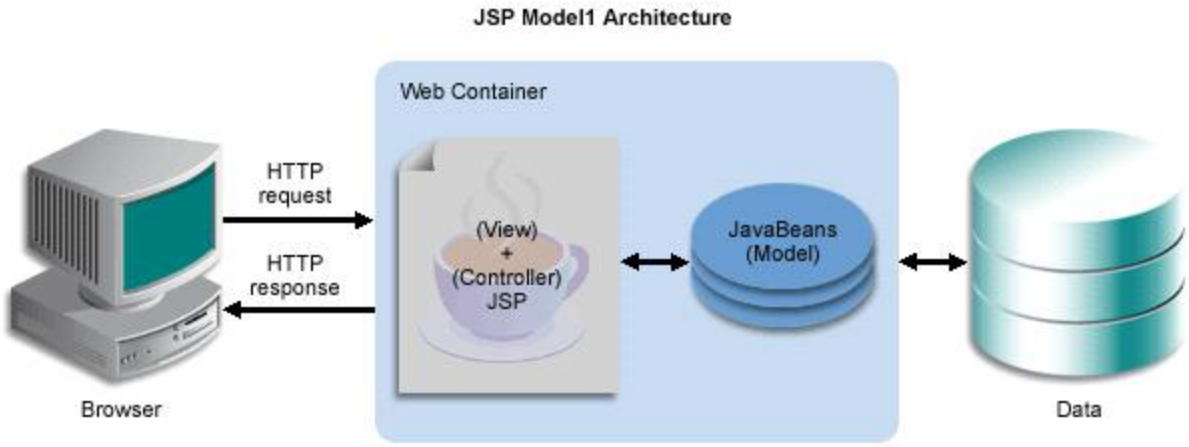
제 9 강 MVC 패턴

1. MVC 1 : Model 1

① Model 1 아키텍처의 구조



- ☐ 90년대 말부터 2000년대 초까지 자바 기반의 웹 애플리케이션 개발에 사용되던 구조
- ☐ Model 1 아키텍처는 JSP와 JavaBeans만을 사용하여 웹을 개발하는 것이다. JavaBeans는 데이터베이스에 연동에 사용되는 자바 객체들이다.
- ☐ Model의 정확한 의미는 데이터베이스 연동 로직을 제공하면서 DB에서 검색한 데이터가 제공되는 자바 객체이다. (VO, DAO)
- ☐ Model 1 구조에서는 JSP가 가장 중요한 역할을 수행하는데 이는 JSP가 Controller와 View의 기능을 모두 처리하기 때문이다.

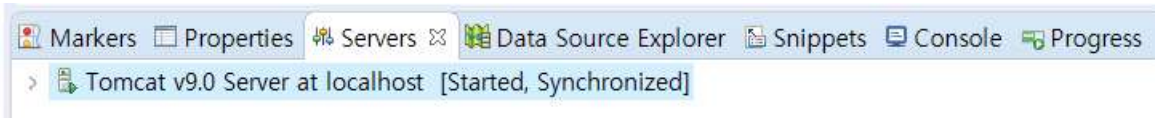
② 로그인 화면 구현

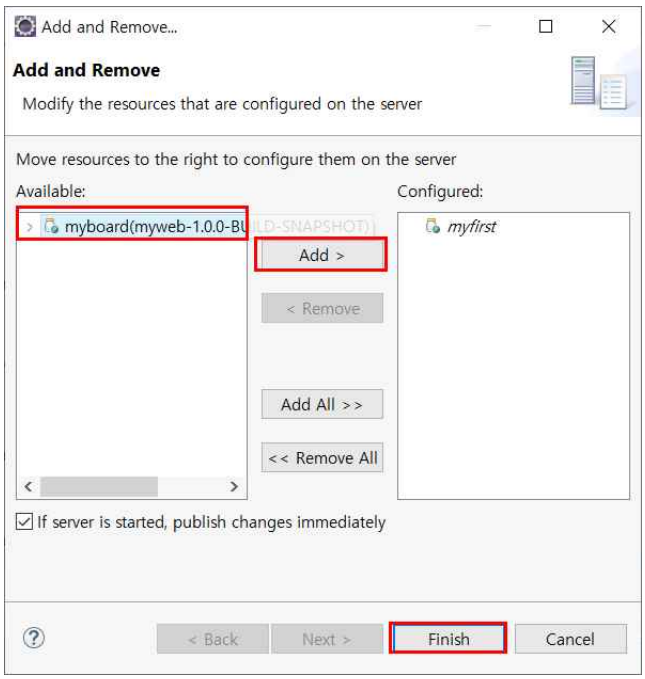
■ src/main/webapp/login.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <!DOCTYPE html>
3  <html>
4  <head>
5  <meta charset="UTF-8">
6  <title>Login</title>
7  </head>
8  <body>
9  <h1>로그인</h1>
10 <hr>
11 <form action="loginProc.jsp" method="post">
12 <table border="1">
13         <tr>
14                 <td>아이디</td>
15                 <td><input type="text" name="id" /></td>
16         </tr>
17         <tr>
```

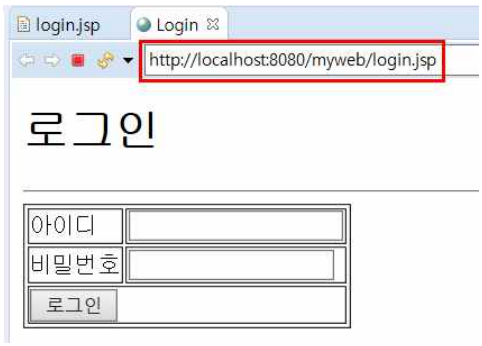
```
18                      <td>비밀번호</td>
19                      <td><input type="password" name="password" /></td>
20              </tr>
21              <tr><td colspan="2"><input type="submit" value="로그인" /></td></tr>
22  </table>
23  </form>
24  </body>
25  </html>
```

▢ 실행하기 전에 Tomcat 서버에서 프로젝트를 등록하자. [Servers] 탭에서 Tomcat 서버를 찍고 우클릭하여 [Add and Remove...]를 선택하여 프로젝트를 바인딩 시켜주자.





▢ 실행하여 결과화면을 확인해 보자.



■ 여기서 문제는 URL 정보를 확인해 보면 프로젝트 이름 대신에 myweb이 컨텍스트 경로로 선택된 것을 볼 수 있다. 이것을 변경하려면 Package Explorer 아래의 Servers 프로젝트에서 server.xml 파일을 수정하면 된다.

```
151
152         <Context docBase="myboard" path="/myboard" reloadable="true" source="org.eclipse.jst.jee.server:myboard"/></Host>
153       </Engine>
154     </Service>
155 </Server>
```

■ 이제 login.jsp를 다시 실행하여 결과를 확인해 보자.

③ 로그인 인증 처리

■ src/main/webapp/loginProc.jsp

```
1  <%@ page import="tommy.spring.web.user.impl.UserDAO"%>
2  <%@ page import="tommy.spring.web.user.UserVO"%>
3  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
4  <%
5          // 1. 사용자 입력 정보 추출
6          String id = request.getParameter("id");
7          String password = request.getParameter("password");
8          // 2. 데이터베이스 연동 처리
9          UserVO vo = new UserVO();
10         vo.setId(id);
11         vo.setPassword(password);
12         UserDAO userDAO = new UserDAO();
13         UserVO user = userDAO.getUser(vo);
14         // 3. 화면 네비게이션
15         if(user != null){
16                 response.sendRedirect("getBoardList.jsp");
17         }else{
18                 response.sendRedirect("login.jsp");
19         }
20 %>
```

▢ 로그인이 성공적으로 이루어지면 getBoardList.jsp로 실패하면 login.jsp로 리다이렉트 하도록 구현하였다. 따라서 정상적으로 실행되려면 getBoardList.jsp를 먼저 구현해야 한다.

④ 글 목록 검색 기능 구현

■ src/main/webapp/getBoardList.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
3  <%@ page import="tommy.spring.web.board.BoardVO"%>
4  <%@ page import="java.util.List"%>
5  <%
6          // 1. 사용자 입력 정보 추출 : 검색 기능은 나중에 구현

7          // 2. 데이터베이스 연동 처리
8          BoardVO vo = new BoardVO();
9          BoardDAO boardDAO = new BoardDAO();
10         List<BoardVO> boardList = boardDAO.getBoardList(vo);

11         // 3. 응답 화면 구성
12 %>
13 <!DOCTYPE html>
14 <html>
15 <head>
16 <meta charset="UTF-8">
17 <title>Board List</title>
18 </head>
19 <body>
20 <h1>글 목록</h1>
21 <h3>테스트 회원님 환영합니다.<a href="logoutProc.jsp">Log-Out</a></h3>
22 <!-- 검색 시작 -->
23 <form action="getBoardList.jsp" method="post">
24 <table border="1">
25 <tr>
26         <td>
27                 <select name="searchCondition">
28                         <option value="TITLE">제목</option>
29                         <option value="CONTENT">내용</option>
30                 </select>
31                 <input type="text" name="searchKeyword" />
32                 <input type="submit" value="검색" />
33         </td>
34 </tr>
35 </table>
36 </form><br/>
37 <!-- 검색 종료 -->
38 <table border="1">
39 <tr>
40         <th>번호</th>
41         <th>제목</th>
42         <th>작성자</th>
43         <th>등록일</th>
44         <th>조회수</th>
45 </tr>
46 <% for(BoardVO board: boardList){ %>
47 <tr>
```

```
48              <td><%=board.getSeq() %></td>
49              <td>
50                  <a href="getBoard.jsp?seq=<%=board.getSeq() %>"><%=board.getTitle() %></a>
51              </td>
52              <td><%=board.getWriter() %></td>
53              <td><%=board.getRegDate() %></td>
54              <td><%=board.getCnt() %></td>
55      </tr>
56      <%} %>
57      </table><br/>
58      <a href="insertBoard.jsp">새글 작성</a>
59      </body>
60      </html>
```

■ login.jsp에서 아이디와 패스워드를 입력하고 로그인 버튼을 클릭해 보자. 로그인이 성공적으로 이루어졌다면 아래와 같이 글 목록 화면이 나올 것이다.

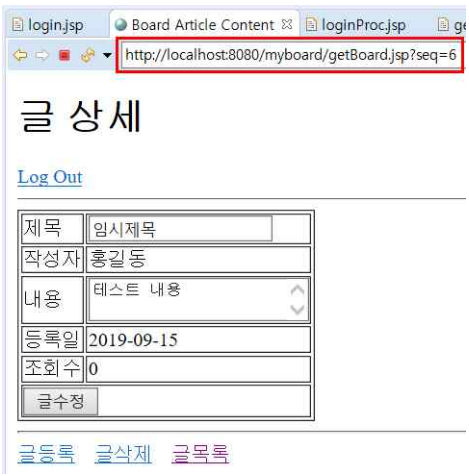
⑤ 글 상세 기능 구현

■ src/main/webapp/getBoard.jsp

```
1   <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
2   <%@ page import="tommy.spring.web.board.BoardVO"%>
3   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
4   <%
5           // 1. 검색할 게시글 번호 추출
6           String seq = request.getParameter("seq");

7           // 2. 데이터베이스 연동 처리
8           BoardVO vo = new BoardVO();
9           vo.setSeq(Integer.parseInt(seq));

10          BoardDAO boardDAO = new BoardDAO();
11          BoardVO board = boardDAO.getBoard(vo);
```

```
12          // 3. 응답 화면 구현
13  %>
14  <!DOCTYPE html>
15  <html>
16  <head>
17  <meta charset="UTF-8">
18  <title>Board Article Content</title>
19  </head>
20  <body>
21  <h1>글 상세</h1>
22  <a href="logoutProc.jsp">Log Out</a><hr>
23  <form action="updateBoardProc.jsp" method="post">
24  <table border="1">
25  <tr>
26          <td>제목</td>
27          <td><input name="title" type="text" value="<%=board.getTitle() %>" /></td>
28  </tr>
29  <tr>
30          <td>작성자</td>
31          <td><%=board.getWriter() %></td>
32  </tr>
33  <tr>
34          <td>내용</td>
35          <td><textarea name="content"><%=board.getContent() %></textarea></td>
36  </tr>
37  <tr>
38          <td>등록일</td>
39          <td><%=board.getRegDate() %></td>
40  </tr>
41  <tr>
42          <td>조회수</td>
43          <td><%=board.getCnt() %></td>
44  </tr>
45  <tr>
46          <td colspan="2"><input type="submit" value="글수정" /></td>
47  </tr>
48  </table>
49  </form><hr>
50  <a href="insertBoard.jsp">글등록</a>   
51  <a href="deleteBoardProc.jsp">글삭제</a>   
52  <a href="getBoardList.jsp">글목록</a>
53  </body>
54  </html>
```

■ 글 목록 화면에서 아무 글이나 제목을 클릭하여 글 상세 화면이 나오는지 실행해 보자.



▢ 글 상세 화면은 게시 글의 상세 내용을 보여줄 뿐만 아니라 수정을 위한 화면이기도 하다. 원래 수정화면과는 분리하여 작성해야 하나 작업을 단순하게 처리하기 위하여 통합하였다.

▢ 아래쪽에 세 개의 링크가 제공되는데 [글 등록] 링크를 클릭하면 글 등록 화면으로 이동하고 [글 삭제] 링크를 클릭하면 현재 보고 있는 게시 글이 삭제되어야 한다. 또 [글 목록] 링크를 클릭하면 다시 글 목록 화면으로 이동하도록 처리해야 한다.

⑥ 글 등록 화면 구현하기

■ src/main/webapp/insertBoard.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <!DOCTYPE html>
3  <html>
4  <head>
5  <meta charset="UTF-8">
6  <title>Insert Board</title>
7  </head>
8  <body>
9  <h1>글등록</h1>
10 <a href="logoutProc.jsp">Log Out</a><hr>
11 <form action="insertBoardProc.jsp" method="post">
12 <table border="1">
13 <tr>
14          <td>제목</td>
15          <td><input type="text" name="title"/></td>
16 </tr>
17 <tr>
18          <td>작성자</td>
19          <td><input type="text" name="writer"/></td>
20 </tr>
21 <tr>
22          <td>내용</td>
23          <td><textarea name="content"></textarea></td>
24 </tr>
25 <tr>
26          <td colspan="2"><input type="submit" value="새글 등록"/></td>
```

```
27  </tr>
28  </table>
29  </form><hr>
30  <a href="getBoardList.jsp">글 목록으로 가기</a>
31  </body>
32  </html>
```

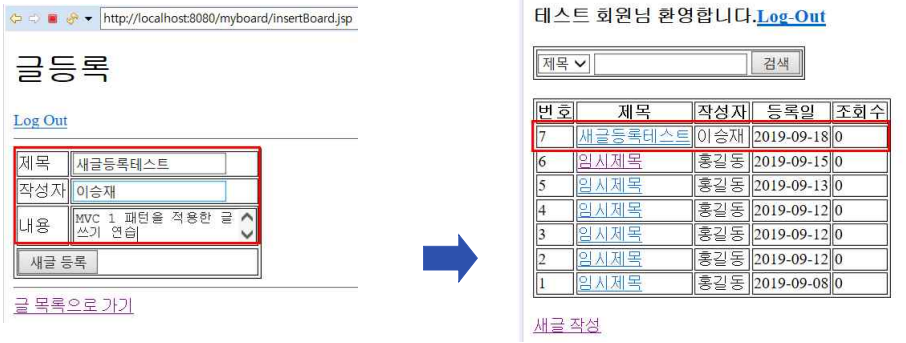
■ [글 등록]을 클릭하여 실행 하여 보자.

⑦ 글 등록 처리하기

■ src/main/webapp/insertBoardProc.jsp

```
1   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2   <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
3   <%@ page import="tommy.spring.web.board.BoardVO"%>
4   <%
5           // 1. 사용자 입력 정보 추출
6           request.setCharacterEncoding("UTF-8");
7           String title = request.getParameter("title");
8           String writer = request.getParameter("writer");
9           String content = request.getParameter("content");

10          // 2. 데이터베이스 연동 처리
11          BoardVO vo = new BoardVO();
12          vo.setTitle(title);
13          vo.setWriter(writer);
14          vo.setContent(content);

15          BoardDAO boardDAO = new BoardDAO();
16          boardDAO.insertBoard(vo);

17          // 3. 화면 네비게이션
18          response.sendRedirect("getBoardList.jsp");
19  %>
```

■ 글 등록 화면에 내용을 입력하고 [새 글 등록]을 클릭하여 글 등록이 정상적으로 수행되는지 확인해 보자.



⑧ 글 수정기능 처리하기

▢ 글 상세 보기 화면에서 글 내용을 수정하고 아래의 [글 수정] 버튼을 클릭하면 해당 글이 수정되도록 처리해야 한다. 따라서 글 수정하기 위해서 원래 글의 Primary Key인 글 번호를 알고 있어야 한다.

■ getBoard.jsp의 내용을 아래와 같이 수정하자.

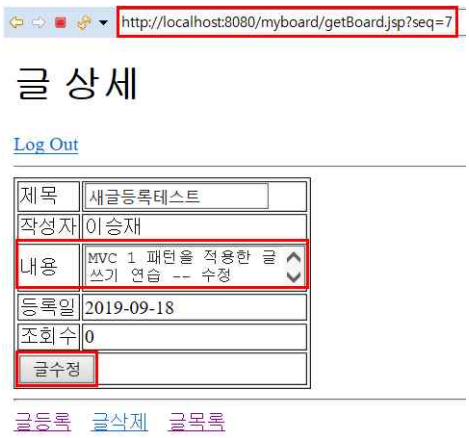
```
1  <!-- 상단 부분 생략 -->
2  <h1>글 상세</h1>
3  <a href="logoutProc.jsp">Log Out</a><hr>
4  <form action="updateBoardProc.jsp" method="post">
5  <input name="seq" type="hidden" value="<%=board.getSeq() %>" />
6  <table border="1">
7  <!-- 하단 부분 생략 -->
```

■ src/main/webapp/updateBoardProc.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
3  <%@ page import="tommy.spring.web.board.BoardVO" %>
4  <%
5          // 1. 사용자 입력 정보 추출
6          request.setCharacterEncoding("UTF-8");
7          String title = request.getParameter("title");
8          String content = request.getParameter("content");
9          String seq = request.getParameter("seq");

10         // 2. 데이터베이스 연동 처리
11         BoardVO vo = new BoardVO();
12         vo.setTitle(title);
13         vo.setContent(content);
14         vo.setSeq(Integer.parseInt(seq));

15         BoardDAO boardDAO = new BoardDAO();
```

```
16          boardDAO.updateBoard(vo);

17          // 3. 화면 네비게이션
18          response.sendRedirect("getBoardList.jsp");
19  %>
```

■ 이제 수정 기능을 수행해 보자.



⑨ 글 삭제 기능 처리하기

▢ 글 상세 보기 화면에서 [글 삭제] 링크를 클릭할 경우 해당 게시 글을 삭제하도록 처리하여야 한다. 이때도 수정처리와 마찬가지로 원래 글의 Primary Key인 글 번호를 알고 있어야 한다.

■ getBoard.jsp의 내용을 아래와 같이 수정하자.

```
1   <!-- 상단 부분 생략 -->
2   <hr>
3   <a href="insertBoard.jsp">글등록</a>   
4   <a href="deleteBoardProc.jsp?seq=<%=board.getSeq()%>">글삭제</a>   
5   <a href="getBoardList.jsp">글목록</a>
6   </body>
7   </html>
```

■ src/main/webapp/deleteBoardProc.jsp

```
1   <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
2   <%@ page import="tommy.spring.web.board.BoardVO"%>
3   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
4   <%
5           // 1. 사용자 입력 정보 추출
6           String seq = request.getParameter("seq");

7           // 2. 데이터베이스 연동 처리
8           BoardVO vo = new BoardVO();
9           vo.setSeq(Integer.parseInt(seq));
```

```
10          BoardDAO boardDAO = new BoardDAO();
11          boardDAO.deleteBoard(vo);

12          // 3. 화면 네비게이션
13          response.sendRedirect("getBoardList.jsp");
14  %>
```

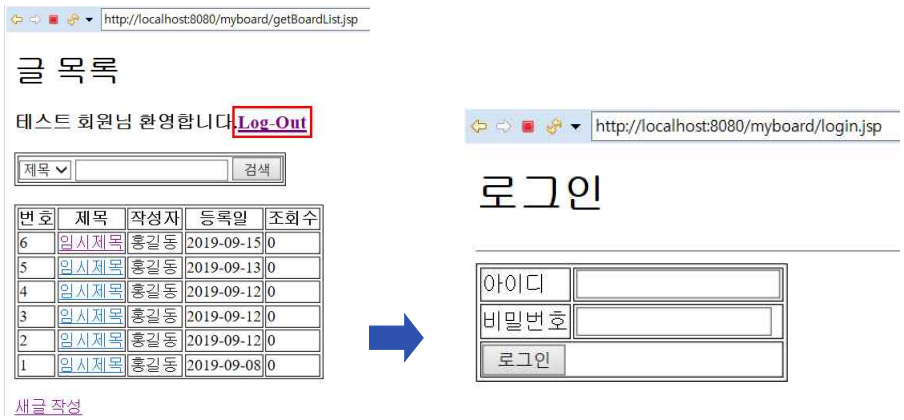
■ 이제 글 삭제 기능을 수행해 보자.

⑩ 로그아웃 기능 구현하기

■ src/main/webapp/logoutProc.jsp

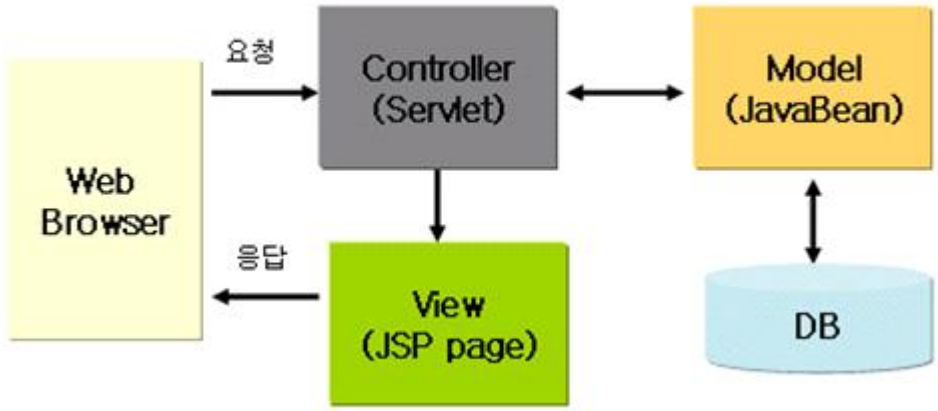
```
1   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2   <%
3           // 1. 브라우저와 연결된 세션 객체를 종료
4           session.invalidate();

5           // 2. 세션 종료 후 메인 화면으로 이동
6           response.sendRedirect("login.jsp");
7   %>
```

■ 로그아웃 기능이 정상 처리되는지 확인해 보자.



2. MVC 2 : Model 2

① Model 2 아키텍처 구조

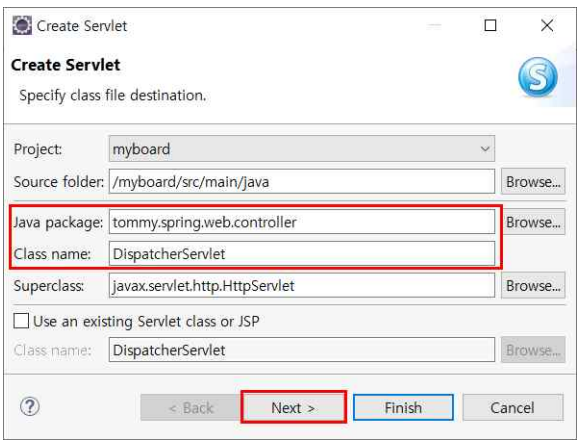


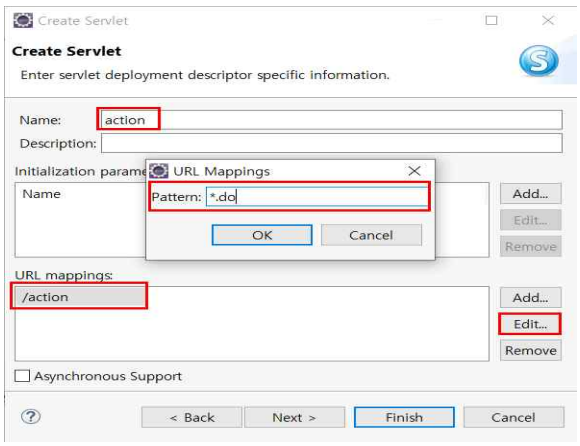
- ☐ Model 1 아키텍처가 엔터프라이즈 시스템에 적합하지 않은 이유는 자바 로직과 화면 디자인이 통합되어 유지보수가 어렵기 때문이다.
- ☐ 이런 Model 1 아키텍처의 문제를 해결하기 위해 고안된 웹 개발 모델이 Model 2 아키텍처 즉 MVC 아키텍처이다.
- ☐ 모델 2 아키텍처에서 가장 중요한 특징은 Controller의 등장이며 기존에 JSP가 담당했던 Controller 로직이 별도의 Controller 기능의 Servlet 클래스로 옮겨졌다.
- ☐ 따라서 기존의 Model 1 아키텍처로 개발한 프로그램에서 JSP 파일에 있는 자바 코드만 Controller로 이동하면 Model 2 아키텍처가 된다.

기 능	구 성 요 소	개 발 주 체
Model	VO, DAO 클래스	자바 개발자
View	JSP 페이지	웹 디자이너, 웹 퍼블리셔, 프론트 엔드 개발자
Controller	Servlet 클래스	자바 개발자 또는 MVC 프레임워크

② Controller 구현하기

- ☐ 프로젝트에서 Java Resources 아래의 src/main/java를 선택 후 우 클릭 [New]-[Servlet] 선택 하여 아래와 같이 진행한다.





■ Controller Servlet 구현하기 : DispatcherServlet

```
1  package tommy.spring.web.controller;
2  import java.io.IOException;
3  import javax.servlet.ServletException;
4  import javax.servlet.annotation.WebServlet;
5  import javax.servlet.http.HttpServlet;
6  import javax.servlet.http.HttpServletRequest;
7  import javax.servlet.http.HttpServletResponse;
8  @WebServlet(name = "action", urlPatterns = { "*.do" })
9  public class DispatcherServlet extends HttpServlet {
10         private static final long serialVersionUID = 1L;
11         protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                                    throws ServletException, IOException {
12                 processRequest(request, response);
13         }
14         protected void doPost(HttpServletRequest request, HttpServletResponse response)
                                                    throws ServletException, IOException {
15                 request.setCharacterEncoding("UTF-8");
16                 processRequest(request, response);
17         }
18         private void processRequest(HttpServletRequest request, HttpServletResponse response)
                                                                throws IOException{
19                 // 1. 클라이언트 정보를 추출한다.
20                 String uri = request.getRequestURI();
21                 String path = uri.substring(uri.lastIndexOf("/"));
22                 System.out.println(path);
23                 // 2. 클라이언트의 요청 path에 따라 적절히 작업을 분기 시켜준다.
24                 if(path.equals("/login.do")) {
25                         System.out.println("로그인 처리");
26                 }else if(path.equals("/logout.do")) {
27                         System.out.println("로그아웃 처리");
28                 }else if(path.equals("/insertBoard.do")) {
29                         System.out.println("글 등록 처리");
30                 }else if(path.equals("/updateBoard.do")) {
31                         System.out.println("글 수정 처리");
32                 }else if(path.equals("/deleteBoard.do")) {
33                         System.out.println("글 삭제 처리");
34                 }else if(path.equals("/getBoard.do")) {
35                         System.out.println("글 상세 보기 처리");
36                 }else if(path.equals("/getBoardList.do")) {
37                         System.out.println("글 목록 검색 처리");
38                 }
39         }
40 }
```

▢ DispatcherServlet에는 Get 방식 요청을 처리하는 doGet() 메서드와 Post 방식을 처리하는 doPost() 메서드가 정의되어 있는데 어떤 방식으로 요청하든 processRequest() 메서드를 통해 처리하도록 구현하였다.

▢ Post 방식의 요청에 대해 doPost() 메서드가 수행되는데 이때 한글이 깨지지 않도록 인코딩을 UTF-8로 처리하도록 하였다.

▢ processRequest() 메서드에서는 가장 먼저 클라이언트의 요청 URI로부터 path 정보를 추출하는데 이때 추출된 path는 URI 문자열에서 마지막 "/XXX.do" 문자열이다. 그리고 추출된 문자열에 따라서 분기처리를 통해 해당 로직을 수행하도록 한다.

■ 결과 실행 : 아래의 주소를 순차적으로 요청해 보자.

```
http://localhost:8080/myboard/login.do
http://localhost:8080/myboard/logout.do
http://localhost:8080/myboard/insertBoard.do
http://localhost:8080/myboard/updateBoard.do
http://localhost:8080/myboard/deleteBoard.do
http://localhost:8080/myboard/getBoard.do
http://localhost:8080/myboard/getBoardList.do
```

```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:₩Program Files₩Java₩jre1.8.0_221₩bin₩
정보: 프로토콜 핸들러 ["ajp-nio-8009"]을(를) 시작합니다.
9월 19, 2019 10:41:22 오전 org.apache.catalina.startup.Catalina start
정보: 서버가 [2,419] 밀리초 내에 시작되었습니다.
/login.do
/logout.do
/insertBoard.do
/updateBoard.do
/deleteBoard.do
/getBoard.do
/getBoardList.do
```

③ 로그인 기능 구현하기

▢ Controller 역할을 수행하는 DispatcherServlet은 *.do 형태의 요청에 대해서만 동작을 한다.

■ login.jsp 수정

```
1  <!-- 상단 부분 생략 -->
2  <body>
3  <h1>로그인</h1>
4  <hr>
5  <form action="login.do" method="post">
6  <table border="1">
7  <tr>
8          <td>아이디</td>
9          <td><input type="text" name="id" /></td>
10 </tr>
11 <!-- 하단 부분 생략 -->
```

■ DispatcherServlet 수정 : loginProc.jsp 내용을 복사

```
1  <!-- 상단 부분 생략 -->
2          // 2. 클라이언트의 요청 path에 따라 적절히 작업을 분기 시켜준다.
3          if(path.equals("/login.do")) {
```

```
4                       System.out.println("로그인 처리");
5                       // 1. 사용자 입력 정보 추출
6                       String id = request.getParameter("id");
7                       String password = request.getParameter("password");

8                       // 2. 데이터베이스 연동 처리
9                       UserVO vo = new UserVO();
10                      vo.setId(id);
11                      vo.setPasswrod(password);

12                      UserDAO userDAO = new UserDAO();
13                      UserVO user = userDAO.getUser(vo);

14                      // 3. 화면 네비게이션
15                      if(user != null){
16                              response.sendRedirect("getBoardList.do");
17                      }else{
18                              response.sendRedirect("login.jsp");
19                      }
20              }else if(path.equals("logout.do")) {
21      <!-- 하단 부분 생략 -->
```

④ 글 목록 검색 기능 구현

■ DispatcherServlet 수정 : getBoardList.jsp Controller에 해당하는 로직을 복사

```
1       <!-- 상단 부분 생략 -->
2               }else if(path.equals("/getBoardList.do")) {
3                               System.out.println("글 목록 검색 처리");
4                               // 1. 사용자 입력 정보 추출 : 검색 기능은 나중에 구현
5                               // 2. 데이터베이스 연동 처리
6                               BoardVO vo = new BoardVO();
7                               BoardDAO boardDAO = new BoardDAO();
8                               List<BoardVO> boardList = boardDAO.getBoardList(vo);

9                               // 3. 응답 화면 구성
10                              HttpSession session = request.getSession();
11                              session.setAttribute("boardList", boardList);
12                              response.sendRedirect("getBoardList.jsp");
13                      }
14              }
15      }
16      <!-- 하단 부분 생략 -->
```

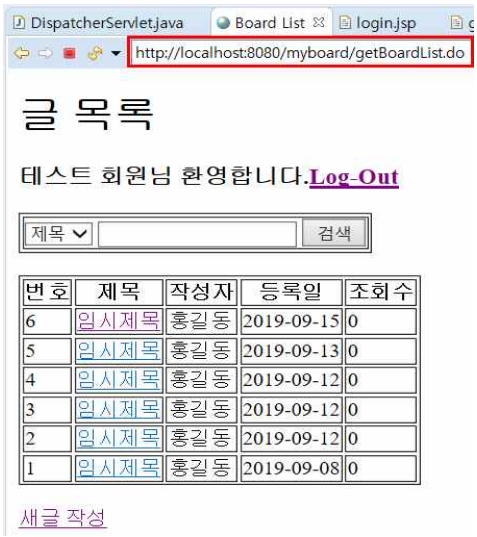
▢ 검색결과를 JSP에서 공유하기 위해서 세션에 저장을 하였는데 원래는 request 내장객체를 이용해야 하지만 지금은 전체적인 구조를 살펴보기 위함이므로 코드를 단순화하기 위하여 안 좋은 방법이지만 사용을 하였다.

■ getBoardList.jsp 수정

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <%@page import="tommy.spring.web.board.BoardVO"%>
3  <%@page import="java.util.List"%>
4  <%
5          // 세션에 저장된 글 목록을 추출
6          List<BoardVO> boardList = (List<BoardVO>) session.getAttribute("boardList");
7  %>
8  <!DOCTYPE html>
9  <!-- 하단 부분 생략 -->
```

▢ 이제 요청 시 getBoardList.jsp 라고 요청을 하면 오류가 발생한다. DispatcherServlet이 *.do 형태의 요청만 처리하기 때문이다.

■ 실행 및 결과 확인



▢ 위 결과의 동작과정을 살펴보면 아래와 같이 진행됨을 알 수 있다.

▣ DispatcherServlet이 클라이언트의 "/getBoardList.do" 요청을 받으면
▣ DispatcherServlet은 BoardDAO 객체를 이용하여 글 목록을 검색한다.
▣ 검색된 글 목록을 세션에 등록하고
▣ getBoardList.jsp 화면을 요청하면
▣ getBoardList.jsp는 세션에 저장된 글 목록을 꺼내어 목록 화면을 구성한다.
▣ 마지막으로 이 응답 화면이 브라우저에 전송된다.

⑤ 글 상세 보기 기능 구현하기

■ getBoardList.jsp의 링크 부분 수정

```
1  <!-- 상단 부분 생략 -->
2  <% for(BoardVO board: boardList){ %>
3  <tr>
4          <td><%=board.getSeq() %></td>
5          <td><a href="getBoard.do?seq=<%=board.getSeq() %>"><%=board.getTitle() %></a></td>
6          <td><%=board.getWriter() %></td>
```

```
7  <!-- 하단 부분 생략 -->
```

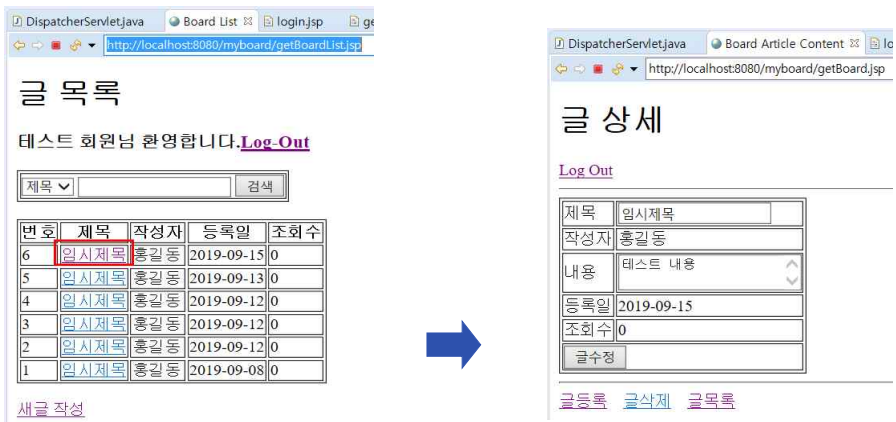
■ DispatcherServlet 수정 : getBoard.jsp Controller에 해당하는 로직을 복사

```
1  <!-- 상단 부분 생략 -->
2          }else if(path.equals("/getBoard.do")) {
3                  System.out.println("글 상세 보기 처리");
4                  // 1. 검색할 게시글 번호 추출
5                  String seq = request.getParameter("seq");

6                  // 2. 데이터베이스 연동 처리
7                  BoardVO vo = new BoardVO();
8                  vo.setSeq(Integer.parseInt(seq));

9                  BoardDAO boardDAO = new BoardDAO();
10                 BoardVO board = boardDAO.getBoard(vo);

11                 // 3. 응답 화면 구현
12                 HttpSession session = request.getSession();
13                 session.setAttribute("board", board);
14                 response.sendRedirect("getBoard.jsp");
15         }else if(path.equals("/getBoardList.do")) {
16 <!-- 하단 부분 생략 -->
```

■ getBoard.jsp 수정 : 세션에 저장된 검색 결과를 얻어오기.

```
1  <%@ page import="tommy.spring.web.board.impl.BoardDAO"%>
2  <%@ page import="tommy.spring.web.board.BoardVO"%>
3  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
4  <%
5          // 세션에 저장한 게시글 정보를 추출한다.
6          BoardVO board = (BoardVO) session.getAttribute("board");
7  %>
8  <!DOCTYPE html>
9  <!-- 하단 부분 생략 -->
```

■ 실행 및 결과 확인



⑥ 글 등록기능 구현하기

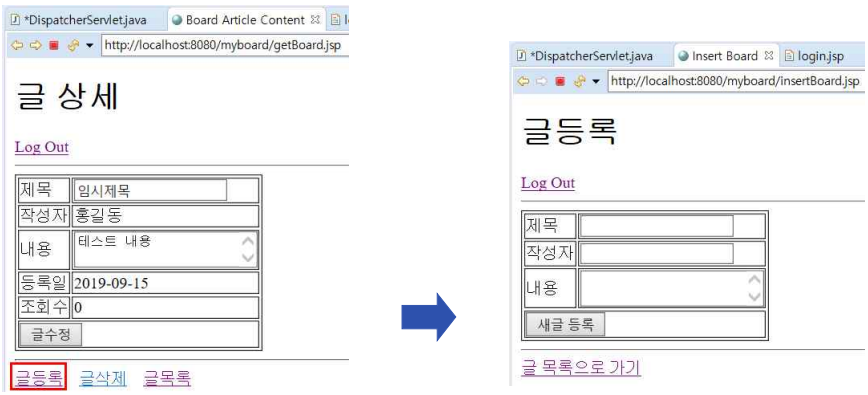
■ insertBoard.jsp 수정하기

```
1  <!-- 상단 부분 생략 -->
2  <body>
3  <h1>글등록</h1>
4  <a href="logoutProc.jsp">Log Out</a><hr>
5  <form action="insertBoard.do" method="post">
6  <table border="1">
7  <!-- 하단 부분 생략 -->
```

■ DispatcherServlet 수정 : insertBoardProc.jsp Controller에 해당하는 로직을 복사

```
1  <!-- 상단 부분 생략 -->
2          }else if(path.equals("/insertBoard.do")) {
3                  System.out.println("글 등록 처리");
4                  // 1. 사용자 입력 정보 추출
                   // request.setCharacterEncoding("UTF-8");
5                  String title = request.getParameter("title");
6                  String writer = request.getParameter("writer");
7                  String content = request.getParameter("content");

8                  // 2. 데이터베이스 연동 처리
9                  BoardVO vo = new BoardVO();
10                 vo.setTitle(title);
11                 vo.setWriter(writer);
12                 vo.setContent(content);

13                 BoardDAO boardDAO = new BoardDAO();
14                 boardDAO.insertBoard(vo);

15                 // 3. 화면 네비게이션
16                 response.sendRedirect("getBoardList.do");
17         }else if(path.equals("/updateBoard.do")) {
18 <!-- 하단 부분 생략 -->
```

■ 실행 및 결과 확인



⑦ 글 수정 기능 구현하기

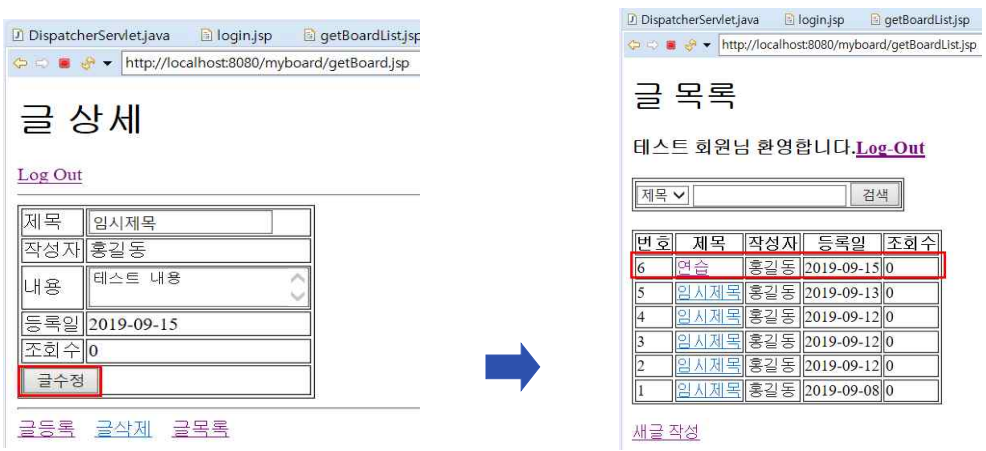
■ getBoard.jsp 수정

```
1  <!-- 상단 부분 생략 -->
2  <body>
3  <h1>글 상세</h1>
4  <a href="logoutProc.jsp">Log Out</a><hr>
5  <form action="updateBoard.do" method="post">
6  <input name="seq" type="hidden" value="<%=board.getSeq() %>" />
7  <!-- 하단 부분 생략 -->
```

■ DispatcherServlet 수정 : updateBoardProc.jsp Controller에 해당하는 로직을 복사

```
1  <!-- 상단 부분 생략 -->
2          }else if(path.equals("/updateBoard.do")) {
3                  // 1. 사용자 입력 정보 추출
                   //request.setCharacterEncoding("UTF-8");
4                  String title = request.getParameter("title");
5                  String content = request.getParameter("content");
6                  String seq = request.getParameter("seq");

7                  // 2. 데이터베이스 연동 처리
8                  BoardVO vo = new BoardVO();
9                  vo.setTitle(title);
10                 vo.setContent(content);
11                 vo.setSeq(Integer.parseInt(seq));

12                 BoardDAO boardDAO = new BoardDAO();
13                 boardDAO.updateBoard(vo);

14                 // 3. 화면 네비게이션
15                 response.sendRedirect("getBoardList.do");
16         }else if(path.equals("/deleteBoard.do")) {
17 <!-- 하단 부분 생략 -->
```

■ 실행 및 결과 확인



⑧ 글 삭제 기능 구현하기

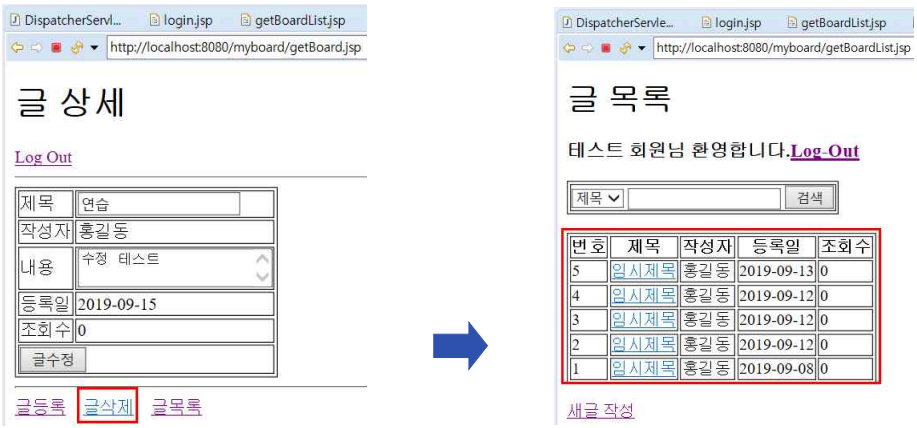
■ getBoard.jsp 수정

```
1  <!-- 상단 부분 생략 -->
2  <hr>
3  <a href="insertBoard.jsp">글등록</a>   
4  <a href="deleteBoard.do?seq=<%=board.getSeq()%>">글삭제</a>   
5  <a href="getBoardList.do">글목록</a>
6  </body>
7  </html>
```

■ DispatcherServlet 수정 : deleteBoardProc.jsp Controller에 해당하는 로직을 복사

```
1  <!-- 상단 부분 생략 -->
2          }else if(path.equals("/deleteBoard.do")) {
3                  System.out.println("글 삭제 처리");
4                  // 1. 사용자 입력 정보 추출
5                  String seq = request.getParameter("seq");
6                  // 2. 데이터베이스 연동 처리
7                  BoardVO vo = new BoardVO();
8                  vo.setSeq(Integer.parseInt(seq));
9                  BoardDAO boardDAO = new BoardDAO();
10                 boardDAO.deleteBoard(vo);
11                 // 3. 화면 네비게이션
12                 response.sendRedirect("getBoardList.do");
13         }else if(path.equals("/getBoard.do")) {
14 <!-- 하단 부분 생략 -->
```

■ 실행 및 결과 확인



⑨ 로그아웃 기능 구현하기

☐ 모든 페이지에서 “logoutProc.jsp” 라는 링크를 모두 “logout.do”로 수정한다.

■ 수정 예 : getBoard.jsp

```
1  <!-- 상단 부분 생략 -->
2  <body>
3  <h1>글 상세</h1>
4  <a href="logout.do">Log Out</a><hr>
5  <form action="updateBoard.do" method="post">
6  <!-- 하단 부분 생략 -->
```

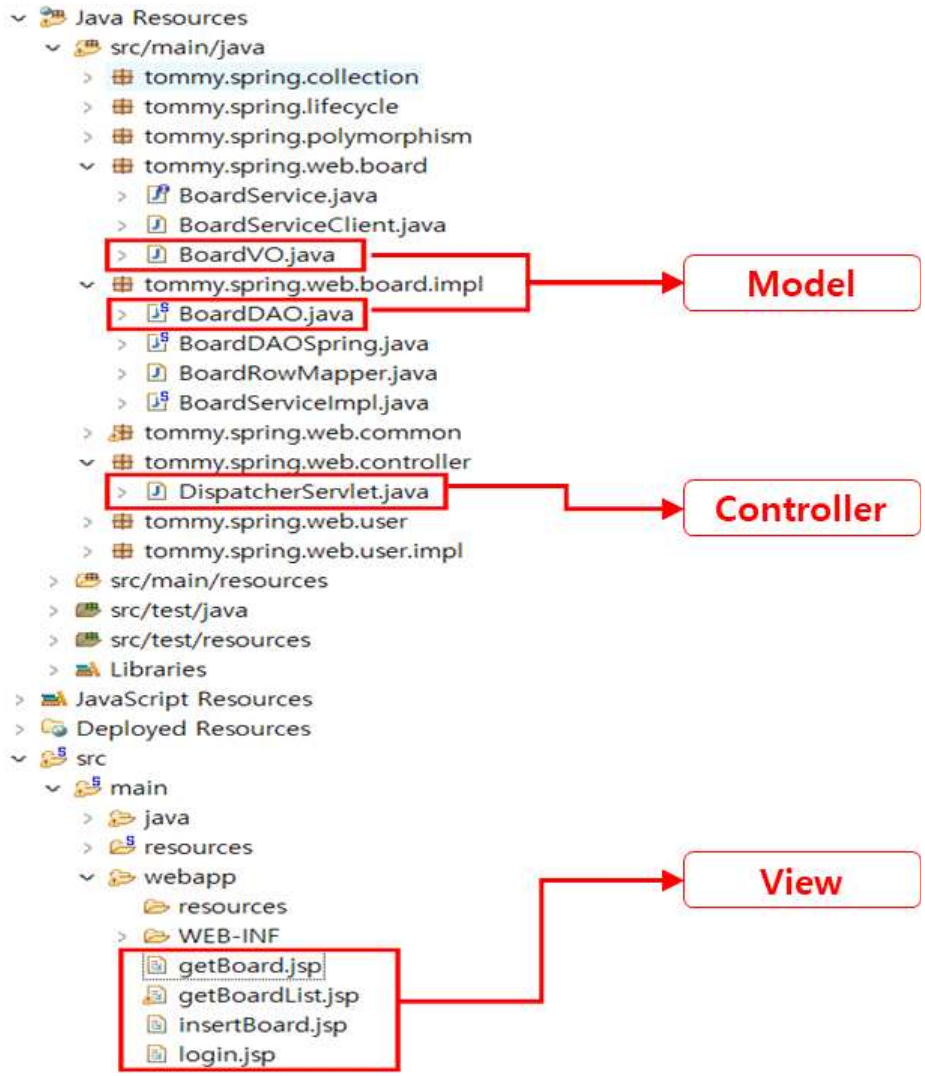
■ DispatcherServlet 수정 : logoutProc.jsp Controller에 해당하는 로직을 복사

```
1  <!-- 상단 부분 생략 -->
2          }else if(path.equals("/logout.do")) {
3                  System.out.println("로그아웃 처리");
4                  // 1. 브라우저와 연결된 세션 객체를 종료
5                  HttpSession session = request.getSession(false);
6                  session.invalidate();
7                  // 2. 세션 종료 후 메인 화면으로 이동
8                  response.sendRedirect("login.jsp");
9          }else if(path.equals("/insertBoard.do")) {
10 <!-- 하단 부분 생략 -->
```

■ 실행 및 결과 확인

3. 결론 및 정리

▢ 이제 게시판의 모든 기능이 MVC 2 구조로 수정되었다. 아래는 변경된 구조이다.



▢ Model 기능에 해당되는 VO, DAO 클래스는 재사용되었고, DispatcherServlet이라는 Controller기능의 Servlet 클래스가 추가되었다.

▢ 여기서 가장 큰 변화는 View 기능의 JSP 파일인데 Controller 기능의 자바 로직을 모두 DispatcherServlet 클래스로 이동 하였다. 따라서 어떠한 JSP 파일에도 더 이상 자바 로직인 존재하지 않는다.

▢ 다만 getBoard.jsp와 getBoardList.jsp에 세션에 저장된 데이터를 꺼내와 for 루프를 이용하여 화면에 출력해 주는 자바 코드가 존재하는데 이것은 핵심적인 Controller의 로직이라고 할 수 없으며 EL, JSTL을 이용하여 모두 제거할 수 있다.

▢ Controller 로직은 사용자의 입력정보를 추출, Model을 이용한 DB연동처리, 화면 내비게이션에 해당하는 자바코드를 의미한다.