

2025년 상반기 K-디지털 트레이닝

# FrontController (심화1)

---

[KB] IT's Your Life

✓ 다음 조건으로 HomeController를 작성하세요.

- 패키지: org.scoula.ex05.controller
- 메서드:
  - 메서드명: getIndex
  - Command 인터페이스와 동일한 구조를 가짐
  - 뷰이름 index 리턴

 WEB-INF/views/index.jsp를 다음과 같이 작성한다.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
  <title>JSP - Hello World</title>
</head>
<body>
  <h1>FrontController</h1>
  <br/>
  <a href="/todo/list">Todo 목록보기</a>
</body>
</html>
```

## FrontController

[Todo 목록보기](#)

- ✓ 다음 조건을 처리하는 부분을 FrontControllerServlet에 추가하세요.
  - 뷰 이름의 접두어는 /WEB-INF/views/, 접미어는 .jsp로 함
  - 필드
    - String prefix
    - String suffix
    - HomeController→ 각각 바로 초기화 한다.
  - init() 메서드에 "/"요청에 대한 HomeController의 getIndex 매핑을 getMap에 추가하세요.

## FrontControllerServlet의 execute() 메서드에 다음을 추가하세요.

```
@WebServlet(name = "frontControllerServlet", value = "/")
public class FrontControllerServlet extends HttpServlet {
    ...
    public void execute(Command command, HttpServletRequest req, HttpServletResponse resp)
        throws IOException, ServletException {

        String viewName = command.execute(req, resp);
        if(viewName.startsWith("redirect:")) { // redirect 처리
            resp.sendRedirect(viewName.substring("redirect:".length()));
        } else { // forward 처리
            String view = prefix + viewName + suffix;
            RequestDispatcher dis = req.getRequestDispatcher(view);
            dis.forward(req, resp);
        }
    }
    ...
}
```

- ✔ URL 맵핑 맵에 없는 경우 404에러 처리를 위한 404.jsp 파일을 작성하세요.

- ✓ FrontController의 doGet() 메서드에서 요청을 처리할 수 없는 경우 404 페이지로 포워딩하는 코드를 추가하세요.

✓ 다음 url을 요청하여 그 결과를 확인하세요.

- <http://localhost:8080/>

## FrontController

[Todo 목록보기](#)

- <http://localhost:8080/test>

# 404

요청하신 페이지가 존재하지 않습니다.



2025년 상반기 K-디지털 트레이닝

# FrontController (심화2)

---

[KB] IT's Your Life

- ✓ 다음 테이블 정보를 이용하여 TodoController를 작성하세요.

METHOD	URL	컨트롤러 메서드	뷰 이름
GET	/todo/list	getList	todo/list
GET	/todo/view	getView	todo/view
GET	/todo/create	getCreate	todo/create
POST	/todo/create	postCreate	redirect:/todo/list
GET	/todo/update	getUpdate	todo/update
POST	/todo/update	postUpdate	redirect:/todo/list
POST	/todo/delete	postdDelete	redirect:/todo/list

- 각 메서드에서는 다음 양식으로 콘솔에 출력한다.
  - [메서드명] [요청 URL]
- getList()에서는 "Todo 1", "Todo 2", "Todo 3"를 List 로 만들고, request 스코프에 속성명 todoList로 저장한다.

- ✓ 다음 조건으로 list.jsp를 작성하세요.
  - 올바른 뷰 위치에 작성함
  - todoList를 EL로 단순 출력하세요.
  - 상세보기 링크 view
  - 새 Todo 링크 create

## Todo 목록 보기

[Todo 1, Todo 2, Todo 3] [상세보기](#)  
[새 Todo](#)

✓ 다음 조건으로 view.jsp를 작성하세요.

- 올바른 뷰 위치에 작성함
- 목록보기 링크 list
- 수정하기 링크 update
- 삭제를 위한 폼
  - action: delete
  - method: POST

## Todo 보기

[목록보기](#) | [수정하기](#)

삭제

✓ 다음 조건으로 create.jsp를 작성하세요.

- 생성 폼 작성
  - action: 현재 url
  - method: POST



The image shows a simple web form. At the top, the text '새 Todo 생성' (Create New Todo) is displayed in a large, bold, black font. Below this text, there is a single button with the text '제출' (Submit) in a smaller, black font. The button has a light gray background and a thin black border. The entire form is enclosed in a white box with a subtle drop shadow.

✓ 다음 조건으로 update.jsp를 작성하세요.

- 수정 폼 작성
  - action: 현재 url
  - method: POST

Todo 수정

제출

- ✓ 각 요청별로 TodoController의 메서드 맵핑관계를 FrontController의 getMap과 postMap에 추가하세요.
- ✓ 각 요청별로 잘 동작하는지 확인하세요.

- ✓ **FrontController의 재사용을 위해 DispatcherServlet 부모 클래스를 작성하세요.**
  - 실제 url 매핑은 createMap() 메서드에서 하게 됨
  - 실제 구현은 자식 클래스가 작성함



- ✓ DispatcherServlet을 상속 받아 작성하는 것으로 FrontControllerServlet를 리팩토링하세요.