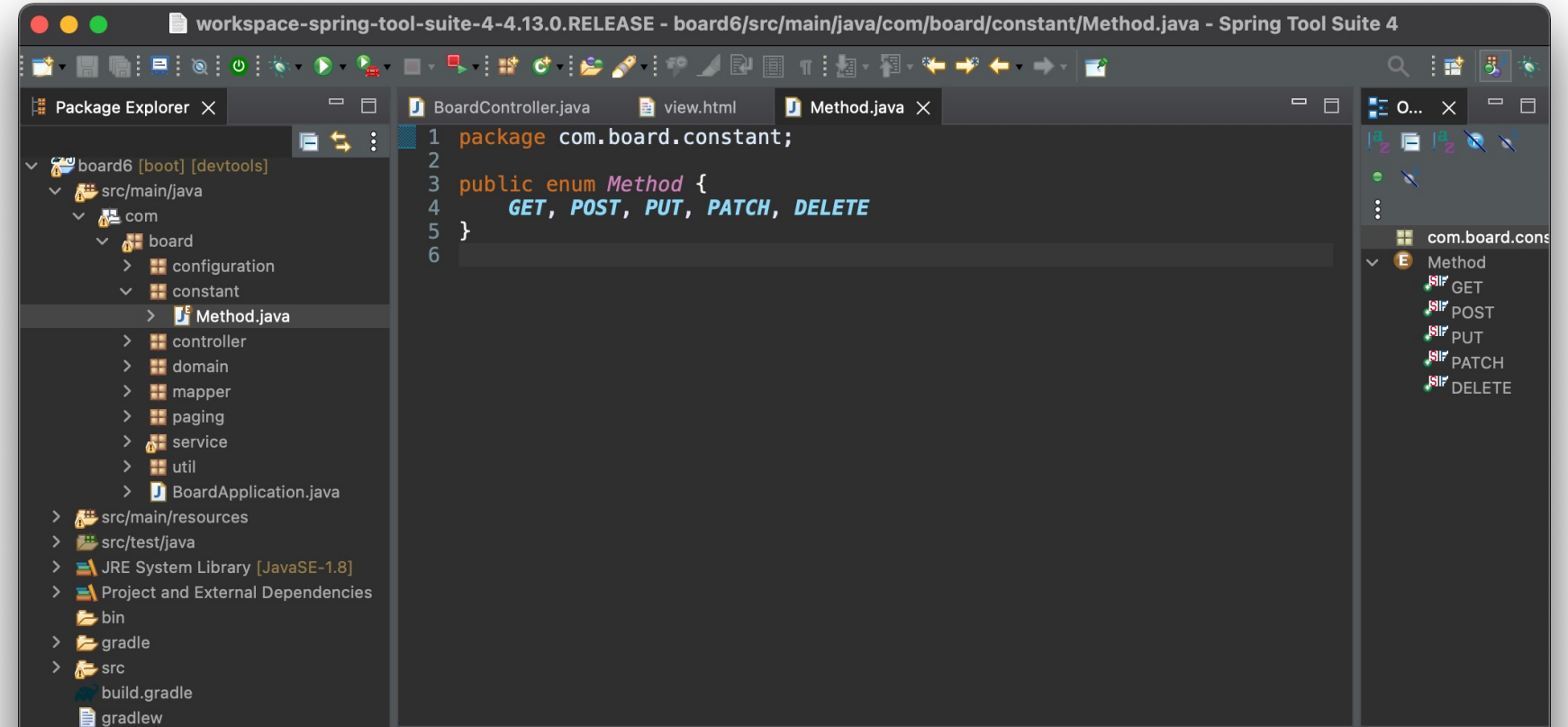
chapter05

Spring Boot

1. 메시지 처리

■ Controller -> View 로 메세지 전달

- com.board 패키지에 constant 패키지를 추가 하고 Method 라는 enum 생성



■ Controller

• com.board 패키지에 util 패키지와 UiUtils 클래스를 추가

```
package com.board.util;

import java.util.Map;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestParam;

import com.board.constant.Method;

@Controller
public class UiUtils {

  public String showMessageWithRedirect(
    @RequestParam(value = "message", required = false) String message,
    @RequestParam(value = "redirectUri", required = false) String redirectUri,
    @RequestParam(value = "method", required = false) Method method,
    @RequestParam(value = "params", required = false) Map<String, Object> params, Model model) {

    model.addAttribute("message", message);
    model.addAttribute("redirectUri", redirectUri);
    model.addAttribute("method", method);
    model.addAttribute("params", params);

    return "utils/message-redirect";
  }

}
```

■ HTML

- src/main/resources 패키지의 templates 폴더에 utils 폴더를 추가 message-redirect.html을 추가

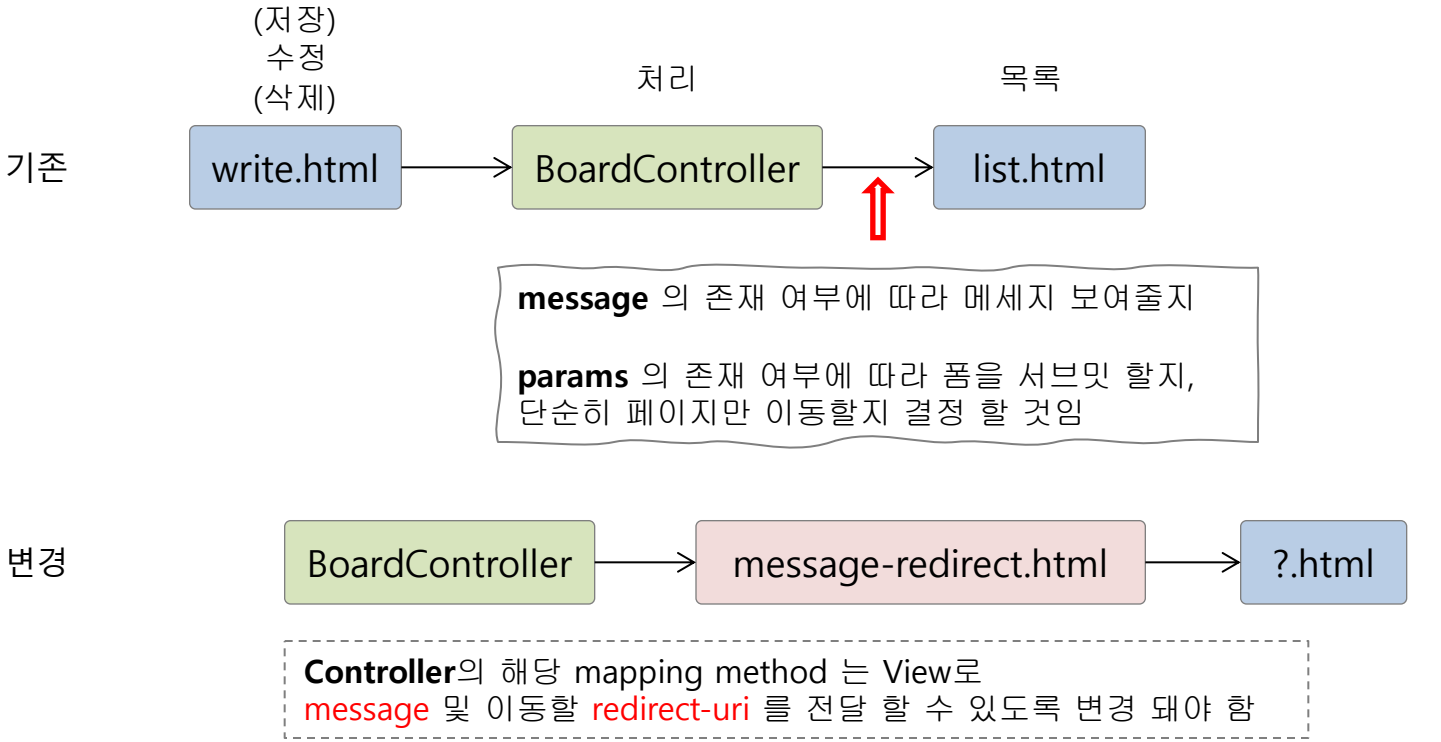
```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org" xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head th:replace="board/fragments/header :: main-head"></head>
<body>
  <form th:if="${not #maps.isEmpty( params )}" name="dataForm" th:action="${redirectUri}" th:method="${method}"
style="display: none;">
  <input th:each="key, status : ${params.keySet()}" type="hidden" th:name="${key}"
th:value="${params.get(key)}" />
  </form>

  <script th:src="@{/scripts/jquery.min.js}"></script>
  <script th:src="@{/scripts/common.js}"></script>
```

■ HTML (message-redirect.hrml)

```
  <th:block layout:fragment="script">
    <script th:inline="javascript">
    /* <![CDATA[ */

    window.onload = function() {
      //var message = /*[[ ${message} ]]*/null;
      var message = JSON.stringify(/*[[ ${message} ]]*/);
      if (isEmpty(message) == false) {
        alert(message);
      }

      var params = /*[[ ${params} ]]*/null;
      if (isEmpty(params) == false) {
        document.dataForm.submit();
      } else {
        var redirectUri = /*[[ ${redirectUri} ]]*/null;
        location.href = redirectUri;
      }
    }

    /* ]]> */
    </script>
  </th:block>
</body>
</html>
```

01 메세지 처리

■ Controller -> View 로 메세지 전달



■ Controller

- BoardController 를 UiUtils 클래스를 상속 받도록 변경

```
@Controller
public class BoardController extends UiUtils{
```

- BoardController . registerBoard() 변경

```
@PostMapping(value = "/board/register.do")
public String registerBoard(final BoardDTO params, Model model) {
  try {
    boolean isRegistered = boardService.registerBoard(params);
    if (isRegistered == false) {
      return showMessageWithRedirect("게시글 등록에 실패하였습니다.", "/board/list.do", Method.GET, null, model);
    }
  } catch (DataAccessException e) {
    return showMessageWithRedirect("데이터베이스 처리 과정에 문제가 발생하였습니다.", "/board/list.do", Method.GET, null, model);

  } catch (Exception e) {
    return showMessageWithRedirect("시스템에 문제가 발생하였습니다.", "/board/list.do", Method.GET, null, model);
  }

  return showMessageWithRedirect("게시글 등록이 완료되었습니다.", "/board/list.do", Method.GET, null, model);
}
```

■ Controller

- BoardController . deleteBoard() 변경

```
@PostMapping(value = "/board/delete.do")
public String deleteBoard(@RequestParam(value = "idx", required = false) Long idx, Model model) {
  if (idx == null) {
    return showMessageWithRedirect("올바르지 않은 접근입니다.", "/board/list.do", Method.GET, null, model);
  }

  try {
    boolean isDeleted = boardService.deleteBoard(idx);
    if (isDeleted == false) {
      return showMessageWithRedirect("게시글 삭제에 실패하였습니다.", "/board/list.do", Method.GET, null, model);
    }
  } catch (DataAccessException e) {
    return showMessageWithRedirect("데이터베이스 처리 과정에 문제가 발생하였습니다.", "/board/list.do", Method.GET, null, model);

  } catch (Exception e) {
    return showMessageWithRedirect("시스템에 문제가 발생하였습니다.", "/board/list.do", Method.GET, null, model);
  }

  return showMessageWithRedirect("게시글 삭제가 완료되었습니다.", "/board/list.do", Method.GET, null, model);
}
```

01 메세지 처리

■ Run App

- 애플리케이션 실행하고 게시글 등록, 삭제 수행