



박경태  
[comsi.java@gmail.com](mailto:comsi.java@gmail.com)

# 고급 자바 프로그래밍

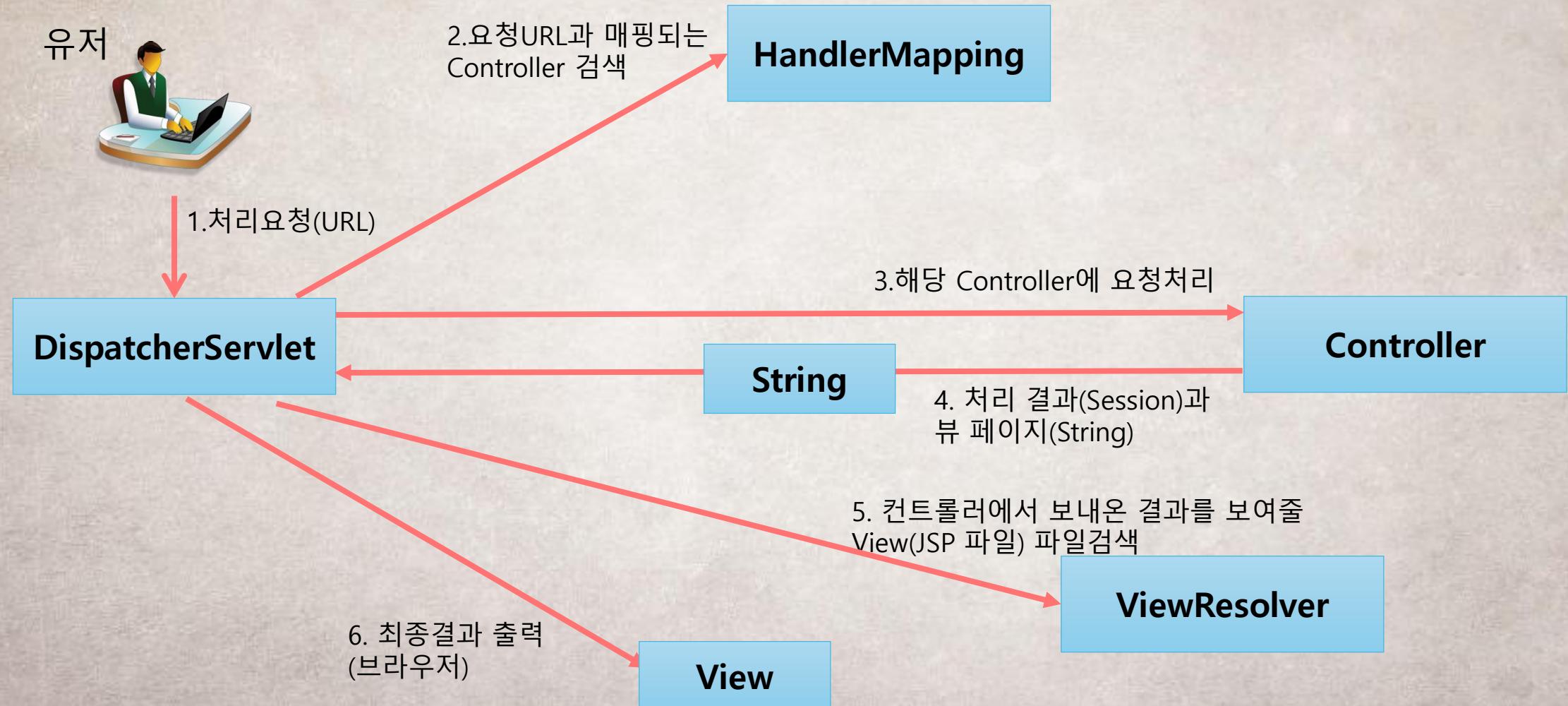
## : STS를 이용한 Spring 프로그래밍

# 4. MVC를 이용한 웹어플리케이션

## 4.1 MVC 프레임워크 구조

# 스프링 MVC의 수행 흐름

## 스프링 MVC의 클라이언트 요청 처리 과정



# 스프링 MVC를 이용한 웹어플리케이션

- 스프링 MVC의 주요 구성 요소

구성요소	설명
<b>DispatcherServlet</b>	유일한 서블릿 클래스로서 모든 클라이언트의 요청을 가장 먼저 처리하는 Front Controller
<b>HandlerMapping</b>	클라이언트의 요청(URL)을 어떤 Controller가 처리할지를 결정
<b>Controller</b>	클라이언트의 요청을 처리한 뒤, 필요시 결과를 세션에 저장하고 뷰페이지(JSP) 정보를 넘김
<b>ViewResolver</b>	Controller가 리턴한 View 이름으로 실행할 JSP경로(접두사+view이름+접미사)를 완성한다.

# web.xml 수정



The screenshot shows a Java IDE interface with three tabs: 'servlet-context.xml', 'BoardDAO.java', and '\*web.xml'. The '\*web.xml' tab is active and displays the XML code for the web application's deployment descriptor. A red rectangular box highlights the section from line 7 to line 17, which defines the DispatcherServlet.

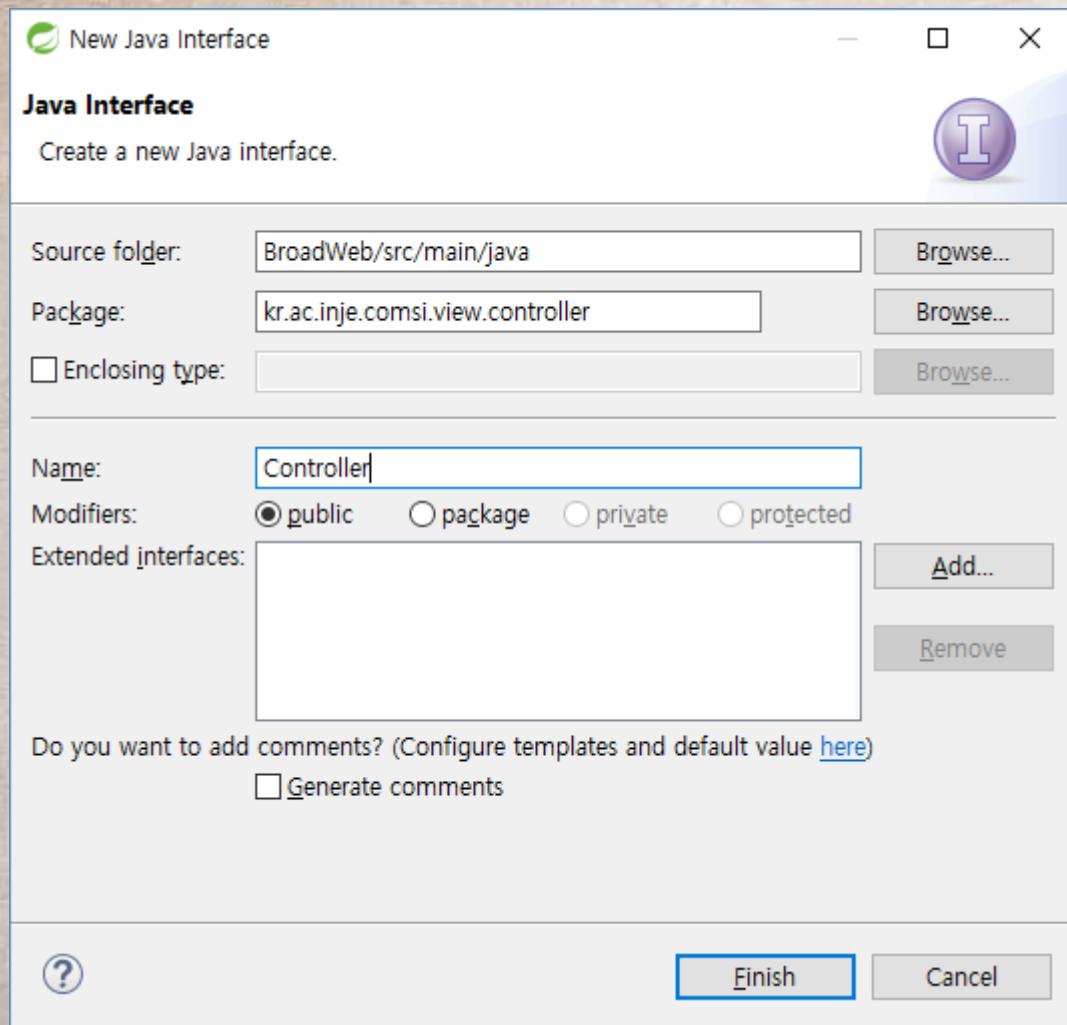
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5                         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
6
7 <servlet>
8     <display-name>DispatcherServlet</display-name>
9     <servlet-name>DispatcherServlet</servlet-name>
10    <servlet-class>kr.ac.inje.comsi.view.controller.DispatcherServlet</servlet-class>
11
12 </servlet>
13 <servlet-mapping>
14    <servlet-name>DispatcherServlet</servlet-name>
15    <url-pattern>*.do</url-pattern>
16 </servlet-mapping>
17 </web-app>
18
19
```

## 4.2 MVC 프레임 구현

# [1] Controller 인터페이스 작성

- DispatchServlet은 클라이언트의 요청을 가장 먼저 받아들임
- 클라이언트의 요청을 처리하기 위해서 하는 일은 거의 없음  
→ 실질적인 요청 처리는 각 Controller가 담당
- 클라이언트의 요청을 받은 DispatcherServlet은 HandlerMapping을 통해 Controller 객체를 검색하고, 검색된 Controller를 실행한다.
- 모든 Controller를 같은 타입으로 관리하기 위한 인터페이스를 만들어야 한다.
  - 즉, 어떤 Controller객체가 검색되더라도 같은 코드로 실행하려면 모든 Controller의 최상위 인터페이스가 필요

# src/main/java 폴더에서



# Controller interface – 모든 Controller가 사용

```
DispatcherServlet.java Controller.java
1 package kr.ac.inje.comsi.view.controller;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 public interface Controller {
7     String handlerRequest(HttpServletRequest request, HttpServletResponse response);
8 }
9
10 |
```

'r'을 제거한다(오타).  
→ 이후 모든 Controller 클래스에서도 제외

## [2] LoginController 구현

The screenshot shows a file tree on the left and a code editor on the right. In the code editor, a partial Java interface definition is visible:

```
1 package kr.ac.inje.comsi.view.controller
2
3 import javax.servlet.http.HttpServletRe
4 import javax.servlet.http.HttpServletRe
5
6 public interface Controller {
7     String handlerRequest(HttpServletRequest
8 }
```

A context menu is open over the interface definition, with the 'New' option highlighted. A secondary submenu is displayed, listing various project and class types. The 'Class' option is also highlighted in this submenu.

File Tree (Left):

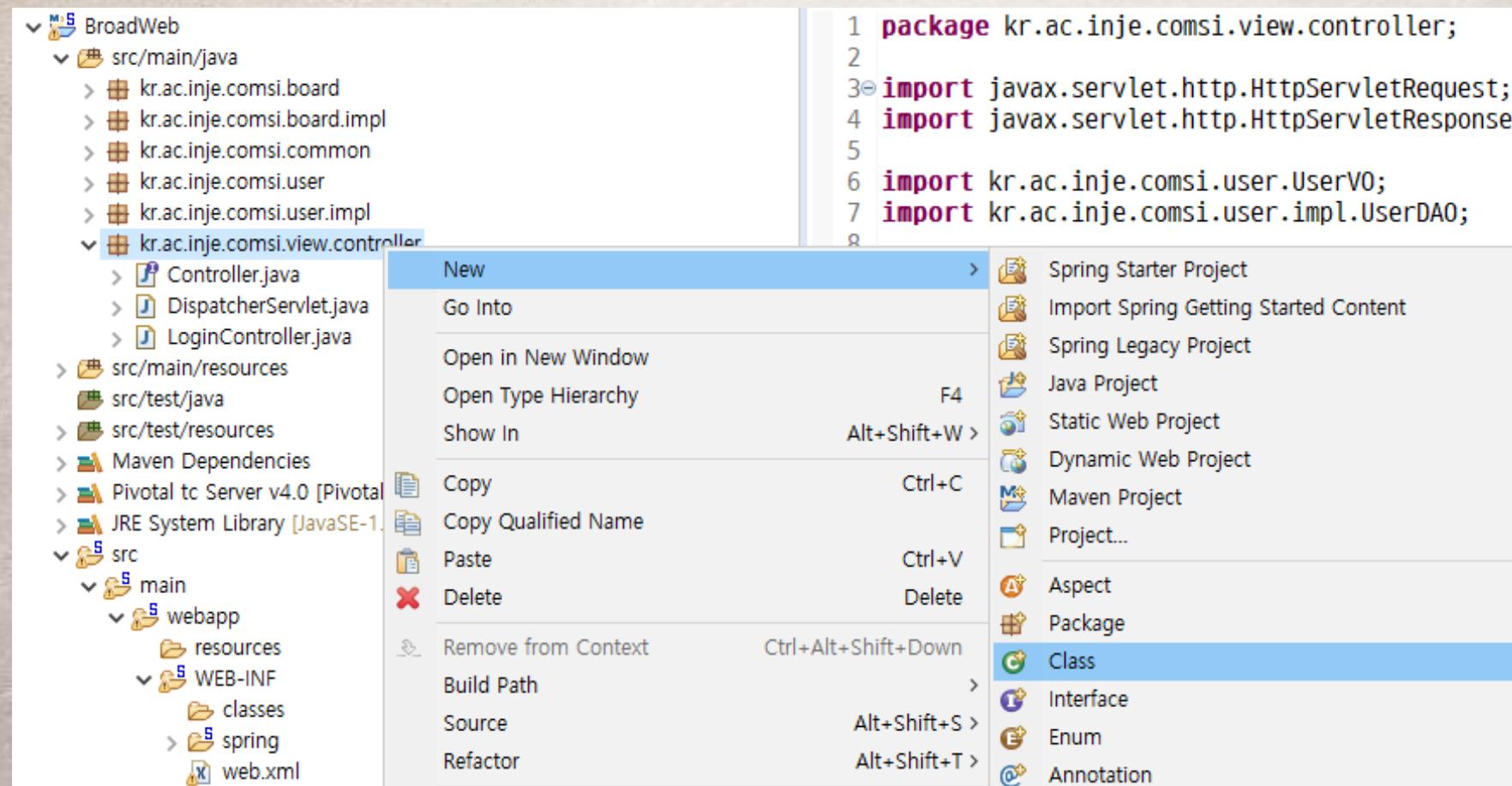
- BroadWeb
- src/main/java
  - kr.ac.inje.comsi.board
  - kr.ac.inje.comsi.board.impl
  - kr.ac.inje.comsi.common
  - kr.ac.inje.comsi.user
  - kr.ac.inje.comsi.user.impl
  - kr.ac.inje.comsi.view.controller
    - Controller
    - Dispatcher
- src/main/resources
- src/test/java
- src/test/resources
- Maven Dependencies
- Pivotal tc Server
- JRE System Library
- src
  - main
    - webapp
      - resources
      - WEB-INF
        - classes
        - spring
        - web
      - getBoa

```
1 package kr.ac.inje.comsi.view.user;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 import kr.ac.inje.comsi.user.UserVO;
7 import kr.ac.inje.comsi.user.impl.UserDAO;
8 import kr.ac.inje.comsi.view.controller.Controller;
9
10 public class LoginController implements Controller {
11
12     @Override
13     public String handlerRequest(HttpServletRequest request, HttpServletResponse response) {
14         System.out.println("로그인 처리");
15
16         // 1. 사용자 입력 정보 추출
17         String id = request.getParameter("id");
18         String password = request.getParameter("password");
19
20         // 2. DB 연동 처리
21         UserVO vo = new UserVO();
22         vo.setId(id);
23         vo.setPassword(password);
24
25         UserDAO userDAO = new UserDAO();
26         UserVO user = userDAO.getUser(vo);
27
28         // 3. 화면 이동
29         if (user != null) {
30             return "getBoardList.do"; // DispatcherServlet에서 받아서 처리
31         } else {
32             return "login"; // ViewResolver 클래스에서 처리
33         }
34     }
35 }
```

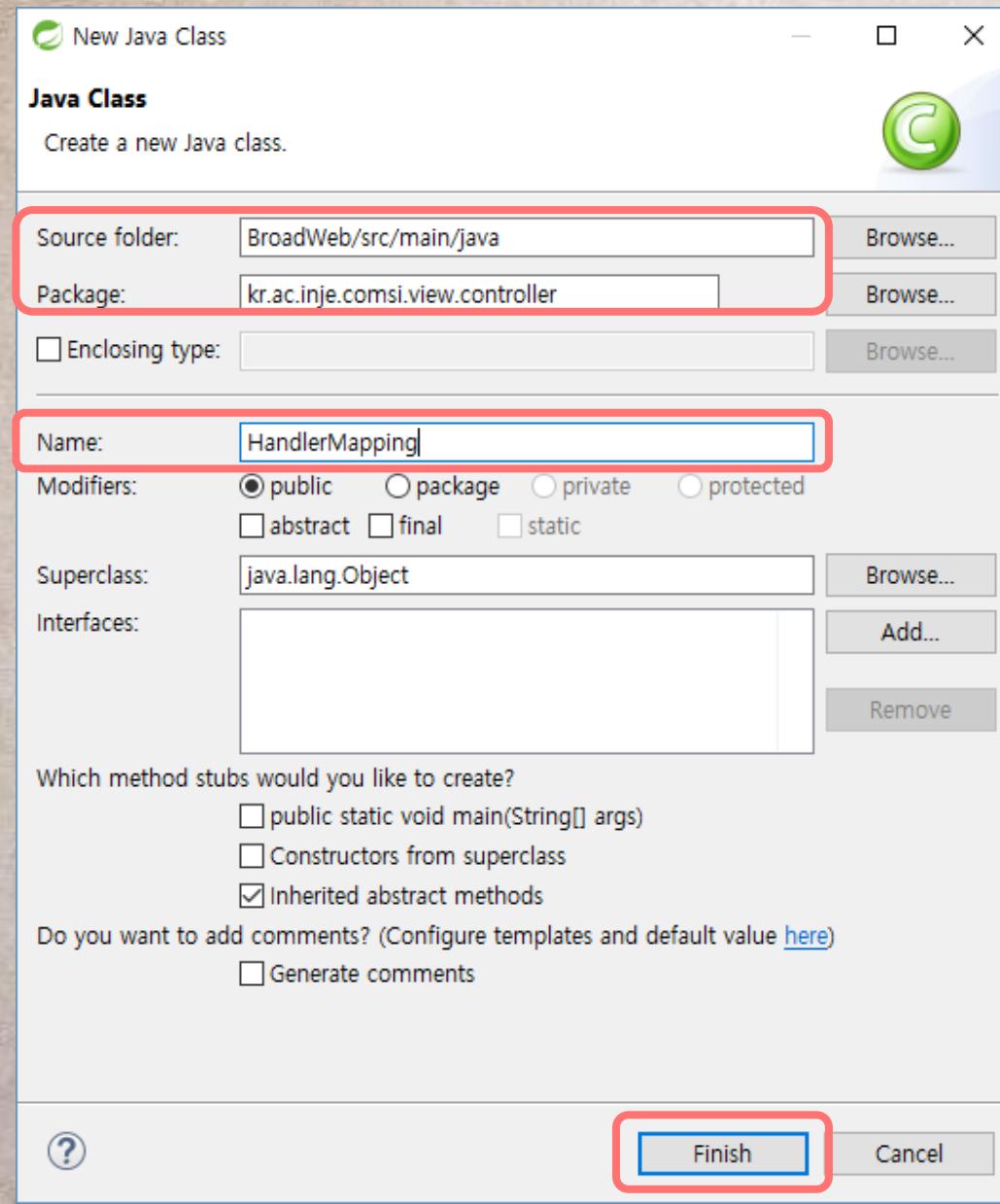
'r'을 제거한다(오타).  
→ 이후 모든 Controller 클래스에서도 제외

### [3] HandlerMapping 클래스 작성

- HandlerMapping은 모든 Controller 객체를 저장하고 있다가 클라이언트의 요청이 들어오면 처리할 특정 Controller를 검색하는 기능
- DispatcherServlet에서 사용하고 init() 메소드가 호출될 때 단 한 번 생성



# 이어서...



The screenshot shows the 'HandlerMapping.java' file in the code editor. The code is:

```
1 package kr.ac.inje.comsi.view.controller;
2
3 public class HandlerMapping {
4
5 }
6
```

A red arrow points from the 'HandlerMapping' entry in the 'New Java Class' dialog's 'Name' field to the corresponding class name in the code editor.

# HandlerMapping.java 클래스

- HandlerMapping은 **Map 타입 컬렉션** 멤버변수를 가지고 게시판 프로그램에 필요한 모든 Controller 객체 등록하고 관리

```
1 package kr.ac.inje.comsi.view.controller;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10    public HandlerMapping() {
11        mappings = new HashMap<String, Controller>();
12        mappings.put("/login.do", new LoginController());
13    }
14
15    // 요청경로에 따른 Controller 얻기
16    public Controller getController(String path) {
17        return mappings.get(path);
18    }
19 }
20
```

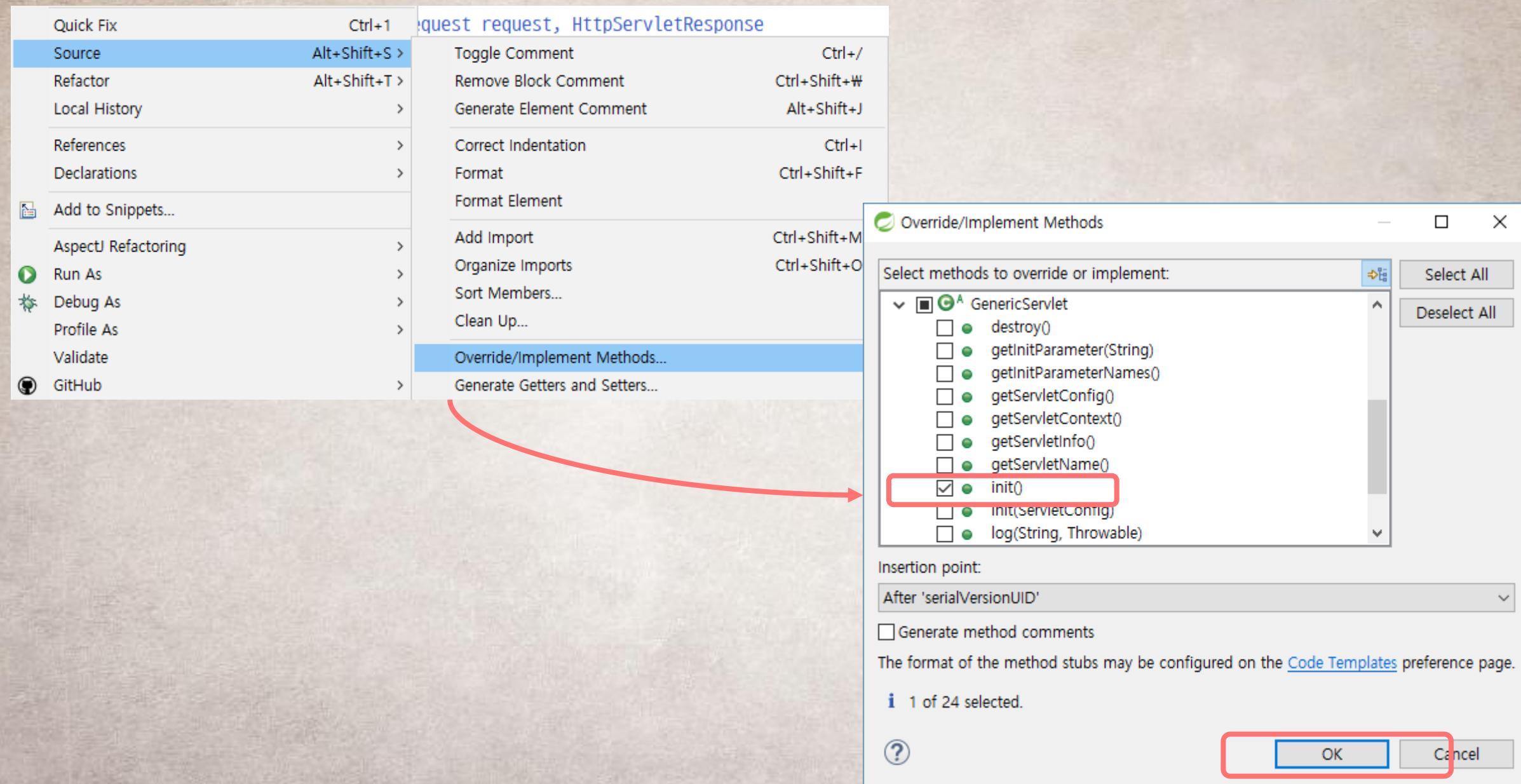
## [4] ViewResolver클래스 작성

- ViewResolver클래스는 Controller가 리턴한 View에 접두사와 접미사를 결합하여 최종 실행된 View 경로와 파일명을 완성한다.
- DispatcherServlet의 init() 메소드가 호출될 때 단 한 번 생성

The screenshot shows a Java code editor with multiple tabs at the top: DispatcherServlet.java, LoginController.java, HandlerMapping.java, and ViewResolver.java. The ViewResolver.java tab is active. The code in the editor is as follows:

```
1 package kr.ac.inje.comsi.view.controller;
2
3 public class ViewResolver {
4     public String prefix;
5     public String suffix;
6
7     public void setPrefix(String prefix) {
8         this.prefix = prefix;
9     }
10
11    public void setSuffix(String suffix) {
12        this.suffix = suffix;
13    }
14
15    // prefix - 파일이 있는 폴더
16    // viewName - 보내질 파일 이름
17    // suffix - 파일 확장자(.jsp)
18    public String getView(String viewName) {
19        return prefix+viewName + suffix;
20    }
21 }
```

# [5] DispatcherServlet 설정



# DispatcherServlet 설정

```
DispatcherServlet.java LoginController.java HandlerMapping.java ViewResolver.java
1 package kr.ac.inje.comsi.view.controller;
2
3 import java.io.IOException;
4
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11
12 public class DispatcherServlet extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14     private HandlerMapping handlerMapping;
15     private ViewResolver viewResolver;
16
17     @Override
18     public void init() throws ServletException {
19         handlerMapping = new HandlerMapping();
20         viewResolver = new ViewResolver();
21         viewResolver.setPrefix("./");
22         viewResolver.setSuffix(".jsp");
23     }
24
25     public DispatcherServlet() {
26         super();
27         // TODO Auto-generated constructor stub
28     }
29
30     protected void doGet(HttpServletRequest request, HttpServletResponse response)
31             throws ServletException, IOException {
32         process(request, response);
33     }
}
```

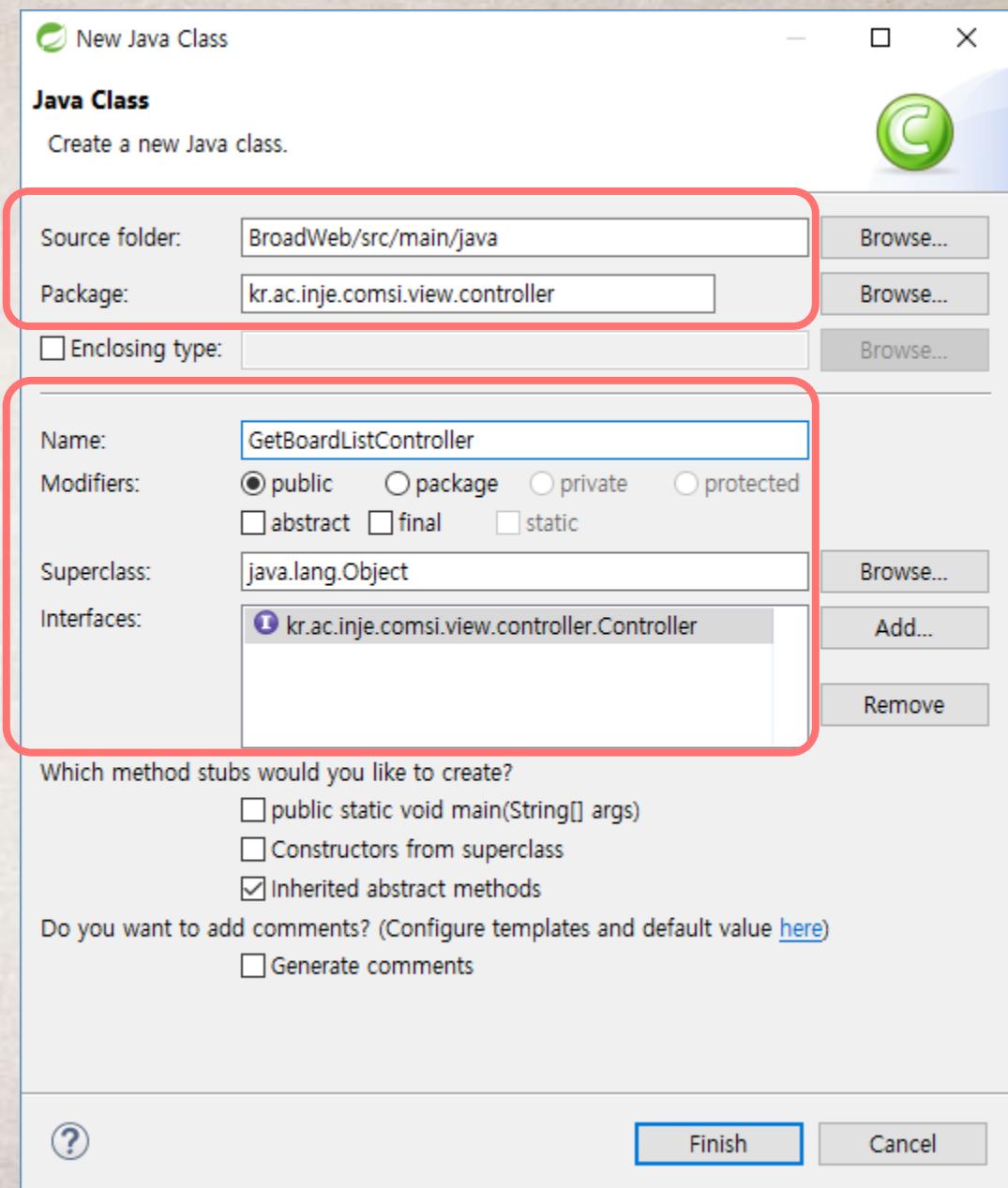
# 이어서...

```
34
35     protected void doPost(HttpServletRequest request, HttpServletResponse response)
36         throws ServletException, IOException {
37         request.setCharacterEncoding("UTF-8");
38         process(request, response);
39     }
40
41     private void process(HttpServletRequest request, HttpServletResponse response) throws IOException {
42         // 1. 클라이언트의 요청 path 정보를 추출한다.
43         String uri = request.getRequestURI();
44         String path = uri.substring(uri.lastIndexOf("/"));
45         System.out.println(path);
46
47         // 2. HandlerMapping을 통해 path에 해당하는 Controller를 검색한다.
48         Controller ctrl = handlerMapping.getController(path);
49
50         // 3. 검색된 Controller를 실행한다.                                'r'을 제거한다(오타).
51         String viewName = ctrl.handlerRequest(request, response);
52
53         // 4. ViewResolver를 통해 viewName에 해당하는 화면을 검색한다.
54         String view = null;
55         if (!viewName.contains(".do")) {
56             view = viewResolver.getView(viewName);
57         } else {
58             view = viewName;
59         }
60
61         // 5. 검색된 화면으로 이동한다.
62         response.sendRedirect(view);
63     }
64 }
```

## 4.3 MVC 프레임워크 적용

# (1) 글 목록 검색 구현

- Controller 인터페이스를 구현한 GetBoardListController 클래스 작성



# GetBoardListController 클래스

```
LogoutController.java GetBoardController.java UpdateBoardController.java GetBoardListController.java

1 package kr.ac.inje.comsi.view.board;
2
3 import java.util.List;
4
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7 import javax.servlet.http.HttpSession;
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 GetBoardListController implements Controller {
14
15     @Override
16     public String handlerRequest(HttpServletRequest request,
17                                  HttpServletResponse response) {
18         System.out.println("글 목록 검색 처리");
19
20         // 1. 사용자 입력 정보 추출(검색 기능은 나중에~~)
21         // 2. DB 연동 처리
22         BoardVO vo = new BoardVO();
23         BoardDAO boardDAO = new BoardDAO();
24         List<BoardVO> boardList = boardDAO.getBoardList(vo);
25
26         // 3. 검색 결과를 세션에 저장하고 목록 화면을 리턴
27         HttpSession session = request.getSession();
28         session.setAttribute("boardList", boardList);
29         return "getBoardList";
30     }
31 }
```

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
DispatcherServlet.java HandlerMapping.java GetBoardListController.java

1 package kr.ac.inje.comsi.view.controller;
2
3+ import java.util.HashMap;
4
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10    public HandlerMapping() {
11        mappings = new HashMap<String, Controller>();
12        mappings.put("/login.do", new LoginController());
13        mappings.put("/getBoardList.do", new GetBoardListController()); ← GetBoardListController 추가
14    }
15
16    // 요청경로에 따른 Controller 얻기
17    public Controller getController(String path) {
18        return mappings.get(path);
19    }
20}
21
```

## [2] 글 상세보기 구현 – GetBoardController 클래스

```
LogoutController.java  GetBoardController.java ✎
1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5 import javax.servlet.http.HttpSession;
6
7 import kr.ac.inje.comsi.board.BoardVO;
8 import kr.ac.inje.comsi.board.impl.BoardDAO;
9 import kr.ac.inje.comsi.view.controller.Controller;
10
11 public class GetBoardController implements Controller {
12
13     @Override
14     public String handlerRequest(HttpServletRequest request,
15                                 HttpServletResponse response) {
16         System.out.println("글 상세 조회 처리");
17
18         // 1. 검색할 게시글 번호 추출
19         String seq = request.getParameter("seq");
20
21         // 2. DB 연동 처리
22         BoardVO vo = new BoardVO();
23         vo.setSeq(Integer.parseInt(seq));
24
25         BoardDAO boardDAO = new BoardDAO();
26         BoardVO board = boardDAO.getBoard(vo);
27
28         // 3. 검색 결과를 세션에 저장하고 목록 화면을 리턴
29         HttpSession session = request.getSession();
30         session.setAttribute("board", board);
31         return "getBoard"; //
32     }
33 }
```

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
HandlerMapping.java ✘ DispatcherServlet.java GetBoardController.java GetBoardListController.java
1 package kr.ac.inje.comsi.view.controller;
2
3+ import java.util.HashMap;└
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10+ public HandlerMapping() {
11     mappings = new HashMap<String, Controller>();
12     mappings.put("/login.do", new LoginController());
13     mappings.put("/getBoardList.do", new GetBoardListController()); ← GetBoardController 추가
14     mappings.put("/getBoard.do", new GetBoardController());
15 }
16
17 // 요청경로에 따른 Controller 얻기
18+ public Controller getController(String path) {
19     return mappings.get(path);
20 }
21 }
22 }
```

### [3] 글 수정 구현 – UpdateBoardController 클래스

```
LogoutController.java GetBoardController.java UpdateBoardController.java

1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 import kr.ac.inje.comsi.board.BoardVO;
7 import kr.ac.inje.comsi.board.impl.BoardDAO;
8 import kr.ac.inje.comsi.view.controller.Controller;
9
10 public class UpdateBoardController implements Controller {
11
12     @Override
13     public String handlerRequest(HttpServletRequest request,
14                                 HttpServletResponse response) {
15         System.out.println("글 수정 처리");
16
17         // 1. 사용자 입력한 정보 추출
18         String title = request.getParameter("title");
19         String content = request.getParameter("content");
20         String seq = request.getParameter("seq");
21
22         // 2. DB 연동 처리
23         BoardVO vo = new BoardVO();
24         vo.setTitle(title);
25         vo.setContent(content);
26         vo.setSeq(Integer.parseInt(seq));
27
28         BoardDAO boardDAO = new BoardDAO();
29         boardDAO.updateBoard(vo);
30
31         // 3. 화면을 리턴
32         return "getBoardList.do"; //
33     }
34 }
```

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
HandlerMapping.java DispatcherServlet.java GetBoardController.java UpdateBoardController.java
1 package kr.ac.inje.comsi.view.controller;
2
3+ import java.util.HashMap;[]
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10+    public HandlerMapping() {
11         mappings = new HashMap<String, Controller>();
12         mappings.put("/login.do", new LoginController());
13         mappings.put("/getBoardList.do", new GetBoardListController());
14         mappings.put("/getBoard.do", new GetBoardController());
15         mappings.put("/updateBoard.do", new UpdateBoardController()); ← UpdateBoardController 추가
16     }
17
18     // 요청경로에 따른 Controller 얻기
19+    public Controller getController(String path) {
20         return mappings.get(path);
21     }
22 }
23
```

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

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
HandlerMapping.java DispatcherServlet.java InsertBoardController.java
1 package kr.ac.inje.comsi.view.controller;
2
3+ import java.util.HashMap;+
4
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10+    public HandlerMapping() {
11        mappings = new HashMap<String, Controller>();
12        mappings.put("/login.do", new LoginController());
13        mappings.put("/getBoardList.do", new GetBoardListController());
14        mappings.put("/getBoard.do", new GetBoardController());
15        mappings.put("/updateBoard.do", new UpdateBoardController());
16        mappings.put("/insertBoard.do", new InsertBoardController()); ← InsertBoardController 추가
17    }
18
19     // 요청경로에 따른 Controller 얻기
20+    public Controller getController(String path) {
21        return mappings.get(path);
22    }
23}
24
```

# [5] 글 삭제-DeleteBoardController 클래스

```
LogoutController.java GetBoardController.java DeleteBoardController.java
1 package kr.ac.inje.comsi.view.board;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 import kr.ac.inje.comsi.board.BoardVO;
7 import kr.ac.inje.comsi.board.impl.BoardDAO;
8 import kr.ac.inje.comsi.view.controller.Controller;
9
10 public class DeleteBoardController implements Controller {
11
12     @Override
13     public String handlerRequest(HttpServletRequest request,
14                                 HttpServletResponse response) {
15         System.out.println("글 삭제 처리");
16
17         // 1. 삭제할 게시글 번호 추출
18         String seq = request.getParameter("seq");
19
20         // 2. DB 연동 처리
21         BoardVO vo = new BoardVO();
22         vo.setSeq(Integer.parseInt(seq));
23
24         BoardDAO boardDAO = new BoardDAO();
25         boardDAO.deleteBoard(vo);
26
27         // 3. 삭제하고 목록 화면을 리턴
28         return "getBoardList.do"; //
29     }
30 }
```

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
HandlerMapping.java DispatcherServlet.java InsertBoardController.java DeleteBoardController.java
1 package kr.ac.inje.comsi.view.controller;
2
3 import java.util.HashMap;
4
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10    public HandlerMapping() {
11        mappings = new HashMap<String, Controller>();
12        mappings.put("/login.do", new LoginController());
13        mappings.put("/getBoardList.do", new GetBoardListController());
14        mappings.put("/getBoard.do", new GetBoardController());
15        mappings.put("/updateBoard.do", new UpdateBoardController());
16        mappings.put("/insertBoard.do", new InsertBoardController());
17        mappings.put("/deleteBoard.do", new DeleteBoardController()); ← DeleteBoardController 추가
18    }
19
20    // 요청경로에 따른 Controller 얻기
21    public Controller getController(String path) {
22        return mappings.get(path);
23    }
24 }
25 }
```

DeleteBoardController 추가

# [6] 로그아웃 구현-LogoutController 클래스

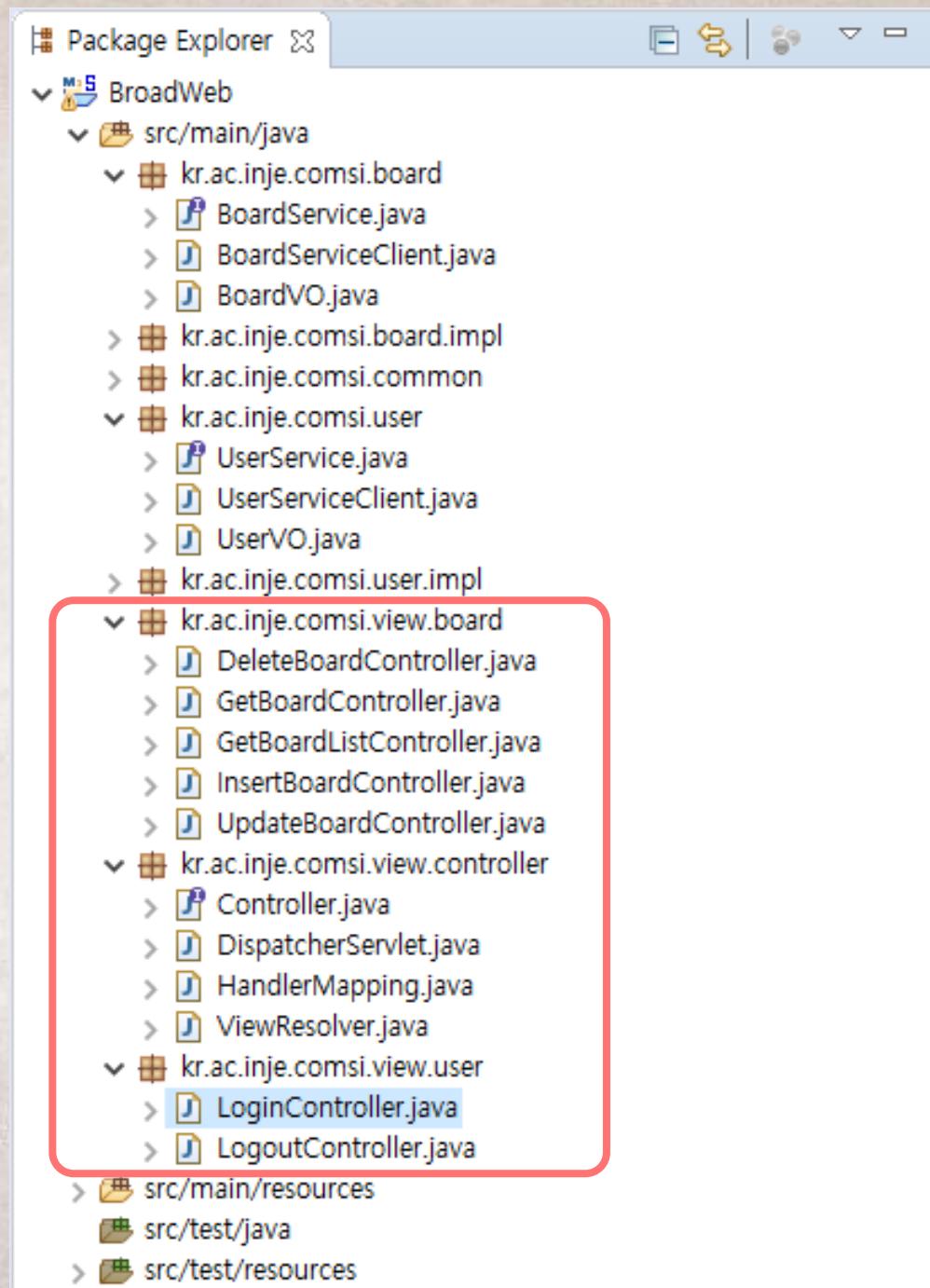
```
HandlerMapping.java DispatcherServlet.java LogoutController.java
1 package kr.ac.inje.comsi.view.user;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5 import javax.servlet.http.HttpSession;
6
7 import kr.ac.inje.comsi.view.controller.Controller;
8
9 public class LogoutController implements Controller {
10
11    @Override
12    public String handlerRequest(HttpServletRequest request,
13                                HttpServletResponse response) {
14        System.out.println("로그아웃 처리");
15
16        // 1. 브라우저와 연결된 세션 객체를 강제 종료
17        HttpSession session = request.getSession();
18        session.invalidate();
19        // 2. 세션 종료후, 메인 화면으로 이동
20        return "login";
21    }
22
23 }
```

# 새로운 Controller를 추가했으므로 HandlerMapping 설정

```
HandlerMapping.java ✘ DispatcherServlet.java ✘ LogoutController.java
1 package kr.ac.inje.comsi.view.controller;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 public class HandlerMapping {
7     private Map<String, Controller> mappings;
8
9     // 요청 경로와 해당 컨트롤러를 HashMap으로 생성
10    public HandlerMapping() {
11        mappings = new HashMap<String, Controller>();
12        mappings.put("/login.do", new LoginController());
13        mappings.put("/getBoardList.do", new GetBoardListController());
14        mappings.put("/getBoard.do", new GetBoardController());
15        mappings.put("/updateBoard.do", new UpdateBoardController());
16        mappings.put("/insertBoard.do", new InsertBoardController());
17        mappings.put("/deleteBoard.do", new DeleteBoardController());
18        mappings.put("/logout.do", new LogoutController()); ← LogoutController 추가
19    }
20
21    // 요청경로에 따른 Controller 얻기
22    public Controller getController(String path) {
23        return mappings.get(path);
24    }
25 }
26 }
```

LogoutController 추가

# 프로젝트 클래스 구조



# 실행결과

로그인

localhost:8080/Board/login.jsp

나의 북마크 VOA 한국어 텐진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 하이브리드 인공지능

# 로그인

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

# 글 목록 보기

글 목록

localhost:8080/Board/getBoardList.jsp

나의 북마크 VOA 한국어 텐진난만 :: 선형 회귀 머신러닝 - 수식 없이 Variability vs. Complexity 조승연의 굿모닝팝스 고급두뇌를 위한 하이브리드 인공지능

## 글 목록

환영합니다. [Logout](#)

번호	제목	작성자	등록일	조회수
5	<a href="#">글 등록</a>	글 등록	2018-09-10	0
4	<a href="#">JDBD 테스트2</a>	관리자	2018-05-27	0
3	<a href="#">JDBD 테스트</a>	관리자	2018-05-27	0
2	<a href="#">임시 제목</a>	홍길동	2018-05-12	0
1	<a href="#">가입인사</a>	관리자	2018-04-08	0

[새글 등록](#)