DAY - 04

Spring MVC (Annotation)

Annotation 기반 Sprnig MVC

- Annotation 설정을 위한 준비 (presentation-layer.xml)

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context-4.2.xsd">

    <context:component-scan base-package="com.springbook.view“/>


</beans>
```

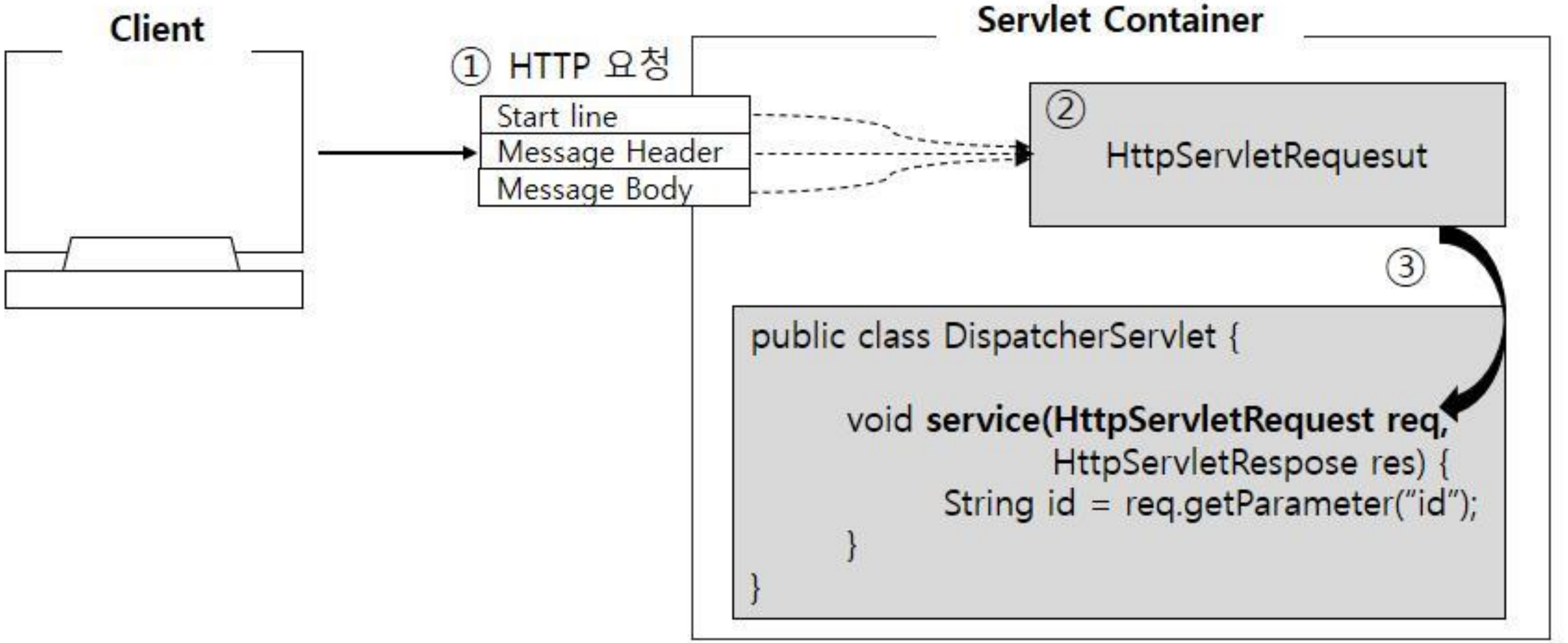
- @Controller Annotation
  - @Controller가 붙은 클래스를 메모리에 생성하고 Controller 객체로 인식하도록 한다.
  - Controller를 POJO(Plain Old Java Object) 스타일로 코딩할 수 있다.

```
@Controller
public class InsertBoardController {

}
```

- @RequestMapping
  - 클라이언트의 요청 path에 대해 실행될 메소드를 매핑한다.

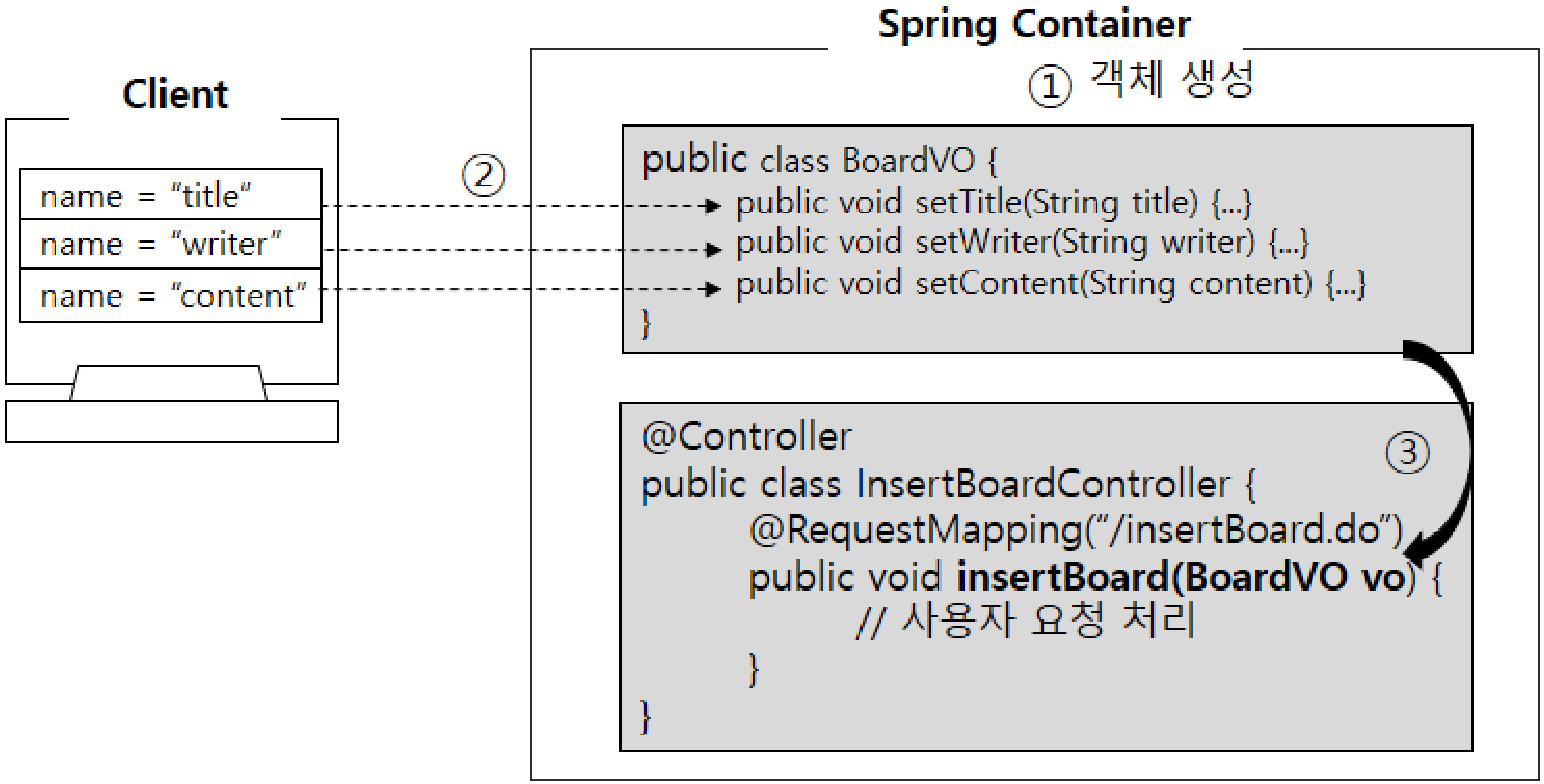
```
@Controller
public class InsertBoardController {

        @RequestMapping(value="/insertBoard.do")
        public void insertBoard(HttpServletRequest request) {

        }
}
```

- 사용자 입력 값 자동 셋팅 원리



```
public class DispatcherServlet {

        void service(HttpServletRequest req,
                     HttpServletRespose res) {
                String id = req.getParameter("id");
        }
}
```

• Command 객체 사용하기



```
public class BoardVO {
        public void setTitle(String title) {...}
        public void setWriter(String writer) {...}
        public void setContent(String content) {...}
}
```

```
@Controller
public class InsertBoardController {
        @RequestMapping("/insertBoard.do")
        public void insertBoard(BoardVO vo) {
                // 사용자 요청 처리
        }
}
```

- 요청 방식에 따른 @RequestMapping 사용

```
@Controller
public class LoginController {
        @RequestMapping(value="/login.do", method=RequestMethod.GET)
        public String loginView(UserVO vo) {
                vo.setId("test");
                vo.setPassword("test123");
                return "login.jsp";
        }

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

- JSP에서 Command 객체 사용하기

```
<table border="1" cellpadding="0" cellspacing="0">
        <tr>
                <td bgcolor="orange">아이디</td>
                <td><input type="text" name="id" value="${userVO.id }"/></td>
        </tr>
        <tr>
                <td bgcolor="orange">비밀번호</td>
                <td><input type="password" name="password"
                                                    value="${userVO.password }"/></td>
        </tr>
        <tr>
                <td colspan="2"><input type="submit" value="로그인"/></td>
        </tr>
</table>
```

• @ModelAttribute 사용

Controller	`@RequestMapping(value="/login.do", method=RequestMethod.GET)`
	`public String loginView(@ModelAttribute("user") UserVO vo) {`
	`        vo.setId("test");`
	`        vo.setPassword("test123");`
	`        return "login.jsp";`
	`}`

JSP	`<tr>`
	`    <td bgcolor="orange">아이디</td>`
	`    <td><input type="text" name="id" value="${user.id }"/></td>`
	`</tr>`
	`<tr>`
	`    <td bgcolor="orange">비밀번호</td>`
	`    <td><input type="password" name="password"`
	`                                        value="${user.password }"/></td>`
	`</tr>`

- Servlet API 사용

  - Servlet 에서 제공하는 HttpServletRequest, HttpServletResponse, HttpSession, Locale 등 다양한 객체를 매개 변수로 받을 수 있다.

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

• Controller 메소드 리턴 타입

리턴타입	소스 비교
ModelAndView	`public ModelAndView login(UserVO vo, UserDAO userDAO, ModelAndView mav) { if(userDAO.getUser(vo) != null) mav.setViewName("getBoardList.jsp"); else mav.setViewName("login.jsp"); return mav; }`
String	`public String login(UserVO vo, UserDAO userDAO) { if(userDAO.getUser(vo) != null) return "getBoardList.jsp"; else return "login.jsp"; }`

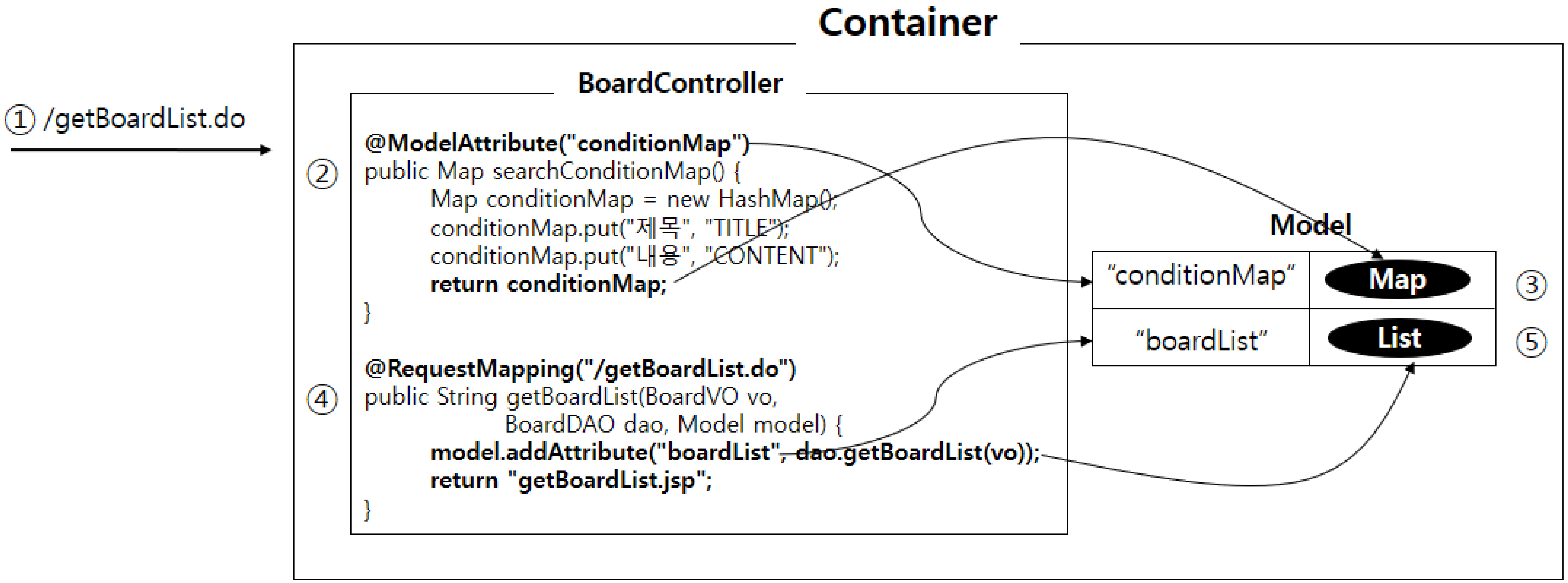
- @RequestParam 사용하기

```
@Controller
public class BoardController {

        @RequestMapping("/getBoardList.do")
        public String getBoardList(
                @RequestParam(value="searchCondition", defaultValue="TITLE",
                                        required=false) String condition,
                @RequestParam(value="searchKeyword", defaultValue="",
                                        required=false) String keyword) {
                System.out.println("검색 조건 : " + condition);
                System.out.println("검색 단어 : " + keyword);

                return "getBoardList.jsp";
        }
}
```

- @ModelAttribute 사용하기 (1)

```
@ModelAttribute("conditionMap")
public Map<String, String> searchConditionMap() {
	Map<String, String> conditionMap = new HashMap<String, String>();
	conditionMap.put("제목", "TITLE");
	conditionMap.put("내용", "CONTENT");
	return conditionMap;
}

@RequestMapping("/getBoardList.do")
public String getBoardList(BoardVO vo, BoardDAO, boardDAO, Model model) {
	// Model 정보 저장
	model.addAttribute("boardList", boardDAO.getBoardList(vo));
	return "getBoardList.jsp";
}
```

- @ModelAttribute 사용하기 (2)



```
@ModelAttribute("conditionMap")
public Map searchConditionMap() {
        Map conditionMap = new HashMap();
        conditionMap.put("제목", "TITLE");
        conditionMap.put("내용", "CONTENT");
        return conditionMap;
}

@RequestMapping("/getBoardList.do")
public String getBoardList(BoardVO vo,
                BoardDAO dao, Model model) {
        model.addAttribute("boardList", dao.getBoardList(vo));
        return "getBoardList.jsp";
}
```

- @ModelAttribute 사용하기 (3)

```
<table border="1" cellpadding="0" cellspacing="0" width="700">
<tr>
        <td align="right">
                <select name="searchCondition">
                <c:forEach items="${conditionMap}" var="option">
                        <option value="${option.value}">${option.key}
                </c:forEach>
                </select>
                <input name="searchKeyword" type="text"/>
                <input type="submit" value="검색"/>
        </td>
</tr>
</table>
```

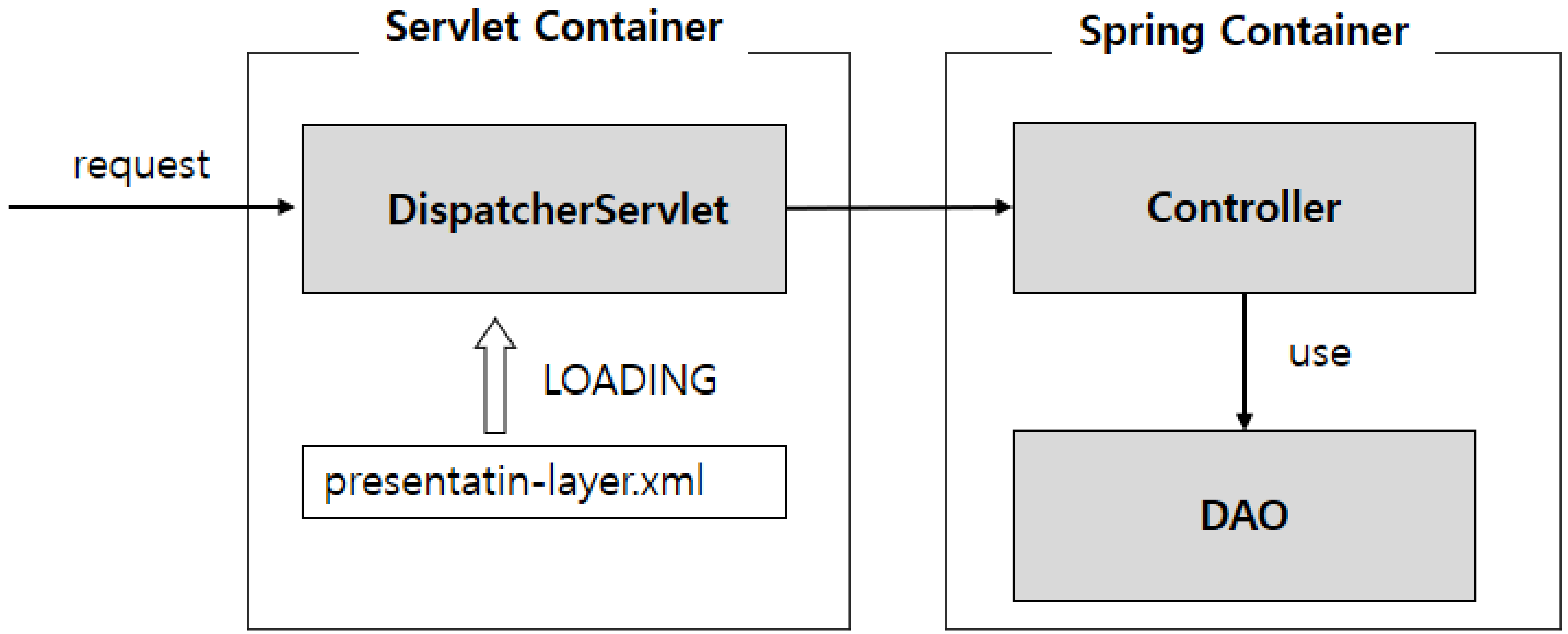
- @SessionAttribute 사용하기 (null 업데이트 방지)

```
@Controller
@SessionAttributes("board")
public class BoardController {
        @RequestMapping("/updateBoard.do")
        public String updateBoard(@ModelAttribute("board") BoardVO vo) {
                System.out.println("BoardVO 상세 정보 : " + vo.toString());
                boardDAO.updateBoard(vo);
                return "getBoardList.do";
        }

        @RequestMapping("/getBoard.do")
        public String getBoard(BoardVO vo, Model model) {
                model.addAttribute("board", boardDAO.getBoard(vo));
                return "getBoard.jsp";
        }
}
```

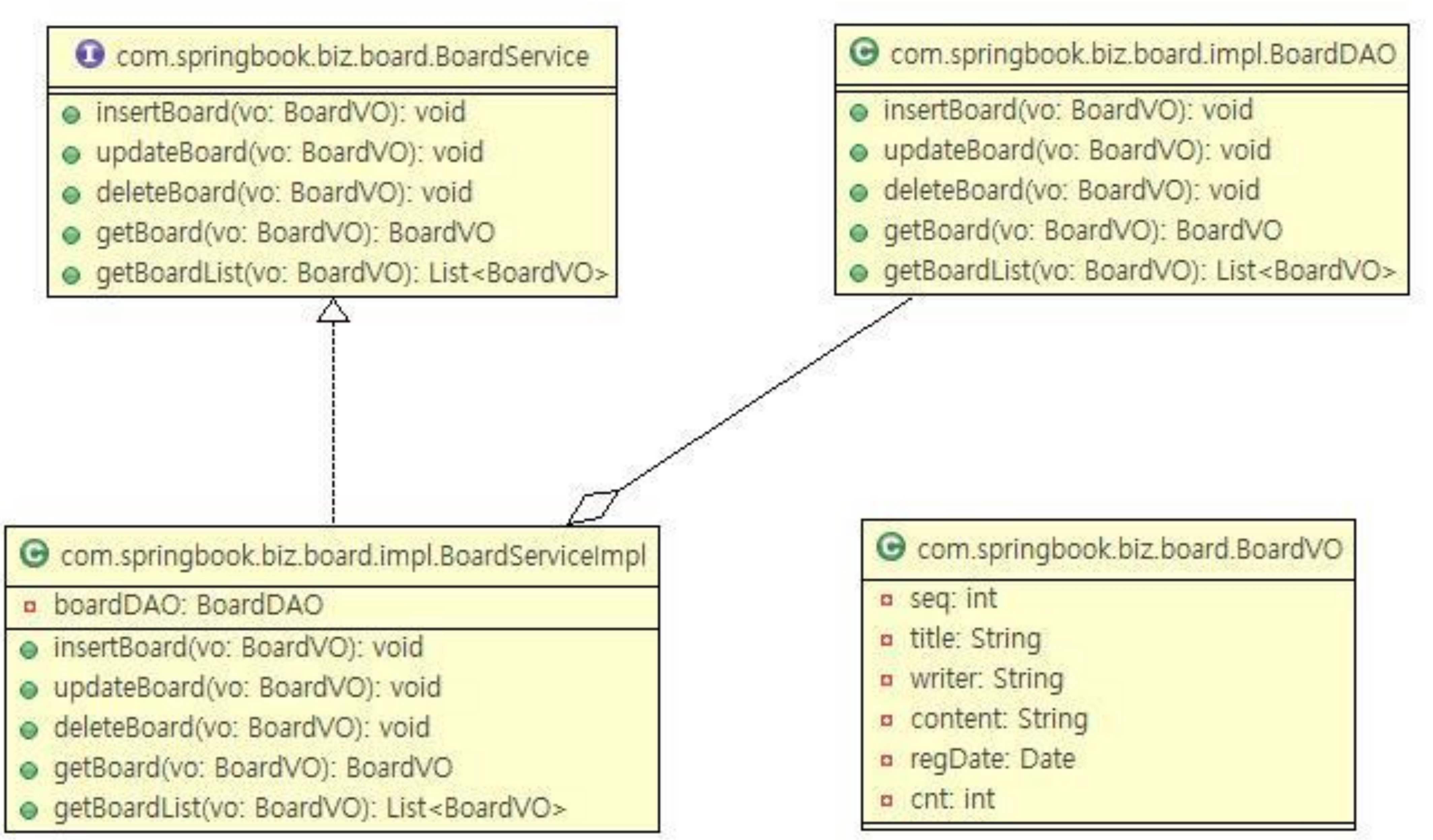
Layer 통합하기

(Presentation-Layer, Business-Layer)

- Layer 통합

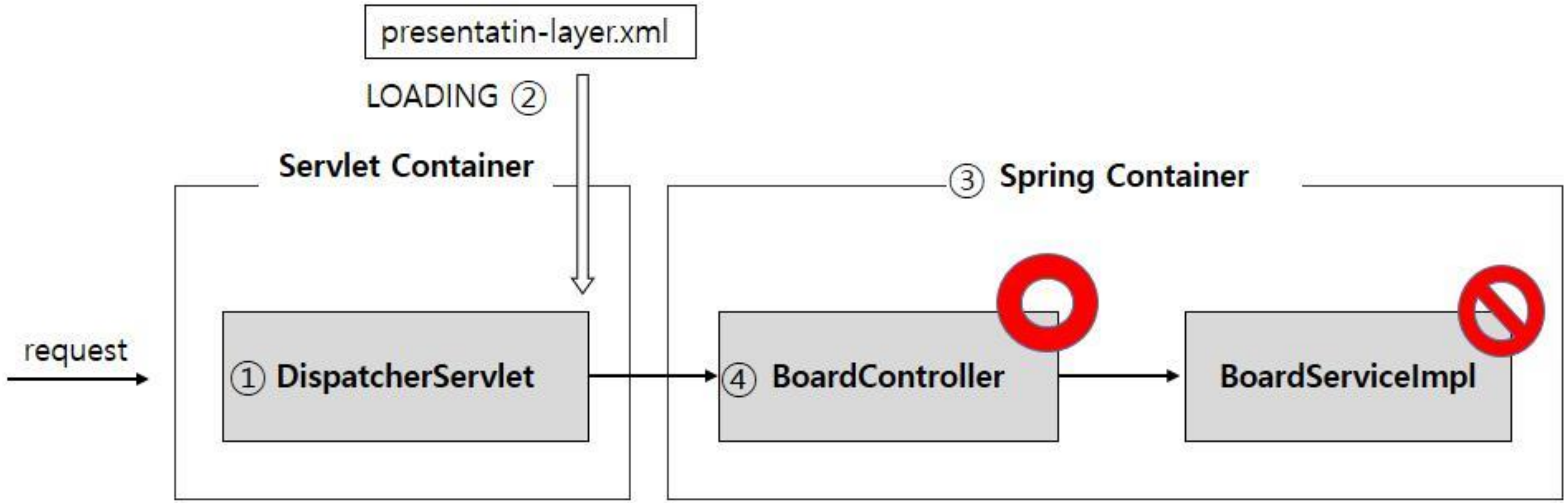

Servlet Container
Spring Container
request
DispatcherServlet
Controller
LOADING
use
presentatin-layer.xml
DAO

- Business Component 구조

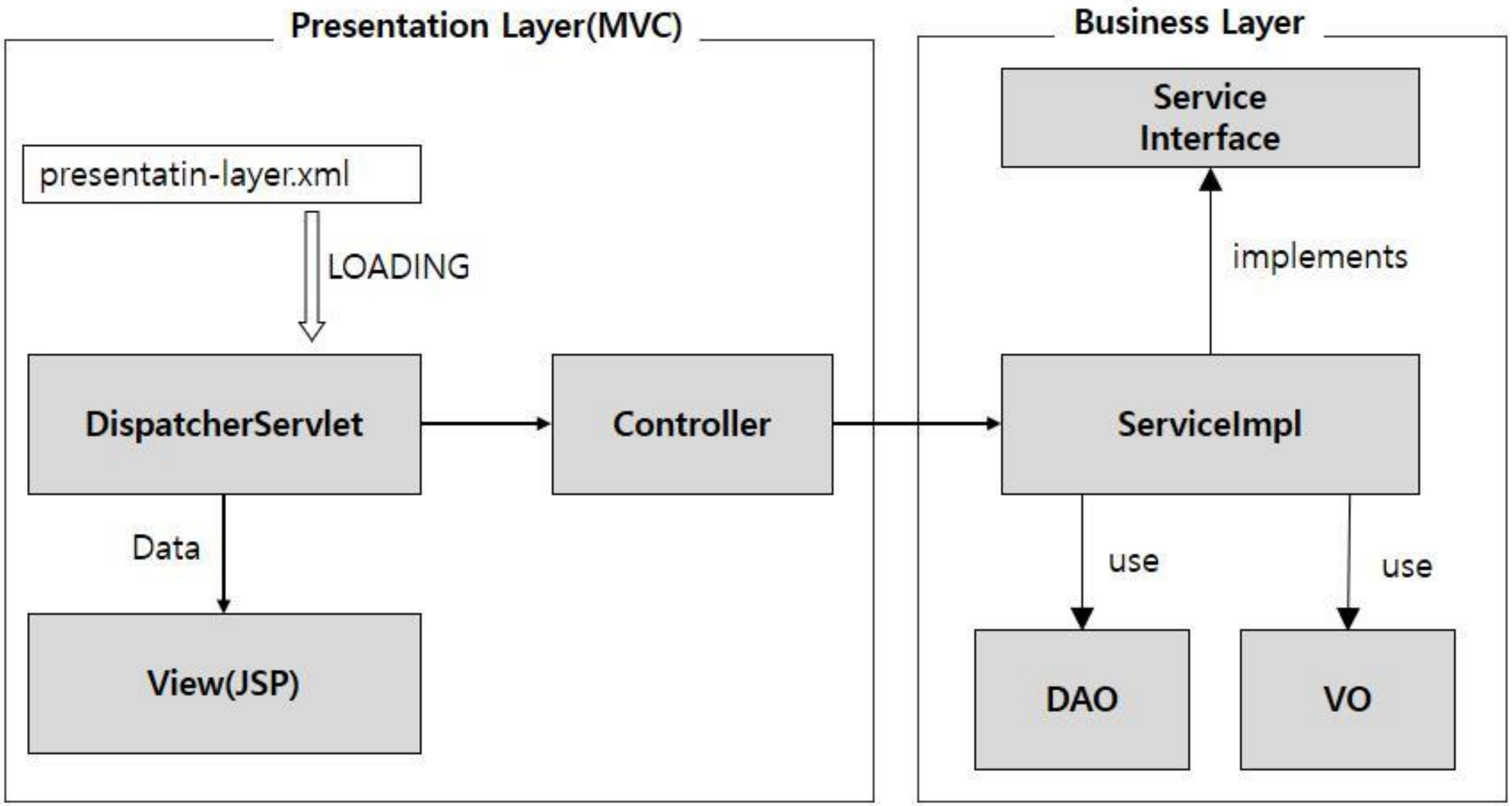


com.springbook.biz.board.BoardService
insertBoard(vo: BoardVO): void
updateBoard(vo: BoardVO): void
deleteBoard(vo: BoardVO): void
getBoard(vo: BoardVO): BoardVO
getBoardList(vo: BoardVO): List<BoardVO>
com.springbook.biz.board.impl.BoardDAO
insertBoard(vo: BoardVO): void
updateBoard(vo: BoardVO): void
deleteBoard(vo: BoardVO): void
getBoard(vo: BoardVO): BoardVO
getBoardList(vo: BoardVO): List<BoardVO>
com.springbook.biz.board.impl.BoardServiceImpl
boardDAO: BoardDAO
insertBoard(vo: BoardVO): void
updateBoard(vo: BoardVO): void
deleteBoard(vo: BoardVO): void
getBoard(vo: BoardVO): BoardVO
getBoardList(vo: BoardVO): List<BoardVO>
com.springbook.biz.board.BoardVO
seq: int
title: String
writer: String
content: String
regDate: Date
cnt: int

• Autowired 실패 원인



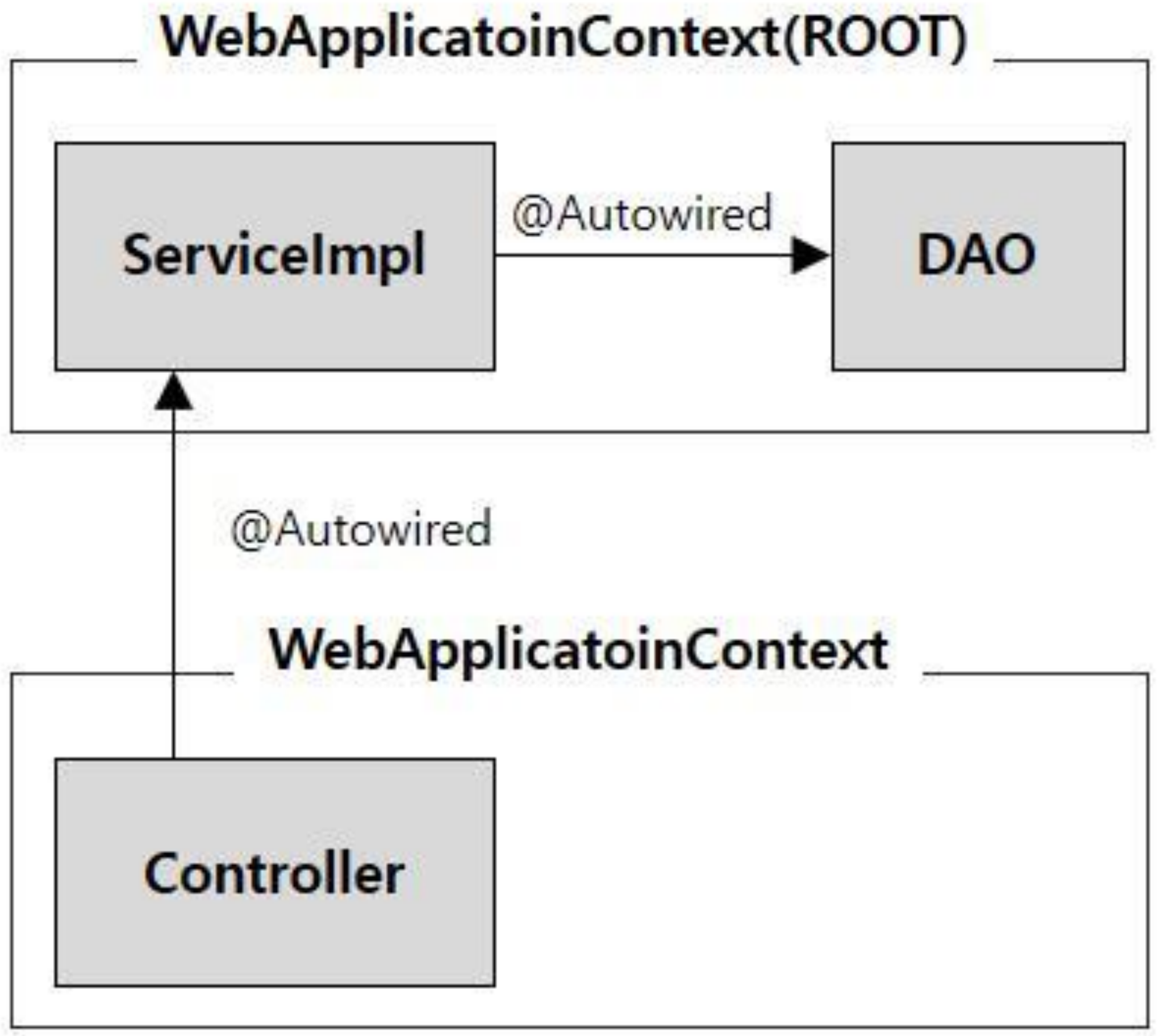
• Layer 연결


Presentation Layer(MVC)
presentatin-layer.xml
LOADING
DispatcherServlet
Controller
Data
View(JSP)
Business Layer
Service Interface
implements
ServiceImpl
use
use
DAO
VO

- ContextLoaderListener 등록 (web.xml)

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                classpath:applicationContext.xml
        </param-value>
</context-param>

<listener>
        <listener-class>
                org.springframework.web.context.ContextLoaderListener
        </listener-class>
</listener>
```

- Container의 관계



File Upload

- 업로드 화면 만들기 (insertBoard.jsp)

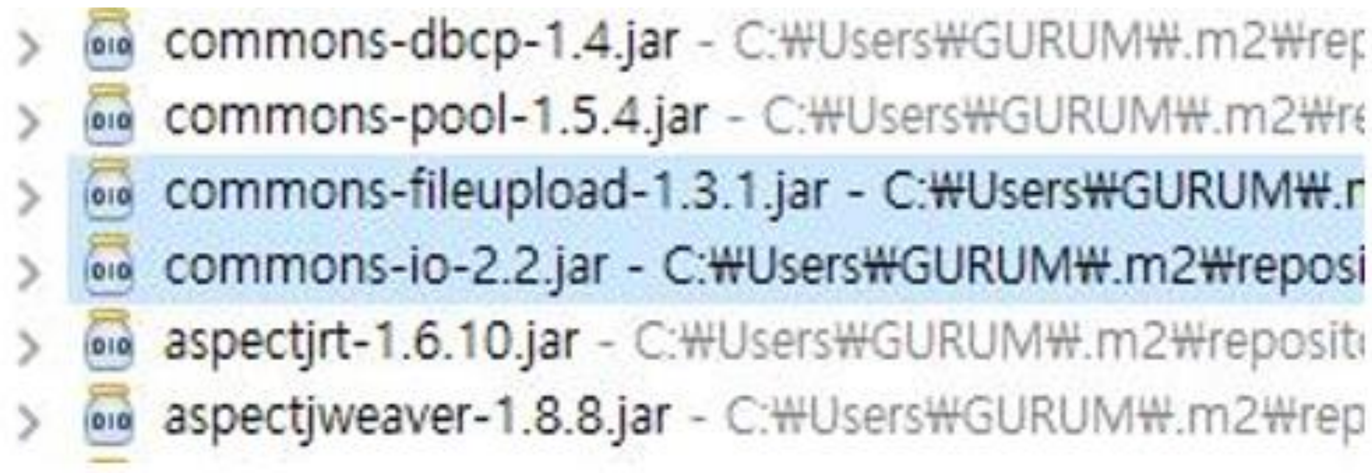
```
<form action="insertBoard.do" method="post" enctype="multipart/form-data">
<table border="1" cellpadding="0" cellspacing="0">
        <tr>
                <td bgcolor="orange" width="70">제목</td><td align="left">
                <input type="text" name="title"/></td>
        </tr>
        <tr>
                <td bgcolor="orange" width="70">업로드</td><td align="left">
                <input type="file" name="uploadFile"/></td>
        </tr>
        <tr>
                <td colspan="2" align="center">
                <input type="submit" value=" 새글 등록 "/></td>
        </tr>
</table>
</form>
```

- VO 클래스 수정 (BoardVO.java)

```
import org.springframework.web.multipart.MultipartFile;

public class BoardVO {
	private int seq;
	private String title;
	private MultipartFile uploadFile;

	~생략~
	public MultipartFile getUploadFile() {
		return uploadFile;
	}
	public void setUploadFile(MultipartFile uploadFile) {
		this.uploadFile = uploadFile;
	}
}
```

• 라이브러리 추가 (pom.xml)

```
<!-- FileUpload -->
<dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
        <version>1.3.1</version>
</dependency>
```



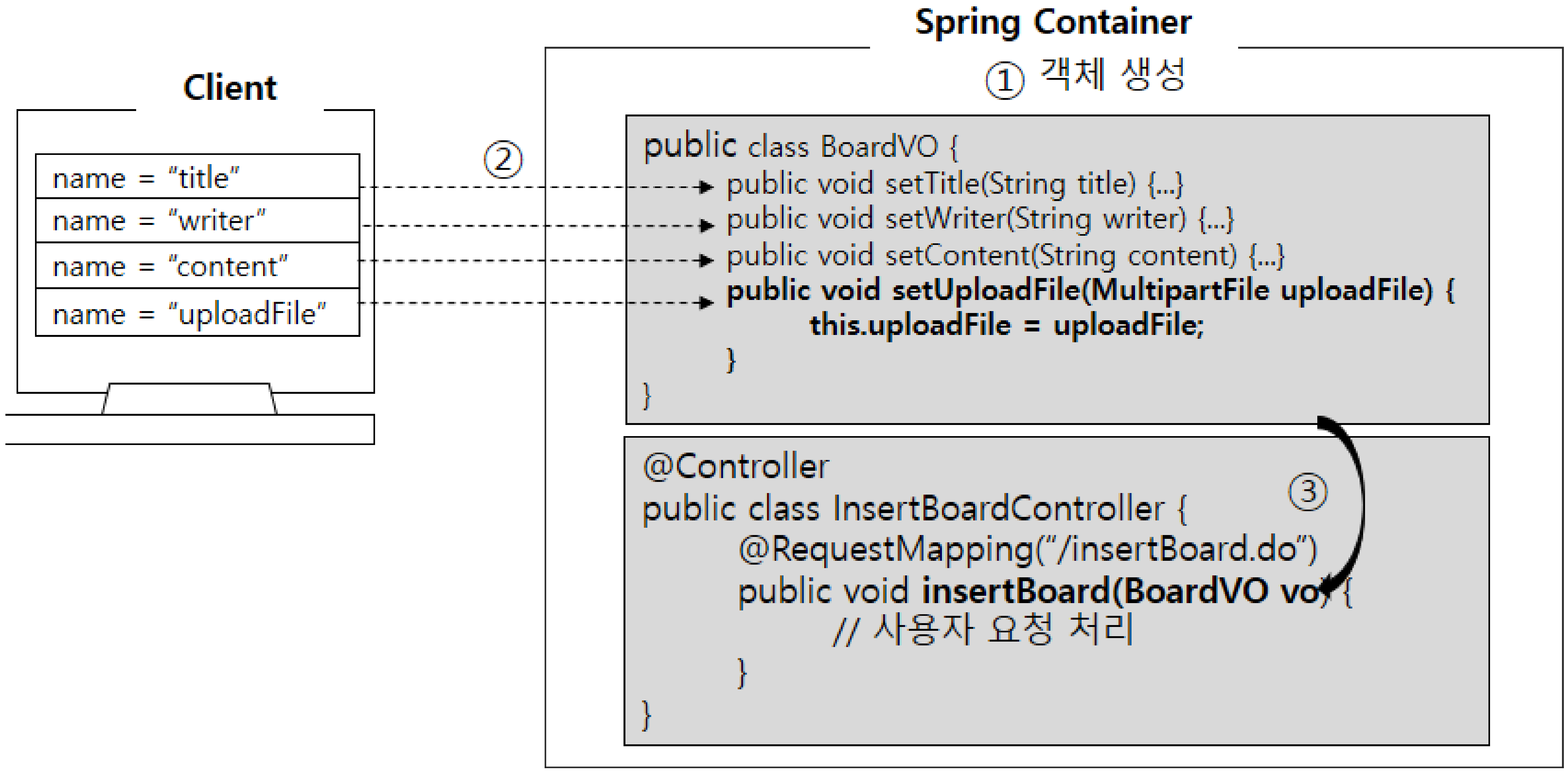
```
commons-dbcp-1.4.jar - C:₩Users₩GURUM₩.m2₩rep
commons-pool-1.5.4.jar - C:₩Users₩GURUM₩.m2₩re
commons-fileupload-1.3.1.jar - C:₩Users₩GURUM₩.r
commons-io-2.2.jar - C:₩Users₩GURUM₩.m2₩reposi
aspectjrt-1.6.10.jar - C:₩Users₩GURUM₩.m2₩reposit
aspectjweaver-1.8.8.jar - C:₩Users₩GURUM₩.m2₩rep
```

- Spring 설정파일 수정 (presentation-layer.xml)

고정된 아이디 사용

```
<!-- 파일 업로드 설정 -->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
	<property name="maxUploadSize" value="100000" />
</bean>
```

• 업로드 정보(MultipartFile)를 Command 객체에 할당하는 과정



```
public class BoardVO {
    public void setTitle(String title) {…}
    public void setWriter(String writer) {…}
    public void setContent(String content) {…}
    public void setUploadFile(MultipartFile uploadFile) {
        this.uploadFile = uploadFile;
    }
}
```

```
@Controller
public class InsertBoardController {
    @RequestMapping("/insertBoard.do")
    public void insertBoard(BoardVO vo) {
        // 사용자 요청 처리
    }
}
```

- MultipartFile 메소드

메소드	설　명
String getOriginalFilename()	업로드한 파일명을 문자열로 리턴
void transferTo(File destFile)	업로드한 파일을 destFile에 저장
boolean isEmpty()	업로드한 파일 존재 여부 리턴(없으면 true 리턴)

- 업로드 처리 (BoardController.java)

```
 // 글 등록
@RequestMapping("/insertBoard.do")
public String insertBoard(BoardVO vo) throws IOException {
        // 파일 업로드 처리
        MultipartFile uploadFile = vo.getUploadFile();
        if(!uploadFile.isEmpty()) {
                String fileName = uploadFile.getOriginalFilename();
                uploadFile.transferTo(new File("C:/" + fileName));
        }

        boardService.insertBoard(vo);
        return "getBoardList.do";
}
```

Exceptioin Handle

• Annotation 기반 (presentation-layer.xml)

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xsi:schemaLocation="http://www.springframework.org/schema/mvc
                http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
                http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd">

    <context:component-scan base-package="com.springbook.view"/>

    <mvc:annotation-driven/>

</beans>
```

- Exception Handler 작성(CommonExcpetionHandler.java)

```
@ControllerAdvice("com.springbook.view")
public class CommonExceptionHandler {
	@ExceptionHandler(ArithmeticException.class)
	public ModelAndView handleArithmeticException(Exception e) {
		ModelAndView mav = new ModelAndView();
		mav.addObject("exception", e);
		mav.setViewName("/common/arithmeticError.jsp");
		return mav;
	}

	@ExceptionHandler(Exception.class)
	public ModelAndView handleException(Exception e) {
		ModelAndView mav = new ModelAndView();
		mav.addObject("exception", e);
		mav.setViewName("/common/error.jsp");
		return mav;
	}
}
```

- Exception Handler Annotation

  - @ControllerAdvice("com.springbook.view")
    - CommonExceptionHandler 객체를 생성하는 Annotation
    - "com.springbook.view" 패키지로 시작하는 컨트롤러에서 예외 발생 시
      @ExceptionHandler 어노테이션으로 지정한 메소드 실행

  - ArithmeticException이 발생하면 handleArithmeticException( ) 메소드 실행

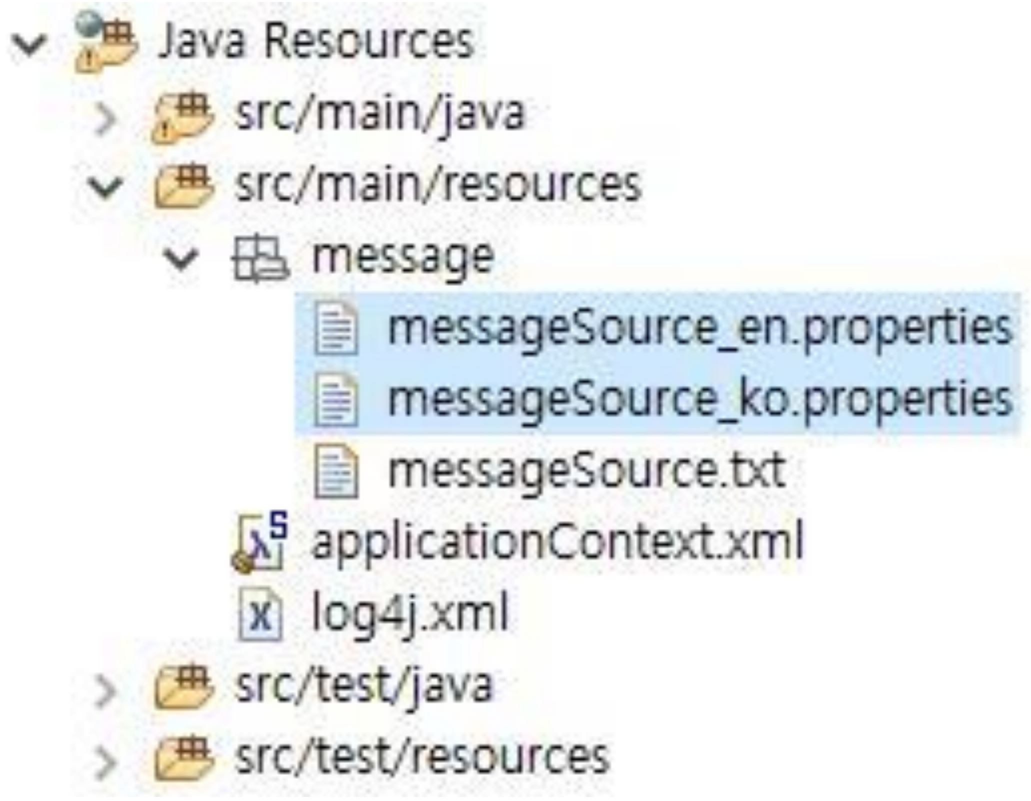
  - handleException( ) 메소드는 기본 예외 핸들러

- XML 기반 (presentation-layer.xml)

```
<bean id="exceptionResolver"
class="org.springframework.web.servlet.handler.SimpleMappingExceptionResol
ver">
        <property name="exceptionMappings">
                <props>
                        <prop key="java.lang.ArithmeticException">
                                common/arithmeticError.jsp
                        </prop>
                        <prop key="java.lang.NullPointerException">
                                common/nullPointerError.jsp
                        </prop>
                </props>
        </property>

        <property name="defaultErrorView" value="common/error.jsp" />
</bean>
```

다국어(국제화)

- Message 파일 작성



```
# login.jsp
message.user.login.title=LOGIN
message.user.login.id=ID
message.user.login.password=PASSWORD
message.user.login.loginBtn=LOG-IN
```

- MessageSource 등록

```
<!-- MessageSource 등록 -->
<bean id="messageSource"
        class="org.springframework.context.support.ResourceBundleMessageSource">
         <property name="basenames">
                  <list>
                           <value>message.messageSource</value>
                  </list>
         </property>
</bean>
```

- LocaleResolver 등록

LocaleResolver 종류	기능 설명
AcceptHeaderLocaleResolver	브라우저에서 전송된 HTTP 요청 헤더에서 Accept-Language에 설정된 Locale로 메시지를 적용한다.
CookieLocaleResolver	Cookie에 저장된 Locale 정보를 추출하여 메시지를 적용한다.
SessionLocaleResolver	HttpSession에 저장된 Locale 정보를 추출하여 메시지를 적용한다.
FixedLocaleResolver	웹 요청과 무관하게 특정 Locale로 고정한다.

- Locale 변경하기

```
<?xml version="1.0" encoding="UTF-8"?>
<beans mvc 네임스페이스 등록>

    <!-- LocaleResolver 등록 -->
    <bean id="localeResolver"
        class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
    </bean>

    <!-- LocaleChangeInterceptor 등록 -->
    <mvc:interceptors>
        <bean
class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
            <property name="paramName" value="lang"/>
        </bean>
    </mvc:interceptors>
</beans>
```

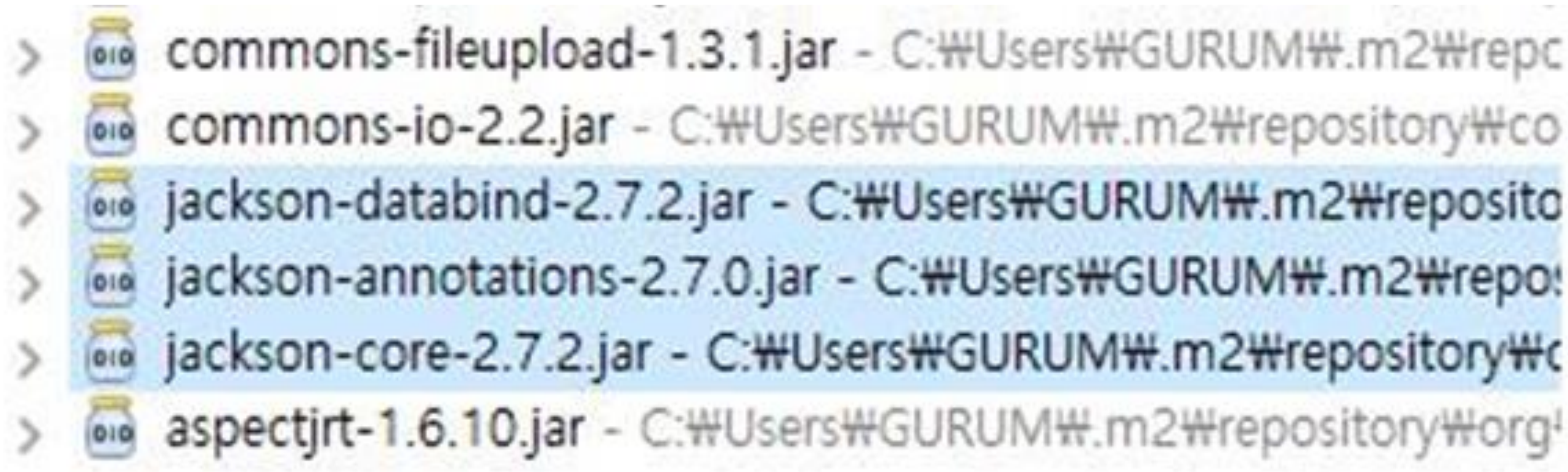
- 다국어 적용

```
<%@page contentType="text/html; charset=EUC-KR"%>
<%@taglib uri="http://www.springframework.org/tags" prefix="spring" %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title><spring:message code="Message Key"/></title>
</head>
<body>
<center>
~~~
</center>
</body>
</html>
```

JSON 변환

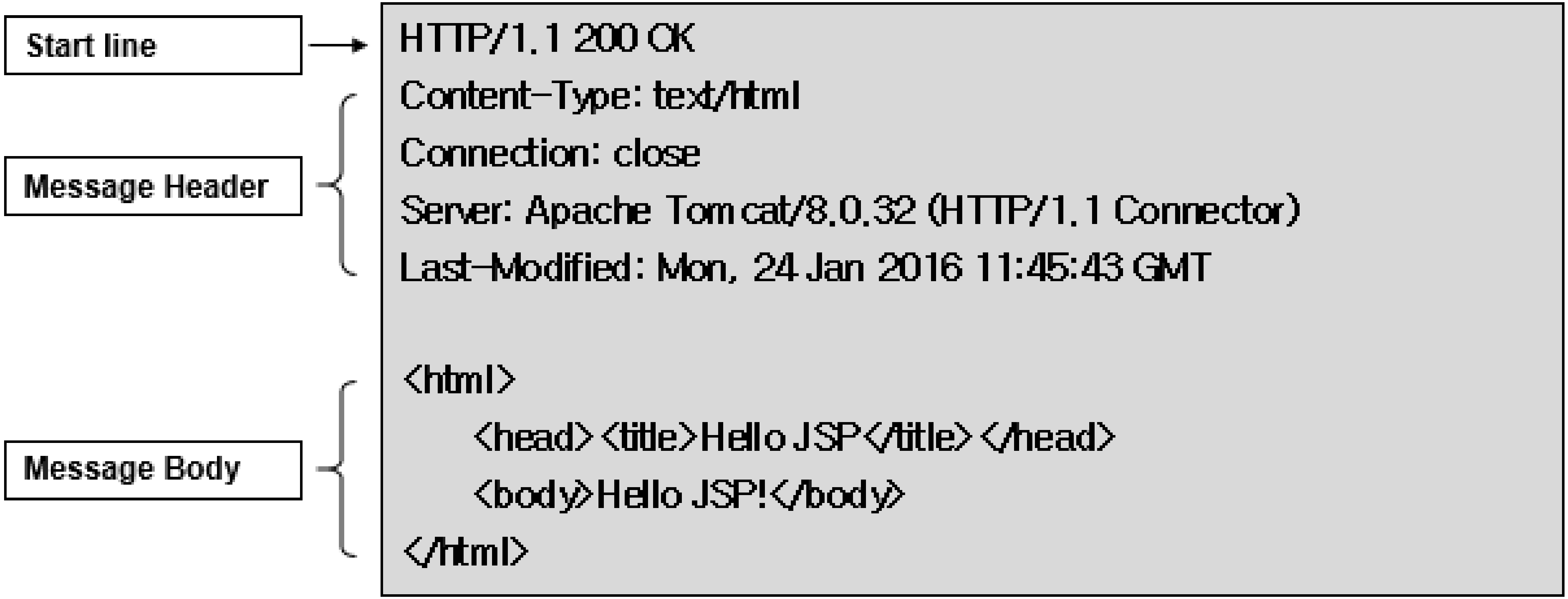
• 라이브러리 추가 (pom.xml)

```
<!-- Jackson2 -->
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-databind</artifactId>
	<version>2.7.2</version>
</dependency>
```



```
> commons-fileupload-1.3.1.jar - C:₩Users₩GURUM₩.m2₩repo
> commons-io-2.2.jar - C:₩Users₩GURUM₩.m2₩repository₩co
> jackson-databind-2.7.2.jar - C:₩Users₩GURUM₩.m2₩reposito
> jackson-annotations-2.7.0.jar - C:₩Users₩GURUM₩.m2₩repo
> jackson-core-2.7.2.jar - C:₩Users₩GURUM₩.m2₩repository₩
> aspectjrt-1.6.10.jar - C:₩Users₩GURUM₩.m2₩repository₩org
```

- HTTP 응답 프로토콜 구조



```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: close
Server: Apache Tomcat/8.0.32 (HTTP/1.1 Connector)
Last-Modified: Mon, 24 Jan 2016 11:45:43 GMT

<html>
      <head><title>Hello JSP</title></head>
      <body>Hello JSP!</body>
</html>
```

- MessageConverter 등록

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:mvc="http://www.springframework.org/schema/mvc"
	xsi:schemaLocation="http://www.springframework.org/schema/mvc
		http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
		http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context
		http://www.springframework.org/schema/context/spring-context-4.2.xsd">

	<mvc:annotation-driven></mvc:annotation-driven>

</beans>
```

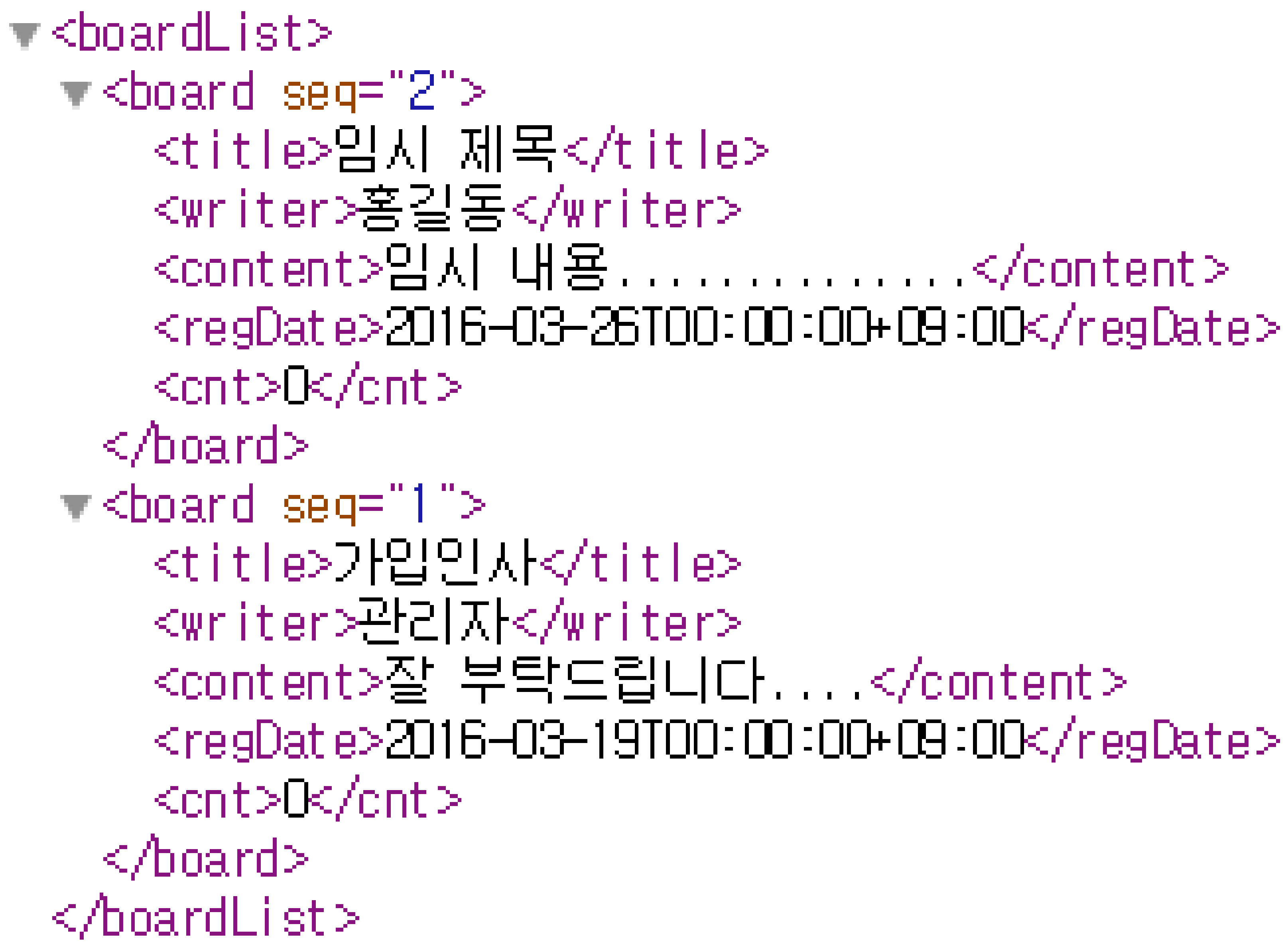
- Controller 수정

```
@Controller
@SessionAttributes("board")
public class BoardController {
    @Autowired
    private BoardService boardService;

    @RequestMapping("/dataTransform.do")
    @ResponseBody
    public List<BoardVO> dataTransform(BoardVO vo) {
        vo.setSearchCondition("TITLE");
        vo.setSearchKeyword("");
        List<BoardVO> boardList = boardService.getBoardList(vo);
        return boardList;
    }
}
```

객체를 HTTP 응답 몸체(Body)로 변환

XML 변환

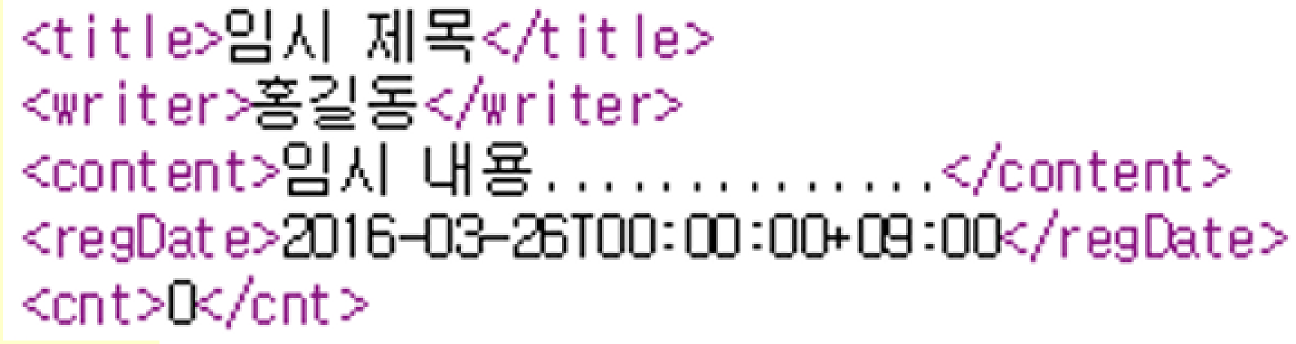
- 목적 XML 데이터 파일



```
▼<boardList>
  ▼<board seq="2">
     <title>임시 제목</title>
     <writer>홍길동</writer>
     <content>임시 내용...............</content>
     <regDate>2016-03-26T00:00:00+09:00</regDate>
     <cnt>0</cnt>
   </board>
  ▼<board seq="1">
     <title>가입인사</title>
     <writer>관리자</writer>
     <content>잘 부탁드립니다....</content>
     <regDate>2016-03-19T00:00:00+09:00</regDate>
     <cnt>0</cnt>
   </board>
 </boardList>
```

- JAXB Annotation 설정

```
@XmlAccessorType(XmlAccessType.FIELD)
public class BoardVO {
        @XmlAttribute
        private int seq;
        private String title;
        private String writer;
        private String content;
        private Date regDate;
        private int cnt;
        @XmlTransient
        private String searchCondition;
        @XmlTransient
        private String searchKeyword;
        @XmlTransient
        private MultipartFile uploadFile;
}
```



```
<title>임시 제목</title>
<writer>홍길동</writer>
<content>임시 내용..............</content>
<regDate>2016-03-26T00:00:00+09:00</regDate>
<cnt>0</cnt>
```

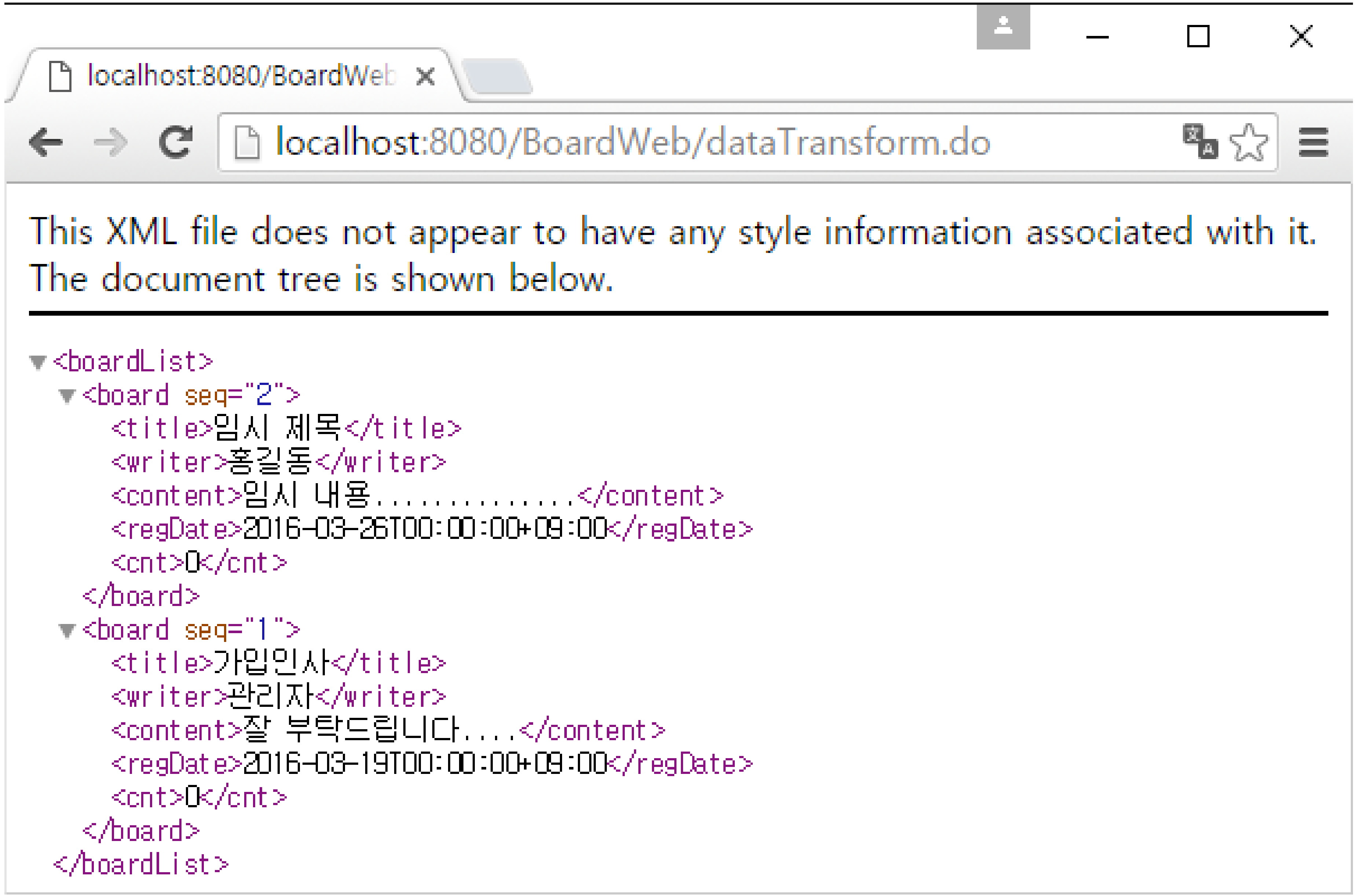
- JAXB Annotation 설정

```
@XmlRootElement(name = "boardList")
@XmlAccessorType(XmlAccessType.FIELD)
public class BoardListVO {
	@XmlElement(name = "board")
	private List<BoardVO> boardList;

	public List<BoardVO> getBoardList() {
		return boardList;
	}

	public void setBoardList(List<BoardVO> boardList) {
		this.boardList = boardList;
	}
}
```

- Controller 수정 (BoardController.java)

```
@Controller
@SessionAttributes("board")
public class BoardController {
    @Autowired
    private BoardService boardService;

    @RequestMapping("/dataTransform.do")
    @ResponseBody
    public BoardListVO dataTransform(BoardVO vo) {
        vo.setSearchCondition("TITLE");
        vo.setSearchKeyword("");
        List<BoardVO> boardList = boardService.getBoardList(vo);
        BoardListVO boardListVO = new BoardListVO();
        boardListVO.setBoardList(boardList);
        return boardListVO;
    }
```

- 실행 결과



localhost:8080/BoardWeb/dataTransform.do

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<boardList>
  ▼<board seq="2">
     <title>임시 제목</title>
     <writer>홍길동</writer>
     <content>임시 내용...............</content>
     <regDate>2016-03-26T00:00:00+09:00</regDate>
     <cnt>0</cnt>
   </board>
  ▼<board seq="1">
     <title>가입인사</title>
     <writer>관리자</writer>
     <content>잘 부탁드립니다....</content>
     <regDate>2016-03-19T00:00:00+09:00</regDate>
     <cnt>0</cnt>
   </board>
 </boardList>
```