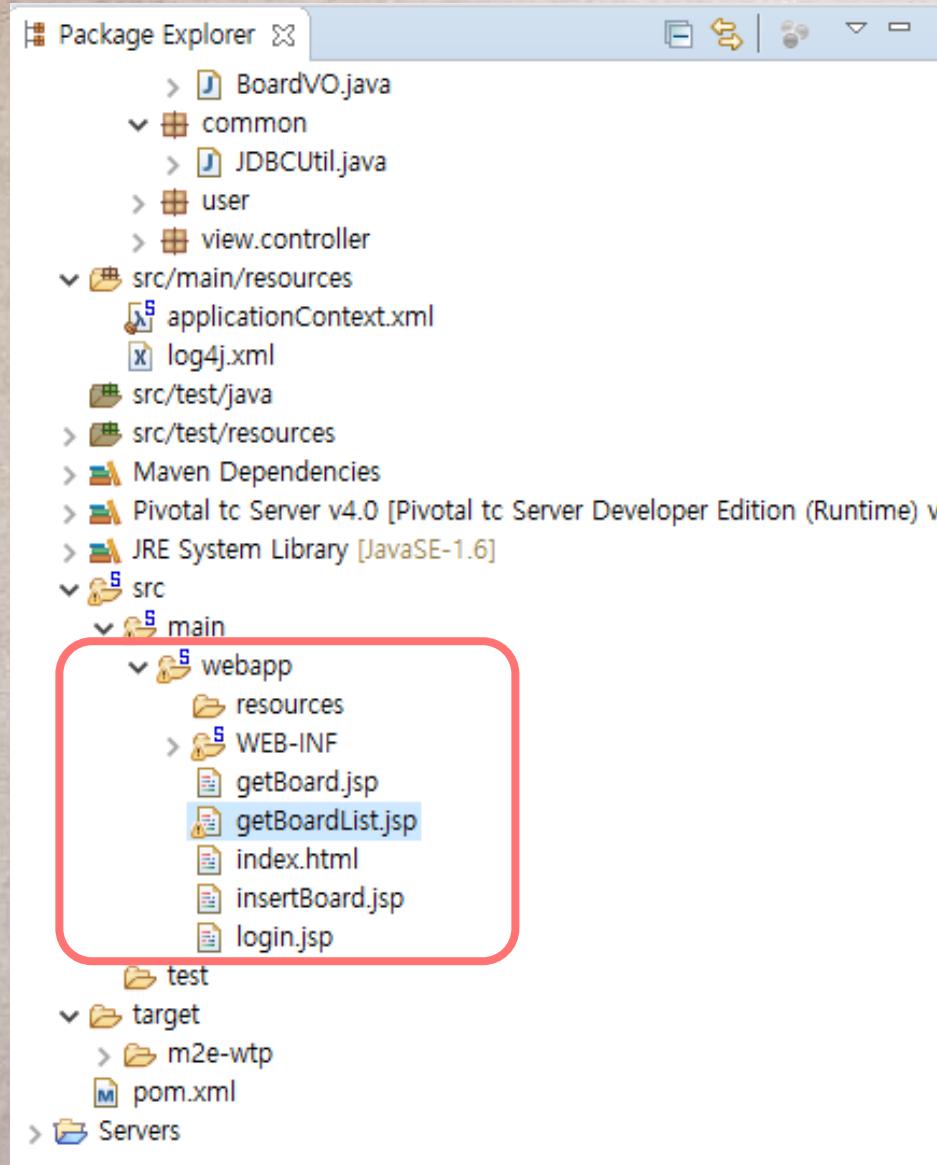
# 이어서...



# login.jsp

```
web.xml *DispatcherServlet.java login.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4<html>
5<head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>로그인</title>
8 </head>
9<body>
10<center>
11 <h1>로그인</h1>
12 <hr>
13<form action="login.do" method="post"> ← action 내용이 "login.do" 호출
14<table border="1" cellpadding="0" cellspacing="0">
15<tr>
16     <td bgcolor="orange">아이디</td>
17     <td><input type="text" name="id"/></td>
18 </tr>
19<tr>
20     <td bgcolor="orange">비밀번호</td>
21     <td><input type="password" name="password" /></td>
22 </tr>
23<tr>
24     <td colspan="2" align="center"><input type="submit" value="로그인"/></td>
25 </tr>
26 </table>
27 </form>
28 <hr>
29 </center>
30 </body>
31 </html>
```

# login.jsp와 이후 게시판 관련 jsp 파일 위치 확인



글 목록 GitHub, Inc. [US] | https://github.com/hopypark/Lecture2018/tree/master/JavaSpring 경태

나의 북마크 편집 난만 :: 선행 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 학습 Time Series Forecast Calculation of Information [Linear Algebra] Basic

Search or jump to... Pull requests Issues Marketplace Explore

hopypark / Lecture2018 Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master Lecture2018 / JavaSpring / Create new file Upload files Find file History

hopypark Add files via upload Latest commit 23a7dbd just now

..

BoardWeb.zip	Add files via upload	13 days ago
Week00.H2Database와 STS설치.pdf	Add files via upload	5 days ago
Week02.스프링과 JDBC.pdf	Add files via upload	4 days ago
Week03.스프링과 MVC.pdf	Add files via upload	a minute ago
h2database_conn.txt	Add files via upload	13 days ago
readme.md	Create readme.md	13 days ago
sql.txt	Add files via upload	13 days ago
user.zip	Add files via upload	6 days ago
webapp_jsp.zip	Add files via upload	just now
readme.md		

JSP 파일

<http://localhost:8080/Board/login.jsp>

A screenshot of a web browser window displaying a login page. The title bar reads "로그인". The address bar shows the URL "localhost:8080/Board/login.jsp". The main content area has a title "로그인" and a form with three fields: "아이디" (ID) and "비밀번호" (Password), both in orange boxes, and a "로그인" (Login) button at the bottom.

→ 아이디와 비밀번호 입력 후 로그인

# DispatcherServlet 클래스의 “/login.do” 호출

```
52 // 2. 클라이언트의 요청 path에 따라 적절히 분기처리 한다.  
53 if(path.equals("/login.do")){  
54     System.out.println("로그인 처리");  
55     // 1. 사용자 입력 정보 추출  
56     String id = request.getParameter("id");  
57     String password = request.getParameter("password");  
58  
59     // 2. DB 연동 처리  
60     UserVO vo = new UserVO();  
61     vo.setId(id);  
62     vo.setPassword(password);  
63  
64     UserDAO userDAO = new UserDAO();  
65     UserVO user = userDAO.getUser(vo);  
66  
67     // 3. 화면 네비게이션  
68     if(user != null){  
69         response.sendRedirect("getBoardList.do");  
70     }else{  
71         response.sendRedirect("login.jsp");  
72     }  
73 }else if(path.equals("/logout.do")){  
74     System.out.println("로그아웃 처리");  
75 }else if(path.equals("/insertBoard.do")){  
76     System.out.println("글 등록 처리");
```

추가

# DispatcherServlet 클래스의 “/getBoardList.do” 호출

```
88 }else if(path.equals("/getBoardList.do")){
89     System.out.println("글 목록 검색 처리");
90     // 1. 사용자 입력 정보 추출(검색 기능은 나중에 구현)
91     // 2. DB 연동 처리
92     BoardVO vo = new BoardVO();
93     BoardDAO boardDAO = new BoardDAO();
94     List<BoardVO> boardList = boardDAO.getBoardList(vo);
95
96     // 3. 검색 결과를 세션에 저장하고 목록 화면으로 이동한다.
97     HttpSession session = request.getSession();
98     session.setAttribute("boardList", boardList);
99     response.sendRedirect("getBoardList.jsp");
100 }
101 }
102 }
```

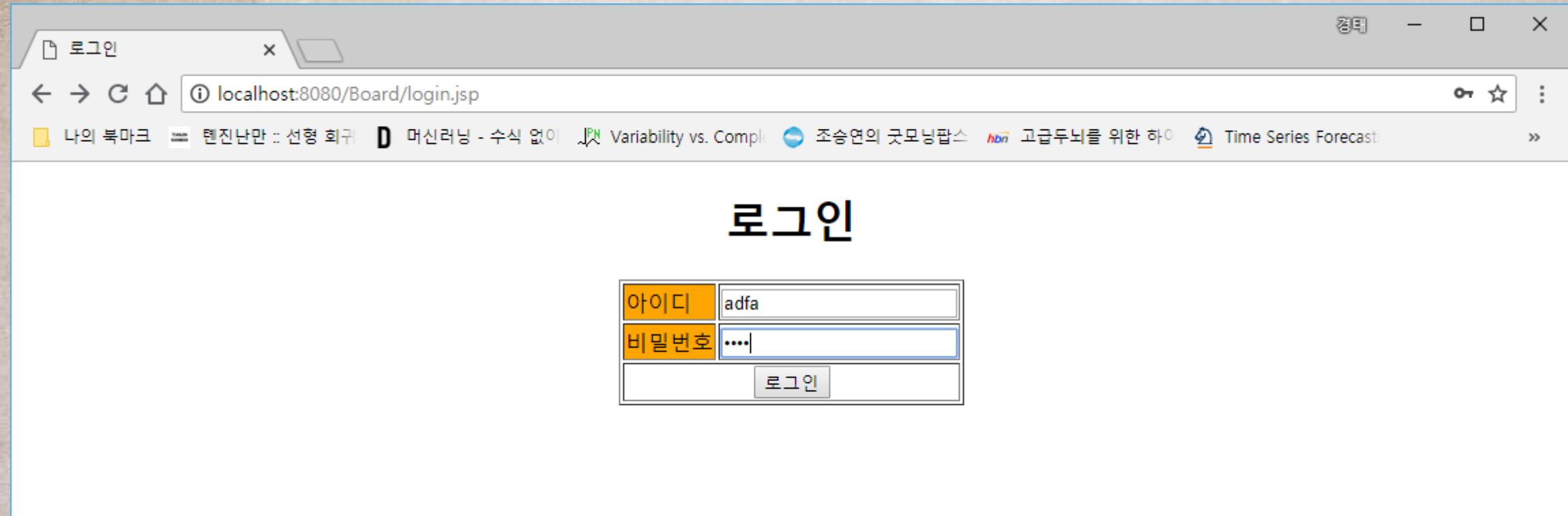
추가

# getBoardList.jsp

```
login.jsp  web.xml  DispatcherServlet.java  getBoardList.jsp
1 <%@ page import="java.util.List" %>
2 <%@ page import="kr.ac.inje.comsi.board.BoardVO" %>
3 <%@ page language="java" contentType="text/html; charset=UTF-8"
4 pageEncoding="UTF-8"%>
5<%
6 // 세션에 저장된 글 목록을 꺼낸다.
7 List<BoardVO> boardList = (List<BoardVO>) session.getAttribute("boardList");
8 %>
9 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
10<html>
11<head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 <title>글 목록</title>
14 </head>
15<body>
16<center>
17     <h1>글 목록</h1>
18     <h3>환영합니다. <a href="logout.do">Logout</a></h3>
19
20     <!-- 검색시작 -->
21<form action="getBoardList.jsp" method="post">
22<table border="1" cellpadding="0" cellspacing="0" width="700">
23<tr>
24<td align="right">
25<select name="searchCondition">
26     <option value="TITLE">제목</option>
27     <option value="CONTENT">내용</option>
28</select>
29     <input name="searchKeyword" type="text"/>
30     <input type="submit" value="검색"/>
31</td>
32</tr>
33</table>
34</form>
35 <!-- 검색종료 -->
```

```
36
37    <table border="1" width="700">
38        <tr>
39            <th bgcolor="orange" width="100">번호</th>
40            <th bgcolor="orange" width="200">제목</th>
41            <th bgcolor="orange" width="150">작성자</th>
42            <th bgcolor="orange" width="150">등록일</th>
43            <th bgcolor="orange" width="100">조회수</th>
44        </tr>
45        <% for(BoardVO board : boardList) { %>
46        <tr>
47            <td align="center"><%= board.getSeq() %></td>
48            <td align="left"><a href="getBoard.do?seq=<%= board.getSeq() %>"><%= board.getTitle() %></a></td>
49            <td align="center"><%= board.getWriter() %></td>
50            <td align="center"><%= board.getRegDate() %></td>
51            <td align="center"><%= board.getCnt() %></td>
52        </tr>
53    <% } %>
54
55    </table>
56    <br>
57    <a href="insertBoard.jsp">새글 등록</a>
58 </center>
59 </body>
60 </html>
```

# 로그인 실패시 → 다시 로그인 페이지로 이동



## 실행 결과

```
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Refreshing root WebApplicationContext: startup date [Sun Sep 09 11:45:21 KST 2018]; parent WebApplicationContext: null
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from ServletContext resource [/WEB-INF/applicationContext.xml]
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization completed in 231 ms
/login.do
로그인 처리
입력내용: ID=adfa, Password=null
===> JDBC로 getUser() 기능 처리
ID=adfa 인 유저가 없습니다. 로그인 페이지로 이동
```

# 로그인 성공 후 글 목록 검색하기

The screenshot shows a web browser window with two tabs: '로그인' and 'H2 콘솔'. The '로그인' tab is active, displaying a login form with fields for '아이디' (test) and '비밀번호' (.....), and a '로그인' button. Below the form, the page content area contains a red-bordered box of server log output.

```
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Closing Root WebApplicationContext: star
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization started
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Refreshing Root WebApplicationContext: s
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from ServletCo
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization completed in 258
/login.do
로그인 처리
입력내용: ID=test, Password=test123
==> JDBC로 getUser() 기능 처리
ID=test, Password=test123 인 유저가 있습니다. 목록 페이지로 이동
/getBoardList.do
글 목록 검색 처리
==> JDBC로 getBoardList() 기능 처리
```

# 로그인 성공 화면

글 목록

localhost:8080/Board/getBoardList.jsp

나의 북마크 렌진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝不尽 고급두뇌를 위한 학습 Time Series Forecast Calculation of Information

## 글 목록

환영합니다. [Logout](#)

번호	제목	작성자	등록일	조회수
4	<a href="#">JDBD 테스트2</a>	관리자	2018-05-27	0
3	<a href="#">JDBD 테스트</a>	관리자	2018-05-27	0
2	<a href="#">임시 제목</a>	홍길동	2018-05-12	0
1	<a href="#">가입인사</a>	관리자	2018-04-08	0

[새글 등록](#)

## 게시판 목록 검색 실행 순서

- ① DispatcherServlet이 클라이언트의 "/getBoardList.do" 요청을 받으면
- ② DispatcherServlet은 BoardDAO 객체를 이용하여 글 목록을 검색 한다.
- ③ 검색된 글 목록을 세션에 등록하고
- ④ DispatcherServlet에서 getBoardList.jsp 화면을 요청하면,
- ⑤ getBoardList.jsp는 세션에 저장된 글 목록을 꺼내어 목록 화면을 구성한다.
- ⑥ 마지막으로 이 응답 화면이 브라우저에 전송된다.

기타 게시판 기능 살펴보기

# 글 상세보기 구현(글 수정하기와 공유)

The screenshot shows a web browser window displaying a board detail page. The URL in the address bar is `localhost:8080/Board/getBoard.jsp`. The page content is as follows:

Log-out

제목	JDBD 테스트2
작성자	관리자
내용	JDBC 테스트2.....
등록일	2018-05-27
조회수	0

[글 수정](#)

[글등록](#) [글삭제](#) [글목록](#)

# 글 상세보기 구현 - DispatcherServlet 클래스 수정

```
91 } else if (path.equals("/getBoard.do")) {  
92     System.out.println("글 상세 조회 처리");  
93     // 1. 검색할 게시글 번호 추출  
94     String seq = request.getParameter("seq");  
95  
96     // 2. DB 연동  
97     BoardVO vo = new BoardVO();  
98     vo.setSeq(Integer.parseInt(seq));  
99  
100    BoardDAO boardDAO = new BoardDAO();  
101    BoardVO board = boardDAO.getBoard(vo);  
102  
103    // 3. 검색 결과를 세션에 저장하고 상세화면으로 이동  
104    HttpSession session = request.getSession();  
105    session.setAttribute("board", board);  
106    response.sendRedirect("getBoard.jsp");  
107  
108 } else if (path.equals("/getBoardList.do")) {
```

1. seq에 해당하는 게시물을 BoardDAO객체를 이용해 가져온다.
2. 가져온 BoardVO객체를 session에 저장하고 getBoard.jsp 파일로 리다이렉트한다.

# getBoard.jsp – 업데이트를 위한 페이지와 공유

```
login.jsp    web.xml    DispatcherServlet.java    getBoardList.jsp    insertBoard.jsp    getBoard.jsp
1 <%@ page import="kr.ac.inje.comsi.board.BoardVO" %>
2 <%@ page language="java" contentType="text/html; charset=UTF-8"
3     pageEncoding="UTF-8"%>
4<%
5     // 세션에 저장된 게시글 정보를 꺼낸다.
6     BoardVO board = (BoardVO) session.getAttribute("board");
7 %>
8 <!DOCTYPE html>
9<html>
10<head>
11 <meta charset="UTF-8">
12 <title>글 상세</title>
13 </head>
14<body>
15<div align="center">
16     <a href="logout.do">Log-out</a>
17     <hr>
18     <form action="updateBoard.do" method="post"> ←
19         <input name="seq" type="hidden" value="<%=board.getSeq()%>" />
20         <table border="1">
21             <tr>
22                 <td bgcolor="orange" width="70">제목</td>
23                 <td align="left"><input name="title" type="text"
24                     value="<%=board.getTitle()%>" /></td>
25             </tr>
26             <tr>
27                 <td bgcolor="orange">작성자</td>
28                 <td align="left"><%=board.getWriter()%></td>
29             </tr>
```

# ....getBoard.jsp 이어서

```
30<tr>
31    <td bgcolor="orange">내용</td>
32    <td align="left"><textarea name="content" cols="80" rows="10">
33        <%=board.getContent()%></textarea></td>
34</tr>
35<tr>
36    <td bgcolor="orange">등록일</td>
37    <td align="left"><%=board.getRegDate()%></td>
38</tr>
39<tr>
40    <td bgcolor="orange">조회수</td>
41    <td align="left"><%=board.getCnt()%></td>
42</tr>
43<tr>
44    <td colspan="2" align="center"><input type="submit"
45        value="글 수정" /></td>
46</tr>
47</table>
48</form>
49<hr>
50<a href="insertBoard.jsp">글등록</a>&ampnbsp&ampnbsp&ampnbsp;
51<a href="deleteBoard.do?seq=<%=board.getSeq()%>">글삭제</a>&ampnbsp&ampnbsp;
52<a href="getBoardList.do">글목록</a>
53</div>
54</body>
55</html>
```

# 글 등록 구현하기

글 목록

localhost:8080/Board/getBoardList.jsp

나의 북마크 웹진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 학습 Time Series Forecast Calculation of Information

## 글 목록

환영합니다. [Logout](#)

번호	제목	작성자	등록일	조회수
5	<a href="#">글 등록</a>	글 등록	2018-09-10	0
4	<a href="#">JDBC 테스트2</a>	관리자	2018-05-27	0
3	<a href="#">JDBC 테스트</a>	관리자	2018-05-27	0
2	<a href="#">임시 제목</a>	홍길동	2018-05-12	0
1	<a href="#">가입인사</a>	관리자	2018-04-08	0

[새글 등록](#)

# 글 등록 구현 - 글 등록 실행 화면

새 글 등록

localhost:8080/Board/insertBoard.jsp

나의 북마크 헌진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 하이브리드 Time Series Forecast Calculation of Information

## 글 등록

[Log-out](#)

제목	<input type="text"/>
작성자	<input type="text"/>
내용	<input type="text"/>

[새 글 등록](#)

[글 목록 가기](#)

# 글 등록 구현 – DispatcherServlet 클래스 수정

```
85 } else if (path.equals("/insertBoard.do")) {  
86     System.out.println("글 등록 처리");  
87  
88     // 1. 사용자 입력 정보 추출  
89     String title = request.getParameter("title");  
90     String writer = request.getParameter("writer");  
91     String content = request.getParameter("content");  
92  
93     // 2. DB 연동처리  
94     BoardVO vo = new BoardVO();  
95     vo.setTitle(title);  
96     vo.setWriter(writer);  
97     vo.setContent(content);  
98  
99     BoardDAO boardDAO = new BoardDAO();  
100    boardDAO.insertBoard(vo);  
101  
102    // 3. 화면 이동  
103    response.sendRedirect("getBoardList.do");  
104 } else if (path.equals("/updateBoard.do")) {
```

1. title, writer, content 파라미터로 받아서 BoardVO 객체를 생성하여 설정
2. 생성된 BoardVO 객체를 BoardDAO 객체를 이용해 DB에 저장한다.

# 글 등록 페이지- insertBoard.jsp 소스

```
login.jsp    web.xml    DispatcherServlet.java    getBoardList.jsp    *insertBoard.jsp    BoardDAO.java
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>새 글 등록</title>
8 </head>
9<body>
10<div align="center">
11     <h1>글 등록</h1>
12     <a href="logout.do">Log-out</a>
13     <hr>
14     <form action="insertBoard.do" method="post">
15         <table border="1" >
16             <tr>
17                 <td bgcolor="orange" width="70">제목</td>
18                 <td align="left"><input type="text" name="title" /></td>
19             </tr>
20             <tr>
21                 <td bgcolor="orange">작성자</td>
22                 <td align="left"><input type="text" name="writer" size="10" /></td>
23             </tr>
24             <tr>
25                 <td bgcolor="orange">내용</td>
26                 <td align="left"><textarea name="content" cols="80" rows="10"></textarea></td>
27             </tr>
28             <tr>
29                 <td colspan="2" align="center"><input type="submit" value="새 글 등록" /></td>
30             </tr>
31         </table>
32     </form>
33     <hr>
34     <a href="getBoardList.do">글 목록 가기</a>
35 </div>
36 </body>
37 </html>
```

# 글 수정하기 – 글 세부내용 보기와 공유

The screenshot shows a web browser window with the title "글 상세" (Post Detail) and the URL "localhost:8080/Board/getBoard.jsp". The page displays a table with the following data:

제목	글 등록
작성자	글 등록
내용	글 수정테스트
등록일	2018-09-10
조회수	0

A red box highlights the "글 수정" (Post Modify) button at the bottom of the table. Below the table, there are three navigation links: "글등록" (Post Register), "글삭제" (Post Delete), and "글목록" (Post List). The browser's address bar shows the URL "localhost:8080/Board/getBoard.jsp".

# 글 수정하기 – DispatcherServlet 클래스 수정

```
104 } else if (path.equals("/updateBoard.do")) {  
105     System.out.println("글 수정 처리");  
106  
107     // 1. 사용자 입력 정보 추출  
108     String title = request.getParameter("title");  
109     String content = request.getParameter("content");  
110     String seq = request.getParameter("seq");  
111  
112     // 2. DB 연동처리  
113     BoardVO vo = new BoardVO();  
114     vo.setTitle(title);  
115     vo.setContent(content);  
116     vo.setSeq(Integer.parseInt(seq));  
117  
118     BoardDAO boardDAO = new BoardDAO();  
119     boardDAO.updateBoard(vo);  
120  
121     // 3. 화면 이동  
122     response.sendRedirect("getBoardList.do");  
123  
124 } else if (path.equals("/deleteBoard.do")) {
```

1. title, writer, content 파라미터로 받아서 BoardVO 객체를 생성하여 설정
2. 생성된 BoardVO 객체로 BoardDAO 객체를 이용해 DB를 수정한다.

## ※ BoardDAO 소스 확인

```
// 글 수정
public void updateBoard(BoardVO vo) {
    System.out.println("==> JDBC로 updateBoard() 기능 처리");
    try {
        conn = JDBCUtil.getConnection();
        pstmt = conn.prepareStatement(BOARD_UPDATE);
        pstmt.setString(1, vo.getTitle());
        pstmt.setString(2, vo.getWriter()); // vo.getWriter()
        pstmt.setInt(3, vo.getSeq());
        pstmt.executeUpdate();
    }catch(Exception e) {
        e.printStackTrace();
    }finally {
        JDBCUtil.close(pstmt, conn);
    }
}
```

# 글 삭제하기 실행화면

글 상세

localhost:8080/Board/getBoard.jsp

나의 북마크 렌진난만 :: 선행 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조증연의 굿모닝팝스 고급두뇌를 위한 하이브리드 Time Series Forecast Calculation of Information

Log-out

제목	null
작성자	null
내용	null
등록일	2018-09-10
조회수	0

글 수정

글등록 글삭제 글목록

# 글 삭제하기 – DispatcherServlet 클래스

```
124 } else if (path.equals("/deleteBoard.do")) {  
125     System.out.println("글 삭제 처리");  
126  
127     // 1. 사용자 입력 정보  
128     String seq = request.getParameter("seq");  
129  
130     // 2. DB 연동  
131     BoardVO vo = new BoardVO();  
132     vo.setSeq(Integer.parseInt(seq));  
133  
134     BoardDAO boardDAO = new BoardDAO();  
135     boardDAO.deleteBoard(vo);  
136  
137     // 3. 화면 이동  
138     response.sendRedirect("getBoardList.do");  
139  
140 } else if (path.equals("/getBoard.do")) {  
141     System.out.println("글 상세 조회 처리");
```

1. seq 파라미터로 받아서 BoardVO 객체를 생성하여 설정
2. 생성된 BoardVO 객체로 BoardDAO 객체를 이용해 DB를 삭제한다.

# 글 삭제 실행 후 화면

글 목록

환영합니다. [Logout](#)

번호	제목	작성자	등록일	조회수
5	<a href="#">글 등록</a>	글 등록	2018-09-10	0
4	<a href="#">JDBD 테스트2</a>	관리자	2018-05-27	0
3	<a href="#">JDBD 테스트</a>	관리자	2018-05-27	0
2	<a href="#">임시 제목</a>	홍길동	2018-05-12	0
1	<a href="#">가입인사</a>	관리자	2018-04-08	0

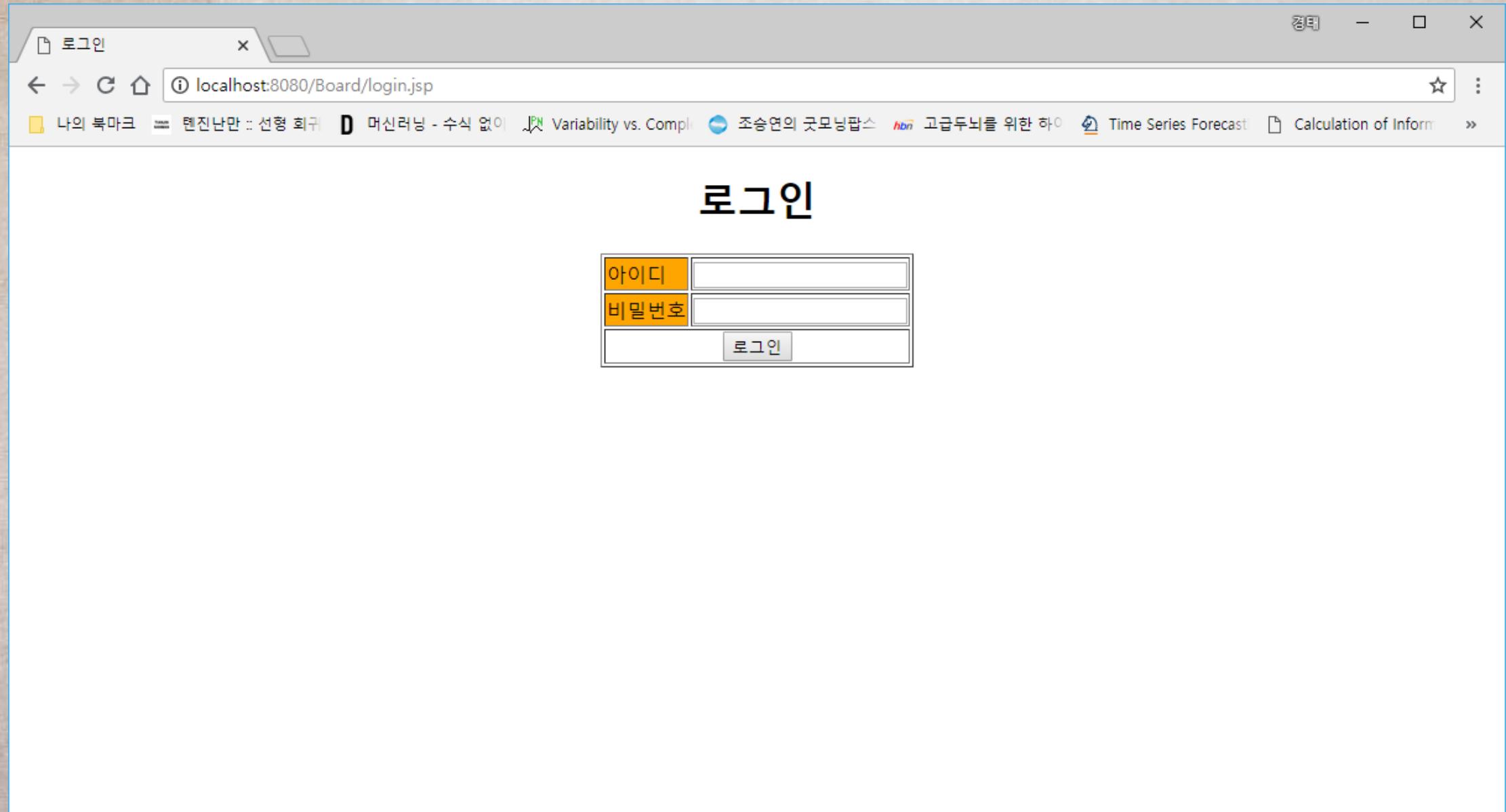
[새글 등록](#)

# 로그아웃 처리

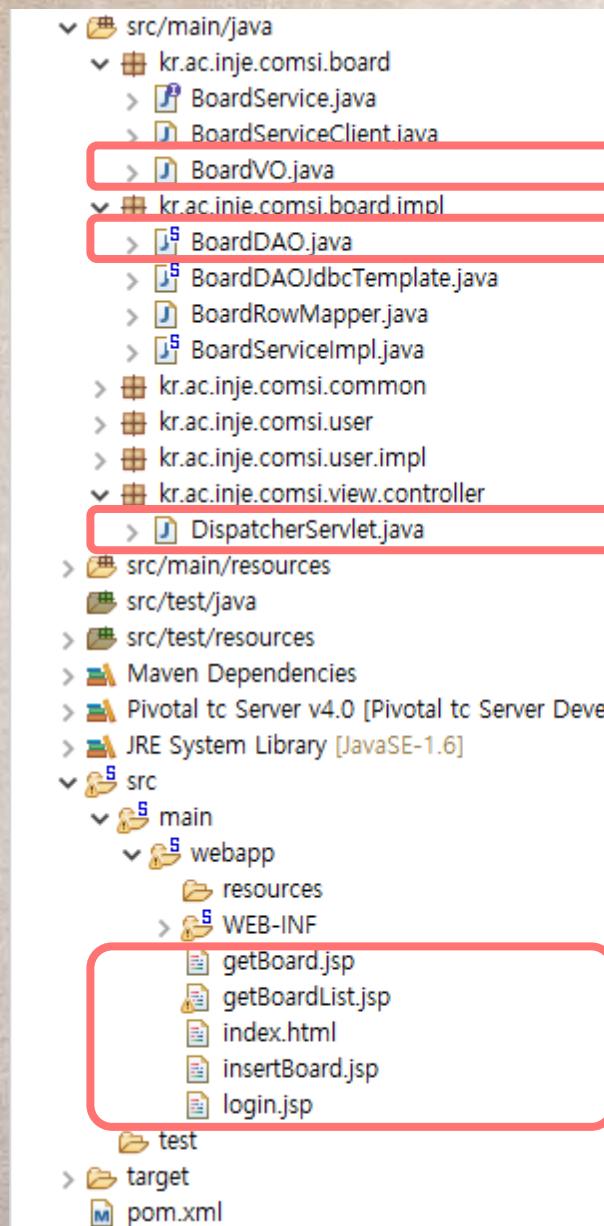
```
83 } else if (path.equals("/logout.do")) {  
84     System.out.println("로그아웃 처리");  
85     // 1. 브라우저와 연결된 세션 객체 강제 종료  
86     HttpSession session = request.getSession();  
87     session.invalidate();  
88  
89     // 2. 세션 종료 후, 메인(로그인) 화면으로 이동  
90     response.sendRedirect("login.jsp");  
91  
92 } else if (path.equals("/insertBoard.do")) {  
93     System.out.println("글 등록 처리");
```

1. HttpSession 객체를 얻어와 무효화
2. 로그인 페이지로 이동

# 로그아웃 후 로그인 화면으로 이동



# MVC 아키텍처에서 본 각 소스의 역할



Model

Controller

View