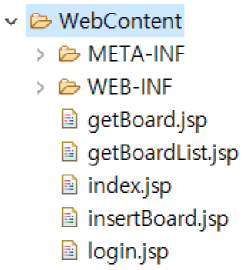
MVC

WebContent 폴더에 넣어야 브라우저에서 요청할 수 있다.

WEB-INF 폴더에는 xml환경설정이나 라이브러리같은 브라우저가 요청할 수 없고 해서는 안되는 파일들을 감추기 위해서 사용하는 폴더.



jsp 문법 🍦

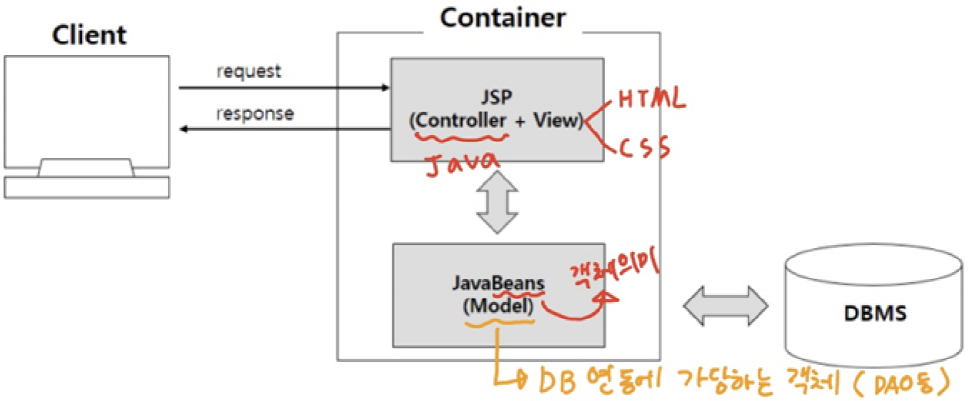
scriptlet (스클립트 릿) = 자바 코드를 묶을 때 씀 <% %>

```
<%
    // 1. 사용자 입력정보 추출
    String id = request.getParameter("id");
    String password = request.getParameter("password");

    // 2. DB 연동 처리
    UserVO vo = new UserVO();
    vo.setId(id);
    vo.setPassword(password);

    UserDAOJDBC dao = new UserDAOJDBC();
    UserVO user = dao.getUser(vo);

    // 3. 화면 이동

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional
```

expression (익스프레션) = 값 넣을 때 <%= %>

```
‹%= user.getName() %>님</font> 환영합니다.‹
```

모델 1

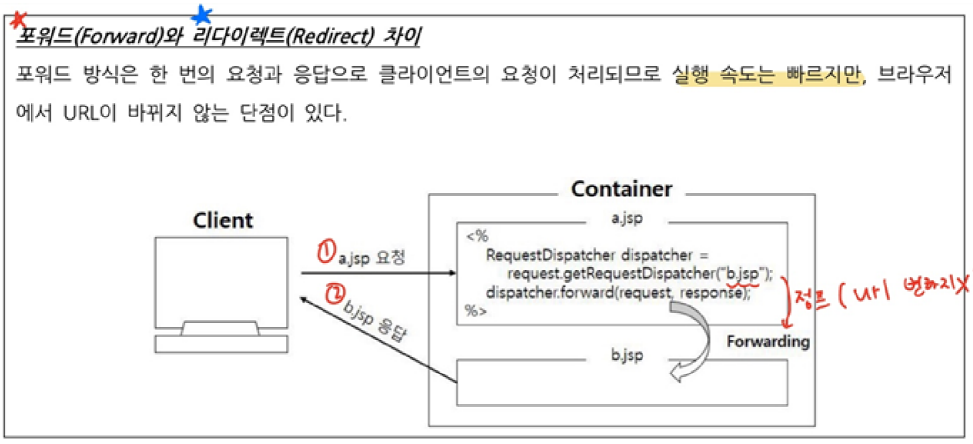


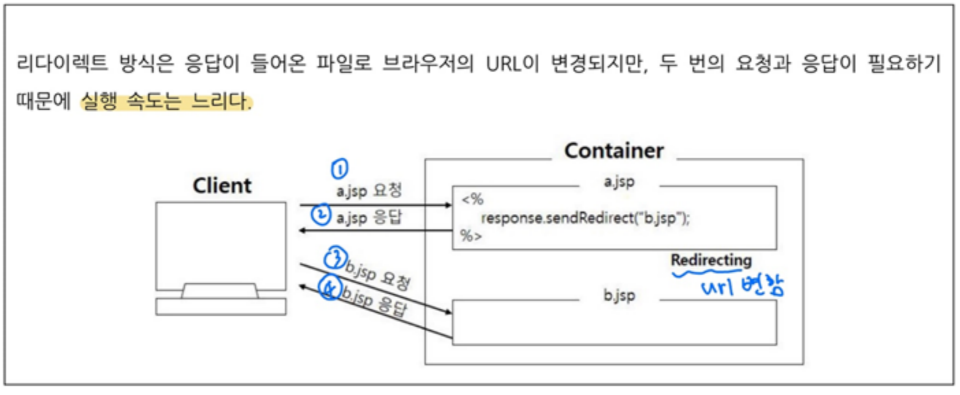
JSP 파일 구조

- 자바 코드와 디자인 관련된 소스들이 뒤섞여 있음.
- 웹디는 자바하기 싫어하고, 자바 개발자는 구림,,
  - 해결법 - 모델 2

화면 네비게이션 방법

포워드와 리다이렉트 차이





대신 forword는 연결되어 있으므로 request로 a, b가 정보 공유를 할 수 있다.

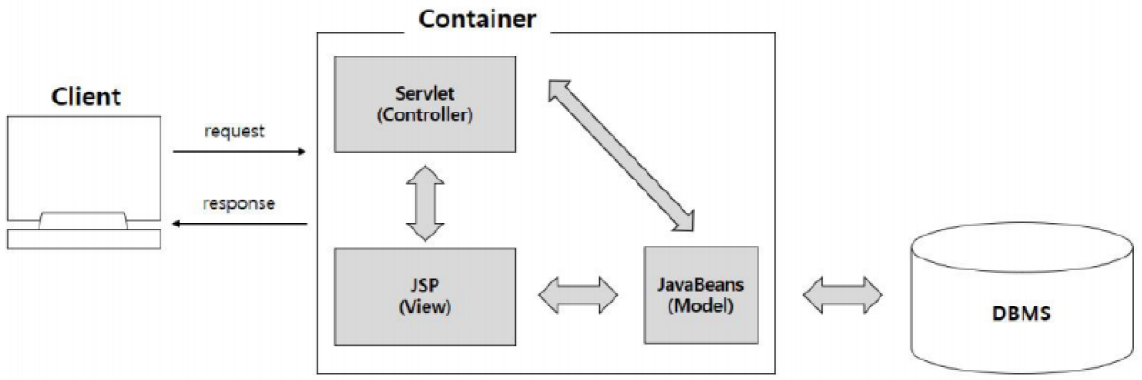
→ redirect는 그래서 session으로 정보 공유를 하는 방법밖에는 없다.

→ 근데 session은 하나만 있으니까 메모리가 너무 커지면 안 좋으므로 안 쓸수록 좋다.

- 로그인 정보 정도만 저장하기.

결론 : 비록 url 주소는 바뀌지 않지만 requst를 통해서 정보공유를 할 수 있으니까 따라서 포워드로 화면을 이동하는 게 좋다. → 세션으로 하는 건 위험하다.

모델 2



jsp파일에 있는 java코드를 controller로 옮기기 위해 모델 2를 만들어 봄.

DispatcherServlet.java : controller 파일

▼ 이런 게 controller 코드임. - 기존에 jsp에 있던 코드 옮겨서 DB연관된 애들은 .do로 가도록 만듦.

```
package com.multicampus.controller;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.multicampus.biz.board.BoardDAOJDBC;
import com.multicampus.biz.board.BoardVO;
import com.multicampus.biz.user.UserDAOJDBC;
import com.multicampus.biz.user.UserVO;

public class DispatcherServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

    public DispatcherServlet() {
        System.out.println("===> DispatcherServlet 생성");
    }

  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // 0. 인코딩 설정
    request.setCharacterEncoding("EUC-KR");

    // 1. 클라이언트 (브라우저)의 요청 path 정보를 추출한다 .
    String path = request.getRequestURI();
    System.out.println("path : " + path);

    // 2. 요청 path에 따라 로직을 분기처리한다 .
    if(path.equals("/login.do")) {
      System.out.println("로그인 기능 처리");

      // 1. 사용자 입력정보 추출
      String id = request.getParameter("id");
      String password = request.getParameter("password");

      // 2. DB 연동 처리
```

```
      UserVO vo = new UserVO();
      vo.setId(id);
      vo.setPassword(password);

      UserDAOJDBC dao = new UserDAOJDBC();
      UserVO user = dao.getUser(vo);

      // 3. 화면 이동
      if(user != null) {  // 로그인 성공
        HttpSession session = request.getSession();
        session.setAttribute("user", user);

        response.sendRedirect("getBoardList.do");
      } else {      // 로그인 실패
        response.sendRedirect("login.jsp");
      }

    } else if(path.equals("/logout.do")) {
      System.out.println("로그아웃 기능 처리 ");

      // 로그아웃 요청한 브라우저와 매핑된 세션을 강제 종료한다 .
      HttpSession session = request.getSession();
      session.invalidate();

      response.sendRedirect("/");

    } else if(path.equals("/insertBoard.do")) {
      System.out.println("글 등록 기능 처리 ");

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

      // 2. DB 연동 처리
      BoardVO vo = new BoardVO();
      vo.setTitle(title);
      vo.setWriter(writer);
      vo.setContent(content);

      BoardDAOJDBC dao = new BoardDAOJDBC();
      dao.insertBoard(vo);

      // 3. 화면 이동
      response.sendRedirect("getBoardList.do");

    } else if(path.equals("/updateBoard.do")) {
      System.out.println("글 수정 기능 처리 ");

      // 1. 사용자 입력정보 추출
      String title = request.getParameter("title");
      String seq = request.getParameter("seq");
      String content = request.getParameter("content");

      // 2. DB 연동 처리
      BoardVO vo = new BoardVO();
      vo.setTitle(title);
      vo.setSeq(Integer.parseInt(seq));
      vo.setContent(content);

      BoardDAOJDBC dao = new BoardDAOJDBC();
      dao.updateBoard(vo);

      // 3. 화면 이동
      response.sendRedirect("getBoardList.do");

    } else if(path.equals("/deleteBoard.do")) {
      System.out.println("글 삭제 기능 처리 ");

      // 1. 사용자 입력정보 추출
      String seq = request.getParameter("seq");

      // 2. DB 연동 처리
      BoardVO vo = new BoardVO();
      vo.setSeq(Integer.parseInt(seq));

      BoardDAOJDBC dao = new BoardDAOJDBC();
      dao.deleteBoard(vo);
```

```
      // 3. 화면 이동
      response.sendRedirect("getBoardList.do");

    } else if(path.equals("/getBoard.do")) {
      System.out.println("글 상세 조회 기능 처리");

      // 1. 사용자 입력정보 추출
      String seq = request.getParameter("seq");

      // 2. DB 연동 처리
      BoardVO vo = new BoardVO();
      vo.setSeq(Integer.parseInt(seq));

      BoardDAOJDBC dao = new BoardDAOJDBC();
      BoardVO board = dao.getBoard(vo);

      // 3. Model(DAO)을 이용하여 검색한 데이터를  View(JSP)가 사용할 수 있도록 세션에 검색 결과를 등록한다 .
      HttpSession session = request.getSession();
      session.setAttribute("board", board);

      response.sendRedirect("getBoard.jsp");

    } else if(path.equals("/getBoardList.do")) {
      System.out.println("글 목록 검색 기능 처리");

      // 1. DB 연동 처리
      BoardVO vo = new BoardVO();
      BoardDAOJDBC dao = new BoardDAOJDBC();
      List<BoardVO> boardList = dao.getBoardList(vo);

      // 2. Model(DAO)을 이용하여 검색한 데이터들  View(JSP)가 사용할 수 있도록 세션에 검색 결과를 등록한다 .
      HttpSession session = request.getSession();
      session.setAttribute("boardList", boardList);

      response.sendRedirect("getBoardList.jsp");

    } else {
      System.out.println("요청  URL을 다시 확인하세요 .");
    }
  }

}
```

위에 보다시피 코드가 너무 길어짐. - 너무 많은 기능 담당 → 그래서 spring MVC를 배워야함 (유지보수 편의성 향상)

EL - Expression Language 란?

JSP 내장객체 (requestt, session, application)에 등록된 데이터에 접근하기 위한 표현 언어

JSTL - JSP Standard Tag Library 란?

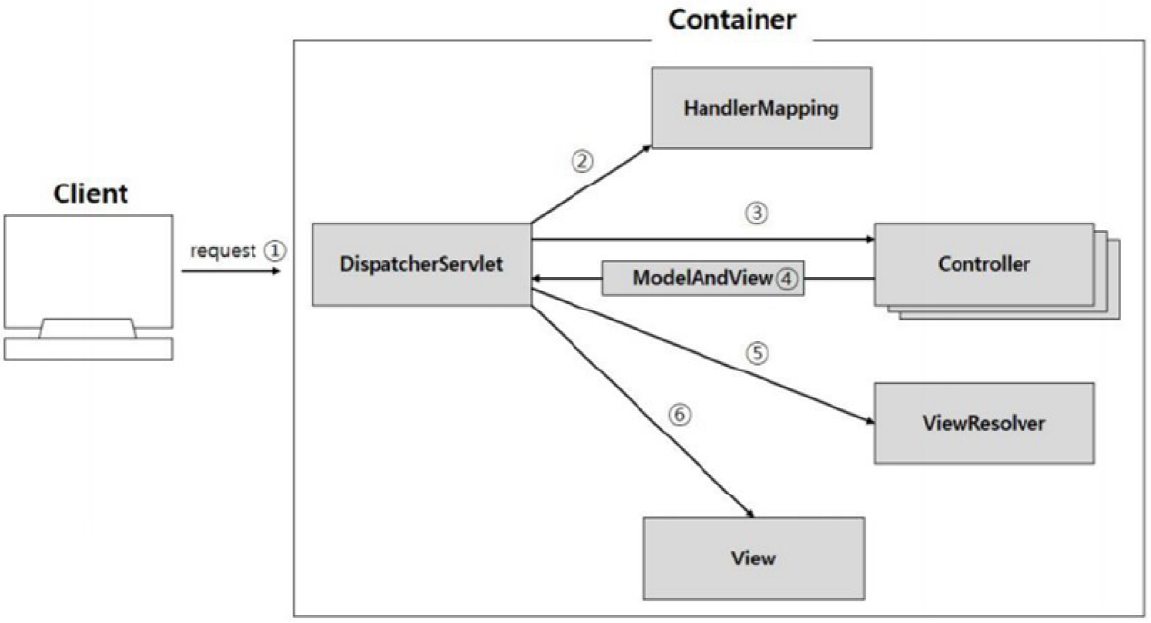
JSP 파일에서 if, for, switch 등과 같은 자바 코드를 대체하는 표준 태그

<%@taglib 어쩌고 %> 사용해야됨.

정리

기 능	구성 요소	개발 주체
Model	DAO, VO 클래스	자바 개발자
View	JSP 페이지	웹 디자이너
Controller	Servlet 클래스	자바 개발자 or MVC 프레임워크

Spring MVC⛱️

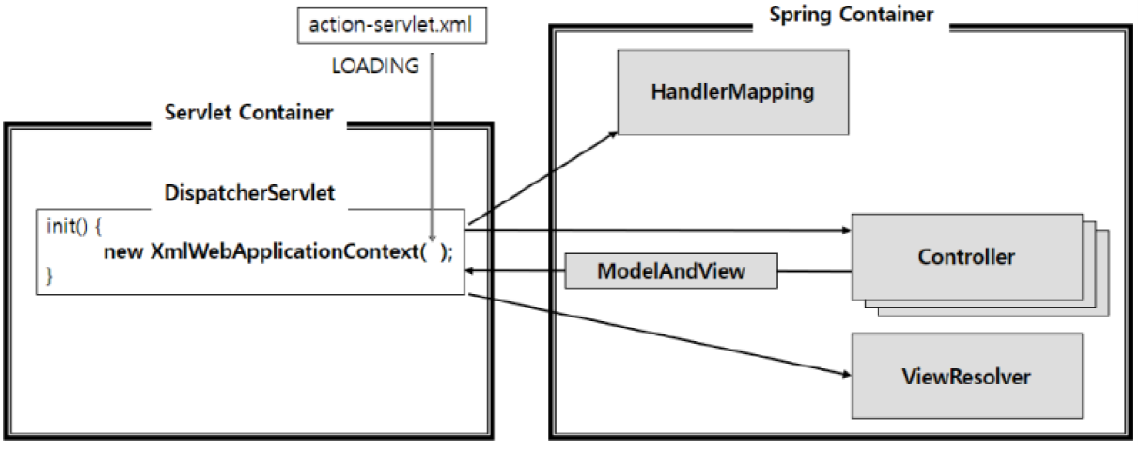


Model = DAO

View = JSP

Controller = 나머지

나머지는 다 class 인데 DispatcherServlet한 Servlet이다.



1. tomcat서버가 생성이 되면 servlet컨테이너를 생성함. 그럼 servlet컨테이너가 web.xml파일을 로딩을 합니다.
2. 그 다음 브라우저를 눌러서 tomcat서버한테 클라이언트가 login.do를 요청을 하면 dispatcherServlet 클래스가 실행이 됩니다.
3. 근데 dispatcherServlet가 다른 일반 자바 클래스로 만든 컨트롤러들이랑 협업을 하기 때문에 필요한 class를 찾아야 함.
4. 근데 다른 class controller는 메모리에 안 뜹니다. 그래서 바로바로 bean등록을 하면 나머지 애들이 메모리에 뜨겠네~

a. □□□ □□ key⚡ □□□

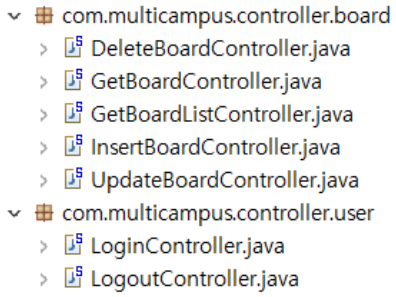
□□

1. DispatcherServlet web.xml□ □□

web.xml | action-servlet.xml | InsertBoardController.java | LoginController.java | LogoutController.java | business-layer.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">

    <display-name>BoardWeb(Model2)</display-name>

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

</web-app>
```

2. class controller □□□□ □□



3. □□□ controller□□ action-servlet.xml□ □□

web.xml | action-servlet.xml | InsertBoardController.java | LoginController.java | LogoutController.java | business-layer.xml

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

    <!-- 모든 controller 클래스들을 등록한다. -->
    <bean id="Login"       class="com.multicampus.controller.user.LoginController"></bean>
    <bean id="Logout"      class="com.multicampus.controller.user.LogoutController"></bean>
    <bean id="insertBoard" class="com.multicampus.controller.board.InsertBoardController"></bean>
    <bean id="updateBoard" class="com.multicampus.controller.board.UpdateBoardController"></bean>
    <bean id="deleteBoard" class="com.multicampus.controller.board.DeleteBoardController"></bean>
    <bean id="getBoard"    class="com.multicampus.controller.board.GetBoardController"></bean>
    <bean id="getBoardList" class="com.multicampus.controller.board.GetBoardListController"></bean>

    <!-- HandlerMapping 등록 -->
    <bean id="handlerMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"></bean>

</beans>
```

4. □□□ HandlerMapping□ □□□□□ □□

```
<!-- 모든 controller 클래스들을 등록한다. -->
<bean id="Login"         class="com.multicampus.controller.user.LoginController"></bean>
<bean id="Logout"        class="com.multicampus.controller.user.LogoutController"></bean>
<bean id="insertBoard"   class="com.multicampus.controller.board.InsertBoardController"></bean>
<bean id="updateBoard"   class="com.multicampus.controller.board.UpdateBoardController"></bean>
<bean id="deleteBoard"   class="com.multicampus.controller.board.DeleteBoardController"></bean>
<bean id="getBoard"      class="com.multicampus.controller.board.GetBoardController"></bean>
<bean id="getBoardList" class="com.multicampus.controller.board.GetBoardListController"></bean>

<!-- HandlerMapping 등록 -->
<bean id="handlerMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="/login.do">login</prop>
            <prop key="/getBoardList.do">getBoardList</prop>
        </props>
    </property>
</bean>
```

1. 브라우저에서 *.do 요청을 보내면 *컨트롤러가 실행되도록 mapping을 해주는 놈
2. 저거 사이에 handlerMapping 끝에 두 글자 앞에를 소문자로 해야 함. 아니면 오류남

```
<!-- HandlerMapping 등록 -->
<bean id="handlerMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"></bean>
```

한글깨질때 - 인코딩 처리 encoding - 177쪽인가에 있음

게시글 목록

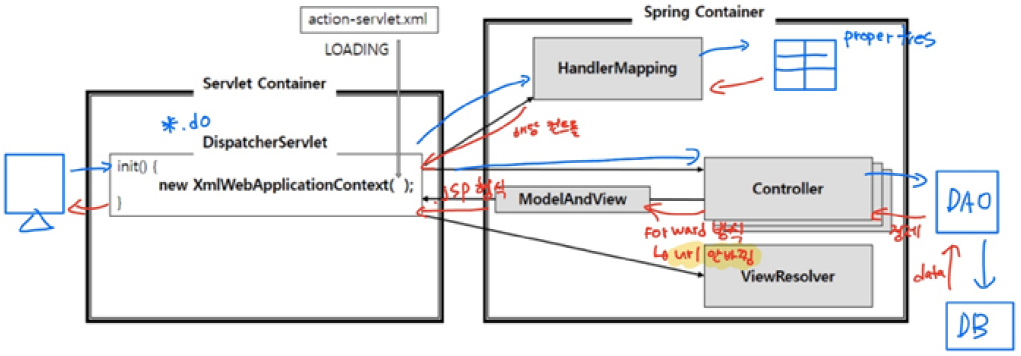
홍길동님 환영합니다!! LOG-OUT

제목 검색

번호	제목	작성자	등록일	조회수
21	¤±¤¤¤·¤?¤·¤±¤?		2023-09-13	0
20	??		2023-09-13	0

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncoding</filter-class>
</filter>
<filter-mapping>
    <filter-name>action</filter-name>
    <url-pattern>*.do</url-pattern>
</filter-mapping>

<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

흐름 정리



POJO는 모든 메서드 시그니쳐를 내 맘대로 할 수 있다.

매개변수를 몇개 받든 순서가 어떻게 되든 상관없다

insertBoard.jsp	BoardVO 클래스

<form action="insertBoard.do">
<table>
<tr>
제목 <input type="text" name="**title**"/>
</tr>
<tr>
작성자 <input type="text" name="**writer**"/>
</tr>
<tr>
내용 <textarea name="**content**"></textarea>
</tr>
<tr>
<input type="submit" value="글등록 "/>
</tr>
</table>
</form>
	public class BoardVO {
	private int seq;
	private String title;
	private String writer;
	private String content;
	
	public void setSeq(int seq) {
	this.seq = seq;
	}
	public void **setTitle**(String title) {
	this.title = title;
	}
	public void **setWriter**(String writer) {
	this.writer = writer;
	}
	public void **setContent**(String content) {
	this.content = content;
	}
	}

- BoardVO class에서 변수 선언할 때 이름은 상관없지만 jsp파일의 set+변수(앞 글자=대문자)는 꼭 맞춰줘야 저 놈으로 값을 받아와서 set해준다.

```java
// 글 등록
@RequestMapping("/insertBoard.do")
public String insertBoard(BoardVO vo, BoardDAOJDBC dao) throws Exception {
    dao.insertBoard(vo);
    return "getBoardList.do";
}

// 글 수정
@RequestMapping("/updateBoard.do")
public String updateBoard(BoardVO vo, BoardDAOJDBC dao) throws Exception {
    dao.updateBoard(vo);
    return "getBoardList.do";
}

// 글 삭제
@RequestMapping("/deleteBoard.do")
public String deleteBoard(BoardVO vo, BoardDAOJDBC dao) throws Exception {
    dao.deleteBoard(vo);
    return "getBoardList.do";
}

// 글 상세 조회
@RequestMapping("/getBoard.do")
public ModelAndView getBoard(BoardVO vo, BoardDAOJDBC dao, ModelAndView mav) throws Exception {
    mav.addObject("board", dao.getBoard(vo)); // Model 정보 저장
    mav.setViewName("getBoard.jsp");   // View  정보 저장
    return mav;
}

// 글 목록 검색
@RequestMapping("/getBoardList.do")
public ModelAndView getBoardList(BoardVO vo, BoardDAOJDBC dao, ModelAndView mav) throws Exception {
    mav.addObject("boardList", dao.getBoardList(vo)); // Model 정보 저장
    mav.setViewName("getBoardList.jsp");   // View  정보 저장
    return mav;
}
```

- 각 메소드는 ModelAndView한테 리턴함.

View 이름 앞에 'forward:'이나 'redirect:'을 붙이면 ViewResolver가 설정되어 있다 하더라도 ViewResolver를 무시한다 . 따라서 로그인 성공 시에 글 목록을 조회하기 위해서 ViewResolver를 적용하지 않는 'getBoardList.do'로 포워드한 것이다 . 그리고 로그인에 실패한 경우에는 다시 로그인 화면으로 이동해야 하는데 로그인 화면 (login.jsp)은 WEB-INF 폴더에 없기 때문에 이 역시 ViewResolver를 적용하지 않아야 한다 .

```java
// 3. 화면 네비게이션
ModelAndView mav = new ModelAndView();
if(user != null) {
        mav.setViewName("forward:getBoardList.do");
} else {
        mav.setViewName("redirect:login.jsp");
}
```

요청 방식에 따른 처리

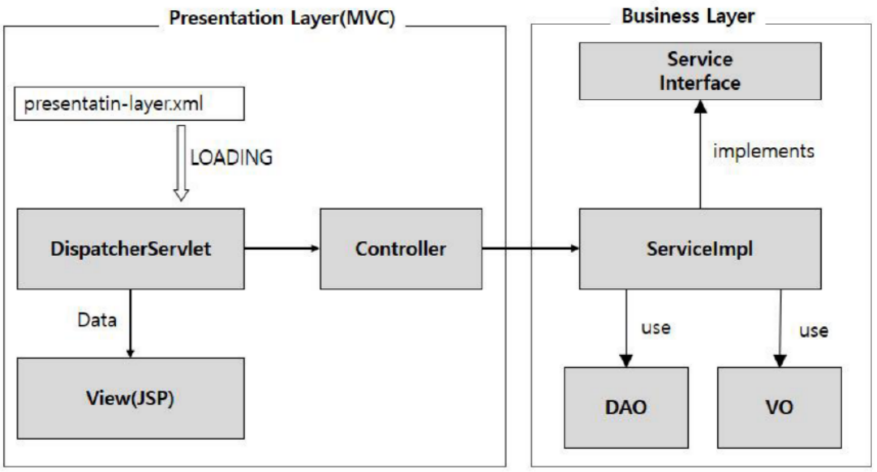
- @RequestMapping method 속성
  - LoginController 클래스에 loginView 메소드를 추가하고 설정을 변경한다.

```java
@Controller
public class LoginController {

	@RequestMapping(value="/login.do", method=RequestMethod.GET)
	public String login(UserVO vo) {
		System.out.println("로그인 화면으로 이동");
		vo.setId("test");
		vo.setPassword("test");
		return "forward:login.jsp";
	}

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

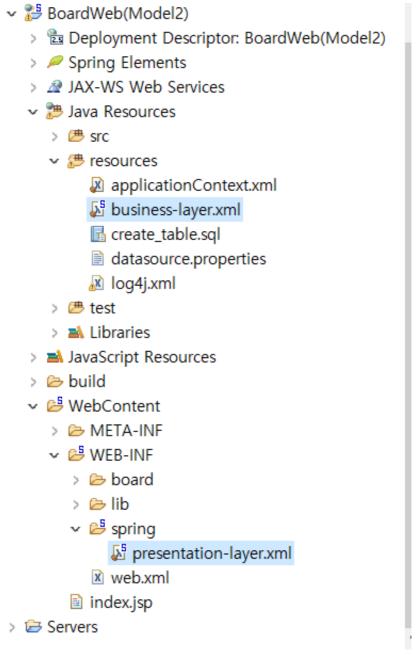
- 검색 버튼을 눌렀을 경우 어떻게 하는지 폼 안에 넣어두기.

```jsp
<!-- 검색 화면 시작 -->
<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>
<!-- 검색 화면 종료 -->
```

🎄말고 🏕️



- aspect의 포인트컷과 어드바이스를 잘 연결해줌
- 메서드를 메모리에 띄우려면 @Service 를 붙여줘야함
- 컨트롤러는 DAO를 이용해서 바로 접근하면 트랜잭션 관리가 안됨.
- 레이어마다 xml파일은 하나씩 있어야함 →
  - 즉, 스프링 컨테이너는 두개 있어야함.
  - 어? 컨테이너 두개 가능?
    - 예스바리~ 컨테이너는 클래스에 불과하다.
- presentation-layer은 servlet이 읽고,
- business-layer은 listener가 읽는다.
- 서블릿, 리스너, 필터 이렇게 있는데 필터가 인코딩 해주는 놈.



Json 👱

예를 들어 vo로 이루어진 검색 데이터를 서버가 이제 데이터로 공유하게 될 때, 여러 플랫폼들이 받게 될 텐데 (예를 들어 C#.NET과 Java - 폰 일수도 있고 컴터 일수도 있고 하니까) 텍스트 데이터로 공유 해야 함.

근데 xml은 시작 태그와 종료 태그로 이루어져 있어서 무겁다.

그래서 key, value로 이루어진 경량의 데이터 포맷인 json이 만들어짐.

제이슨 연동하는 법

- ResponseBody : 리턴되는 자바 객체를 JSON 데이터로 변환하여 HTTP 응답 프로토콜 Body에 출력해줌.

```java
@Controller
public class BoardController {

    @Autowired
    private BoardService boardService;

    // json 변환
    @RequestMapping("/json.do")
    public @ResponseBody BoardVO json(BoardVO vo) throws Exception {
        return boardService.getBoard(vo);
    }

    // 글 등록 화면으로 이동
    @RequestMapping("/insertBoardView.do")
    public String insertBoardView() throws Exception {
        return "insertBoard";
    }


  // json 변환
  // ResponseBody : 리턴되는 자바 객체를 JSON 데이터로 변환하여 HTTP 응답 프로토콜 Body에 출력해줌.
  @RequestMapping("/json.do")
  public @ResponseBody List<BoardVO> json(BoardVO vo) throws Exception {
      vo.setSearchCondition("TITLE");
      vo.setSearchKeyword("");
      return boardService.getBoardList(vo);
  }

  // json 변환
```

```
List<BoardVO> getBoardList(BoardVO vo);
Press 'F2' for focus
```