
박 경 태
comsi.java@gmail.com

# 고급 자바 프로그래밍
## : STS를 이용한 Spring 프로그래밍

# 강의 내용

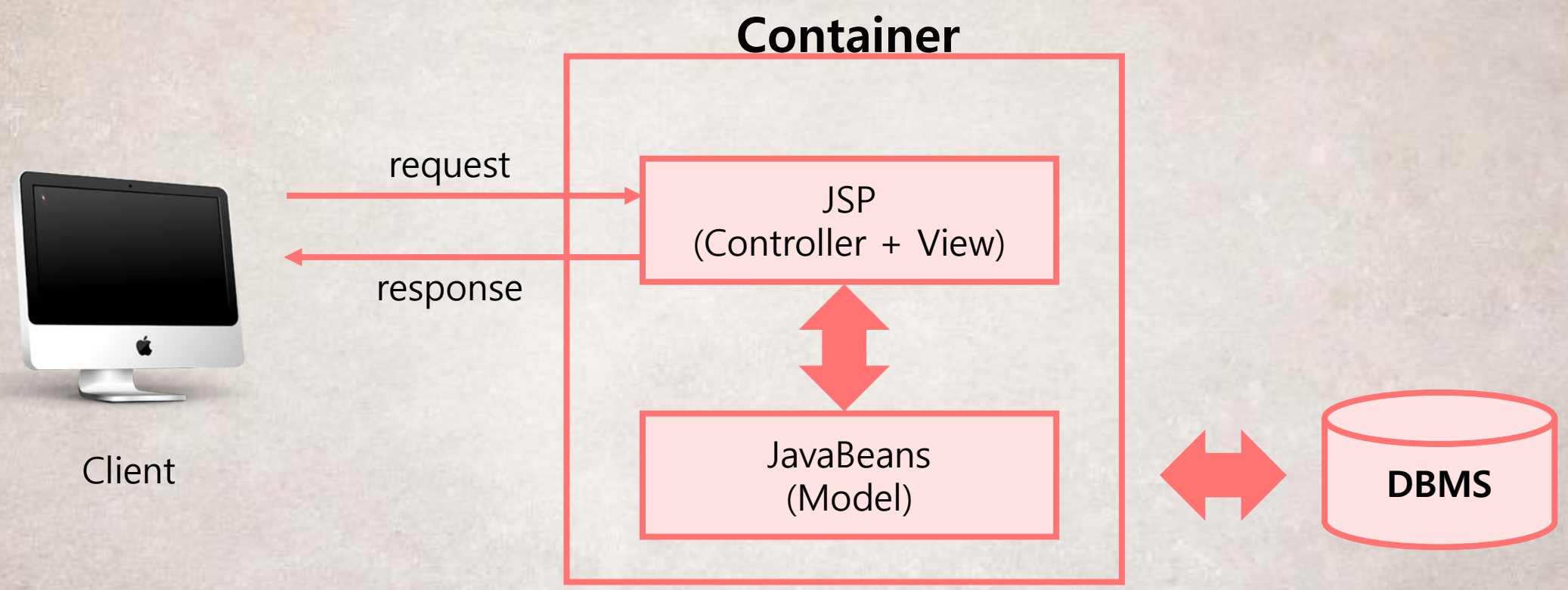
| 순서 | 내 용 |
|---|---|
| 1 | • Spring IoC를 이용한 비즈니스 컴포넌트 만들기 |
| 2 | • Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기<br>• Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리<br>  • JdbcTemplate 클래스를 이용한 JDBC의 반복적 코드 제거와 SQL분리 |
| 3 | • Spring MVC를 이용한 MVC 아키텍쳐 적용하기 |
| 4 | • Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등) |
| 5 | • Spring과 MyBatis 연동하기<br>• Spring과 JPA 연동하기<br>  • 표준 ORM(Object Relational Mapping)을 이용한 RDB와의 객체 지향적 연결과 관리 |

http://github.com/hopypark

# Model 1 아키텍처로 게시판 개발

# Model 1 아키텍처 구조

- Model 1 아키텍처는 JSP와 JavaBean만 사용하여 웹 개발(200년대 초~)




**➔ 자바에서 Bean은 객체를 의미함.**

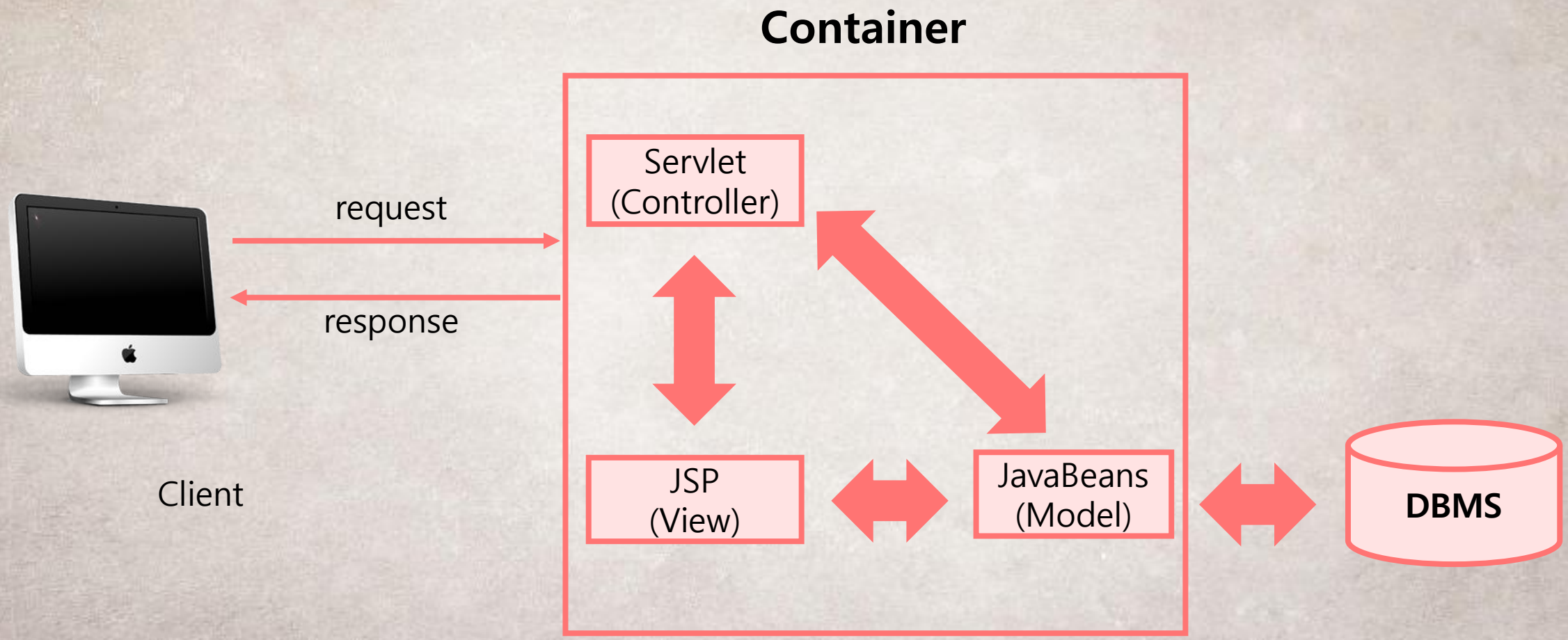
➔ **자바 서버 페이지(Java Server Pages, JSP)**는 HTML내에 자바 코드를 삽입하여 웹 서버에서 **동적**으로 웹 페이지를 생성하여 웹 브라우저에 돌려주는 언어

# Model 1 아키텍처 구조

- **Model** : 데이터베이스 연동 로직을 제공하면서 DB에서 검색한 데이터가 저장되는 자바 객체(정확한 의미)
  - 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 **Controller의 등장**이며 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

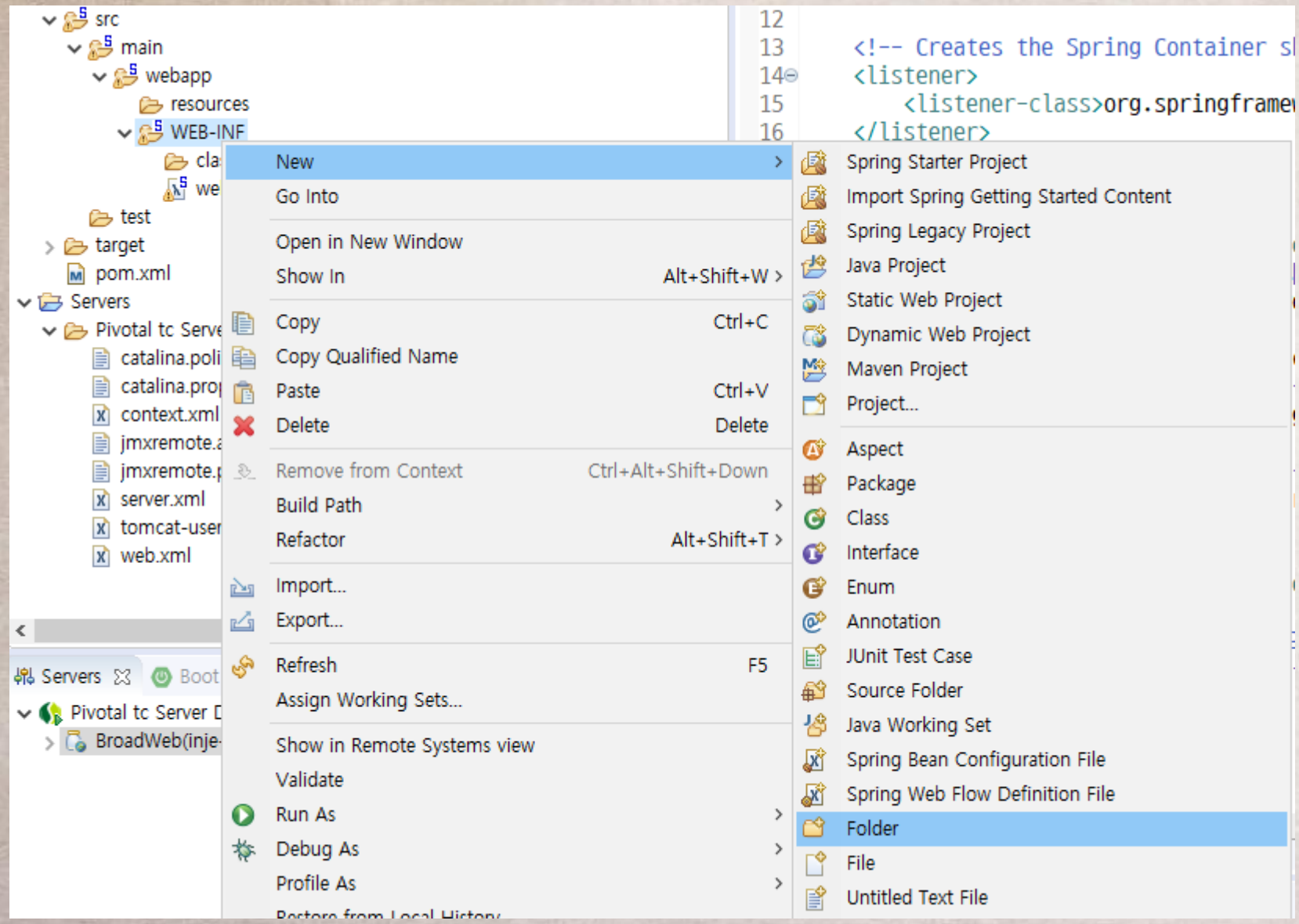
servlet-context.xml | index.html | Insert title here | web.xml | root-context.xml

```
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>
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>
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>
```

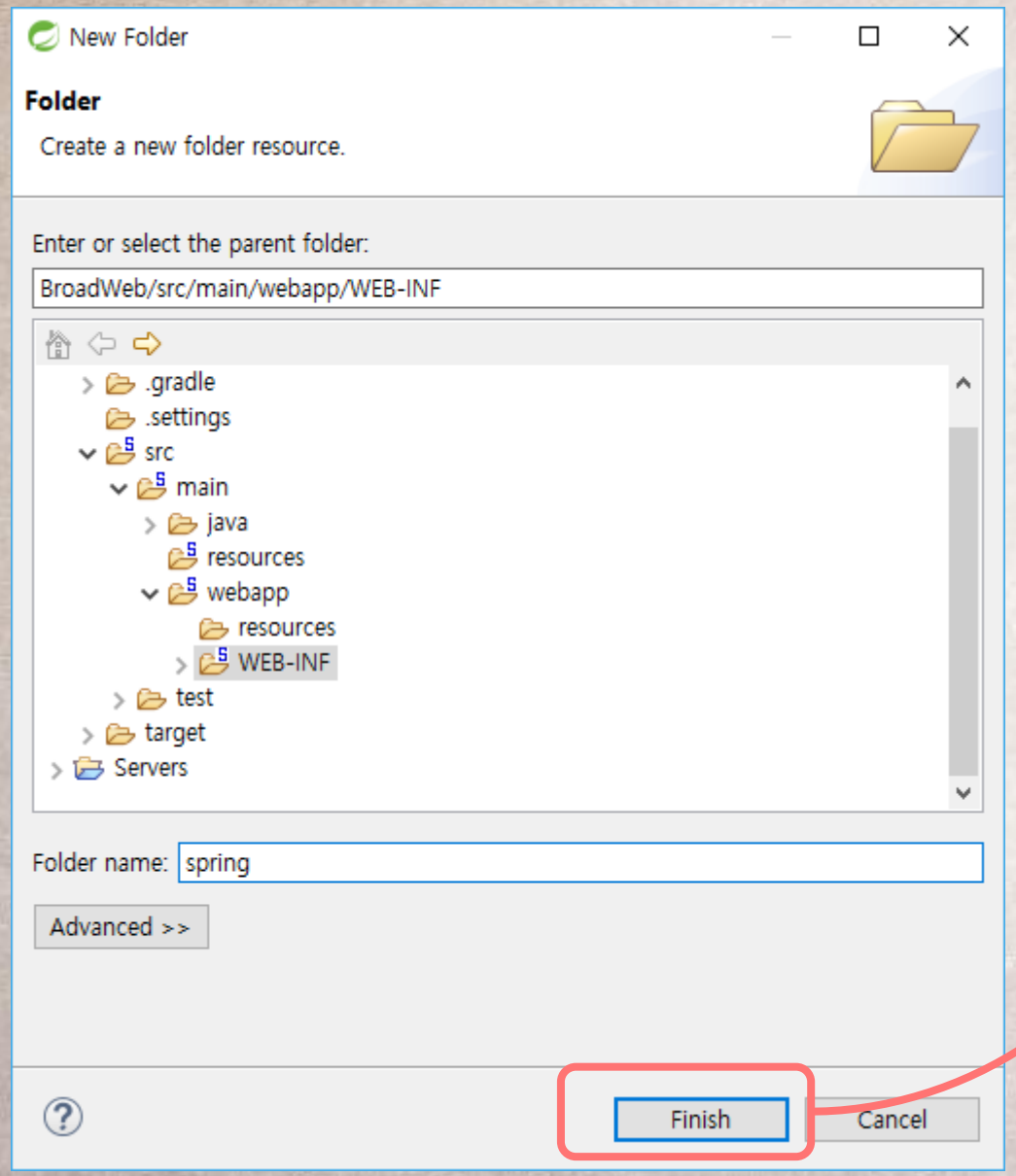
파일 추가

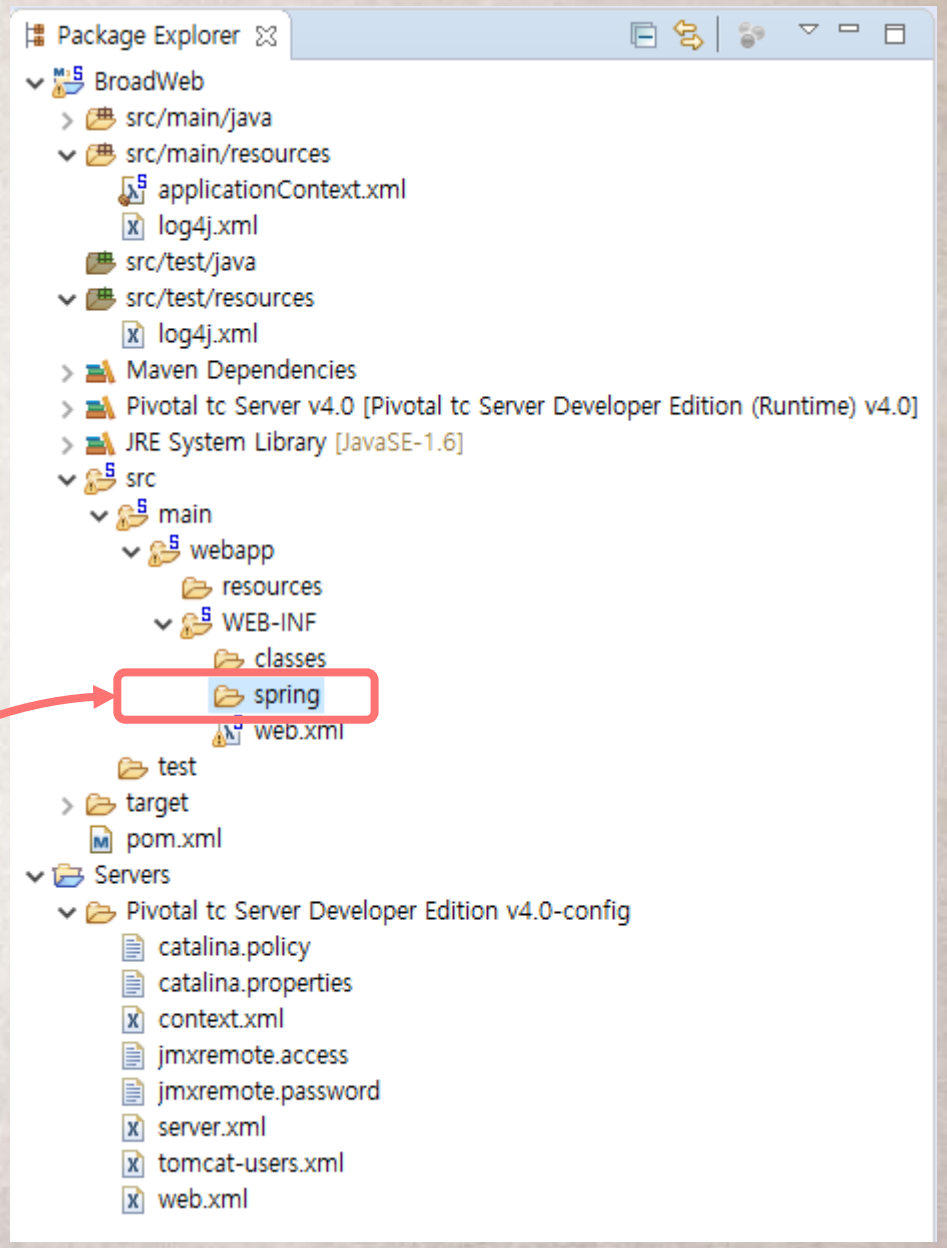
파일 추가

# /WEB-INF/spring/root-context.xml(스프링 설정파일)

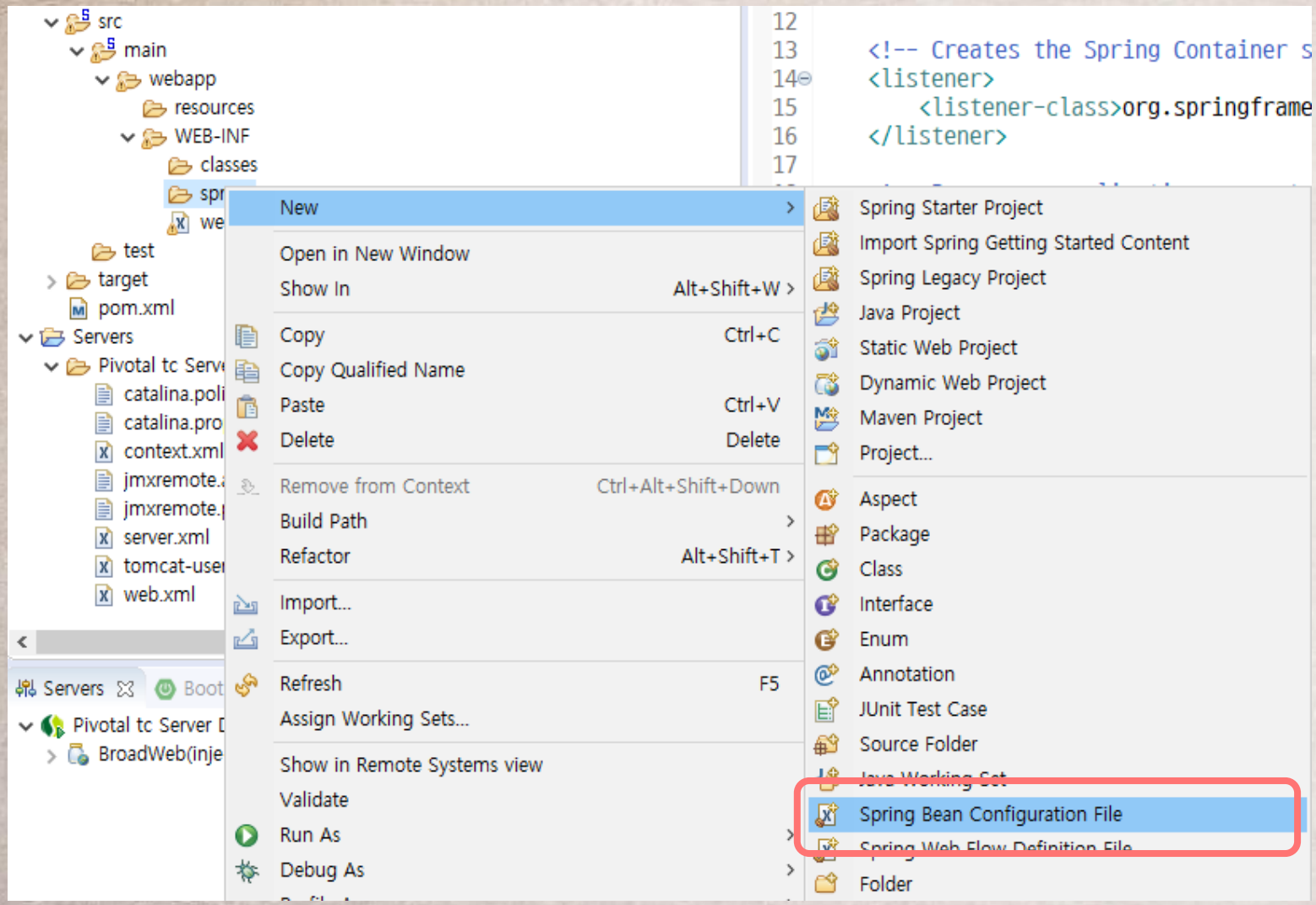


# /WEB-INF/spring/root-context.xml(스프링 설정파일)

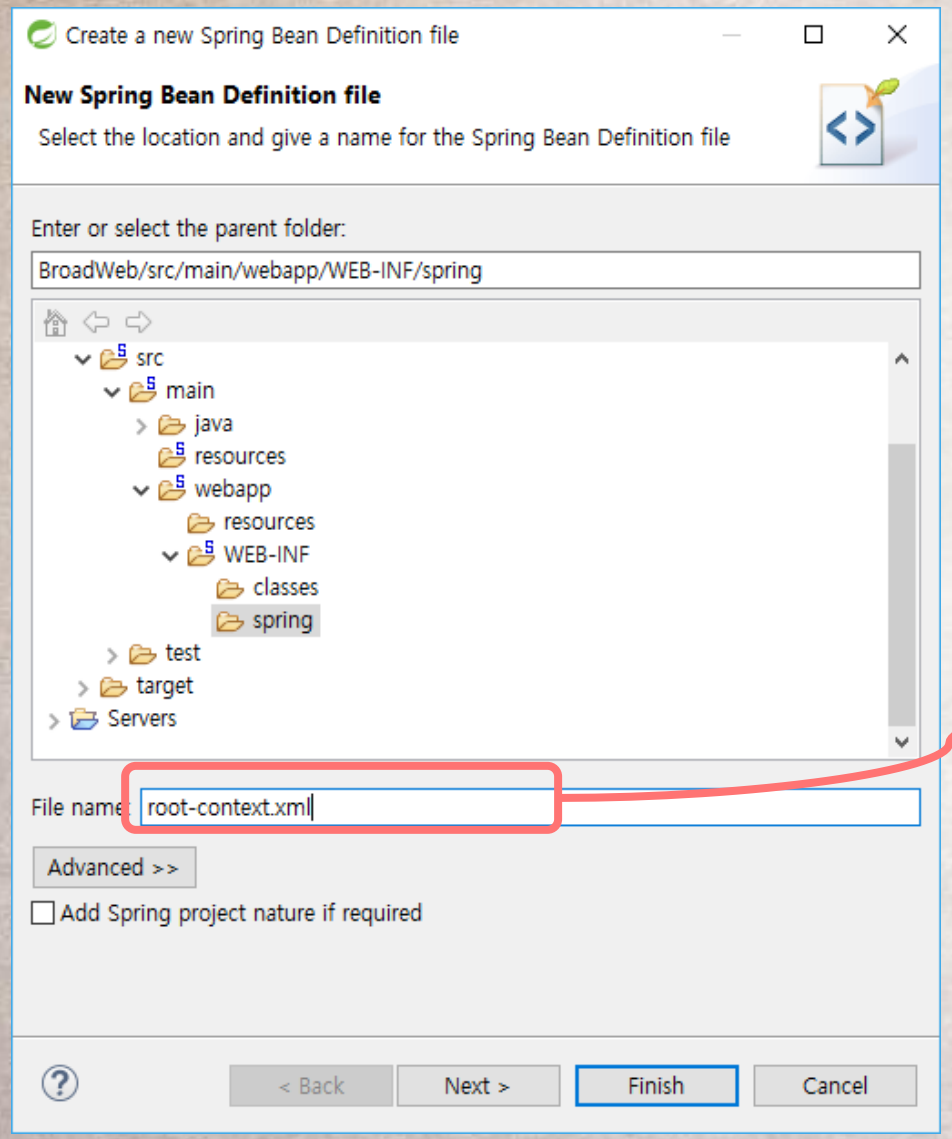


# /WEB-INF/spring/root-context.xml(스프링 설정파일)

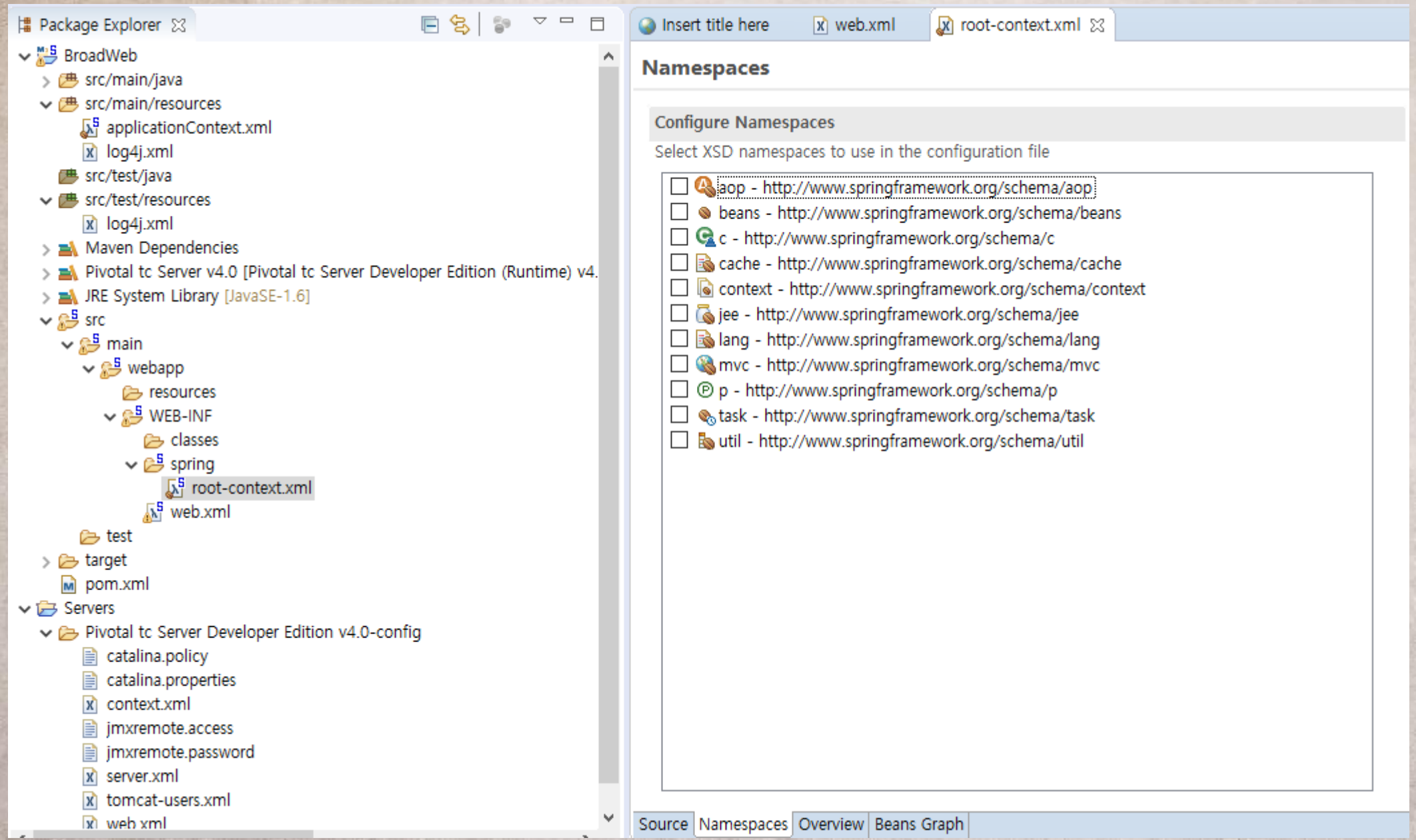


# /WEB-INF/spring/root-context.xml(스프링 설정파일)

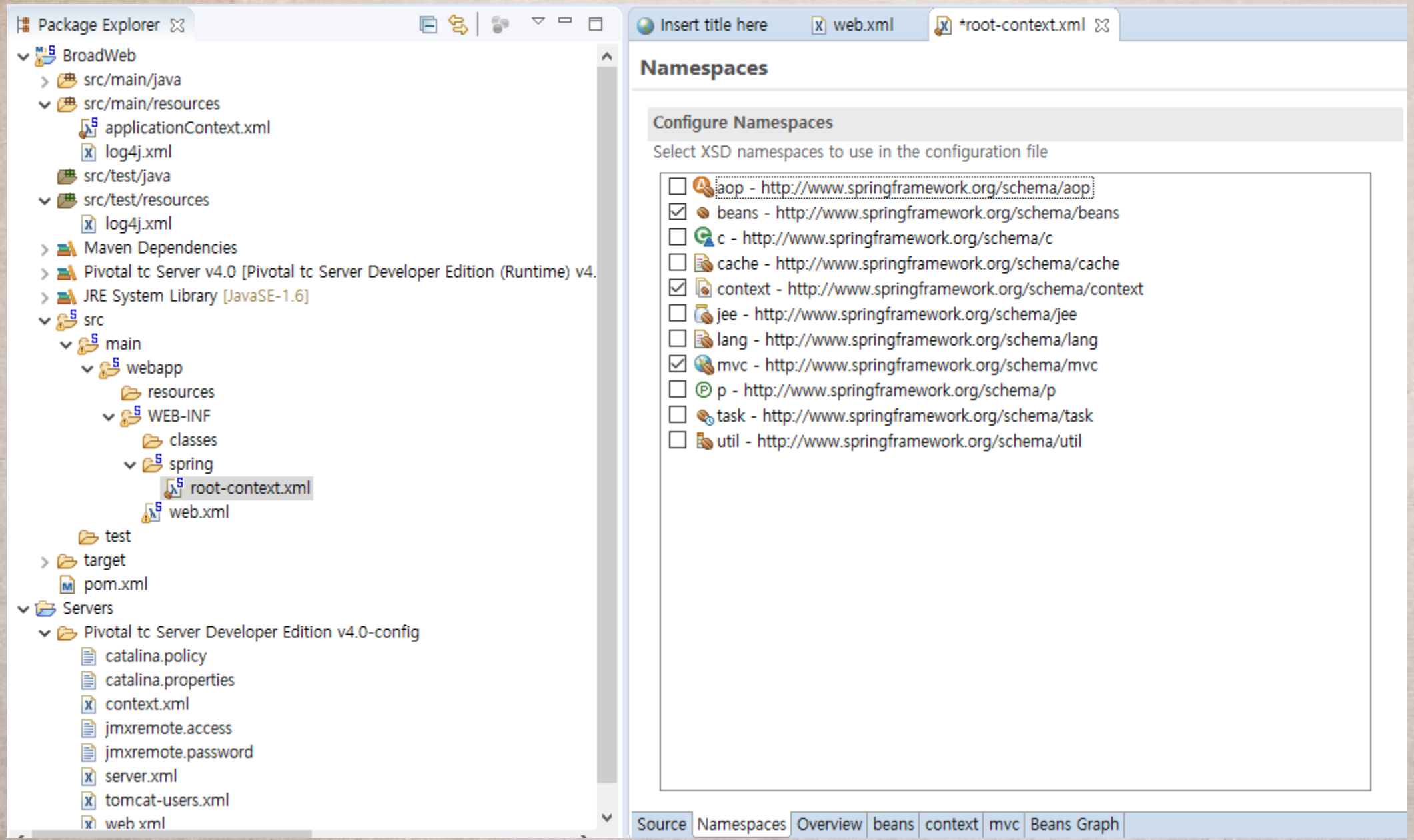



root-context.xml

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



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

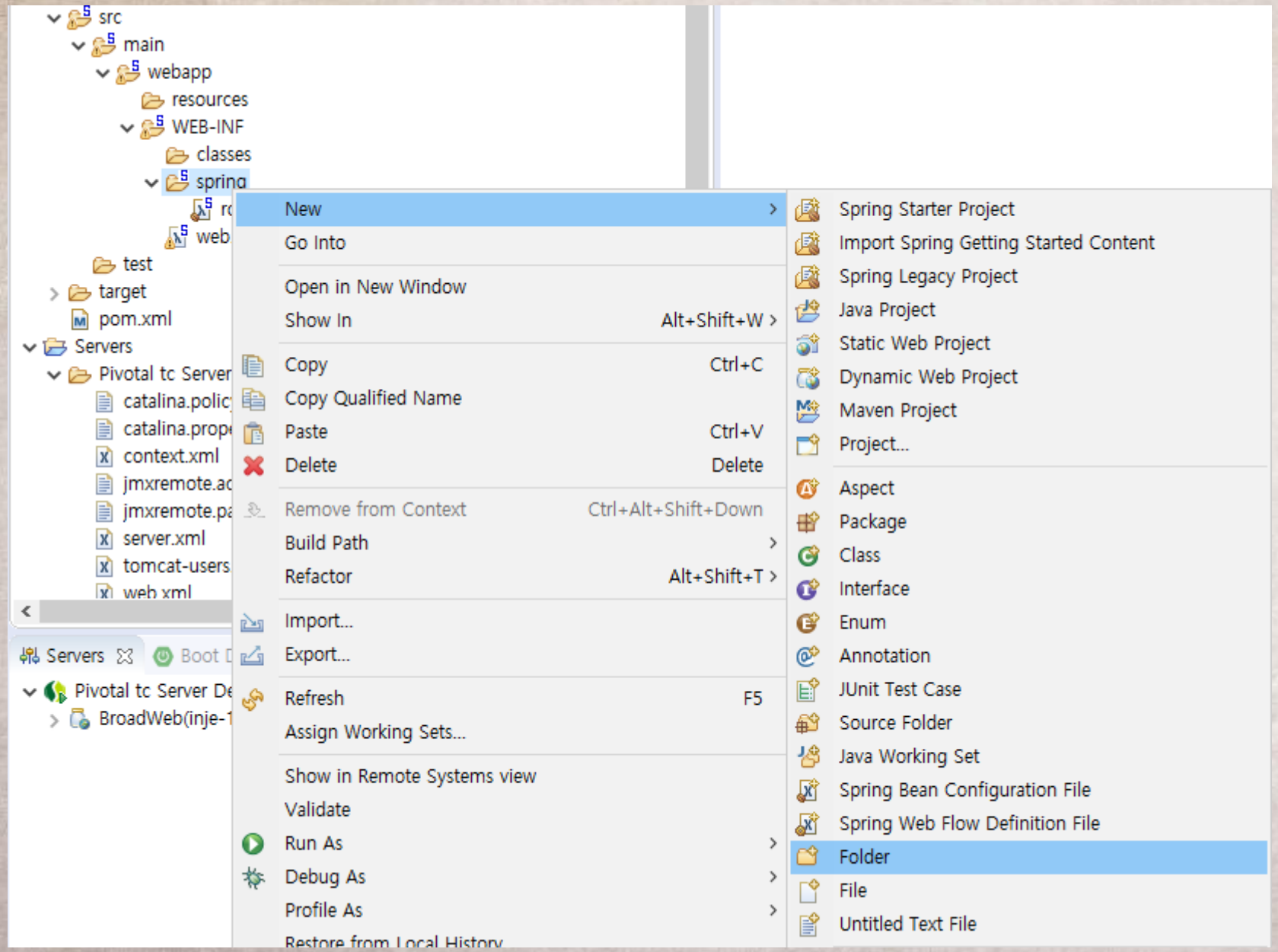


# root-context.xml 파일 설정

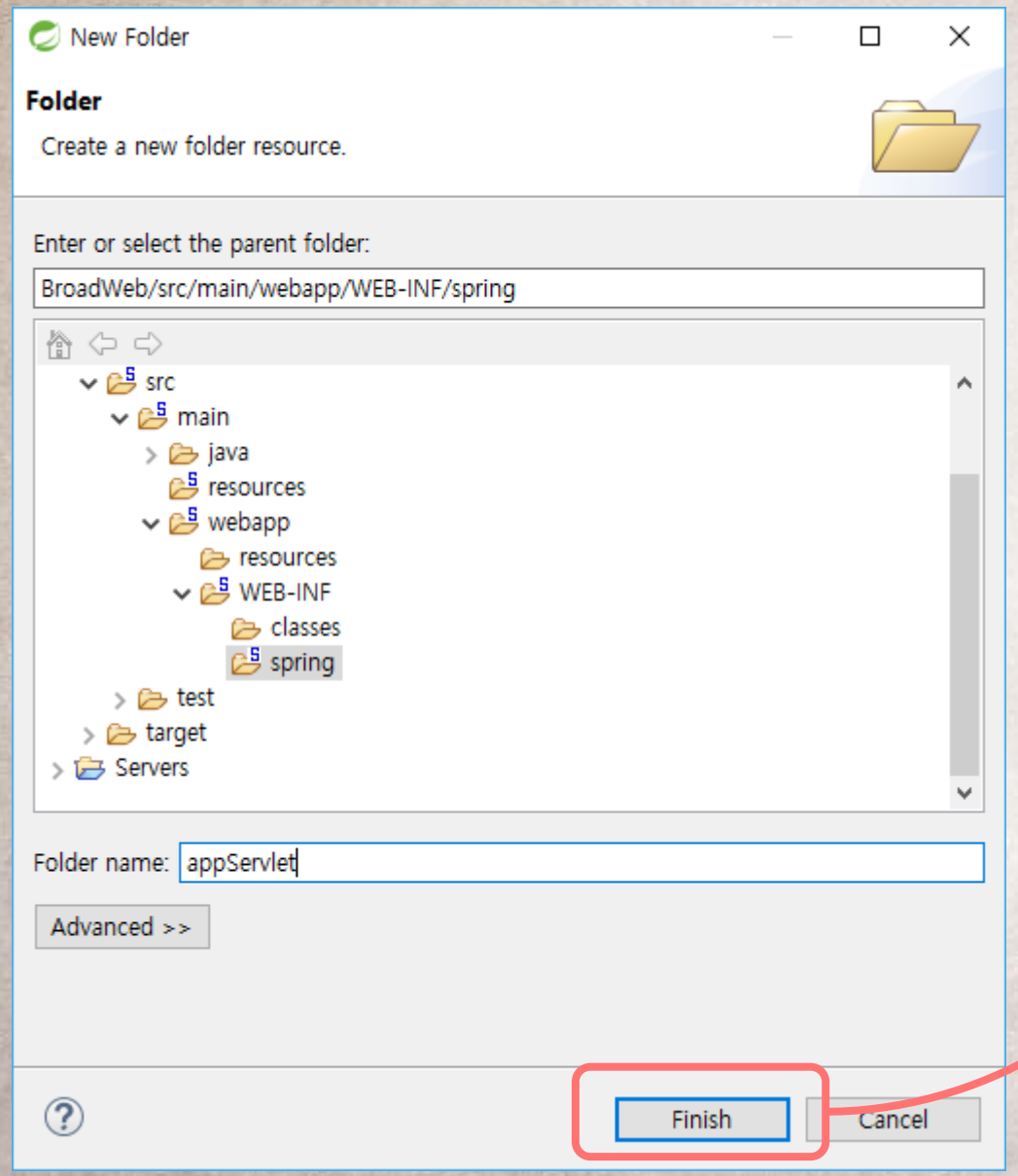
web.xml | servlet-context.xml | root-context.xml

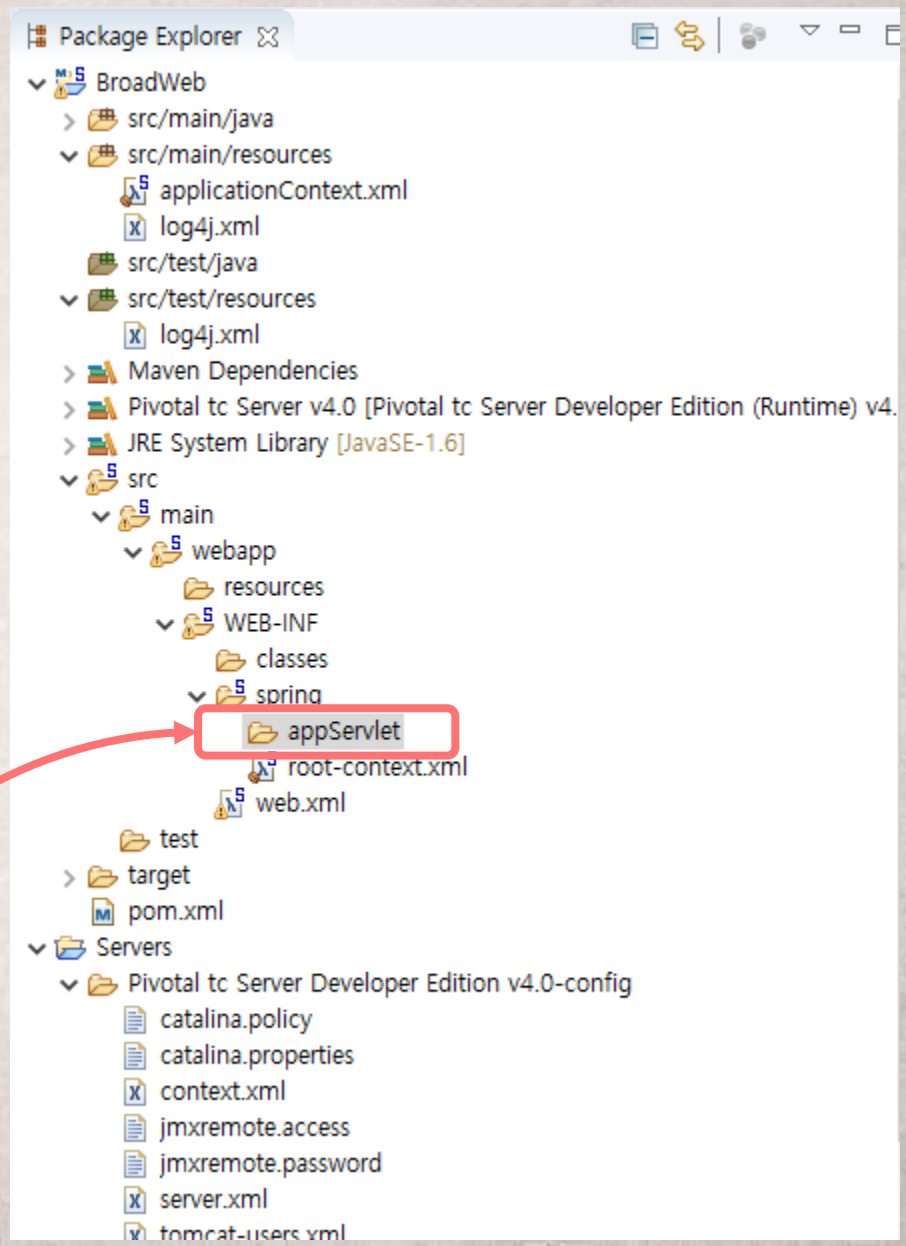
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:context="http://www.springframework.org/schema/context"
5      xmlns:mvc="http://www.springframework.org/schema/mvc"
6      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
11
12 </beans>
13
14
```

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

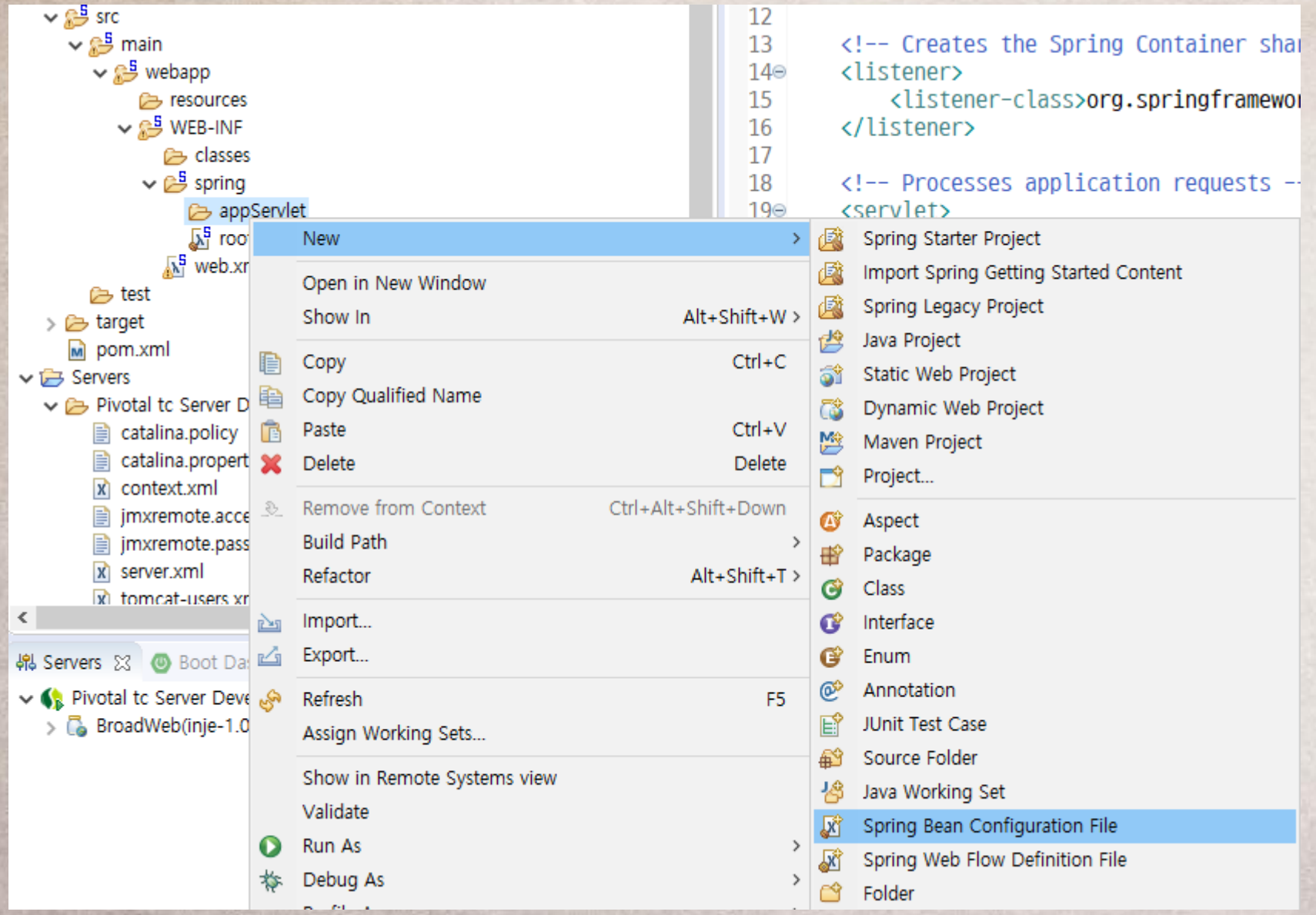


# /WEB-INF/spring/appServlet/servlet-context.xml

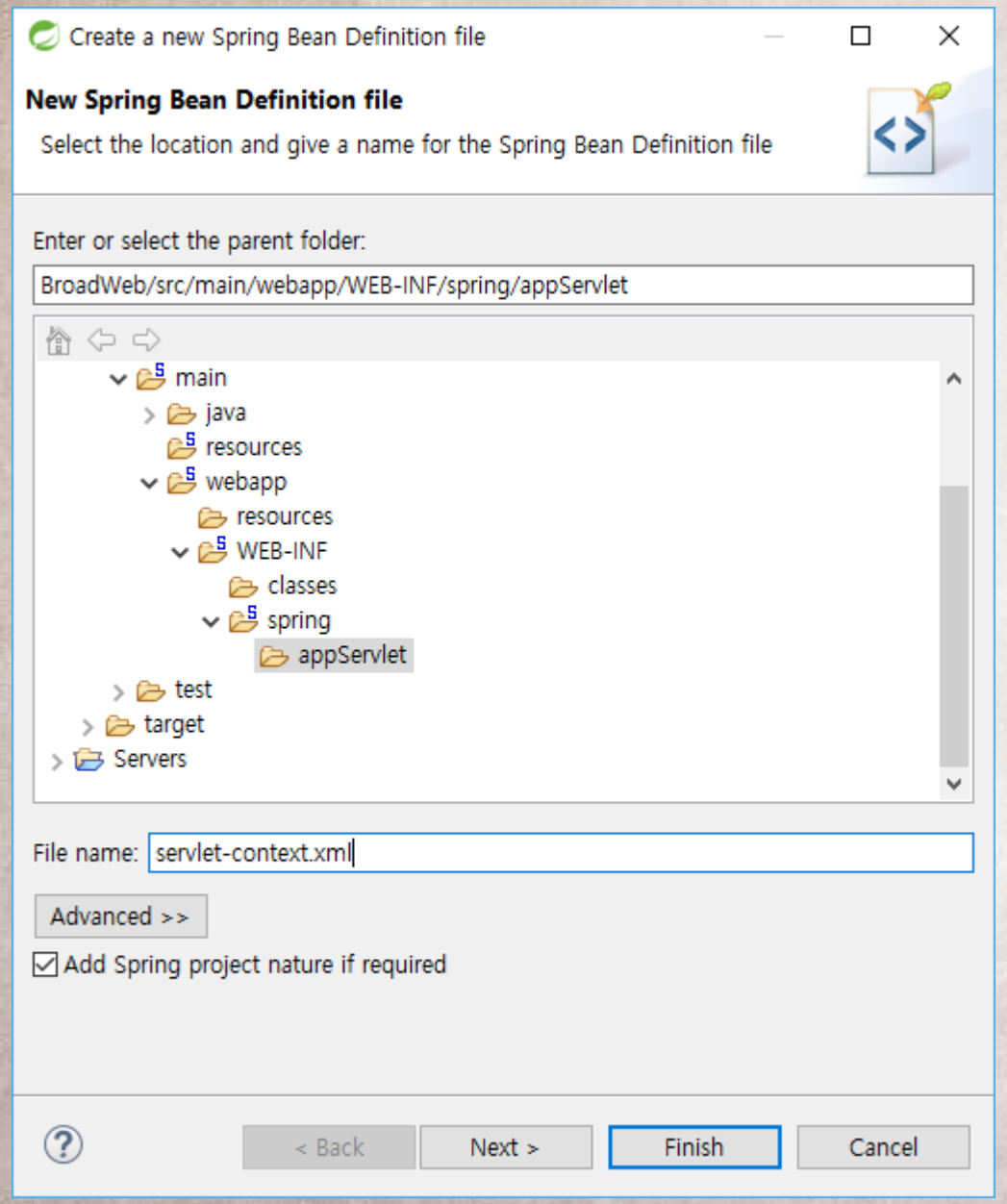


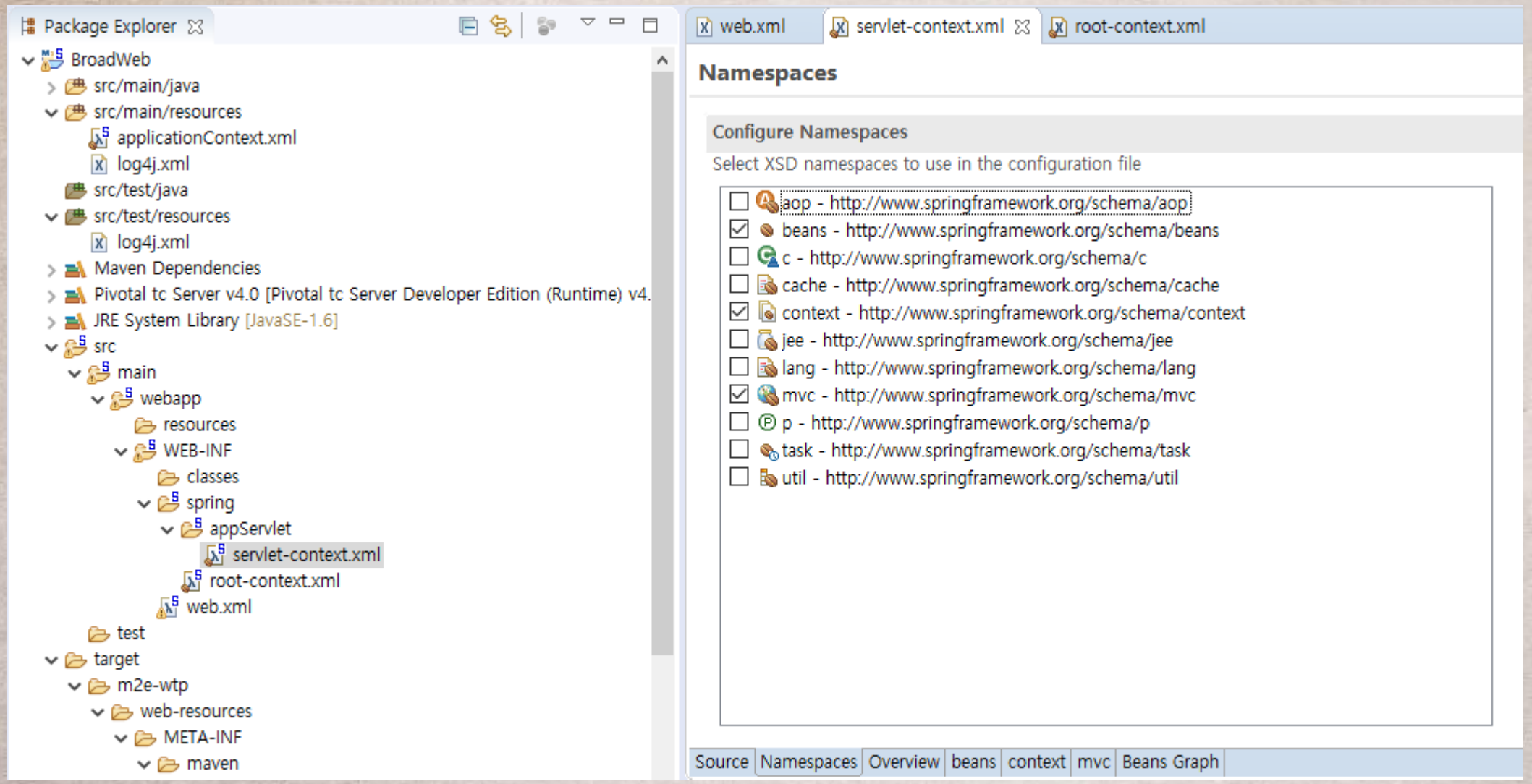
# /WEB-INF/spring/appServlet/servlet-context.xml



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



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



# /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 사용가능하도록 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

web.xml | servlet-context.xml | root-context.xml | index.html | Insert title here | BroadWeb/pom.xml

```
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>
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>
```

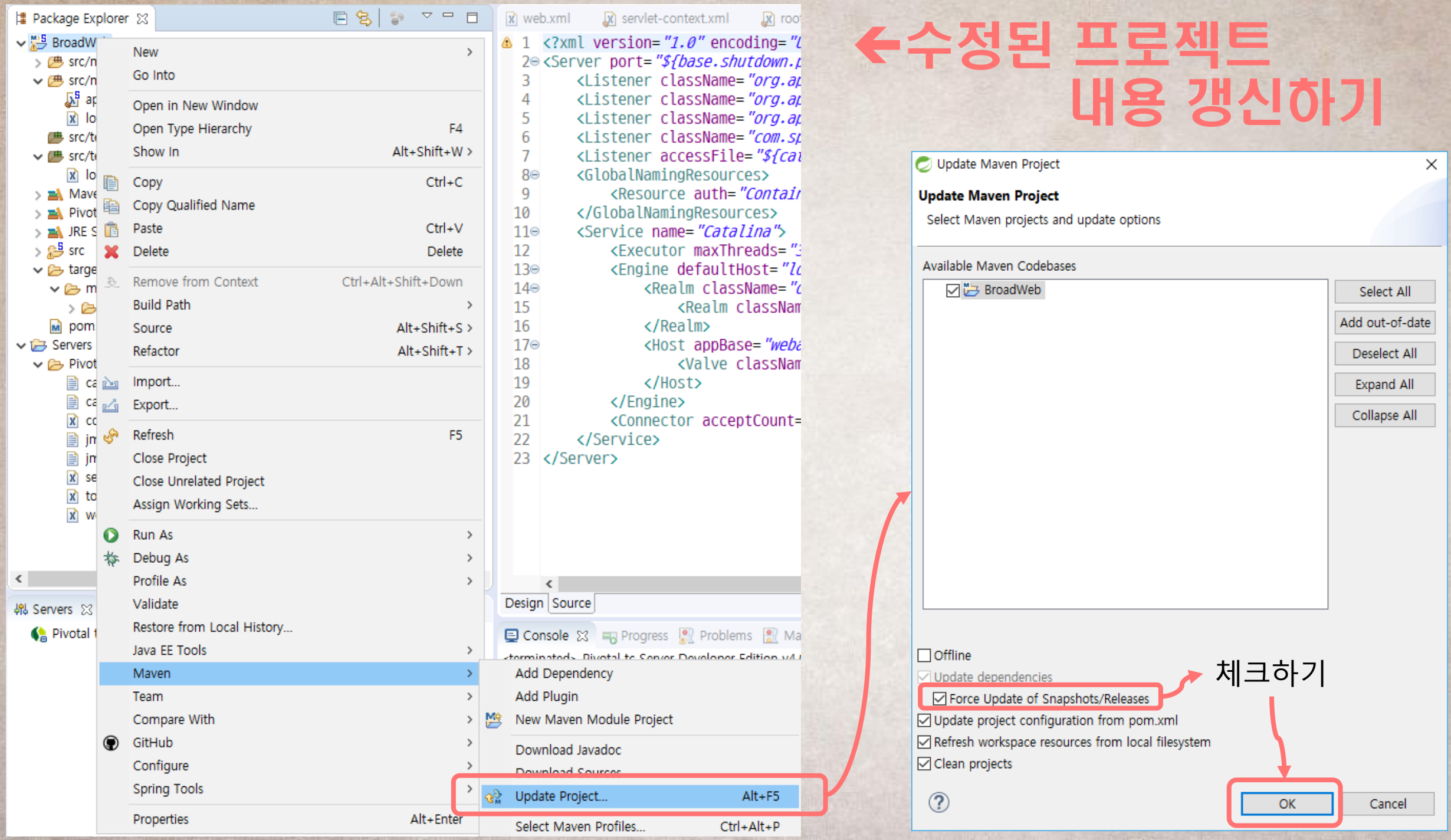
<artifactId> 태그 내의 값을
"Board"로 변경

web.xml | servlet-context.xml | root-context.xml | index.html | Insert title here

http://localhost:8080/Board/

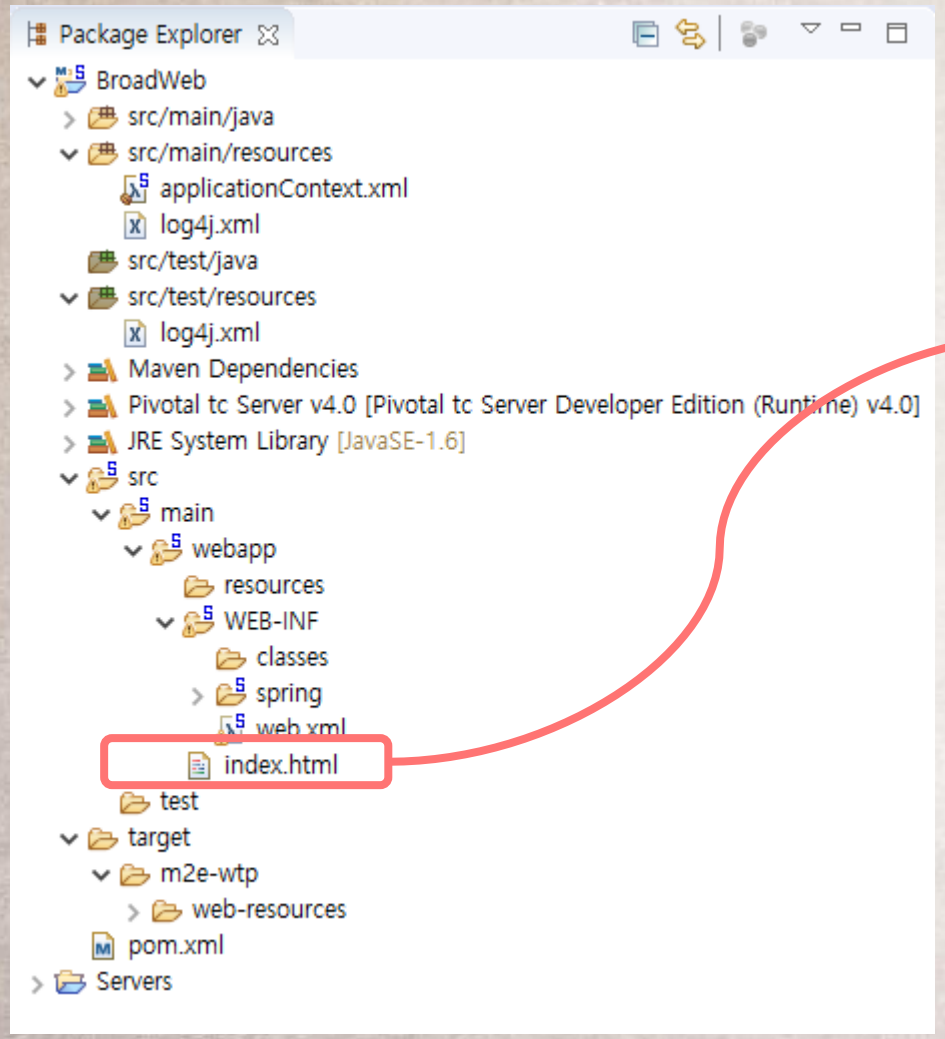
Hello World~2

Board가 프로젝트
폴더로 추가됨

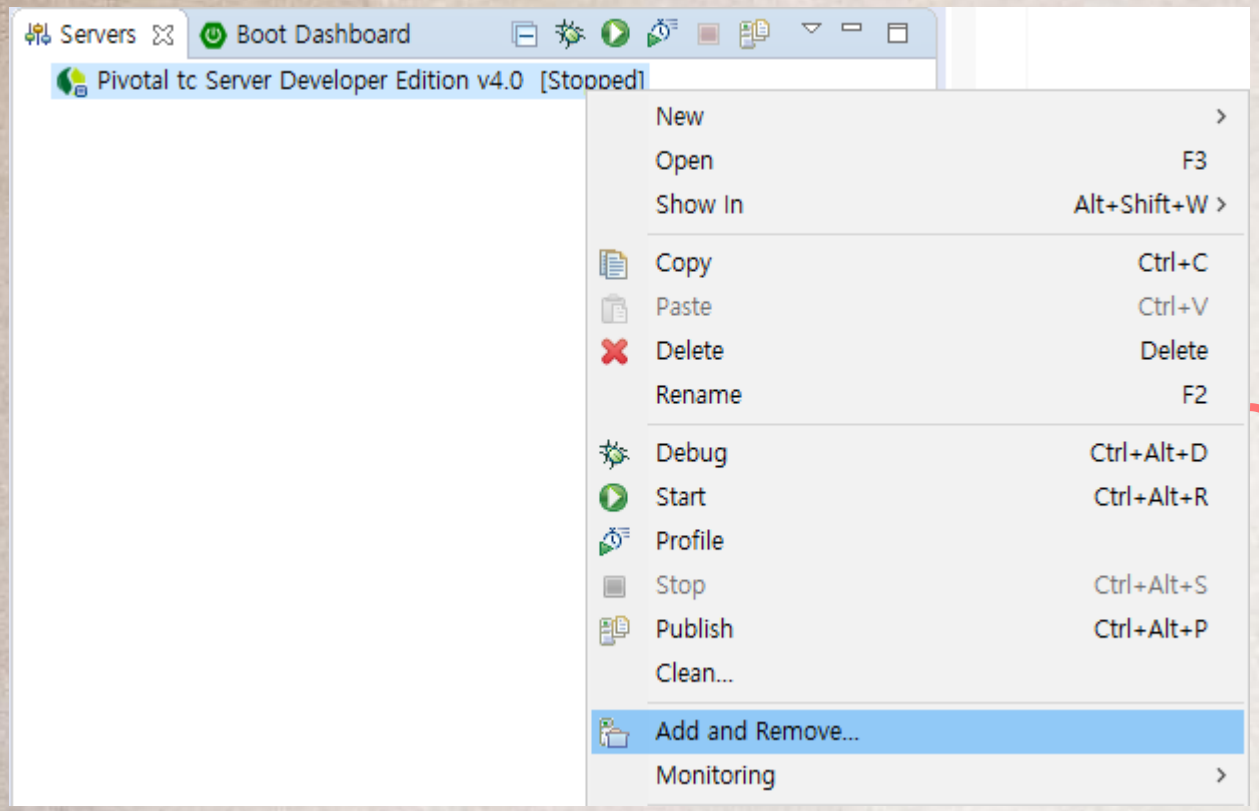

수정된 프로젝트
내용 갱신하기
Package Explorer
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Project
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Restore from Local History...
Java EE Tools
Maven
Team
Compare With
GitHub
Configure
Spring Tools
Properties Alt+Enter
Add Dependency
Add Plugin
New Maven Module Project
Download Javadoc
Update Project... Alt+F5
Select Maven Profiles... Ctrl+Alt+P
web.xml
servlet-context.xml
Design
Source
Console
Progress
Problems
Servers
Update Maven Project
Update Maven Project
Select Maven projects and update options
Available Maven Codebases
BroadWeb
Select All
Add out-of-date
Deselect All
Expand All
Collapse All
Offline
Update dependencies
Force Update of Snapshots/Releases
Update project configuration from pom.xml
Refresh workspace resources from local filesystem
Clean projects
체크하기
OK
Cancel

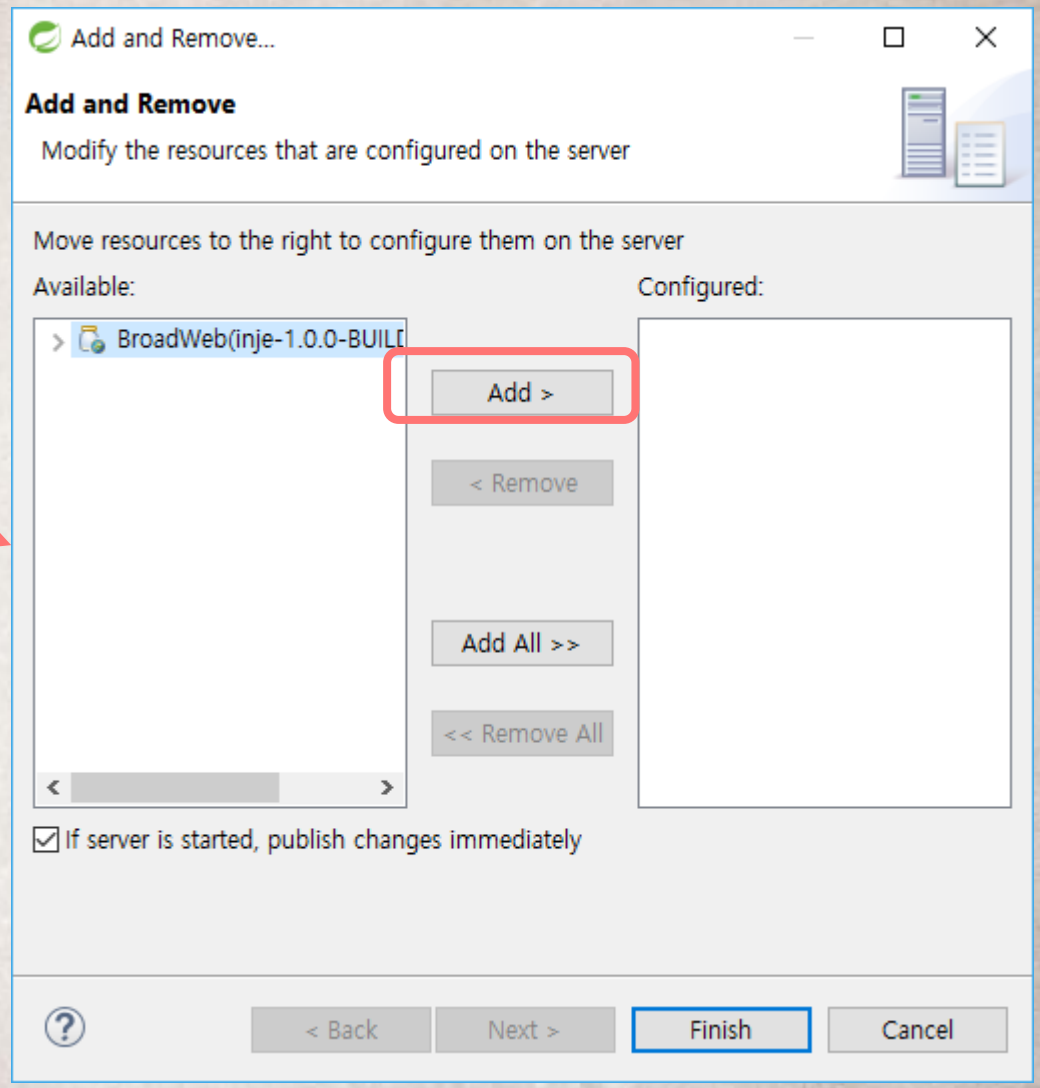
# html(JSP) 페이지 추가와 서버 구동

# webapp폴더에 index.html 파일 추가

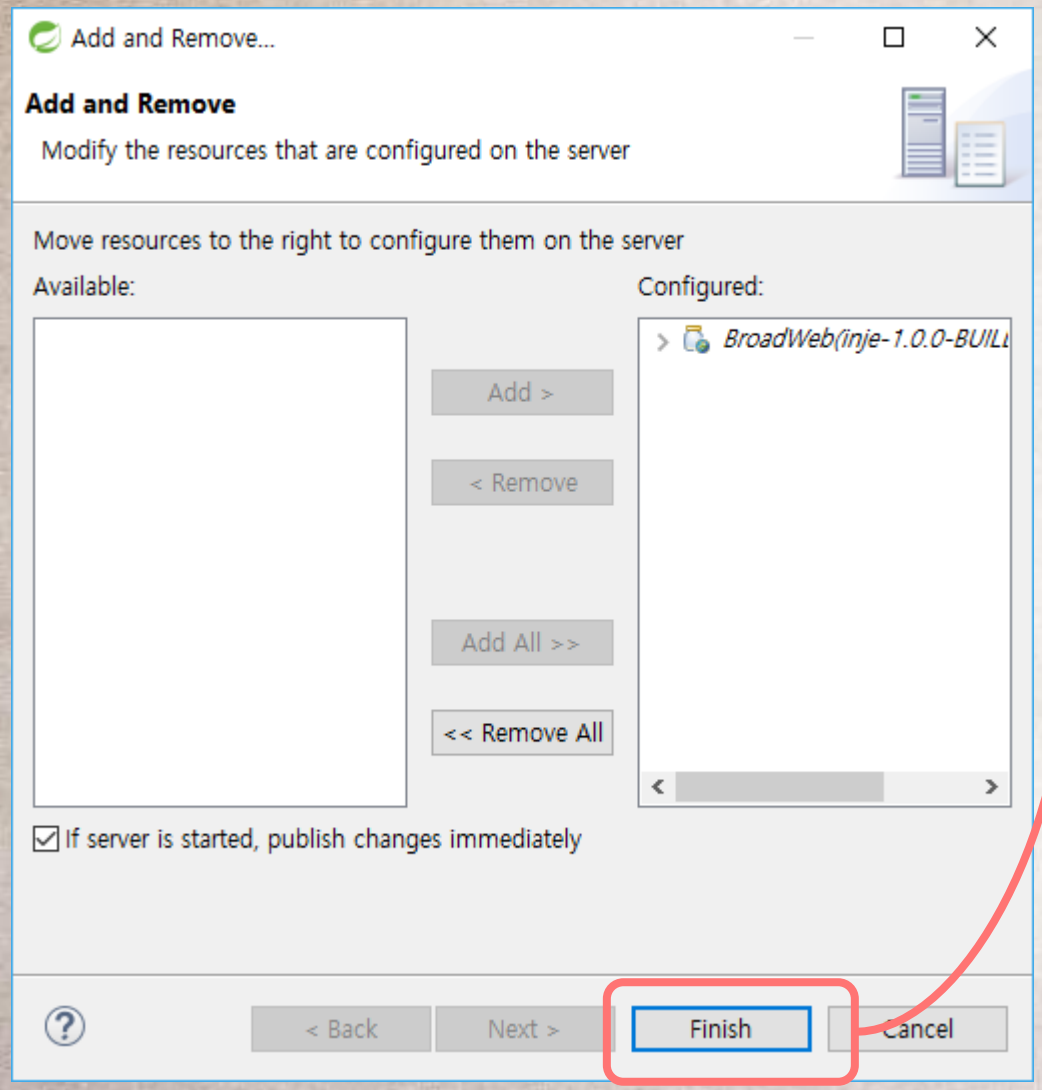


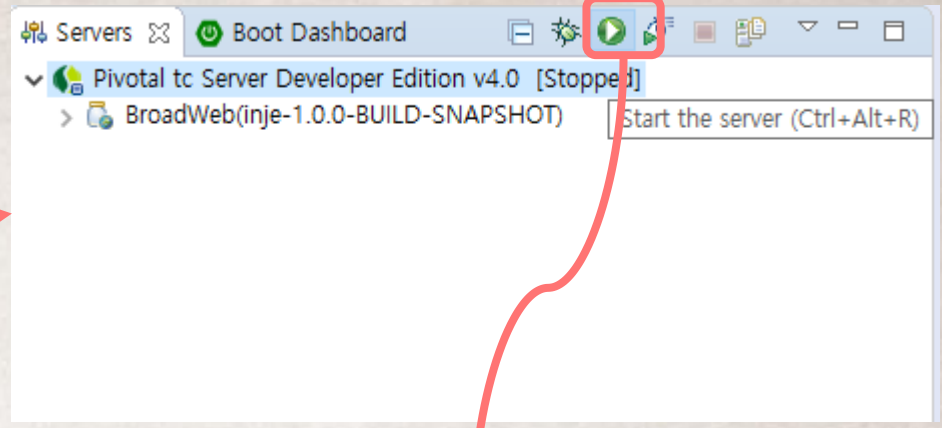

# 웹 서버 설정하기

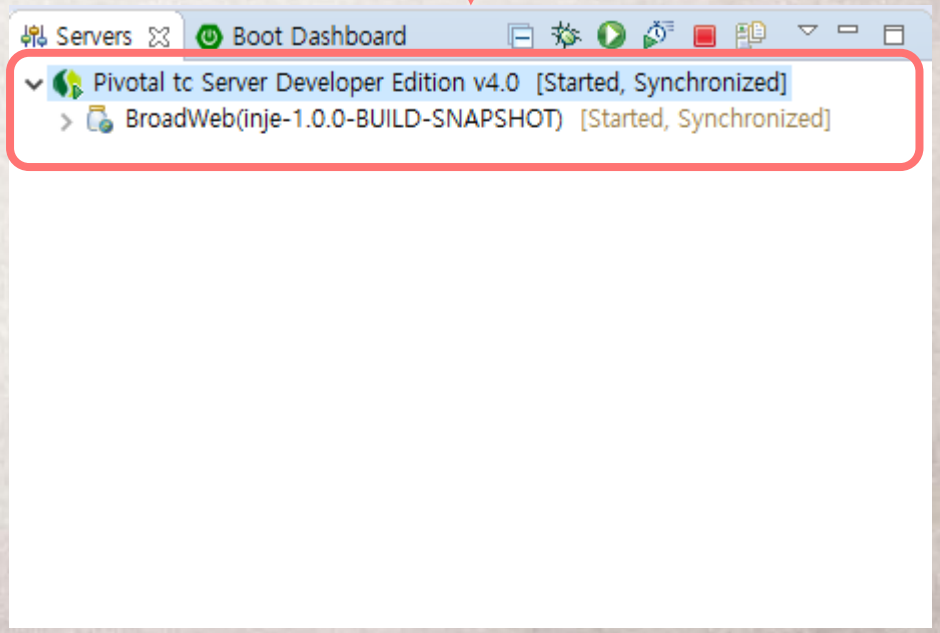

Servers
Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Stopped]
New
Open F3
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Rename F2
Debug Ctrl+Alt+D
Start Ctrl+Alt+R
Profile
Stop Ctrl+Alt+S
Publish Ctrl+Alt+P
Clean...
Add and Remove...
Monitoring

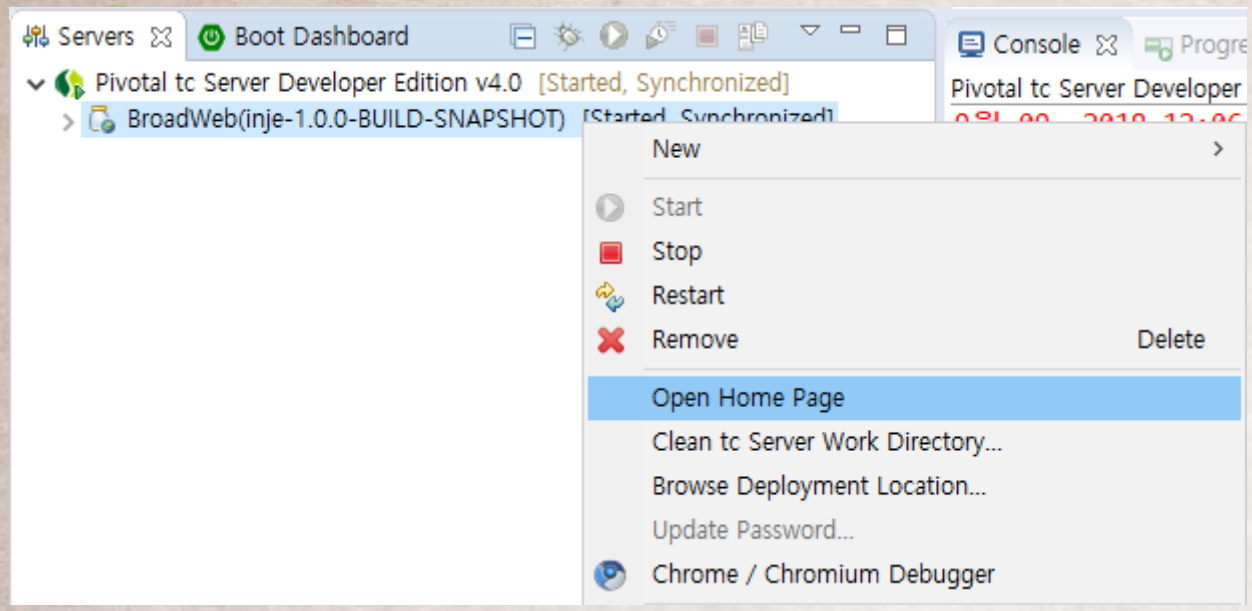

Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
BroadWeb(inje-1.0.0-BUILD
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Finish
Cancel

# 웹 서버 설정하기

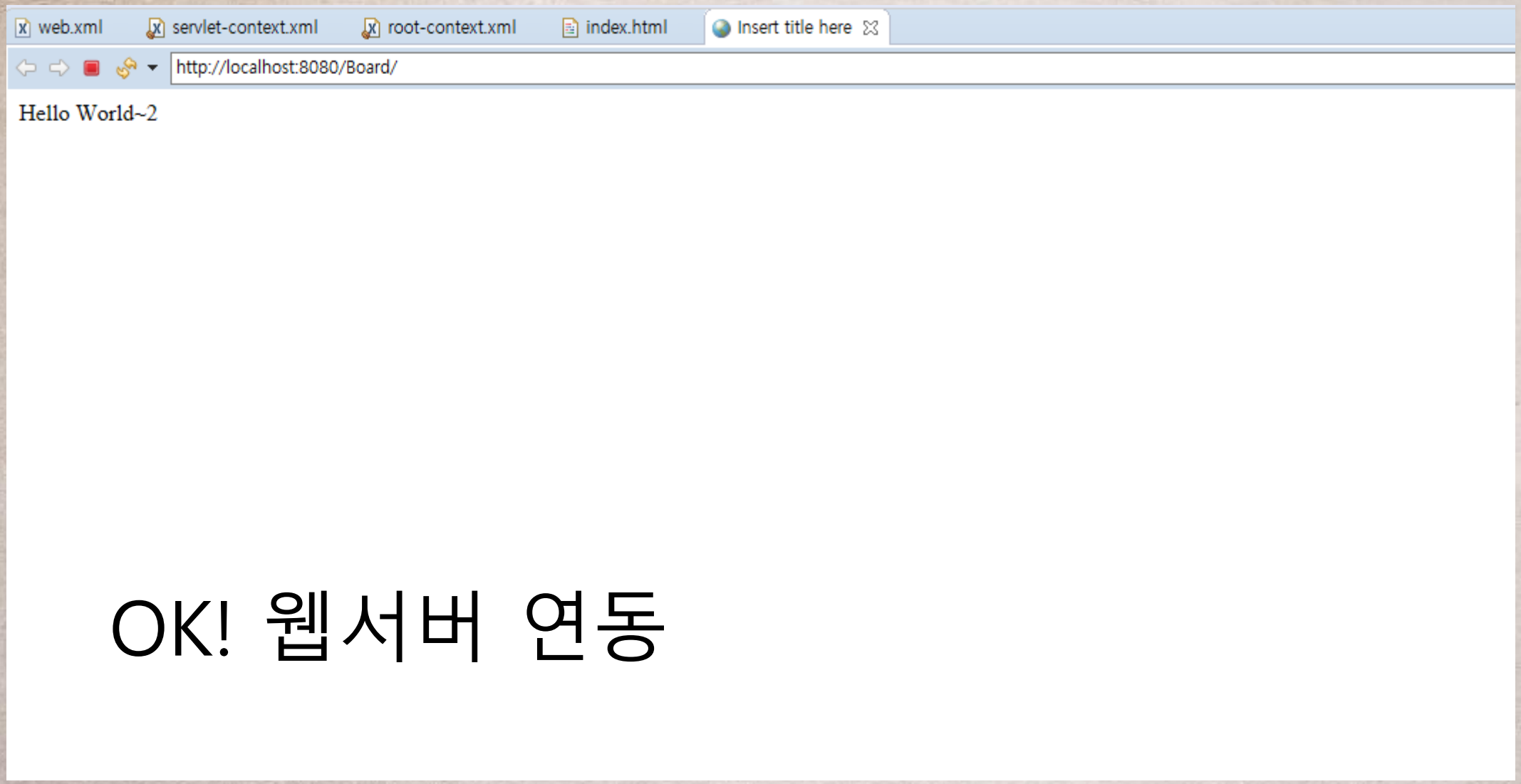

Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
BroadWeb(inje-1.0.0-BUILD
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Finish
Cancel


Servers
Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Stopped]
BroadWeb(inje-1.0.0-BUILD-SNAPSHOT)
Start the server (Ctrl+Alt+R)


Servers
Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Started, Synchronized]
BroadWeb(inje-1.0.0-BUILD-SNAPSHOT) [Started, Synchronized]


Servers
Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Started, Synchronized]
BroadWeb(inje-1.0.0-BUILD-SNAPSHOT)
Console
Pivotal tc Server Developer
New
Start
Stop
Restart
Remove
Delete
Open Home Page
Clean tc Server Work Directory...
Browse Deployment Location...
Update Password...
Chrome / Chromium Debugger

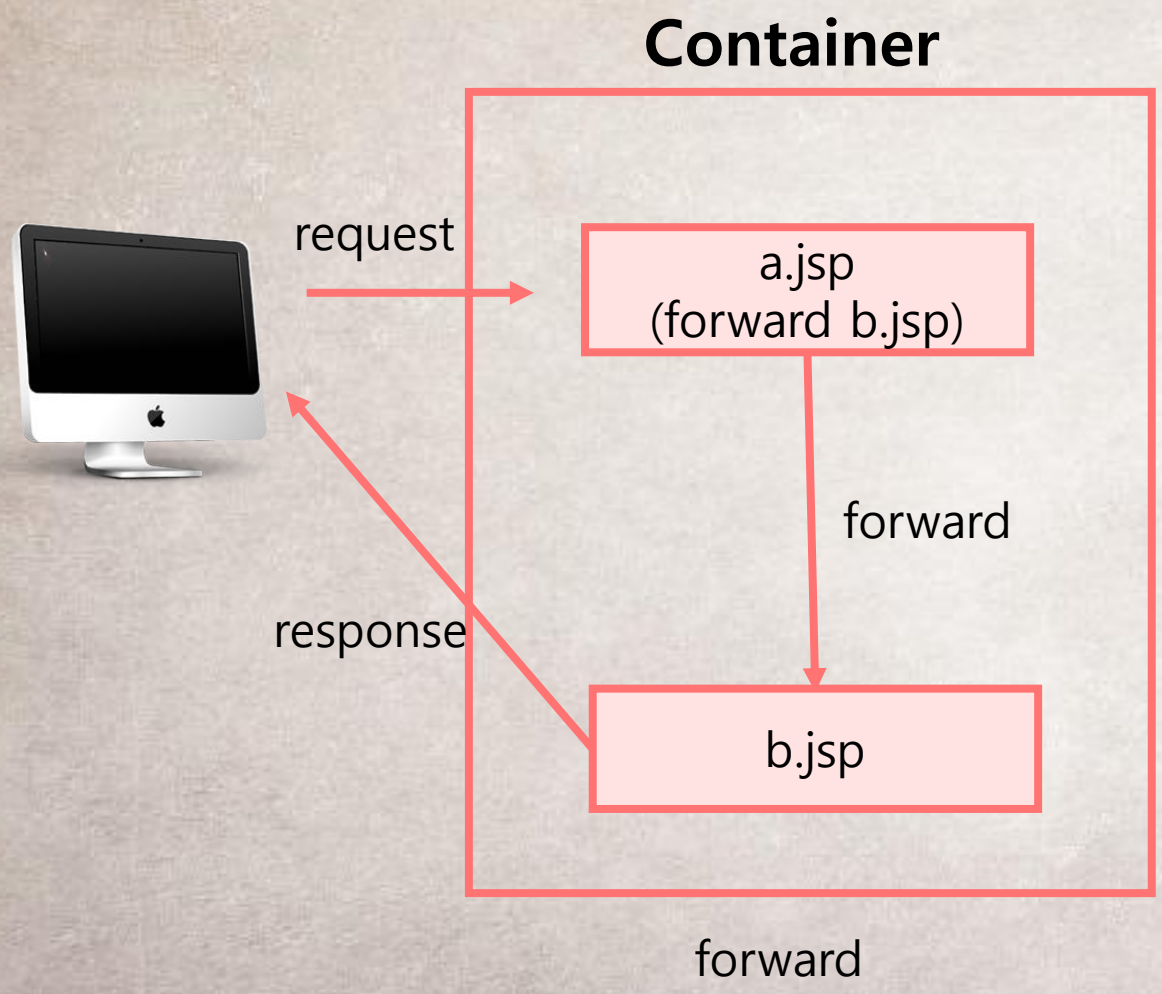
# 웹서버에서 전송된 기본 html 파일-index.html

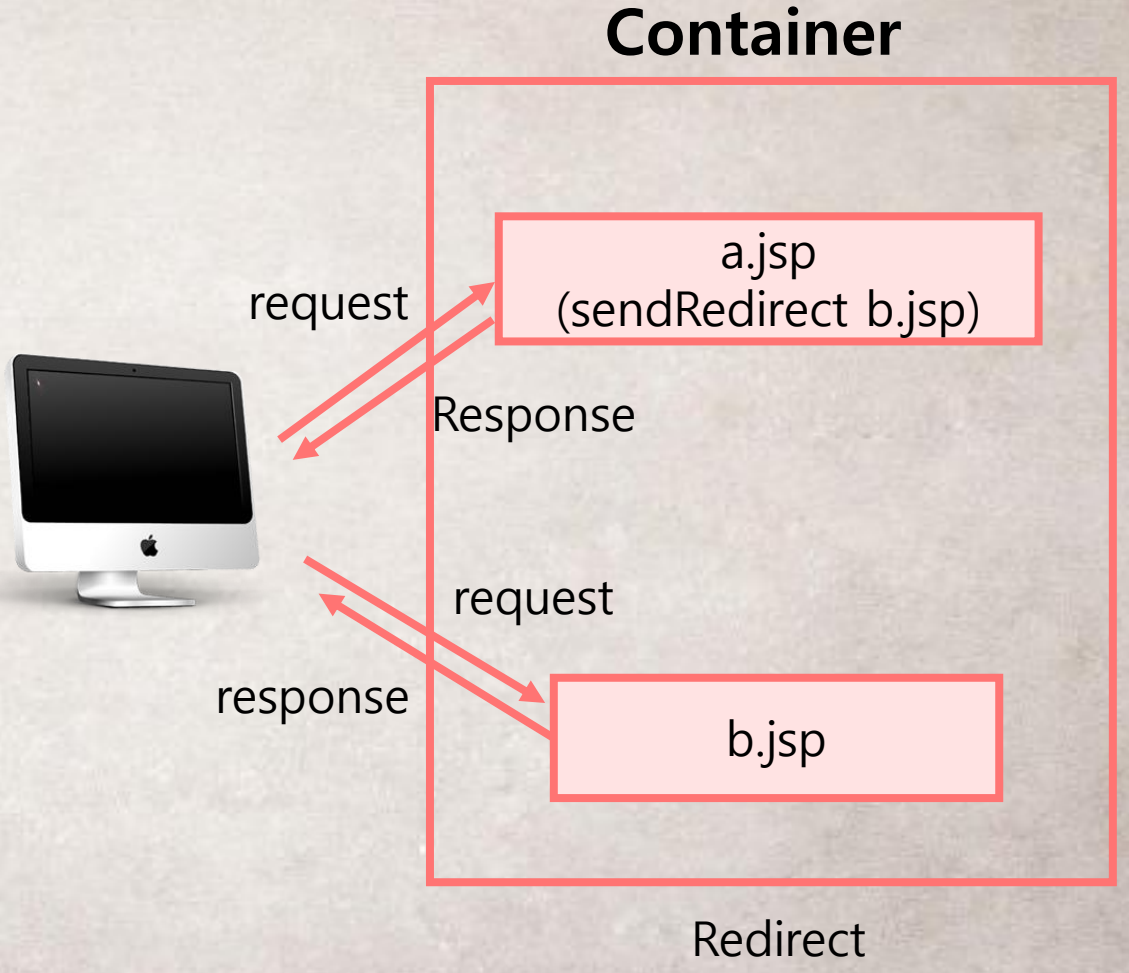



OK! 웹서버 연동

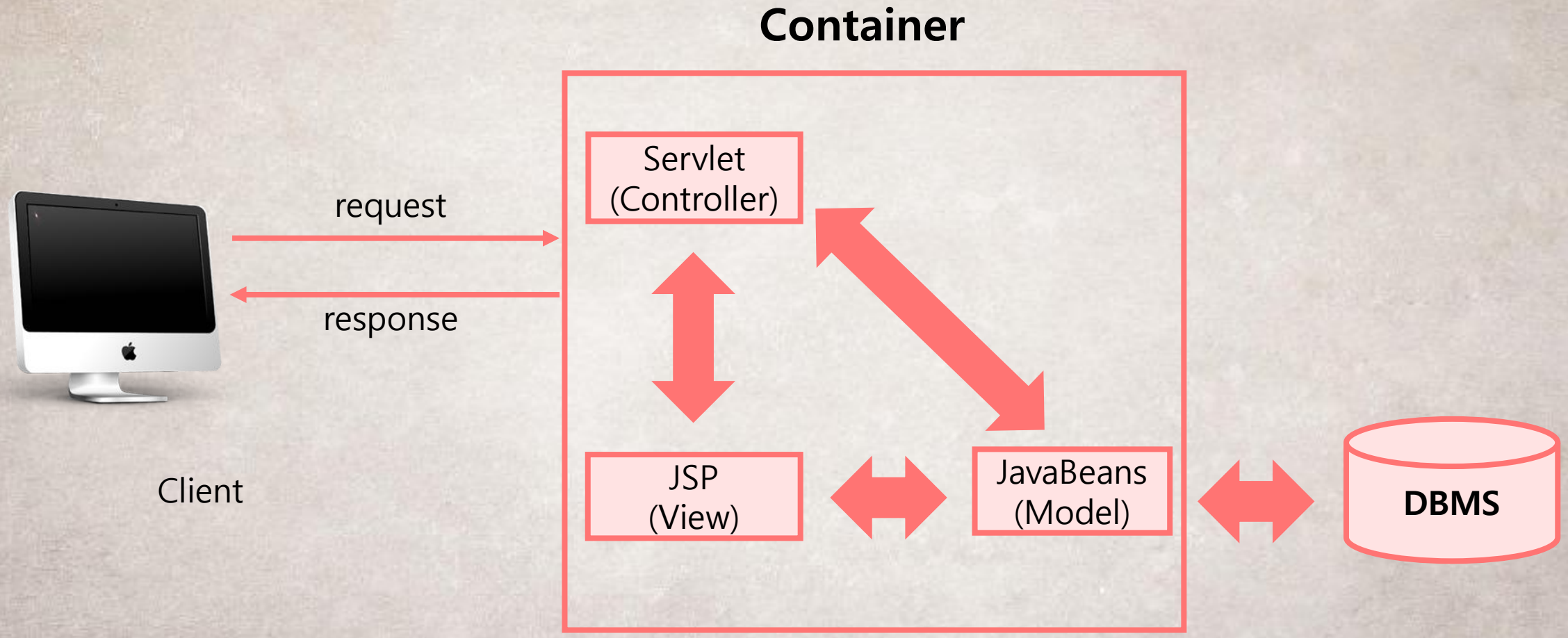
# 게시판 모델2 작업

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



# Model2 아키텍처


Container
Servlet
(Controller)
request
response
JSP
(View)
JavaBeans
(Model)
DBMS
Client

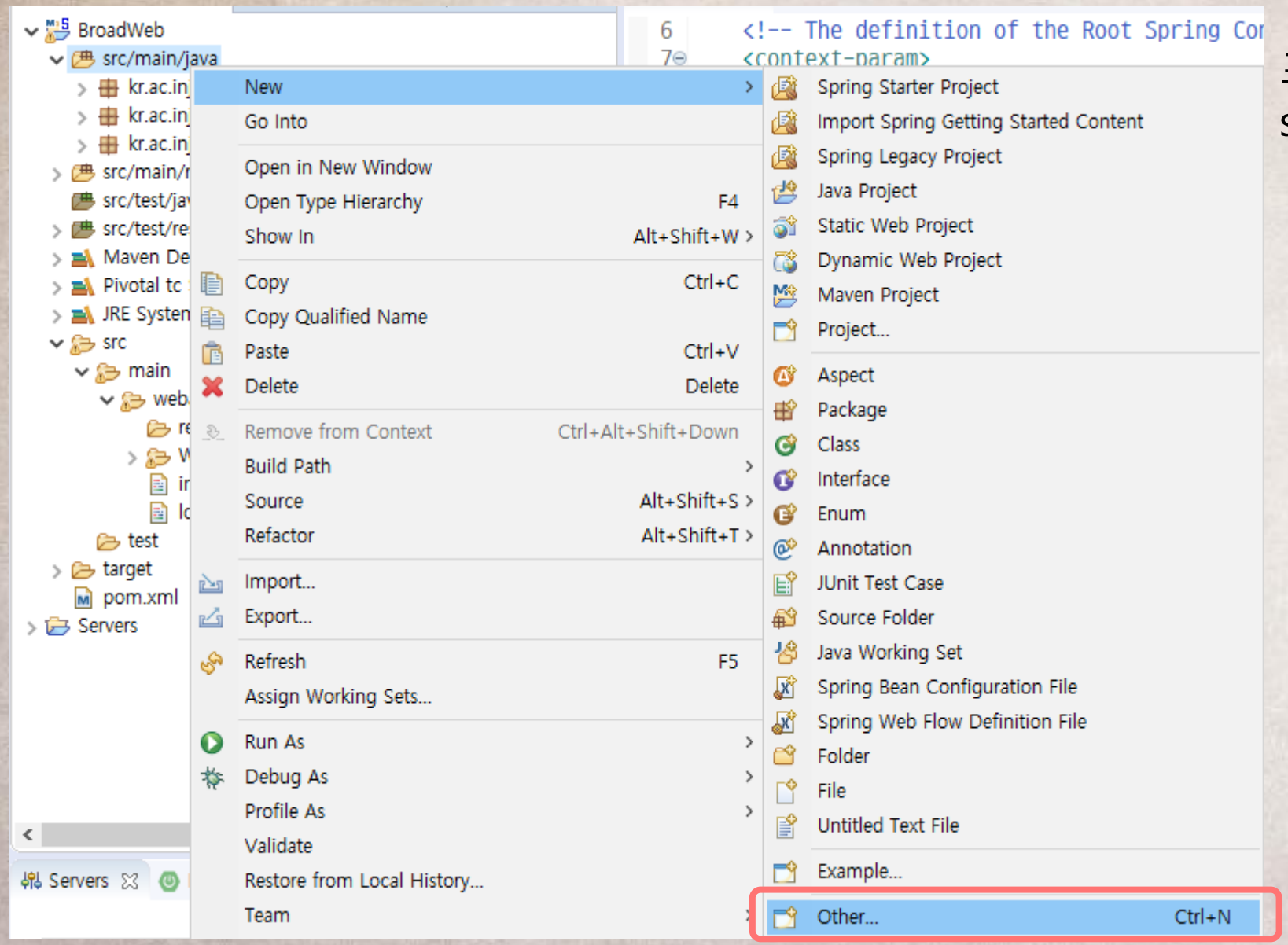
# web.xml 수정

login.jsp | web.xml

```
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>
```

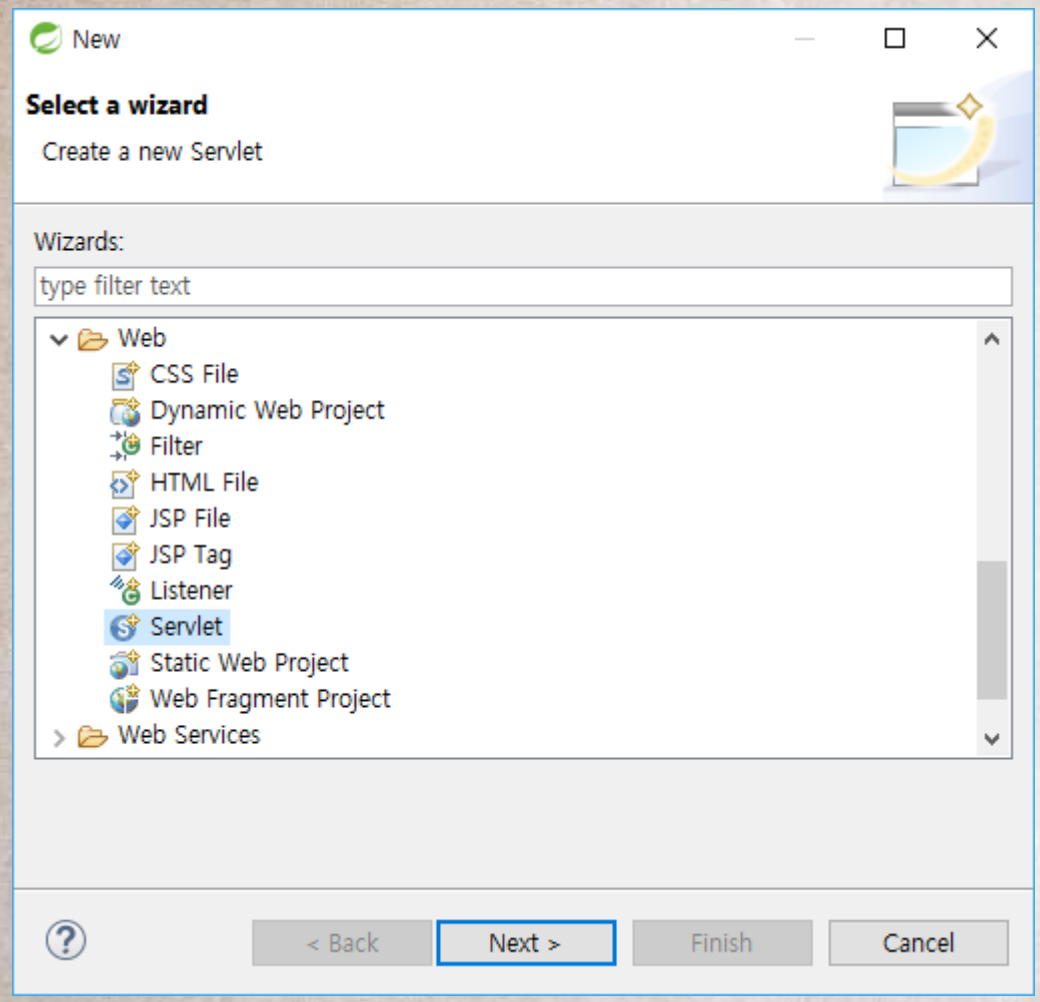
주석처리

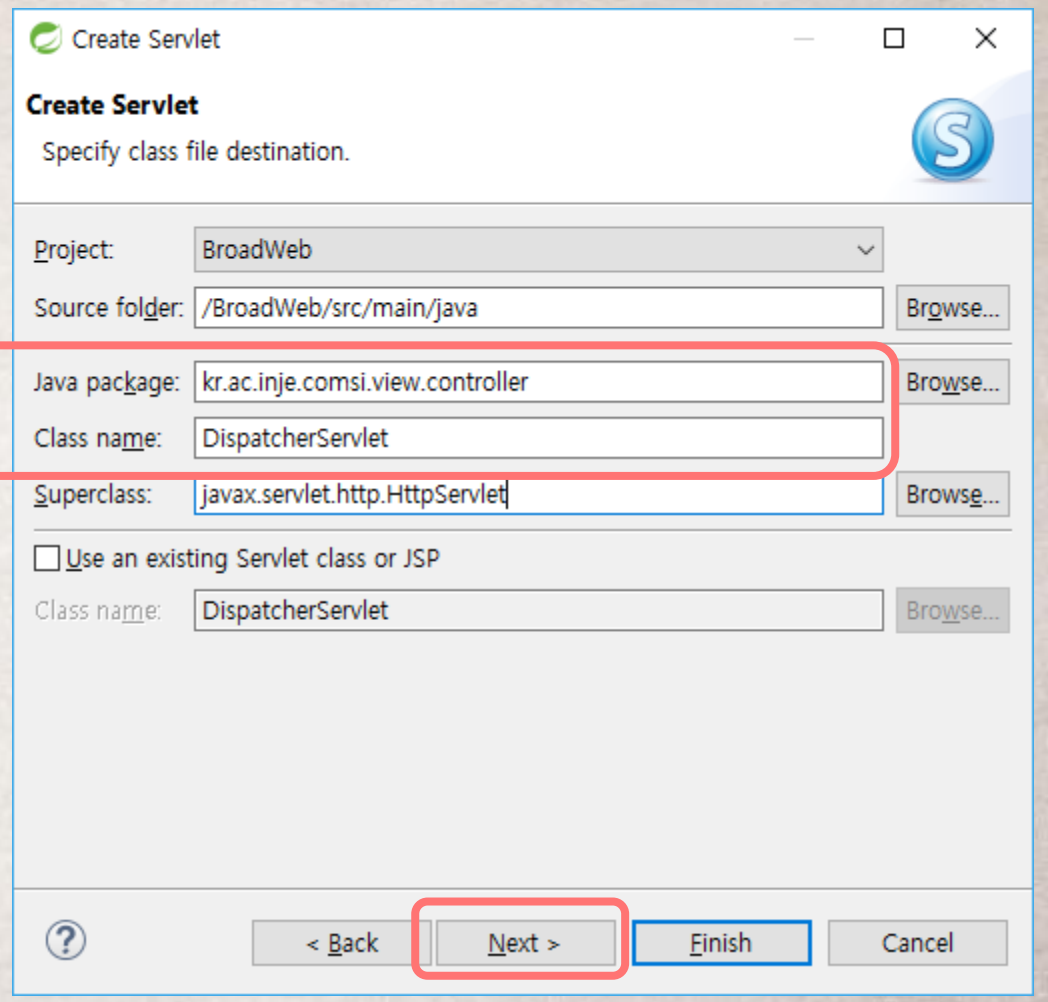
# Controller 구현 – DispatcherServlet




프로젝트 탐색창에서
src/main/java 폴더에서

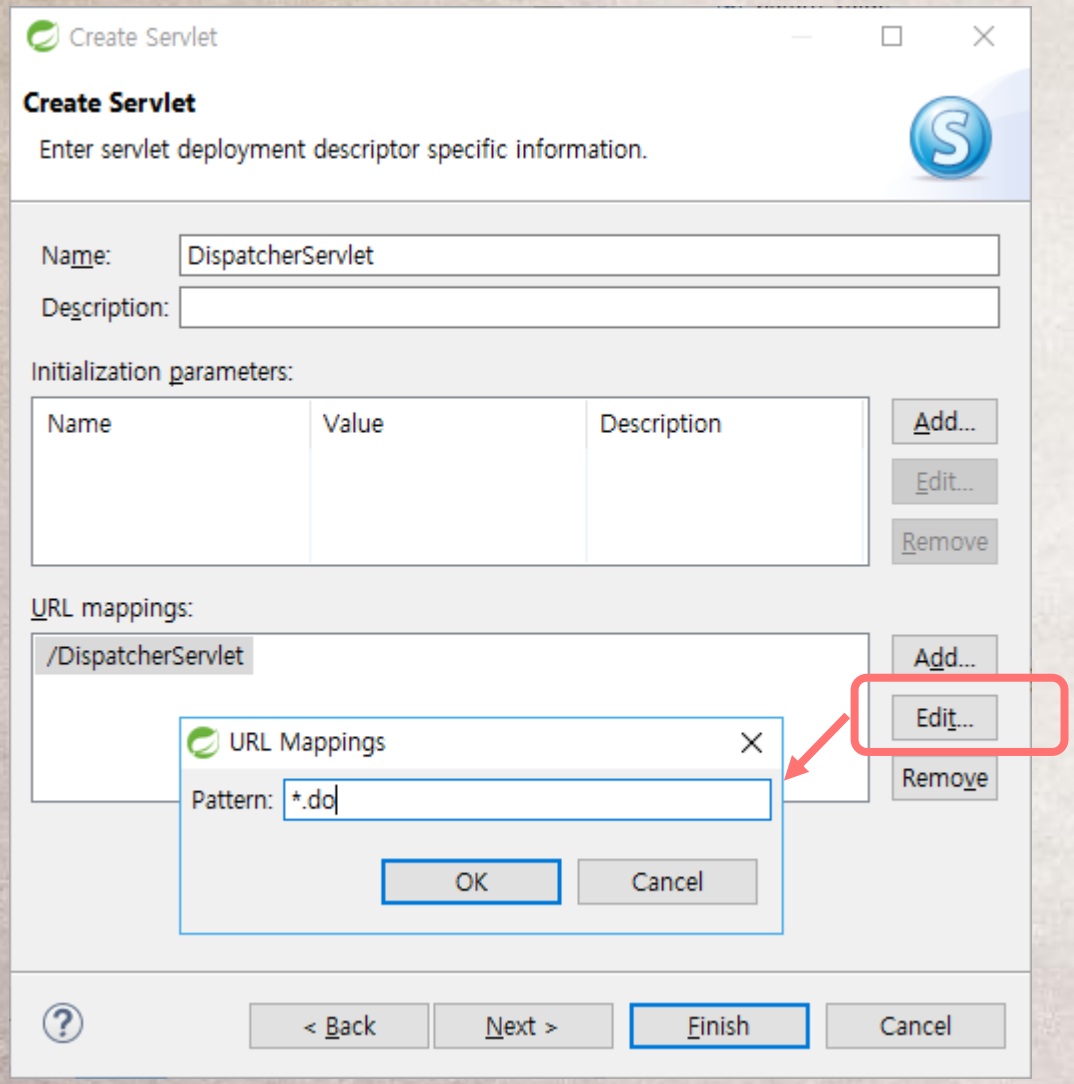
# Controller 구현 – DispatcherServlet 클래스

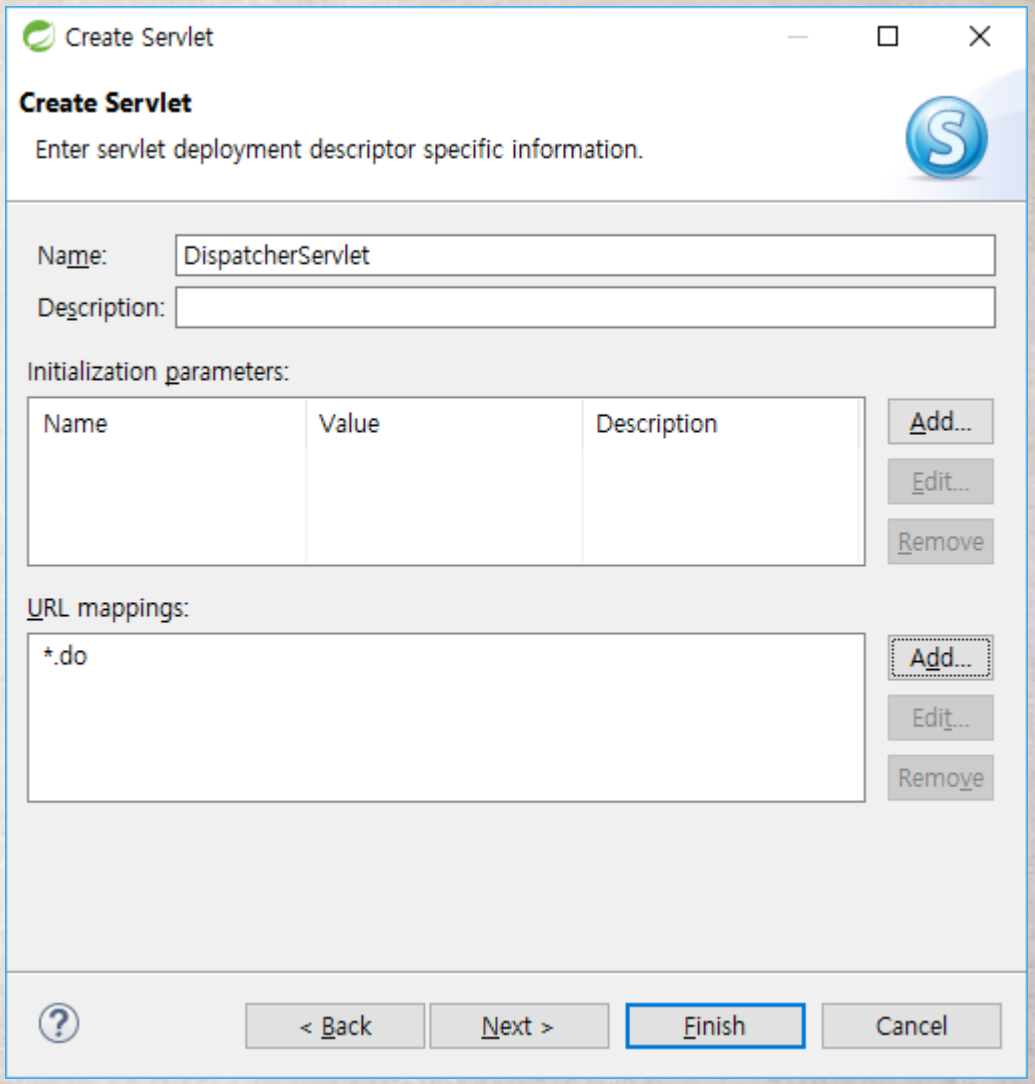



Java package – kr.ac.inje.comsi.view.controller
Class name - DispatcherServlet

# 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

# DispatcherServlet객체 생성 및 활성화

# 생성된 DispatcherServlet 클래스

login.jsp | web.xml | DispatcherServlet.java

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

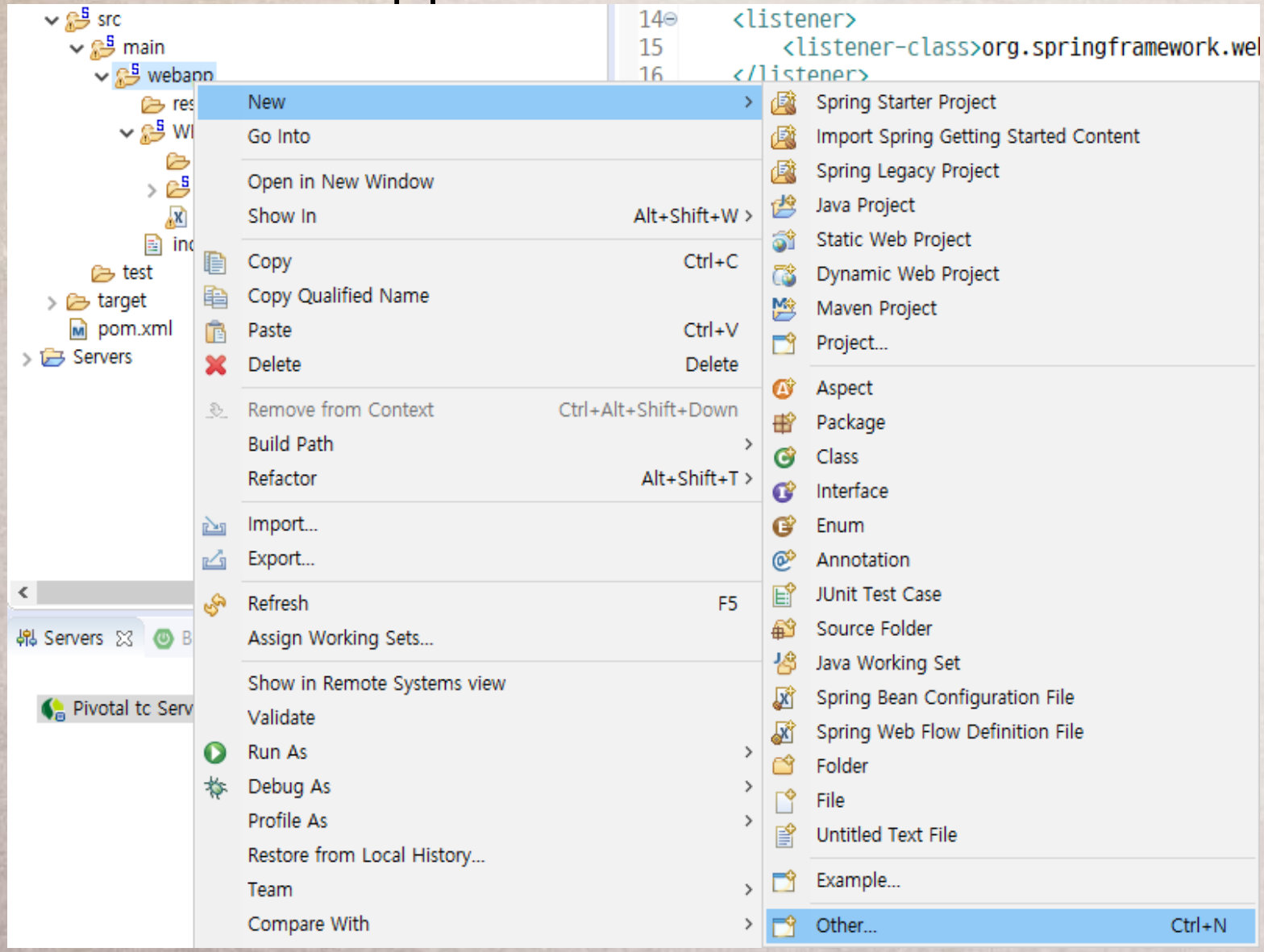
```
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Closing Root WebApplicationConte
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization started
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Refreshing 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

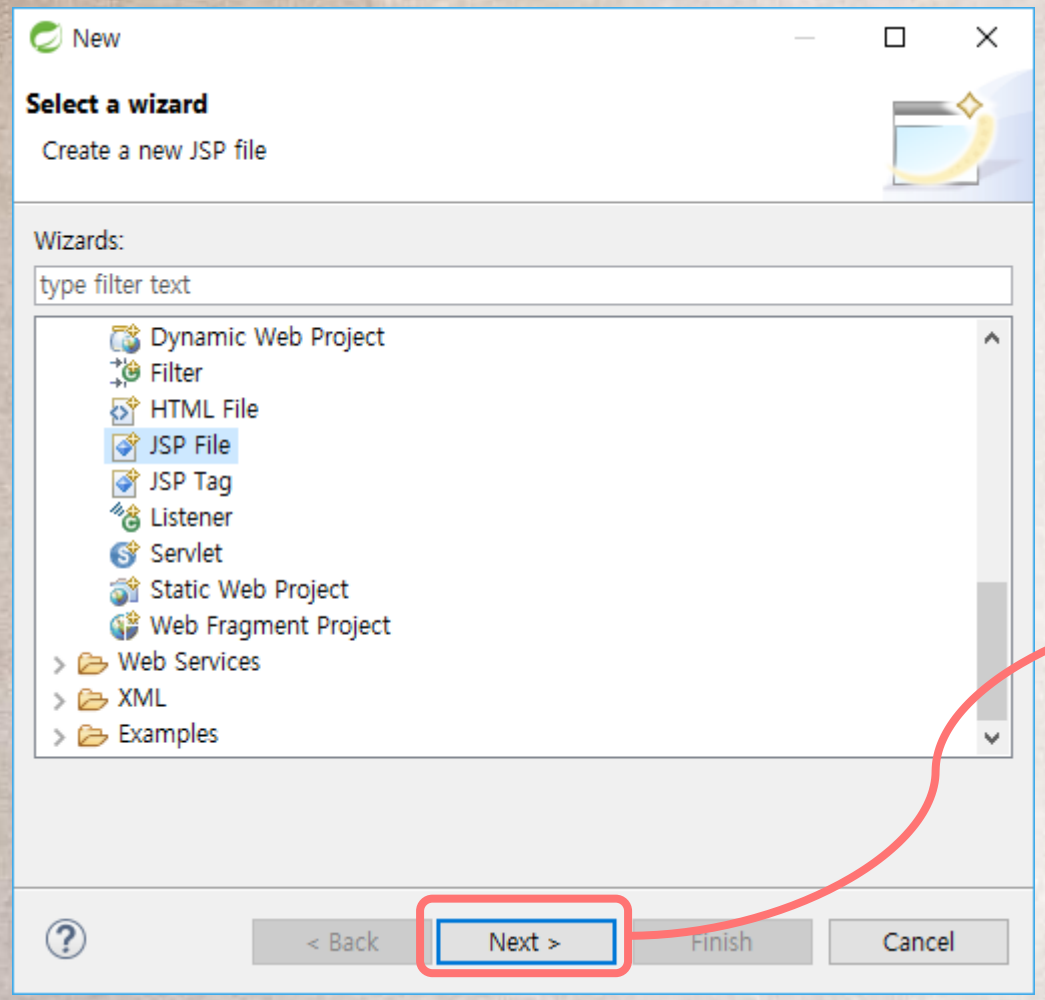
# 로그인 기능 구현

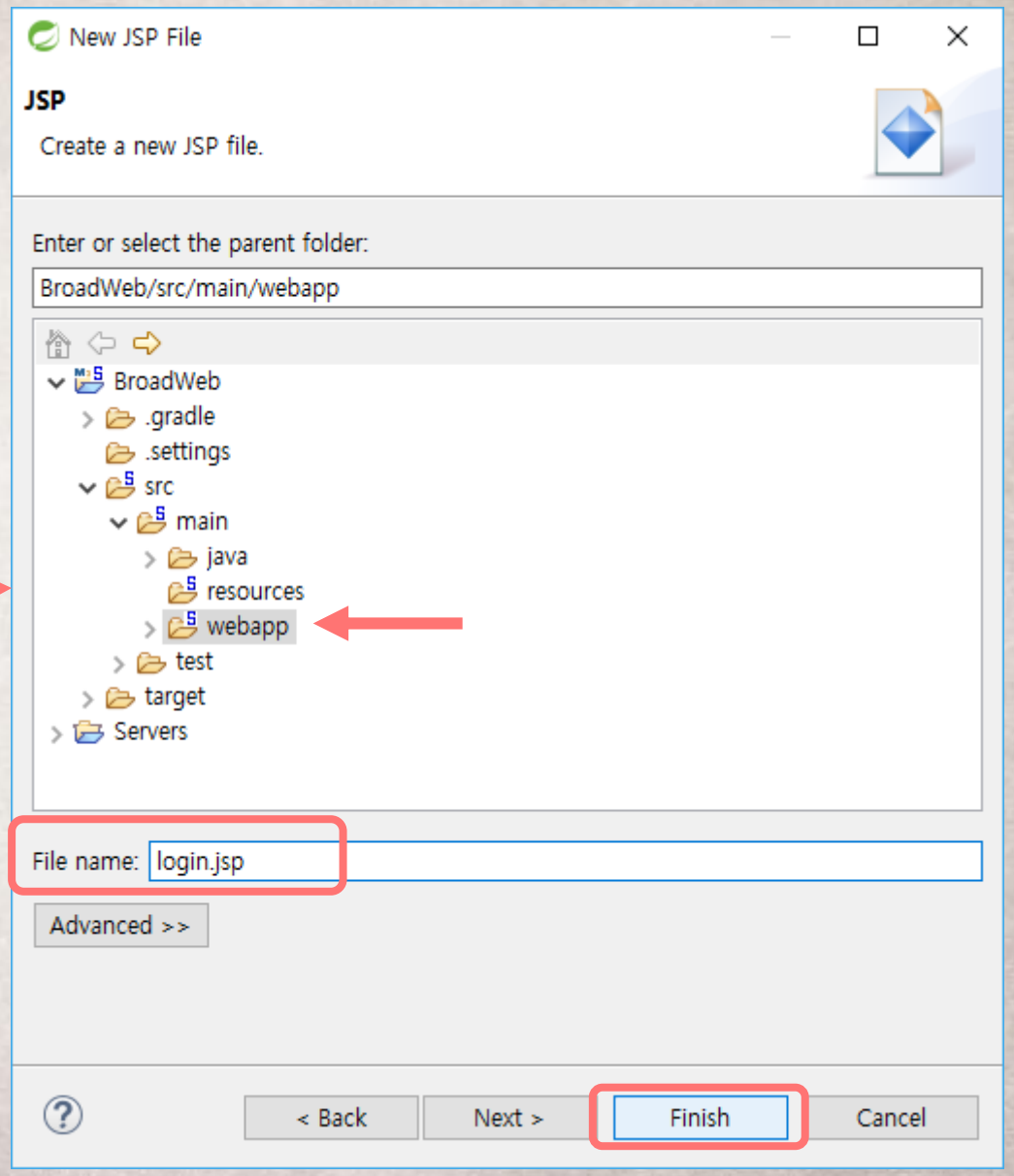
# login.jsp – JSP 파일(View 구현)

- src/main/webapp 폴더에 등록해야 함.



# 이어서…


New
Select a wizard
Create a new JSP file
Wizards:
type filter text
Dynamic Web Project
Filter
HTML File
JSP File
JSP Tag
Listener
Servlet
Static Web Project
Web Fragment Project
Web Services
XML
Examples
< Back
Next >
Finish
Cancel


New JSP File
JSP
Create a new JSP file.
Enter or select the parent folder:
BroadWeb/src/main/webapp
BroadWeb
.gradle
.settings
src
main
java
resources
webapp
test
target
Servers
File name: login.jsp
Advanced >>
< Back
Next >
Finish
Cancel

# 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">
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>
```

action 내용이 "login.do" 호출

# http://localhost:8080/Board/login.jsp


➔ 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

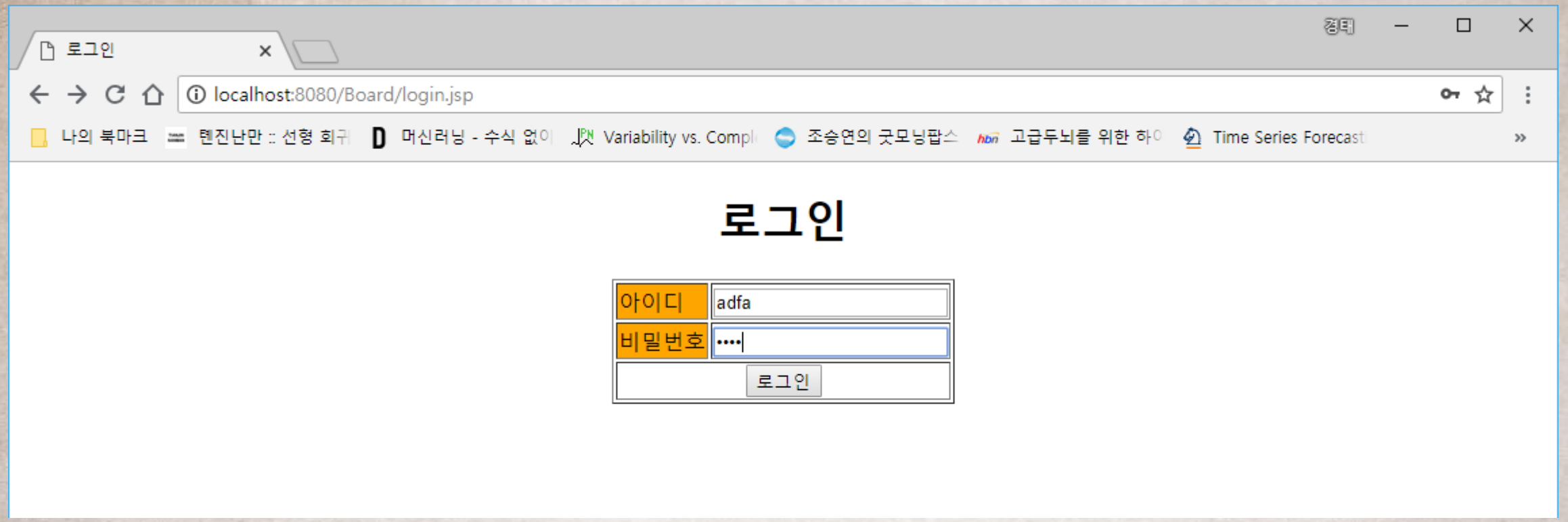
login.jsp | web.xml | DispatcherServlet.java | getBoardList.jsp

```jsp
<%@ page import="java.util.List" %>
<%@ page import="kr.ac.inje.comsi.board.BoardVO" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // 세션에 저장된 글 목록을 꺼낸다.
    List<BoardVO> boardList = (List<BoardVO>) session.getAttribute("boardList");
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>글 목록</title>
</head>
<body>
<center>
    <h1>글 목록</h1>
    <h3>환영합니다. <a href="logout.do">Logout</a></h3>

    <!-- 검색시작 -->
    <form action="getBoardList.jsp" method="post">
    <table border="1" cellpadding="0" cellspacing="0" width="700">
    <tr>
        <td align="right">
            <select name="searchCondition">
                <option value="TITLE">제목</option>
                <option value="CONTENT">내용</option>
            </select>
            <input name="searchKeyword" type="text"/>
            <input type="submit" value="검색"/>
        </td>
    </tr>
    </table>
    </form>
    <!-- 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.do">새글 등록</a>
58  </center>
59  </body>
60  </html>
```

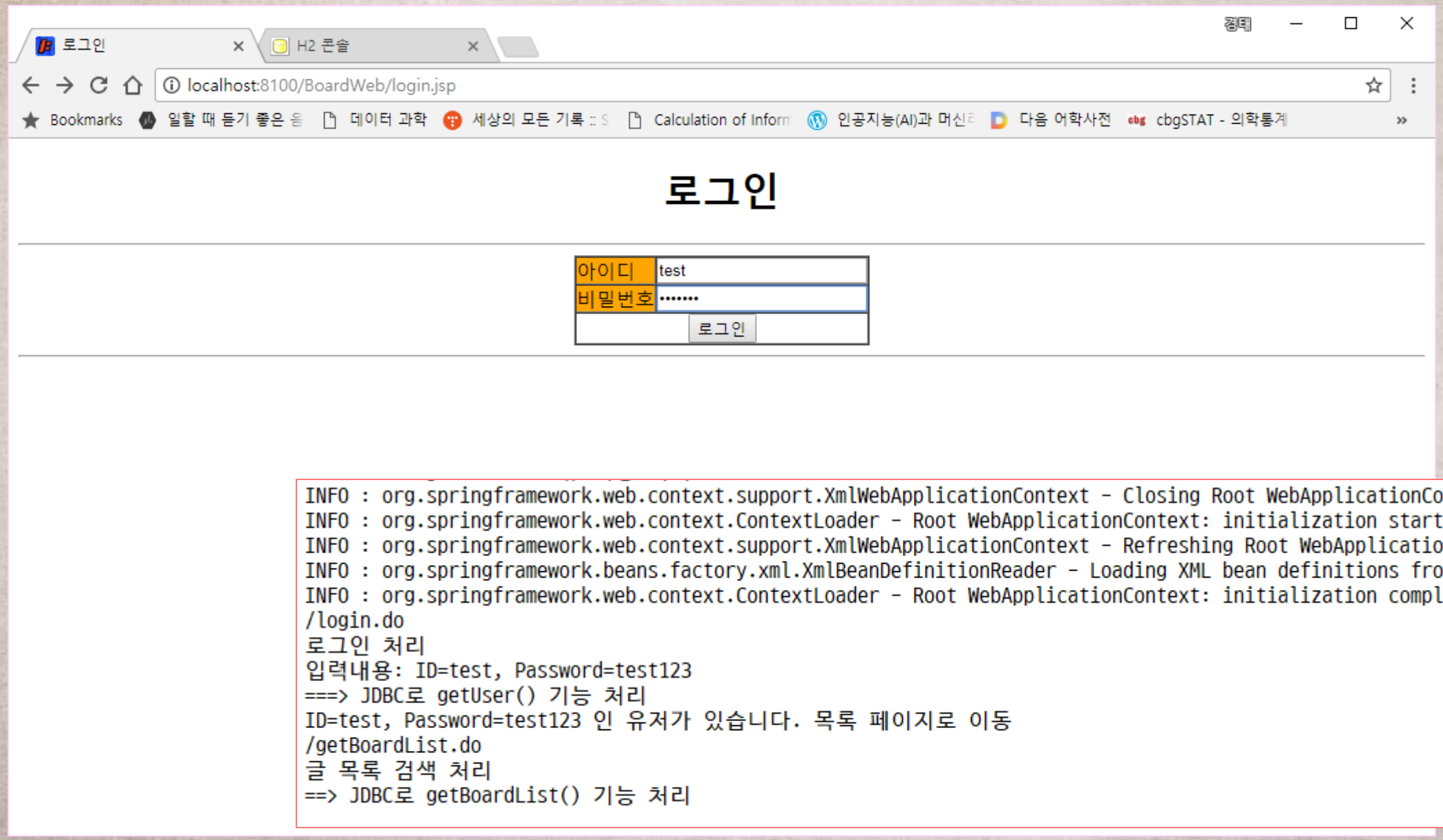
# 로그인 실패시 → 다시 로그인 페이지로 이동




## 실행 결과

```
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Refreshing Root WebApplicationContext: startup date [Sun Sep 09
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from ServletContext resource [/WEB-INF
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




```
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
텐진난만 :: 선형 회귀
머신러닝 - 수식 없이
Variability vs. Compl
조승연의 굿모닝팝스
고급두뇌를 위한 하이
Time Series Forecast
Calculation of Inform
글 목록
환영합니다. Logout
제목
검색
번호 제목 작성자 등록일 조회수
4 JDBD 테스트2 관리자 2018-05-27 0
3 JDBD 테스트 관리자 2018-05-27 0
2 임시 제목 홍길동 2018-05-12 0
1 가입인사 관리자 2018-04-08 0
새글 등록

# 게시판 목록 검색 실행 순서

① DispatcherServlet이 클라이언트의 “/getBoardList.do”요청을 받으면

② DispatcherSerlvet은 BoardDAO 객체를 이용하여 글 목록을 검색한다.

③ 검색된 글 목록을 세션에 등록하고

④ getBoardList.jsp 화면을 요청하면,

⑤ getBoardList.jsp는 세션에 저장된 글 목록을 꺼내어 목록 화면을 구성한다.

⑥ 마지막으로 이 응답 화면이 브라우저에 전송된다.