



박 경 태
comsi.java@gmail.com

고급 자바 프로그래밍

: STS를 이용한 Spring 프로그래밍

어노테이션 기반 MVC 개발

1.1 어노테이션 관련 설정[servlet-context.xml 수정]

- <bean> 등록을 모두 삭제하고 아래처럼 만 남긴다.



```
BoardDAO.java servlet-con... root-context... web.xml getBoard.jsp LoginControl... applicationC... InsertBoardC... »4
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>
```

```
BoardDAO.java servlet-con... root-context... web.xml getBoard.jsp LoginControl... applicationC... InsertBoardC... »4
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.view"></context:component-scan>
11
12 </beans>
13
14 |
```

추가하기

1.2 @Controller 사용하기

- **@Controller**는 **@Component**를 상속한 것으로 클래스의 객체를 메모리에 생성하는 기능을 제공과 함께 **DispatcherServlet**이 인식하는 **Controller**객체로 만들어 준다.
- 만약, **@Controller**를 사용하지 않으면 스프링이 제공하는 **Controller** **인터페이스**를 반드시 구현해야 하고, **handleRequest()** 메소드를 반드시 재정의하여 **DispatcherServlet**이 이를 호출할 수 있도록 해야한다.
 - 하지만 이렇게 구현하면 스프링 프레임워크가 지향하는 POJO스타일의 클래스가 아니다. 즉, “**implements Controller**” 제거
 - “**@Controller**”를 선언해야 함.

→ 글 등록 기능의 **InsertBoardController** 클래스를 POJO스타일로 구현해보자.

기존 InsertBoardController 클래스

```
1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 import org.springframework.web.servlet.ModelAndView;
7 import org.springframework.web.servlet.mvc.Controller;
8
9 import kr.ac.inje.comsi.board.BoardVO;
10 import kr.ac.inje.comsi.board.impl.BoardDAO;
11 //import kr.ac.inje.comsi.view.controller.Controller;
12
13 public class InsertBoardController implements Controller {
14
15     @Override
16     public ModelAndView handleRequest(HttpServletRequest request,
17                                     HttpServletResponse response) {
18         System.out.println("글 등록 처리");
19
20         // 1. 사용자 입력한 정보 추출
21         String title = request.getParameter("title");
22         String content = request.getParameter("content");
23         String writer = request.getParameter("writer");
24
25         // 2. DB 연동 처리
26         BoardVO vo = new BoardVO();
27         vo.setTitle(title);
28         vo.setContent(content);
29         vo.setWriter(writer);
30
31         BoardDAO boardDAO = new BoardDAO();
32         boardDAO.insertBoard(vo);
33
34         // 3. 화면을 리턴
35         ModelAndView mav = new ModelAndView();
36         mav.setViewName("getBoardList.do");
37         return mav; //
38     }
39 }
```

어노테이션 기반으로 변경된 후

```
 servlet-context.xml InsertBoardController.java
1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4
5 import org.springframework.stereotype.Controller; @Controller에 대한 import 클래스
6
7 import kr.ac.inje.comsi.board.BoardVO;
8 import kr.ac.inje.comsi.board.impl.BoardDAO;
9
10 @Controller
11 public class InsertBoardController {
12
13     public void insertBoard(HttpServletRequest request) { return값이 없다.
14         System.out.println("글 등록 처리");
15
16         // 1. 사용자 입력한 정보 추출
17         String title = request.getParameter("title");
18         String content = request.getParameter("content");
19         String writer = request.getParameter("writer");
20
21         // 2. DB 연동 처리
22         BoardVO vo = new BoardVO();
23         vo.setTitle(title);
24         vo.setContent(content);
25         vo.setWriter(writer);
26
27         BoardDAO boardDAO = new BoardDAO();
28         boardDAO.insertBoard(vo);
29     }
30 }
```

1.3 @RequestMapping 사용하기 – HandlerMapping 대체

```
1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4
5 import org.springframework.stereotype.Controller;
6 import org.springframework.web.bind.annotation.RequestMapping;
7
8 import kr.ac.inje.comsi.board.BoardVO;
9 import kr.ac.inje.comsi.board.impl.BoardDAO;
10
11 @Controller
12 public class InsertBoardController {
13
14     @RequestMapping(value="/insertBoard.do")
15     public void insertBoard(HttpServletRequest request) {
16         System.out.println("글 등록 처리");
17
18         // 1. 사용자 입력한 정보 추출
19         String title = request.getParameter("title");
20         String content = request.getParameter("content");
21         String writer = request.getParameter("writer");
22
23         // 2. DB 연동 처리
24         BoardVO vo = new BoardVO();
25         vo.setTitle(title);
26         vo.setContent(content);
27         vo.setWriter(writer);
28
29         BoardDAO boardDAO = new BoardDAO();
30         boardDAO.insertBoard(vo);
31     }
32 }
```

HandleMapping을 대체함

이전의 경우

```
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  <!-- HandlerMapping 등록 -->
12<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
13  <property name="mappings">
14    <props>
15      <prop key="/login.do">login</prop>
16      <prop key="/getBoardList.do">getBoardList</prop>
17      <prop key="/getBoard.do">getBoard</prop>
18      <prop key="/insertBoard.do">insertBoard</prop>
19    </props>
20  </property>
21</bean>
22<bean id="login" class="kr.ac.inje.comsi.view.user.LoginController"></bean>
23<bean id="getBoardList" class="kr.ac.inje.comsi.view.board.GetBoardListController"></bean>
24<bean id="getBoard" class="kr.ac.inje.comsi.view.board.GetBoardController"></bean>
25<bean id="insertBoard" class="kr.ac.inje.comsi.view.board.InsertBoardController"></bean>
26</beans>
27
```

어노테이션이
이 부분을 대체

1.4 클라이언트 요청 처리 – Command 객체 사용

- 사용자의 정보는 HttpServletRequest의 getParameter() 메소드 사용

```
// 1. 사용자 입력한 정보 추출  
String title = request.getParameter("title");  
String content = request.getParameter("content");  
String writer = request.getParameter("writer");
```

글 등록 작업 처리

- 문제는?** 사용자가 입력하는 정보가 많거나 변경되는 상황
→ 입력정보 증가에 따른 자바 코드 증가, 변경될 때마다 Controller 수정

• Command 객체를 이용한 문제 해결

- Command 객체는 Controller 메소드의 매개 변수로 받은 VO(value object) 객체
- InsertBoardController 클래스의 insertBoard 메소드를 Command 객체를 이용하여 구현

Command 객체를 이용한 사용자 입력정보 처리

servlet-context.xml InsertBoardController.java

```
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 import kr.ac.inje.comsi.board.BoardVO;
7 import kr.ac.inje.comsi.board.impl.BoardDAO;
8
9 @Controller
10 public class InsertBoardController {
11
12     @RequestMapping(value="/insertBoard.do")
13     public void insertBoard(BoardVO vo) {
14         System.out.println("글 등록 처리");
15
16         BoardDAO boardDAO = new BoardDAO();
17         boardDAO.insertBoard(vo);
18     }
19 }
20 }
```

getParameter()와 BoardVO 객체
설정 부분이 삭제됨

- insertBoard() 매소드의 매개변수로 사용자가 입력한 값을 매핑할 BoardVO 클래스를 선언하면 스프링 컨테이너가 insertBoard() 매소드를 실행할 때, Command 객체를 생성하여 넘겨 준다.
- 또한, 이때 사용자가 입력한 값들을 Command 객체에 세팅까지 해서 넘겨 준다.

→ 사용자가 입력한 정보 추출과 VO객체 생성, 그리고 값 설정 → 컨테이너
가 처리

HTML의 Form태그와 Command 객체 설정

```
<form action="insertBoard.do" method="post">
  <table border="1" >
    <tr>
      <td bgcolor="orange" width="70">제목</td>
      <td align="left"><input type="text" name="title" /></td>
    </tr>
    <tr>
      <td bgcolor="orange">작성자</td>
      <td align="left"><input type="text" name="writer" size="10" /></td>
    </tr>
    <tr>
      <td bgcolor="orange">내용</td>
      <td align="left"><textarea name="content" cols="80" rows="10"></textarea></td>
    </tr>
    <tr>
      <td colspan="2" align="center"><input type="submit" value=" 새 글 등록 " /></td>
    </tr>
  </table>
</form>
```

```
private int seq;
private String title;
private String writer;
private String content;
private Date regDate;
private int cnt;

public int getSeq() {
  return seq;
}
public void setSeq(int seq) {
  this.seq = seq;
}
public String getTitle() {
  return title;
}
public void setTitle(String title) {
  this.title = title;
}
public String getWriter() {
  return writer;
}
public void setWriter(String writer) {
  this.writer = writer;
}
```

Form 태그의 name 이름과 VO객체의 매개 변수 이름이 같아야 함.
그리고, VO클래스에 setter 매소드가 있어야 인젝션이 발생함

어노테이션으로 게시판
프로그램 구현하기

servlet-context.xml 설정

```
root-context.xml servlet-context.x... InsertBoardContr... GetBoardListContr... GetBoardControle... DeleteBoardContr... UpdateBoardContr...
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.view"></context:component-scan> ← 어노테이션 기반으로
11
12<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
13  <property name="prefix" value="/WEB-INF/board/"></property>
14  <property name="suffix" value=".jsp"></property>
15</bean>
16</beans>
17
18|
```

어노테이션 기반으로
Bean 자동 생성

2.1 글 등록 기능 구현하기

2.2 글 목록 검색 구현하기

```
InsertBoardController.java GetBoardListController.java GetBoardController.java DeleteBoardController.java UpdateBoardController.java
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.servlet.ModelAndView;
6
7 import kr.ac.inje.comsi.board.BoardVO;
8 import kr.ac.inje.comsi.board.impl.BoardDAO;
9
10 @Controller
11 public class GetBoardListController{
12
13     @RequestMapping("/getBoardList.do")
14     public ModelAndView getBoardList(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
15
16         mav.addObject("boardList", boardDAO.getBoardList(vo));
17         mav.setViewName("getBoardList");
18         return mav;
19     }
20 }
21
22 }
```

ModelAndView 객체도 매개변수로 받아서 컨테이너가 생성하도록 설정

2.3 글 상세보기 구현하기

```
InsertBoardController.java GetBoardListController.java GetBoardController.java DeleteBoardController.java UpdateBoardController.java
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.servlet.ModelAndView;
6
7 import kr.ac.inje.comsi.board.BoardVO;
8 import kr.ac.inje.comsi.board.impl.BoardDAO;
9
10 @Controller
11 public class GetBoardController {
12
13     @RequestMapping("/getBoard.do")
14     public ModelAndView getBoard(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
15
16         mav.addObject("board", boardDAO.getBoard(vo)); // View 정보 저장
17         mav.setViewName("getBoard"); // View wjdqh wjwkd
18         return mav; //
19     }
20 }
21 }
```

ModelAndView 객체도 매개변수로 받아서
컨테이너가 생성하도록 설정

2.4 글 수정 기능 구현하기

```
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 import kr.ac.inje.comsi.board.BoardVO;
7 import kr.ac.inje.comsi.board.impl.BoardDAO;
8
9 @Controller
10 public class UpdateBoardController {
11
12     @RequestMapping("/updateBoard.do")
13     public String updateBoard(BoardVO vo, BoardDAO boardDAO) {
14
15         boardDAO.updateBoard(vo);
16
17         return "redirect:getBoardList.do";
18     }
19 }
20
```

BoardDAO 객체도 매개변수로 받고
리턴타입은 String으로 변경

2.5 글 삭제 기능 구현하기

InsertBoardController.java GetBoardListController.java GetBoardController.java DeleteBoardController.java

```
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 import kr.ac.inje.comsi.board.BoardVO;
7 import kr.ac.inje.comsi.board.impl.BoardDAO;
8
9 @Controller
10 public class DeleteBoardController{
11
12     @RequestMapping("/deleteBoard.do")
13     public String deleteBoard(BoardVO vo, BoardDAO boardDAO) {
14
15         boardDAO.deleteBoard(vo);
16         return "redirect:getBoardList.do";
17     }
18 }
19 }
```

2.6 로그인 기능 구현하기

The screenshot shows a Java IDE interface with multiple tabs at the top: InsertBoardController.java, GetBoardListController.java, GetBoardController.java, DeleteBoardController.java, and LoginController.java. The LoginController.java tab is active, displaying the following Java code:

```
1 package kr.ac.inje.comsi.view.user;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 import kr.ac.inje.comsi.user.UserVO;
7 import kr.ac.inje.comsi.user.impl.UserDAO;
8
9 @Controller
10 public class LoginController{
11
12     @RequestMapping("/login.do")
13     public String login(UserVO vo, UserDAO userDAO) {
14         if (userDAO.getUser(vo) != null) return "redirect:getBoardList.do";
15         else return "redirect:login.jsp";
16     }
17 }
18
19 }
```

2.7 로그아웃 기능 구현하기

The screenshot shows a Java code editor with five tabs at the top: InsertBoardController.java, DeleteBoardController.java, LoginController.java, LogoutController.java (which is currently selected), and another tab that is mostly obscured. The code in the editor is for a LogoutController:

```
1 package kr.ac.inje.comsi.view.user;
2
3+import javax.servlet.http.HttpSession;
4
5 @Controller
6 public class LogoutController{
7
8     @RequestMapping("/logout.do")
9     public String logout(HttpSession session) {
10         session.invalidate();
11         return "redirect:login.jsp";
12     }
13 }
14
15 }
```

The code defines a LogoutController with a single method named logout. This method takes an HttpSession parameter and invalidates it. It then returns a string indicating a redirect to the login page.

2.8 컨트롤러 통합하기

```
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.servlet.ModelAndView;
6
7 import kr.ac.inje.comsi.board.BoardVO;
8 import kr.ac.inje.comsi.board.impl.BoardDAO;
9
10 @Controller
11 public class BoardController {
12
13     // 글 상세 가져오기
14     @RequestMapping("/getBoard.do")
15     public ModelAndView getBoard(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
16
17         mav.addObject("board", boardDAO.getBoard(vo)); // View 정보 저장
18         mav.setViewName("getBoard"); // View wjdqh wjwkd
19         return mav; //
20     }
21
22     // 여기에 다른 함수의 메인 기능 부분을 추가.
23
24     // 글 목록 가져오기
25     @RequestMapping("/getBoardList.do")
26     public ModelAndView getBoardList(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
27
28         mav.addObject("boardList", boardDAO.getBoardList(vo));
29         mav.setViewName("getBoardList");
30         return mav;
31     }
32 }
```

로그인 화면

로그인

아이디	<input type="text"/>
비밀번호	<input type="password"/>
<input type="button" value="로그인"/>	

글 목록 가져오기

The screenshot shows a Java-based application interface. At the top, there is a toolbar with several icons and labels: InsertBoardCo..., DeleteBoardCo..., LoginControlle..., LogoutControll..., 글 목록 (highlighted in blue), BoardControlle..., GetBoardContr..., and GetBoardListC... . Below the toolbar is a browser-style header with back, forward, stop, and refresh buttons, and a URL field containing <http://localhost:8080/Board/getBoardList.do>. The main content area displays a title "글 목록" and a message "환영합니다. [Logout](#)". Below this is a table listing five board entries:

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

At the bottom center of the page is a link [새글 등록](#).

추가 어노테이션과
프로젝트 계층 분리

기타 어노테이션 – 요청에 따른 처리

- @RequestMapping을 이용할 때, 클라이언트의 요청 방식(GET/POST) 방식에 따라 수행될 메소드를 다르게 설정

```
1 package kr.ac.inje.comsi.view.user;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 import kr.ac.inje.comsi.user.UserVO;
7 import kr.ac.inje.comsi.user.impl.UserDAO;
8
9 @Controller
10 public class LoginController{
11
12     @RequestMapping(value="/login.do")
13     public String login(UserVO vo, UserDAO userDAO) {
14         if (userDAO.getUser(vo) != null) return "redirect:getBoardList.do";
15         else return "redirect:login.jsp";
16     }
17 }
18 }
```

```
@RequestMapping(value="/login.do", method=RequestMethod.POST)
public String login(UserVO vo, UserDAO userDAO) {
    if (userDAO.getUser(vo) != null) return "redirect:getBoardList.do";
    else return "redirect:login.jsp";
}
```

LoginController 설정

getBoardList.jsp BoardController.java BoardServiceImpl.java LoginController.java LogoutController.java applicationContext.xml

```
1 package kr.ac.inje.comsi.view.user;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6
7 import kr.ac.inje.comsi.user.UserVO;
8 import kr.ac.inje.comsi.user.impl.UserDAO;
9
10 @Controller
11 public class LoginController{
12
13     @RequestMapping(value="/login.do", method=RequestMethod.GET)
14     public String loginView(UserVO vo) {
15         System.out.println("로그인 화면으로 이동");
16         vo.setId("test");
17         vo.setPassword("test123");
18         return "redirect:login.jsp";
19     }
20
21     @RequestMapping(value="/login.do", method=RequestMethod.POST)
22     public String login(UserVO vo, UserDAO userDAO) {
23         System.out.println("로그인 인증처리 이동");
24         if (userDAO.getUser(vo) != null) return "redirect:getBoardList.do";
25         else return "redirect:login.jsp";
26     }
27 }
28 }
```

} UserVO클래스의 Command 객체 사용

index.jsp 파일 생성 – webapp 폴더에 생성

- 나머지 모든 jsp파일을 WEB-INF/board/ 폴더로 이동한다.



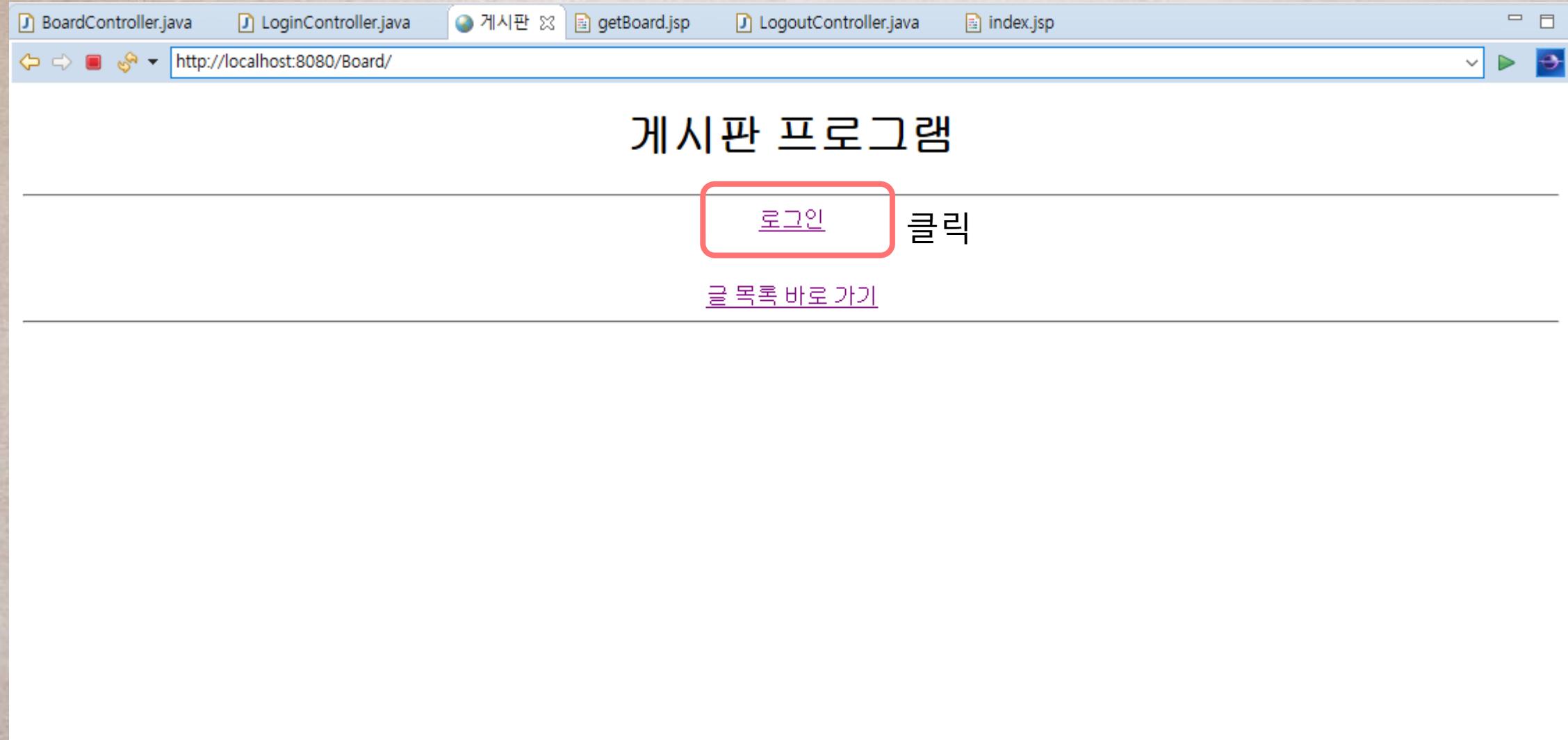
The screenshot shows a Java IDE interface with the following tabs in the top bar:

- BoardController.java
- LoginController.java
- 로그인 (Login)
- getBoard.jsp
- LogoutController.java
- index.jsp (highlighted in blue)

The content of the index.jsp file is displayed in the main editor area:

```
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>게시판</title>
8 </head>
9<body>
10<center>
11     <h1>게시판 프로그램</h1>
12     <hr>
13         <a href="login.do">로그인</a><br><br><br>
14         <a href="getBoardList.do">글 목록 바로 가기</a>
15     <hr>
16 </center>
17 </body>
18 </html>
```

시작 페이지 – index.jsp



http://localhost:8080/Board/login.do



/login.do(**GET방식**)의 url mappin으로 처리된 loginView(UserVO vo) 함수 실행
→ UserVO 객체가 Command객체로 사용되어 JSP에 적용됨
→ 기본 Command 객체명은 첫글자가 소문자인 Command객체 클래스명(userVO)

login.jsp

BoardController.java 로그인 LogoutController.java index.jsp login.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4<html>
5<head>
6 <meta charset="UTF-8">
7 <title>로그인</title>
8 </head>
9<body>
10<div align="center">
11     <h1>로그인</h1>
12     <form action="login.do" method="post">
13         <table border="1" >
14             <tr>
15                 <td bgcolor="orange">아이디</td>
16                 <td><input type="text" name="id" value="${userVO.id}" /></td>
17             </tr>
18             <tr>
19                 <td bgcolor="orange">비밀번호</td>
20                 <td><input type="password" name="password" value="${userVO.password}" /></td>
21             </tr>
22             <tr>
23                 <td colspan="2" align="center"><input type="submit" value="로그인"/></td>
24             </tr>
25         </table>
26     </form>
27 </div>
28 </body>
29 </html>
```

기본 Command 객체의 이름은 클래스 이름의 첫 글자를
소문자로 변경한 이름을 자동으로 설정

@SessionAttributes와 @ModelAttribute 사용하기

- `@SessionAttributes("board")` – Model에 "board"라는 이름이 저장되어 있다면 그 데이터를 세션에 자동으로 저장하라는 설정
- `@ModelAttribute("board")` – Model에서 "board"이름 객체를 가져옴 (함수 매개변수)

```
// 글 상세 보기
@RequestMapping(value="/getBoard.do")
public ModelAndView getBoard(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
    mav.addObject("board", boardDAO.getBoard(vo)); // View 정보 저장
    mav.setViewName("getBoard"); // View 정보
    return mav;
}
```

↓ Model과 View를 분리하여 자료 전달

```
// 글 상세 보기
@RequestMapping(value="/getBoard.do")
public String getBoard(BoardVO vo, BoardDAO boardDAO, Model model) {
    model.addAttribute("board", boardDAO.getBoard(vo)); → 모델과 뷰 분리한 후 board객체를 모델에 올림
    return "getBoard";
}
```

BoardController 클래스 – @SessionAttribute 적용

```
BoardController.java ✘ 로그인 LogoutController.java index.jsp login.jsp BoardDAO.java

1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.SessionAttributes;
7 import org.springframework.web.servlet.ModelAndView;
8
9 import kr.ac.inje.comsi.board.BoardVO;
10 import kr.ac.inje.comsi.board.impl.BoardDAO;
11
12 @Controller
13 @SessionAttributes("board")
14 public class BoardController {
15
16     // 글 목록 보기
17+    public ModelAndView getBoardList(BoardVO vo, BoardDAO boardDAO, ModelAndView mav) {
18+
19
20     // 글 추가하기
21+    public String insertBoard(BoardVO vo, BoardDAO boardDAO) {
22+
23
24     // 글 수정하기
25+    public String updateBoard(BoardVO vo, BoardDAO boardDAO) {
26+
27
28     // 글 상세 보기
29+    public String getBoard(BoardVO vo, BoardDAO boardDAO, Model model) {
30+
31
32     // 글 삭제
33+    public String deleteBoard(BoardVO vo, BoardDAO boardDAO) {
34+
35
36
37
38
39
40
41     // 글 상세 보기
42+    public String getBoard(BoardVO vo, BoardDAO boardDAO, Model model) {
43+
44
45
46
47
48
49     // 글 삭제
50+    public String deleteBoard(BoardVO vo, BoardDAO boardDAO) {
51+
52
53
54
55
56 }
```

getBoard.jsp-게시물 내용보기

[Log-out](#)

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

[글등록](#) [글삭제](#) [글목록](#)

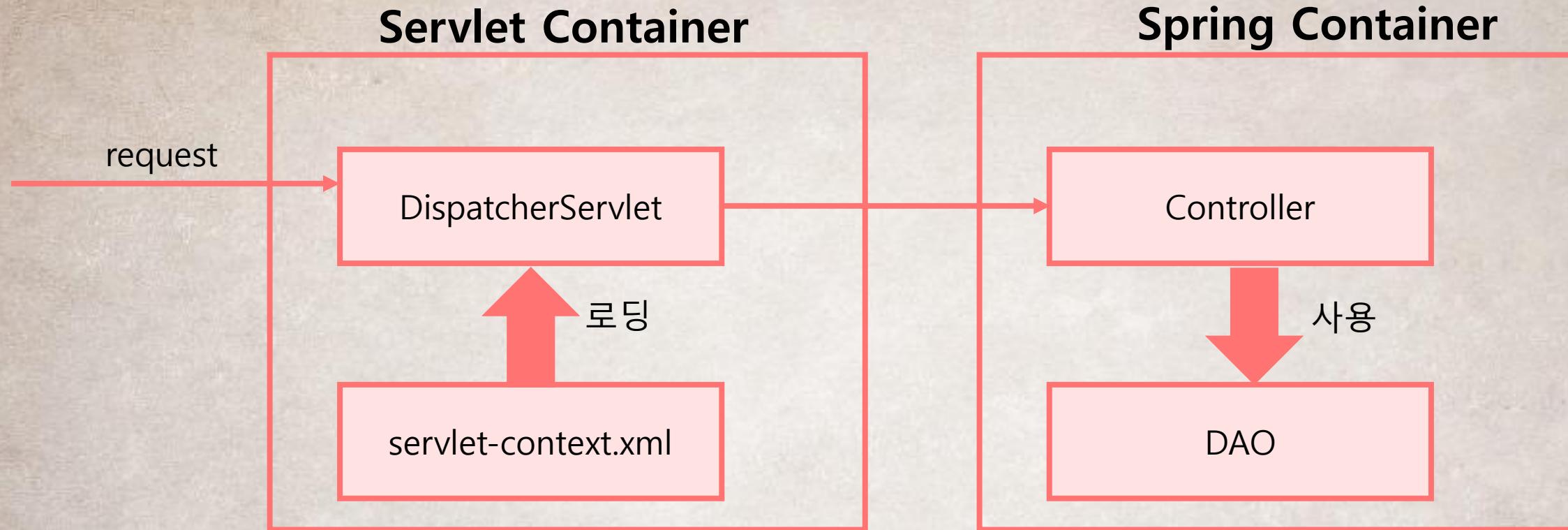
getBoard.jsp- 게시물 내용보기

```
BoardController.java  글 목록 LogoutController.java BoardDAO.java getBoard.jsp index.jsp
12      <a href="logout.do">Log-out</a>
13      <br>
14      <form action="updateBoard.do" method="post">
15          <input name="seq" type="hidden" value="${board.seq}" />
16          <table border="1">
17              <tr>
18                  <td bgcolor="orange" width="70">제목</td>
19                  <td align="left"><input name="title" type="text" value="${board.title}" /></td>
20              </tr>
21              <tr>
22                  <td bgcolor="orange">작성자</td>
23                  <td align="left">${board.writer}</td>
24              </tr>
25              <tr>
26                  <td bgcolor="orange">내용</td>
27                  <td align="left"><textarea name="content" cols="80" rows="10">${board.content}</textarea></td>
28              </tr>
29              <tr>
30                  <td bgcolor="orange">등록일</td>
31                  <td align="left">${board.regDate}</td>
32              </tr>
33              <tr>
34                  <td bgcolor="orange">조회수</td>
35                  <td align="left">${board.cnt}</td>
36              </tr>
37              <tr>
38                  <td colspan="2" align="center"><input type="submit" value="글 수정" /></td>
39              </tr>
40          </table>
41      </form>
```

03. 프리젠테이션레이어와 비즈니스 레이어 통합

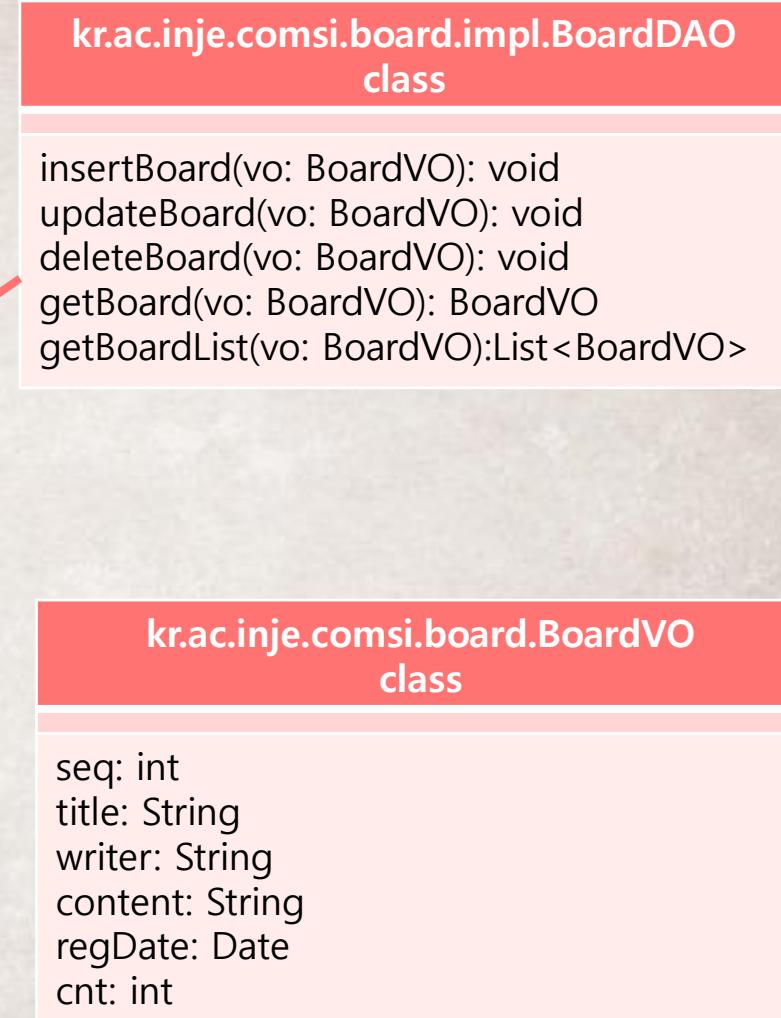
DAY 4

스프링 MVC 구조와 실행 순서



- Controller가 모든 사용자 처리를 위해 직접 DAO객체를 사용
- 프로그램이 실행되지 않거나 심각한 문제가 되지 않는다.
- 하지만, Controller는 DAO객체를 직접이용해서는 안되고 비즈니스 콤포넌트를 이용해야 한다.

3.1 비즈니스 컴포넌트 사용



- BoardService 비즈니스 컴포넌트 클래스 다이어그램

Controller가 DAO메소드를 직접 호출하면 안 되는 이유

- 유지보수 과정에서 DAO클래스를 다른 클래스로 쉽게 교체하기 위해
 - BoardController의 모든 메소드가 boardDAO 객체를 받아서 DB 연동 처리
 - 만약, DAO 클래스를 변경하거나 앞으로 추가될 다른 DAO 클래스로 변경하고자 한다면 BoardController의 모든 메소드를 수정
 - Controller가 하나면 괜찮으나 여러 개라면 일일이 수정해야 함
- 유지보수의 어려움
- 그러면, 비즈니스 컴포넌트가 수정되더라도 Controller가 수정되지 않도록 하려면 어떻게 해야 할까?
 - 비즈니스 컴포넌트 입장에서 자신을 사용해주는 클라이언트는 Controller임
 - 클라이언트가 인터페이스를 통해서 비즈니스 컴포넌트를 이용하면 컴포넌트의 구현 클래스를 수정하거나 다른 클래스로 대체해도 이를 사용하는 클라이언트는 수정하지 않아도 됨 → 다형성(객체지향 언어의 특징)

BoardController를 수정(DAO 클래스 교체)

BoardController.java 글 목록 LogoutController.java BoardDAO.java BoardServiceImpl.java

```
1 package kr.ac.inje.comsi.view.board;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.SessionAttributes;
8 import org.springframework.web.servlet.ModelAndView;
9
10 import kr.ac.inje.comsi.board.BoardService;
11 import kr.ac.inje.comsi.board.BoardVO;
12
13 @Controller
14 @SessionAttributes("board")
15 public class BoardController {
16     @Autowired
17     private BoardService boardService; // 1. 추가(BoardService 타입으로 이름이 "boardService"인 객체가 주입 -BoardServiceImpl 클래스)
18
19
20     // 글 목록 보기
21     @RequestMapping(value="/getBoardList.do")
22     public ModelAndView getBoardList(BoardVO vo, ModelAndView mav) {
23
24         mav.addObject("boardList", boardService.getBoardList(vo));
25         mav.setViewName("getBoardList");
26         return mav; // 2. BoardDAO 매개변수를 삭제하고 boardDAO객체 대신에
27     } boardService 객체 사용
28 }
```

1. 추가(BoardService 타입으로 이름이 "boardService"인 객체가 주입
-BoardServiceImpl 클래스)

2. BoardDAO 매개변수를 삭제하고 boardDAO객체 대신에
boardService 객체 사용

이어서...

```
29 // 글 추가하기
30 @RequestMapping(value="/insertBoard.do")
31 public String insertBoard(BoardVO vo) {
32     boardService.insertBoard(vo);
33     return "redirect:getBoardList.do";
34 }
35
36 // 글 수정하기
37 @RequestMapping(value="/updateBoard.do")
38 public String updateBoard(BoardVO vo) {
39     boardService.updateBoard(vo);
40
41     return "redirect:getBoardList.do";
42 }
43
44 // 글 상세 보기
45 @RequestMapping(value="/getBoard.do")
46 public String getBoard(BoardVO vo, Model model) {
47     System.out.println("Model과 @SessionAttributes 적용");
48     model.addAttribute("board", boardService.getBoard(vo));
49     return "getBoard";
50 }
51
52 // 글 삭제
53 @RequestMapping(value="/deleteBoard.do")
54 public String deleteBoard(BoardVO vo) {
55
56     boardService.deleteBoard(vo);
57     return "redirect:getBoardList.do";
58 }
59 }
60 }
```

BoardServiceImpl 클래스 수정

The screenshot shows a Java code editor with the following details:

- File tabs:** BoardController.java, 글 목록, LogoutController.java, BoardDAO.java, BoardServiceImpl.java (selected).
- Code content:** The code is annotated with line numbers (1-30) and highlights specific lines (e.g., line 18). It defines a class BoardServiceImpl that implements the BoardService interface. The class uses autowiring to inject a BoardDAO dependency and provides implementations for insertBoard, updateBoard, and deleteBoard methods.

```
1 package kr.ac.inje.comsi.board.impl;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import kr.ac.inje.comsi.board.BoardService;
9 import kr.ac.inje.comsi.board.BoardVO;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14     @Autowired
15     private BoardDAO boardDAO;
16
17     @Override
18     public void insertBoard(BoardVO vo) {
19         boardDAO.insertBoard(vo);
20     }
21
22     @Override
23     public void updateBoard(BoardVO vo) {
24         boardDAO.updateBoard(vo);
25     }
26
27     @Override
28     public void deleteBoard(BoardVO vo) {
29         boardDAO.deleteBoard(vo);
30     }
}
```

이어서...

```
31
32 @Override
33     public BoardVO getBoard(BoardVO vo) {
34         return boardDAO.getBoard(vo);
35     }
36
37 @Override
38     public List<BoardVO> getBoardList(BoardVO vo) {
39         return boardDAO.getBoardList(vo);
40     }
41 }
42 }
```

DAO 클래스가 변경된다면

- BoardDAO를 다른 DAO 클래스로 변경한다면?

```
mybatis-config.xml root-context.xml servlet-context.xml BoardServiceImpl.java
1 package kr.ac.inje.comsi.board.impl;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import kr.ac.inje.comsi.board.BoardService;
9 import kr.ac.inje.comsi.board.BoardVO;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14     @Autowired
15     private BoardDAO boardDAO;
16
17     @Override
18     public void insertBoard(BoardVO vo) {
19         boardDAO.insertBoard(vo);
20     }
21 }
```

@Autowired
private BoardDAOSpring boardDAO ;

게시판 실행하기....에러~~~~~

Type Exception Report

Message Servlet.init() for servlet [DispatcherServlet] threw exception

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
javax.servlet.ServletException: Servlet.init() for servlet [DispatcherServlet] threw exception
    org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:491)
    org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
    org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:668)
    org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343)
    org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:408)
    org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:66)
    org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:764)
    org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1388)
    org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
    java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
    java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
    org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
    java.lang.Thread.run(Unknown Source)
```

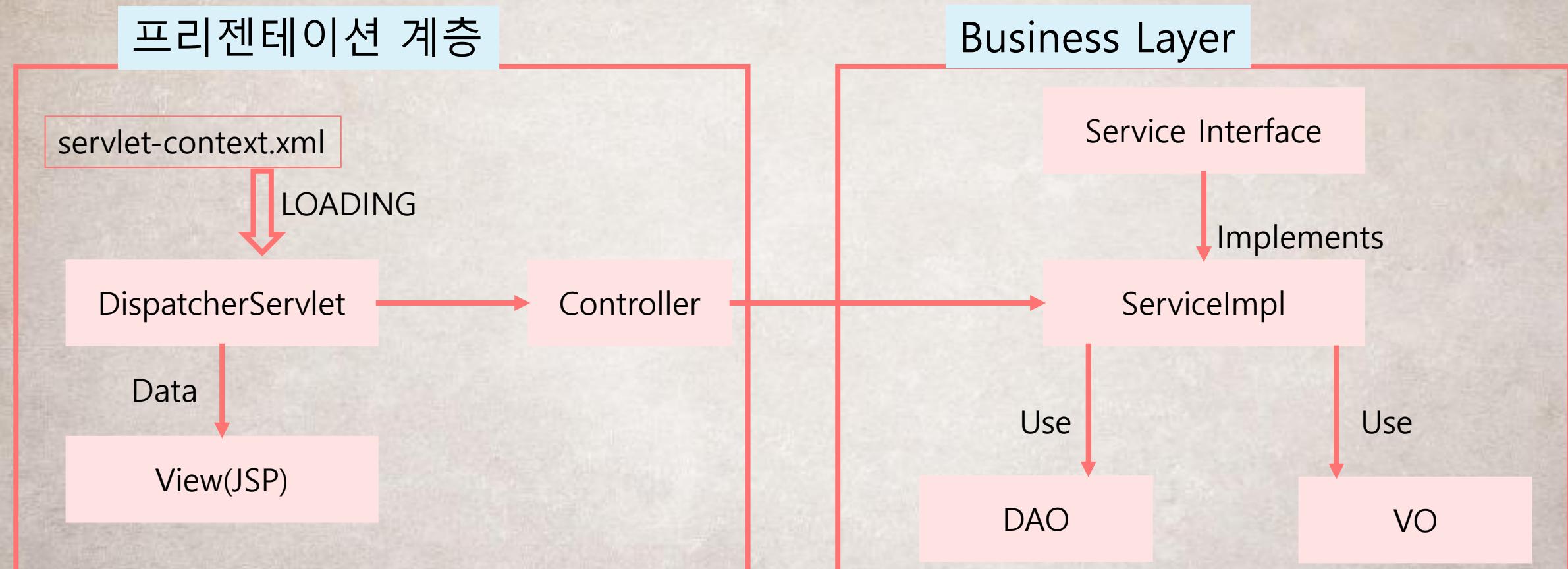
Root Cause

boardController 빈이 생성되지 않았다는 것

```
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'boardController': Unsatisfied dependency expressed through field 'listService'. Dependency annotations: @org.springframework.beans.factory.annotation.Autowired(required=true)
    org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:615)
    org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetadata.java:88)
    org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor.postProcessPropertyValues(AutowiredAnnotationBeanPostProcessor.java:397)
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1268)
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:553)
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:483)
    org.springframework.beans.factory.support.AbstractBeanFactory$1.getObject(AbstractBeanFactory.java:312)
    org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:230)
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:308)
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:197)
    ...
```

시스템의 전체 구조를 두 개의 레이아웃으로 표현

- 프리젠테이션 레이어–사용자와의 커뮤니케이션 담당
- 비즈니스 레이어–사용자의 요청에 대한 비즈니스 로직 처리 담당



프리젠테이션계층의 Controller가 생성되기 전에 비즈니스 계층 컴포넌트가 먼저 생성되어야 함 → 스프링의 ContextLoaderListener가 제공(web.xml)

비즈니스 계층 컴포넌트가 먼저 생성되도록 설정하기

- web.xml 파일 수정
- 스프링의 ContextLoaderListener 항목을 추가

```
37<listener>
38    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
39</listener>
40<context-param>
41    <param-name>contextConfigLocation</param-name>
42    <param-value>/WEB-INF/spring/root-context.xml</param-value>
43</context-param>
44
45
46</web-app>
```

ContextLoaderListener에서 설정한 파일 생성

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure. A red box highlights the `spring/appServlet/root-context.xml` file under the `WEB-INF/classes/spring/appServlet` directory. On the right, the `Configure Namespaces` dialog is open, listing various Spring namespaces with checkboxes. Several checkboxes are checked, including `beans`, `context`, `mvc`, and `util`.

Package Explorer

- BoardController.java
- kr.ac.inje.comsi.view.user
- LoginController.java
- LogoutController.java
- src/main/resources
 - applicationContext.xml
 - log4j.xml
 - src/test/java
 - src/test/resources
- Maven Dependencies
- JRE System Library [JavaSE-1.6]
- src
 - main
 - webapp
 - resources
 - WEB-INF
 - board
 - getBoard.jsp
 - getBoardList.jsp
 - insertBoard.jsp
 - login.jsp
 - classes
 - spring
 - appServlet
 - root-context.xml
 - web.xml
 - index.jsp
 - test
- target
- pom.xml

root-context.xml 생성후 namespace 추가

root-context.xml

- component-scan 항목의 base-package 설정
- BoardServiceImpl이 포함된 경로



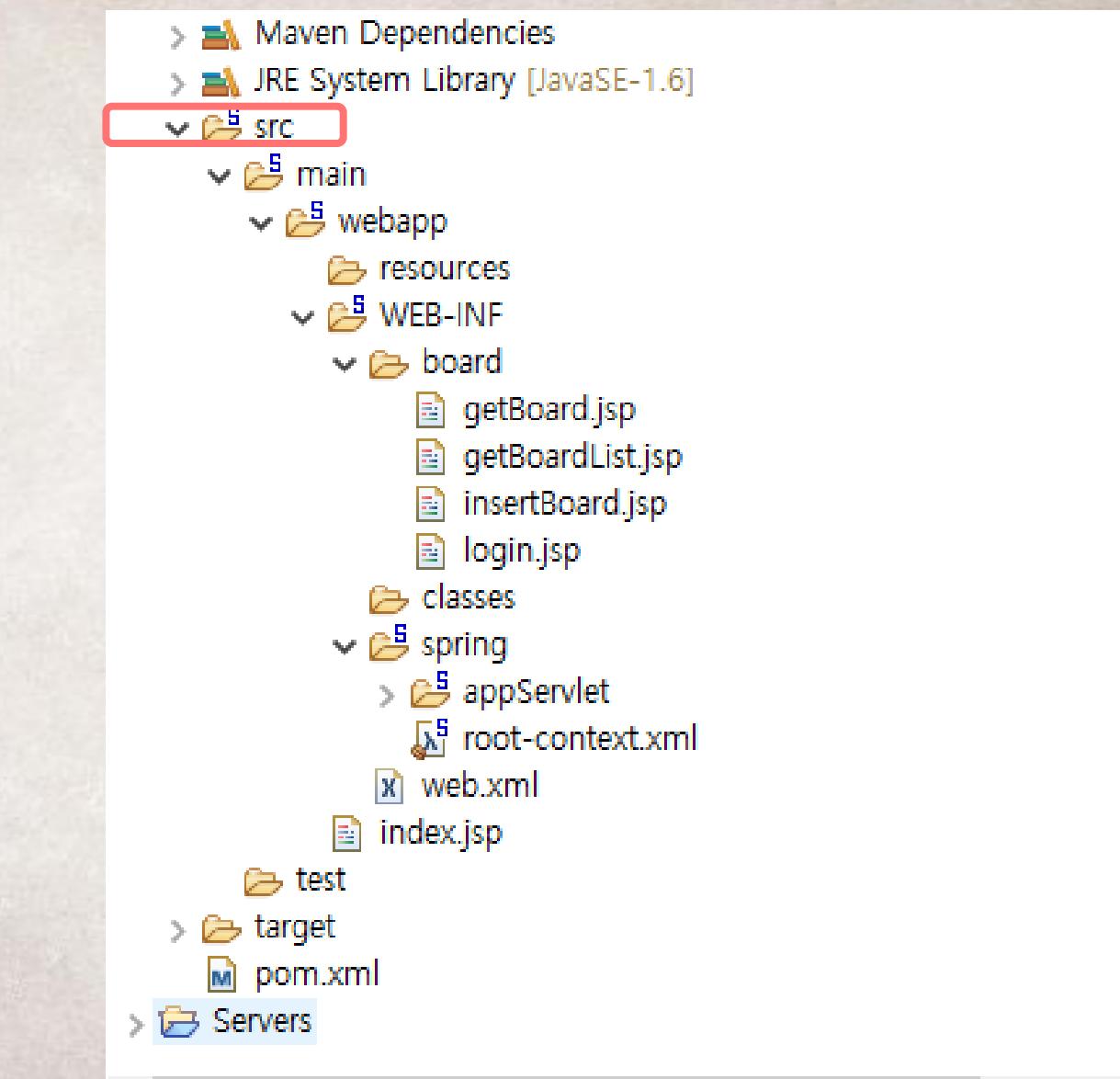
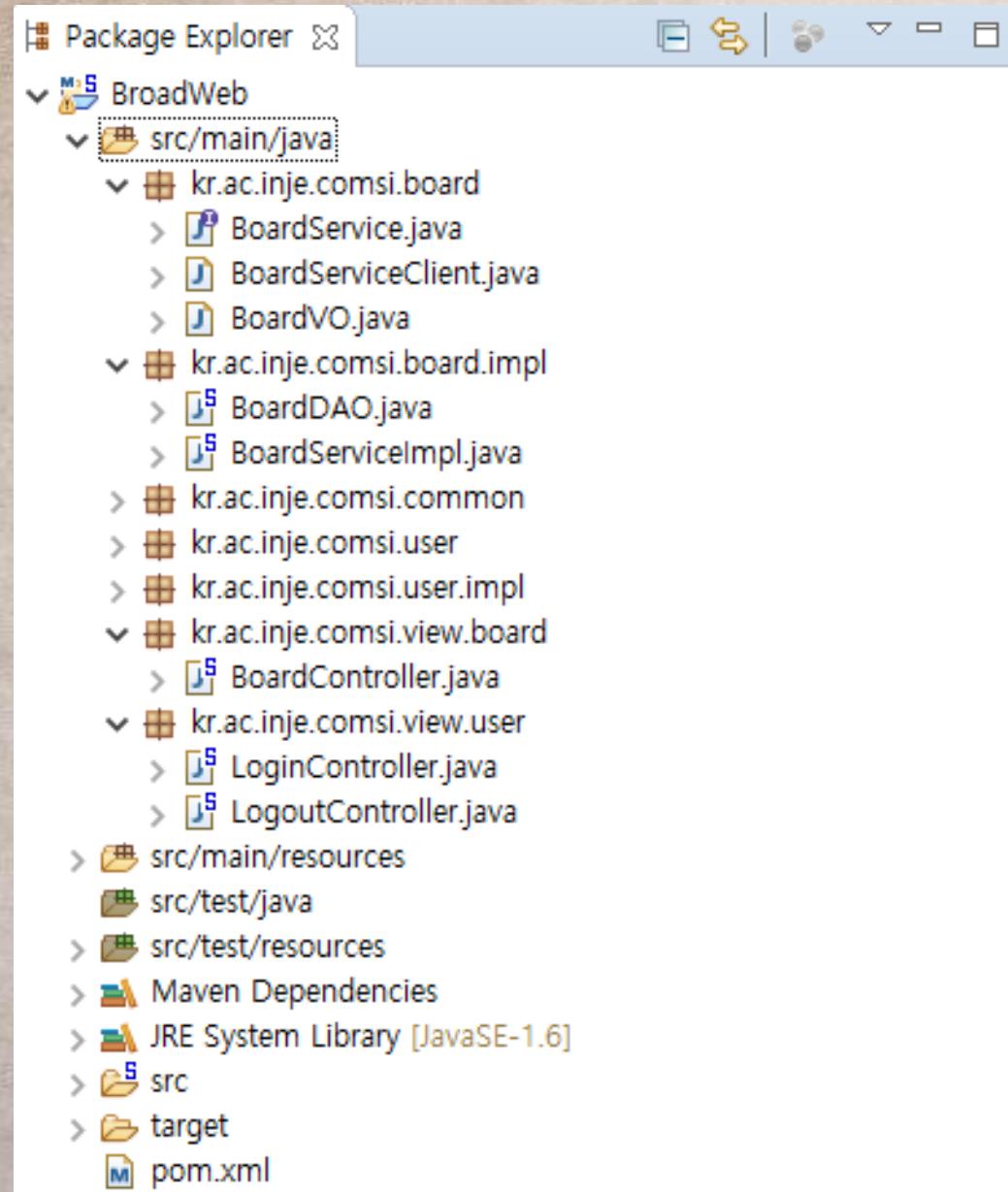
The screenshot shows a code editor window with several tabs at the top: '글 목록', 'servlet-context.xml', 'BoardDAO.java', and 'root-context.xml'. The 'root-context.xml' tab is active, and the code is displayed below:

```
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.board"></context:component-scan>
11</beans>
12
13 |
```

servlet-context.xml의 component-scan 확인

```
글 목록 servlet-context.xml BoardDAO.java 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.view"></context:component-scan>
11
12<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
13  <property name="prefix" value="/WEB-INF/board/"></property>
14  <property name="suffix" value=".jsp"></property>
15</bean>
16</beans>
17
18 |
```

프로젝트 구조



스프링 컨테이너의 관계

1) Servlet Container

servlet-context.xml 로딩

request

DispatcherServlet

ContextLoaderListener

root-context.xml 로딩

3) Spring Container

Controller

2) Spring Container (ROOT)

ServiceImpl

사용

DAO

- ① web.xml을 로딩하여 서블릿 컨테이너가 구동
- ② web.xml에 등록된 ContextLoaderListener 객체를 생성
(root-context.xml을 로딩하여 스프링 컨테이너 구동 → ROOT 컨테이너로 Service 구현 클래스나 DAO 객체들을 메모리에 생성)
- ③ *.do 요청이 서버에 들어오면 서블릿 컨테이너는 DispatcherServlet 객체를 생성하고 servlet-context.xml 파일을 로딩하여 두 번째 스프링 컨테이너를 구동(Controller 객체를 메모리에 로딩)