

2024년 상반기 K-디지털 트레이닝

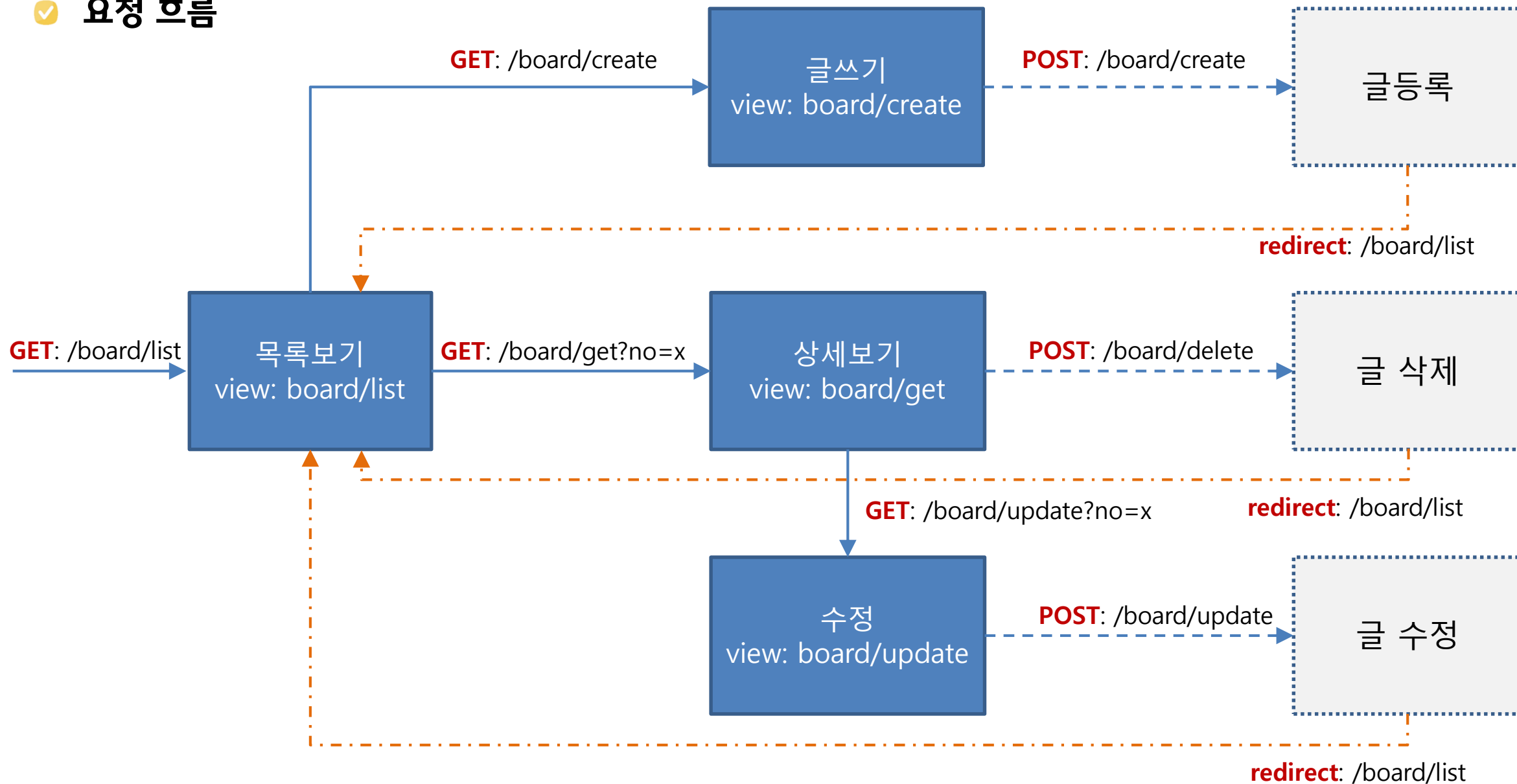
프레젠테이션(웹) 계층의 CRUD 구현

[KB] IT's Your Life

✔ BoardController의 분석

Task	URL	Method	Parameter	Form	URL 이동
전체 목록	/board/list	GET			
등록 처리	/board/create	POST	모든 항목	입력화면 필요	이동
조회	/board/get	GET	no=123		
수정 처리	/board/update	POST	no	입력화면 필요	이동
삭제 처리	/board/delete	POST	모든 항목	입력화면 필요	이동

✓ 요청 흐름



BoardController.java

```
package org.scoula.board.controller;

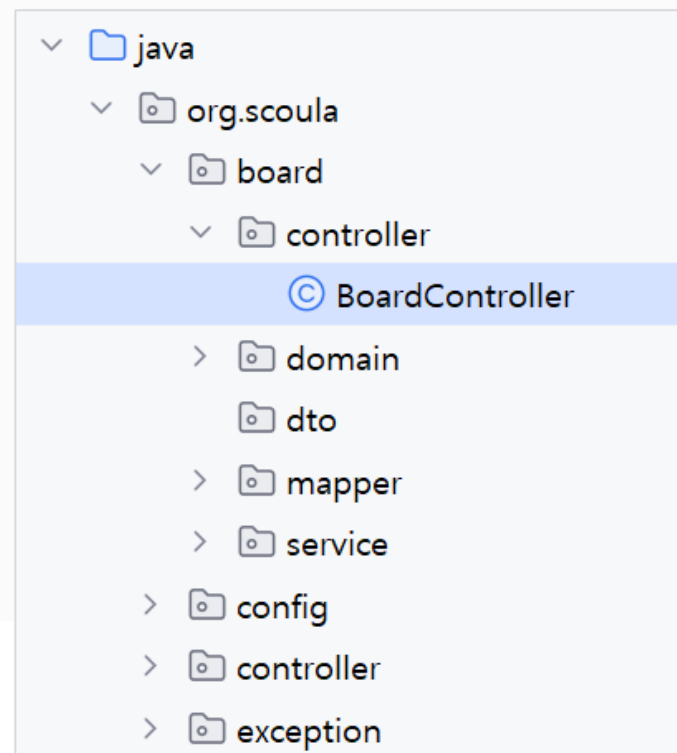
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import lombok.extern.log4j.Log4j;

@Controller
@Log4j
@RequestMapping("/board")
@RequiredArgsConstructor
public class BoardController {

    final private BoardService service;

}
```



ServletConfig.java

```
...

@EnableWebMvc
@ComponentScan(basePackages = {
    "org.scoula.exception",
    "org.scoula.controller",
    "org.scoula.board.controller"
})
public class ServletConfig implements WebMvcConfigurer {
    ...
}
```

✓ 컨트롤러 테스트

- 웹 요청을 실제로 하는 것이 아님
- MockMvc
 - 웹 서버에 요청을 보낸 것과 같은 효과를 내는 메서드 제공
 - 메서드 호출로 웹 요청(GET/POST/UPDATE/DELETE 등)을 수행
- 테스트 마다 새로운 MockMvc 객체 생성이 필요
→ @BeforeEach 어노테이션 메서드 설정

test :: BoardControllerTest.java

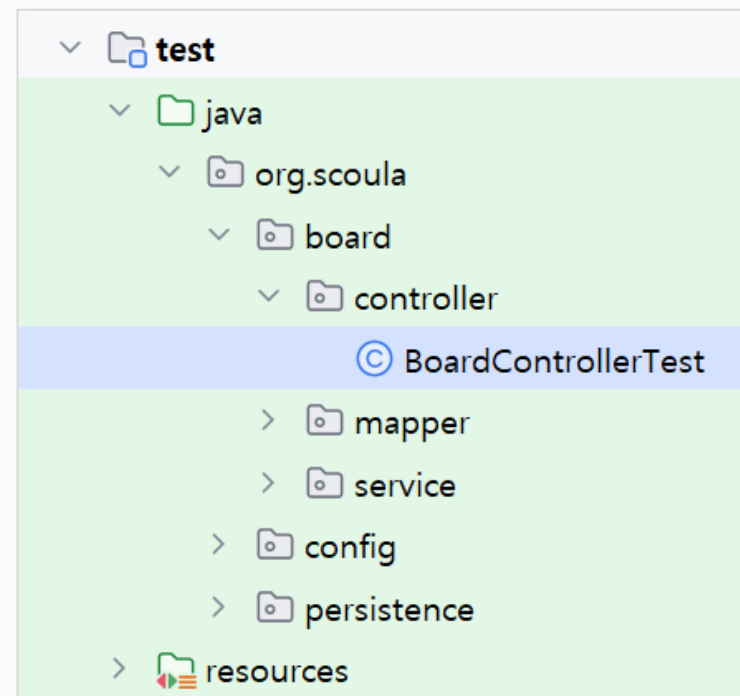
```
...

@ExtendWith(SpringExtension.class)
@WebAppConfiguration
@ContextConfiguration(classes = {
    RootConfig.class,
    ServletConfig.class
})
@Log4j
public class BoardControllerTest {
    @Autowired
    BoardService service;

    @Autowired
    private WebApplicationContext ctx;

    private MockMvc mockMvc;

    @BeforeEach
    public void setup() {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(ctx).build();
    }
}
```




✓ 목록 요청

○ GET 요청

- url: /board/list
- Model 구성
 - 속성명 : list
 - 속성값: service의 getList()로 목록
- View 이름: board/list

BoardController.java

```
...
@RequestMapping("/board")
public class BoardController {
    ...

    @GetMapping("/list") 
    public void list(Model model) {

        log.info("list");
        model.addAttribute("list", service.getList());

    }
}
```

✓ BoardController의 작성 - 목록에 대한 처리와 테스트

○ Get 요청 만들기

- MockMvcRequestBuilders.get(url문자열)

○ MockMvc

- .perform(요청빌더)
 - 지정된 요청을 스프링 MVC가 처리
→ 지정된 URL을 처리하는 컨트롤러 메서드 호출
 - ResultActions 객체 리턴
 - ResultActions의 andReturn(): 컨트롤러의 처리를 결과를 리턴
- MvcResult
 - 컨트롤러에 의해 구성된 Model 및 View에 대한 정보 및 처리결과(상태 코드) 등을 가짐
 - getModelAndView() : ModelAndView 객체 리턴

✎ test :: BoardControllerTest.java

```
@Test
public void list() throws Exception {

    log.info(
        mockMvc.perform(MockMvcRequestBuilders.get("/board/list")) // ResultActions 리턴
                .andReturn() // MvcResult 리턴
                .getModelAndView() // ModelAndView 리턴
                .getModelMap() // Model 리턴
    );
}
```

```
INFO : org.scoula.board.controller.BoardController - list
INFO : org.scoula.board.service.BoardServiceImpl - getList.....
INFO : jdbc.sqlonly - select * from tbl_board
```

```
INFO : org.scoula.board.controller.BoardControllerTest - {list=[BoardDTO(no=1, title=제목 수정합니다., content=테스트 내용1,
writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 14:15:10 KST 2024), BoardDTO(no=3, title=테스트 제
목3, content=테스트 내용3, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024),
BoardDTO(no=4, title=테스트 제목4, content=테스트 내용4, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue
Jul 02 13:37:07 KST 2024), BoardDTO(no=5, title=수정된 제목, content=수정된 내용, writer=user00, regDate=Tue Jul 02 13:37:07
KST 2024, updateDate=Tue Jul 02 13:58:14 KST 2024), BoardDTO(no=6, title=새로 작성하는 글, content=새로 작성하는 내용,
writer=user0, regDate=Tue Jul 02 13:56:51 KST 2024, updateDate=Tue Jul 02 13:56:51 KST 2024), BoardDTO(no=7, title=새로 작성하
는 글, content=새로 작성하는 내용, writer=user1, regDate=Tue Jul 02 14:11:06 KST 2024, updateDate=Tue Jul 02 14:11:06 KST
2024)]}
```

```
> Task :test
```

✓ 새 글 등록

○ GET 요청

- url: /board/create
- view 이름: board/create

○ POST 요청

- url: /board/create
- 파라미터: BoardDTO
- 처리: service.create()로 실제 등록
- view 이름 : redirect:/board/list

BoardController.java

```
...
@GetMapping("/create") GET :: /board/create
public void create() {
    log.info("create");
}

@PostMapping("/create") POST :: /board/create
public String create(BoardDTO board) {

    log.info("create: " + board);

    service.create(board);

    return "redirect:/board/list";
}
```

✓ MockMvc로 Post 요청 테스트하기

```
MockMvcRequestBuilders.post(url 문자열)
    .param(키1, 값1)        // form 요소
    .param(키2, 값2)
```

test :: BoardControllerTest.java

```
@Test
public void create() throws Exception {

    String resultPage = mockMvc
        .perform(
            MockMvcRequestBuilders.post("/board/create")
                .param("title", "테스트 새글 제목")
                .param("content", "테스트 새글 내용")
                .param("writer", "user1"))
        .andReturn()
        .getModelAndView()
        .getViewName();

    log.info(resultPage);
}
```

INFO : org.scoula.board.controller.BoardController - create: BoardDTO(no=null, title=테스트 새글 제목, content=테스트 새글 내용, writer=user1, regDate=null, updateDate=null)

INFO : org.scoula.board.service.BoardServiceImpl - create.....BoardDTO(no=null, title=테스트 새글 제목, content=테스트 새글 내용, writer=user1, regDate=null, updateDate=null)

INFO : jdbc.sqlonly - insert into tbl_board (title, content, writer) values ('테스트 새글 제목', '테스트 새글 내용', 'user1')

INFO : jdbc.sqlonly - SELECT LAST_INSERT_ID()

INFO : org.scoula.board.controller.BoardControllerTest - **redirect:/board/list**

> Task :test

✓ 글 상세 보기(조회)

○ GET 요청

- url: /board/get
- 파라미터
 - no: 글 번호
- Model 구성
 - 속성명 : board
 - 속성값: service.get()으로 BoardDTO 획득
- View 이름: board/get

○ 수정의 GET 요청 처리와 동일

- url: /board/update

BoardController.java

```
...  
@GetMapping({ "/get", "/update" })  
public void get(@RequestParam("no") Long no, Model model) {  
  
    log.info("/get or update");  
    model.addAttribute("board", service.get(no));  
}
```

GET :: /board/get
GET :: /board/update

✓ Get 요청의 쿼리 파라미터 테스트

```
MockMvcRequestBuilders.get(url 문자열)
    .param(키1, 값1)      // 쿼리 파라미터
    .param(키2, 값2)
```

test :: BoardControllerTest.java

```
@Test
public void get() throws Exception {

    log.info(
        mockMvc.perform(MockMvcRequestBuilders.get("/board/get").param("no", "1"))
            .andReturn()
            .getModelAndView()
            .getModelMap()
    );
}
```

INFO : org.scoula.board.controller.BoardController - /get or update

INFO : org.scoula.board.service.BoardServiceImpl - get.....1

INFO : **jdbc.sqlonly - select * from tbl_board where no = 1**

INFO : org.scoula.board.controller.BoardControllerTest - {board=BoardDTO(no=1, title=제목 수정합니다., content=테스트 내용1, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 14:15:10 KST 2024), org.springframework.validation.BindingResult.board=org.springframework.validation.BeanPropertyBindingResult: 0 errors}
> Task :test

✓ 글 수정 처리

○ GET 요청

- url: /board/update
- BoardController의 get()에서 처리

○ POST 요청

- url: /board/update
- 파라미터: BoardDTO
- 처리: service.update()로 실제 수정
- view 이름 : redirect:/board/list

BoardController.java

```
@PostMapping("/update") POST :: /board/update
public String update(BoardDTO board) {
    log.info("update:" + board);

    service.update(board);

    return "redirect:/board/list";
}
```

test :: BoardControllerTest.java

```
@Test
public void update() throws Exception {

    String resultPage = mockMvc.perform(
        MockMvcRequestBuilders.post("/board/update")
            .param("no", "1")
            .param("title", "수정된 테스트 새글 제목")
            .param("content", "수정된 테스트 새글 내용")
            .param("writer", "user00"))
        .andReturn()
        .getModelAndView()
        .getViewName();

    log.info(resultPage);

}
```

INFO : org.scoula.board.controller.BoardController - update:BoardDTO(no=1, title=수정된 테스트 새글 제목, content=수정된 테스트 새글 내용, writer=user00, regDate=null, updateDate=null)
INFO : org.scoula.board.service.BoardServiceImpl - update.....BoardDTO(no=1, title=수정된 테스트 새글 제목, content=수정된 테스트 새글 내용, writer=user00, regDate=null, updateDate=null)
INFO : jdbc.sqlonly - **update tbl_board set title = '수정된 테스트 새글 제목', content = '수정된 테스트 새글 내용', writer = 'user00', update_date = now() where no = 1**

INFO : org.scoula.board.controller.BoardControllerTest - **redirect:/board/list**
> Task :test

✓ 삭제 처리

- GET 요청 없음
- POST 요청
 - url: /board/delete
 - 파라미터: no - 삭제할 글 번호
 - 처리: service.delete()로 실제 삭제
 - view 이름 : redirect:/board/list

BoardController.java

```
...
@PostMapping("/delete") POST :: /board/delete
public String delete(@RequestParam("no") Long no) {

    log.info("delete..." + no);
    service.delete(no);

    return "redirect:/board/list";
}
```


test :: BoardControllerTest.java

```
@Test
public void delete() throws Exception {
    // 삭제전 데이터베이스에 게시물 번호 확인할 것
    String resultPage = mockMvc.perform(
        MockMvcRequestBuilders
            .post("/board/delete")
            .param("no", "25") )
        .andReturn()
        .getModelAndView()
        .getViewName();

    log.info(resultPage);
}
```

INFO : org.scoula.board.controller.BoardController - delete...25

INFO : org.scoula.board.service.BoardServiceImpl - delete....25

INFO : jdbc.sqlonly - **delete from tbl_board where no = 25**

INFO : org.scoula.board.controller.BoardControllerTest - **redirect:/board/list**

> Task :test