KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

비즈니스 계층 (심화 1)

[KB] IT's Your Life

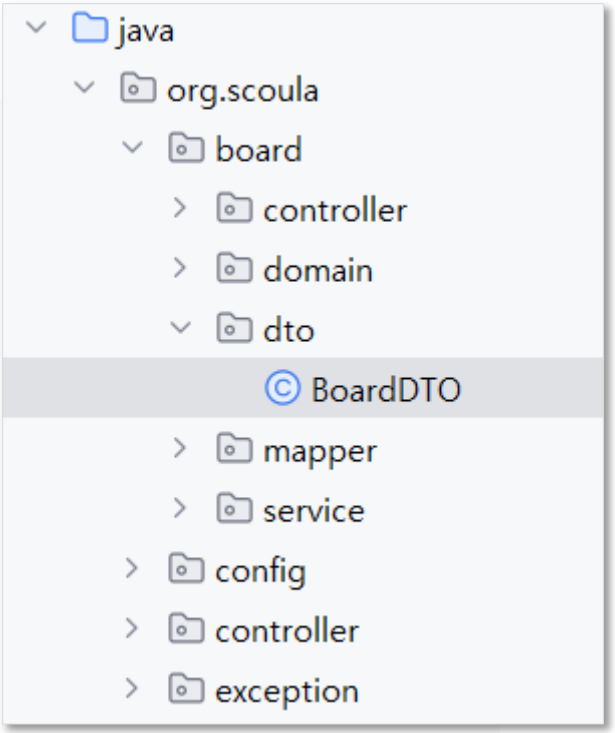
KB 국민은행

비즈니스 계층

- **다음과 같은 조건을 가지는 BoardDTO 클래스를 정의하세요**
  - 패키지: org.scoula.board.dto
  - BoradVO에 대한 DTO 객체
  - BoardVO로부터 DTO 객체를 생성하는 팩토리 메서드 of() 정의
  - BoardVO로 변환하는 toVo() 메서드

BoardDTO.java

```
package org.scoula.board.dto;
...

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class BoardDTO {
    private Long no;
    private String title;
    private String content;
    private String writer;
    private Date regDate;
    private Date updateDate;
```



BoardDTO.java

```
    // VO → DTO 변환
    public static BoardDTO of(BoardVO vo) {
        return vo == null ? null : BoardDTO.builder()
                .no(vo.getNo())
                .title(vo.getTitle())
                .content(vo.getContent())
                .writer(vo.getWriter())
                .regDate(vo.getRegDate())
                .updateDate(vo.getUpdateDate())
                .build();
    }

    // DTO → VO 변환
    public BoardVO toVo() {
        return BoardVO.builder()
                .no(no)
                .title(title)
                .content(content)
                .writer(writer)
                .regDate(regDate)
                .updateDate(updateDate)
                .build();
    }
}
```

비즈니스 계층

- RootConfig의 컴포넌트 스캔 패키지에 org.scoula.board.service를 추가하세요.

RootConfig.java

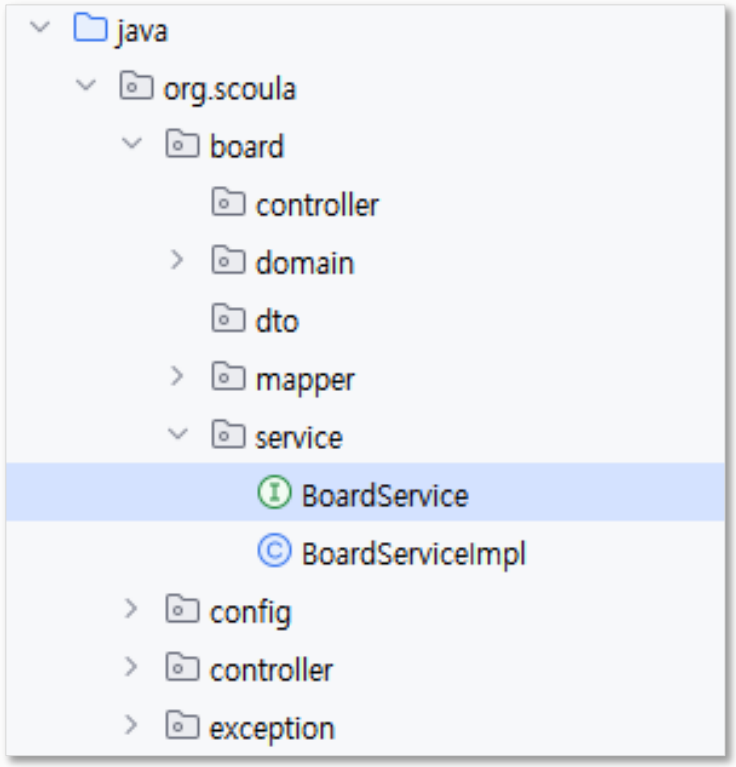
```
package org.scoula.config;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

@Configuration
@PropertySource({"classpath:/application.properties"})
@ComponentScan(basePackages={ "org.scoula.board.service" })
@MapperScan(basePackages  = {"org.scoula.board.mapper"})
public class RootConfig {

        ...
}
```

비즈니스 계층

- Board에 대한 CRUD를 수행하는 메서드를 가지는 BoardService 인터페이스를 정의하세요.
  - 패키지: org.scoula.board.service

BoardService.java

```
package org.scoula.board.service;

import org.scoula.board.dto.BoarDTO;

import java.util.List;
import java.util.Optional;

public interface BoardService {
    public List<BoardDTO> getList();

    public BoardDTO get(Long no);

    public void create(BoardDTO board);

    public boolean update(BoardDTO board);

    public boolean delete(Long no);

}
```



비즈니스 계층

- **BoardService 인터페이스의 구현체 BoardServiceImpl 클래스를 정의하세요.**
  - 멤버변수 BoardMapper는 생성자 주입으로 받음
  - 실제 메서드 구현은 추후 진행 예정

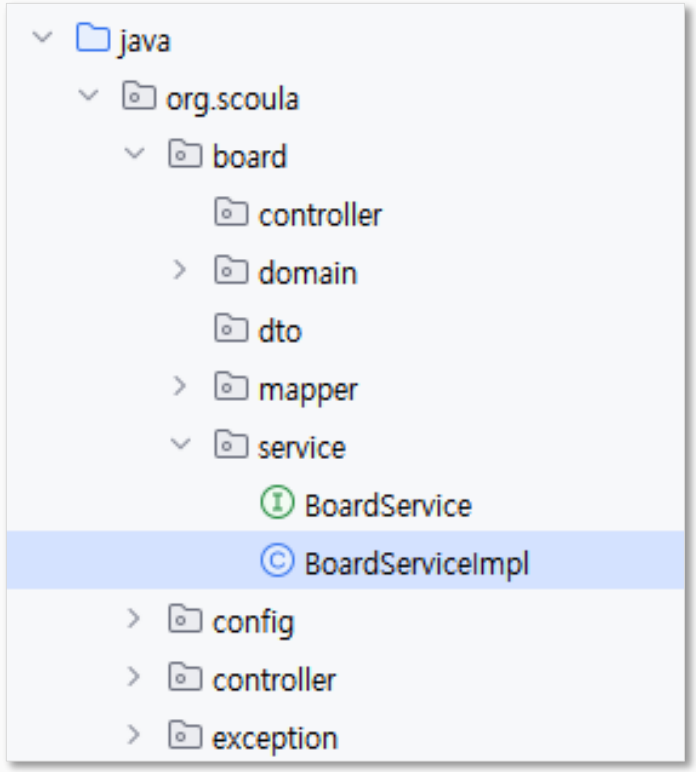
  - 테스트 클래스 BoardServiceTest의 기본 골격 작성

BoardServiceImpl.java

```
package org.scoula.board.service;

import java.util.List;

import org.springframework.stereotype.Service;
import org.scoula.board.domain.BoardVO;
import org.scoula.board.mapper.BoardMapper;

import lombok.AllArgsConstructor;
import lombok.extern.log4j.Log4j2;

@Log4j2
@Service
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService {

    final private BoardMapper mapper;
```

final 멤버를 인자로 가지는 생성자 추가

생성자가 1개인 경우 생성자 주입으로 초기화

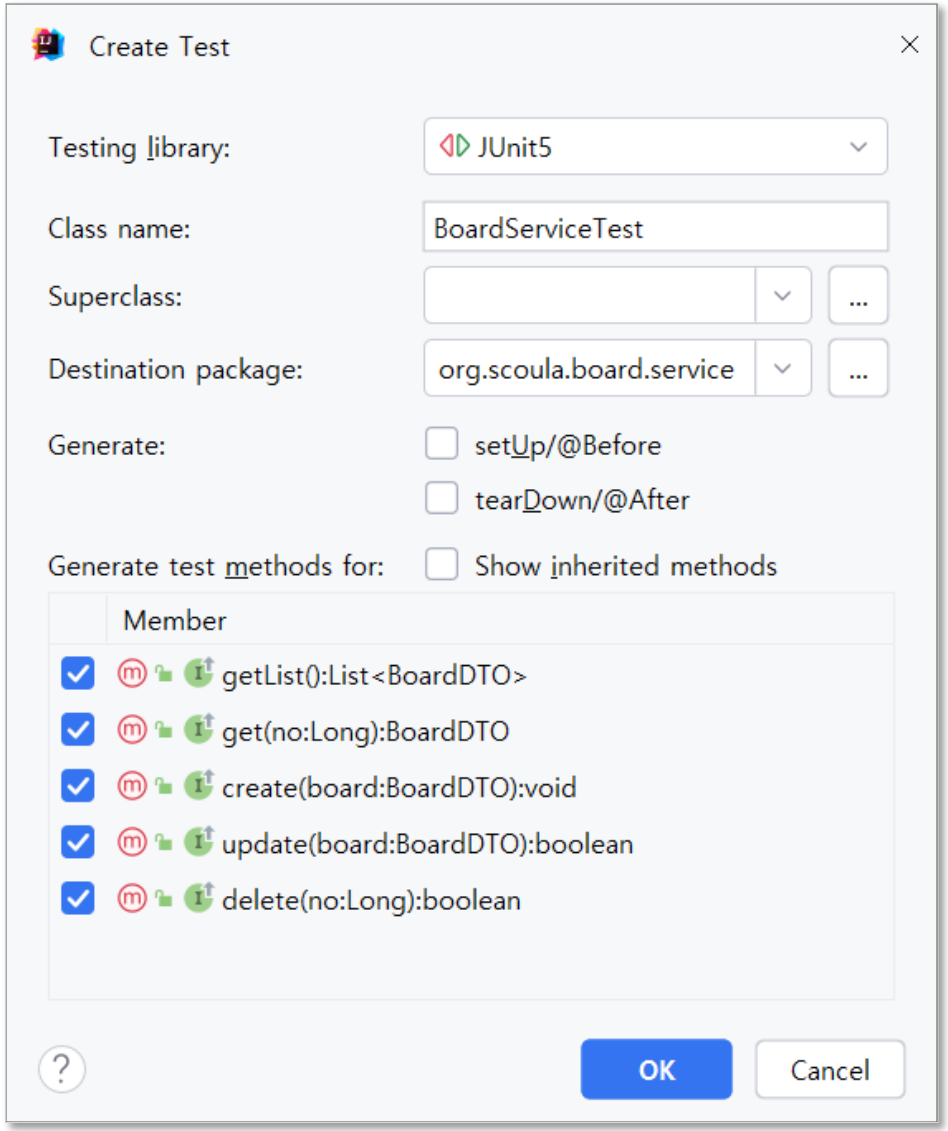


BoardServiceImpl.java

```
        @Override
        public List<BoardDTO> getList() { … }

        @Override
        public BoardDTO get(Long no) { … }

        @Override
        public void create(BoardDTO board) { … }

        @Override
        public boolean update(BoardDTO board) { … }

        @Override
        public boolean delete(Long no) { … }
        }

}
```

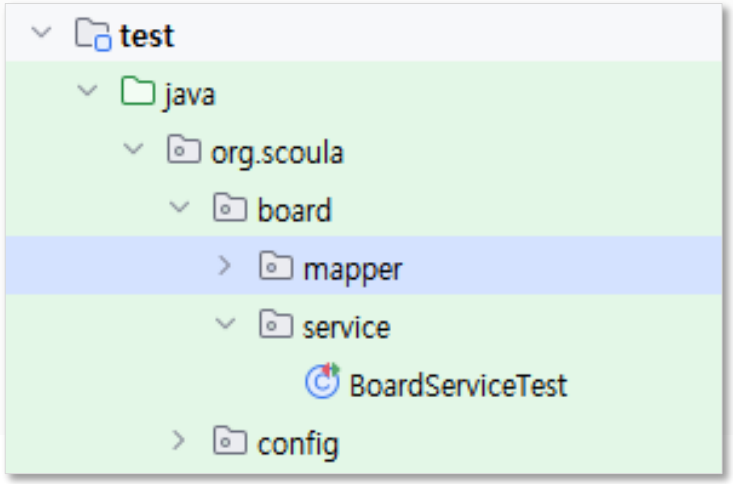
비즈니스 계층

test :: BoardServiceTest.java



test :: BoardServiceTest.java

```
…

@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = {RootConfig.class} )
@Log4j2
public class BoardServiceTest {

        @Autowired
        private BoardService service;

        …

}
```



비즈니스 계층

- 게시글 목록을 추출하는 getList() 메서드를 정의하고, 단위 테스트를 진행하세요.

BoardServiceImpl.java

```
@Override
public List<BoardVO> getList() {

        log.info("getList.....");

    return mapper.getList().stream()        // BoardVO의 스트림
        .map(BoardDTO::of)                  // BoardDTO의 스트림
        .toList();                          // List<BoardDTO> 변환
}
```

test :: BoardServiceTest.java

```
@Test
public void getList() {

        for(BoardDTO board: service.getList()) {
                log.info(board);
        }

}
```

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

INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024)
INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(no=2, title=테스트 제목2, content=테스트 내용2, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024)
INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(no=3, title=테스트 제목3, content=테스트 내용3, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024)
INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(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)
INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(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)
INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(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)
> Task :test

비즈니스 계층

- **게시글 하나를 추출하는 get(Long no) 메서드를 정의하고, 단위 테스트를 진행하세요.**
  - Optional을 이용하여 지정한 번호의 글이 없는 경우 NoSuchElementException 발생

BoardServiceImpl.java

```
@Override
public BoardDTO get(Long no) {

        log.info("get....." + no);

        BoardDTO board = BoardDTO.of(mapper.get(no));
        return Optional.ofNullable(board)
                .orElseThrow(NoSuchElementException::new);
 }
```

test :: BoardServiceTest.java

```
@Test
void get() {
    log.info(service.get(1L));
}
```

INFO : org.scoula.board.service.BoardServiceImpl - get.....1
INFO : **jdbc.sqlonly - select * from tbl_board where no = 1**

INFO : org.scoula.board.service.BoardServiceImplTest - BoardVO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024)
> Task :test

비즈니스 계층

- 게시글 하나를 생성하는 create(BoardDTO) 메서드를 정의하고, 단위 테스트를 진행하세요.

BoardServiceImpl.java

```
@Override
public void create(BoardDTO board) {

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

        mapper.create(board.toVo());
}
```

test :: BoardServiceTest.java

```
@Test
public void create() {

        BoardDTO board = new BoardDTO();
        board.setTitle("새로 작성하는 글");
        board.setContent("새로 작성하는 내용");
        board.setWriter("user1");

        service.create(board);

        log.info("생성된 게시물의 번호: " + board.getNo());
}
```

INFO : org.scoula.board.service.BoardServiceImpl - create.....BoardVO(no=null, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user1, regDate=null, updateDate=null)
INFO : jdbc.sqlonly - **insert into tbl_board (title, content, writer) values ('새로 작성하는 글', '새로 작성하는 내용', 'user1')**

비즈니스 계층

- 게시글 하나를 수정하는 update(BoardDTO) 메서드를 정의하고, 단위 테스트를 진행하세요.

BoardServiceImpl.java

```
@Override
public boolean update(BoardDTO board) {

        log.info("update....." + board);

        return mapper.update(board.toVo()) == 1;
}
```

test :: BoardServiceTest.java

```
@Test
public void update() {

    BoardDTO board = service.get(1L);

    board.setTitle("제목 수정합니다.");
    log.info("update RESULT: " + service.update(board));
}
```

INFO : org.scoula.board.service.BoardServiceImpl - update.....BoardVO(no=1, title=제목 수정합니다., content=테스트 내용1, writer=user00, regDate=Tue Jul 02 13:37:07 KST 2024, updateDate=Tue Jul 02 13:37:07 KST 2024)
INFO : jdbc.sqlonly - **update tbl_board set title = '제목 수정합니다.', content = '테스트 내용1', writer = 'user00', update_date = now() where no = 1**

INFO : org.scoula.board.service.BoardServiceImplTest - update RESULT: true
> Task :test

비즈니스 계층

- 게시글 하나를 삭제하는 delete(Long no) 메서드를 정의하고, 단위 테스트를 진행하세요.

BoardServiceImpl.java

```
@Override
public boolean delete(Long no) {

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

        return mapper.delete(no) == 1;
}
```

test :: BoardServiceTest.java

```
@Test
public void delete() {

        // 게시물 번호의 존재 여부를 확인하고 테스트할 것
        log.info("delete RESULT: " + service.delete(2L));

}
```

INFO : org.scoula.board.service.BoardServiceImpl - delete....2
INFO : jdbc.sqlonly - **delete from tbl_board where no = 2**

INFO : org.scoula.board.service.BoardServiceImplTest - delete RESULT: true
> Task :test

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

비즈니스 계층 (심화 2)

[KB] IT's Your Life

KB 국민은행

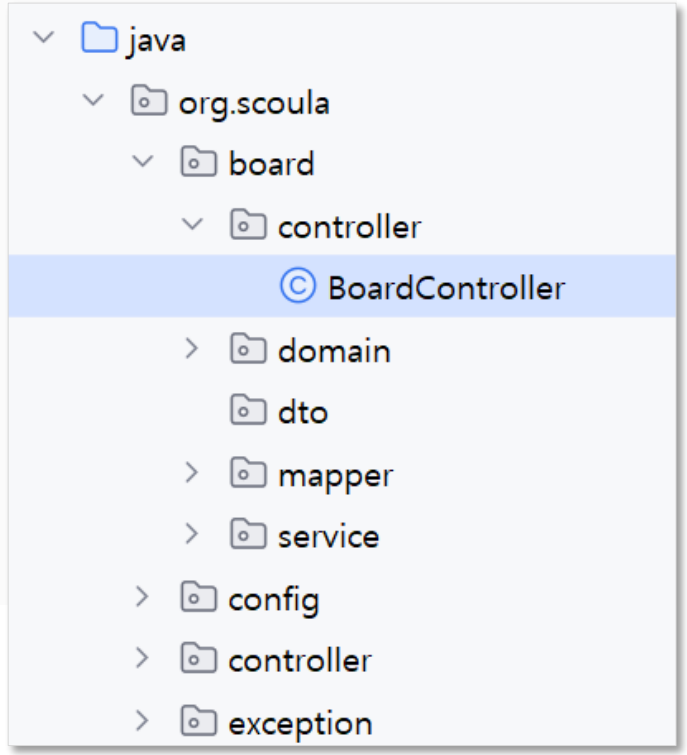
프레젠테이션(웹) 계층의 CRUD 구현

- **BoardController 클래스의 기본 모양을 정의하세요.**
  - 패키지: org.scoula.board.controller
  - BoardService는 생성자 주입을 통해 설정
  - 공통 url: /board

- **ServletConfig에서 컴포넌트 스캔에 탐지되도록 설정하세요.**

BoardController.java

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

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

import lombok.extern.log4j.Log4j2;

@Controller
@Log4j2
@RequestMapping("/board")
@RequiredArgsConstructor
public class BoardController {


        final private BoardService service;

}
```



프레젠테이션(웹) 계층의 CRUD 구현

ServletConfig.java

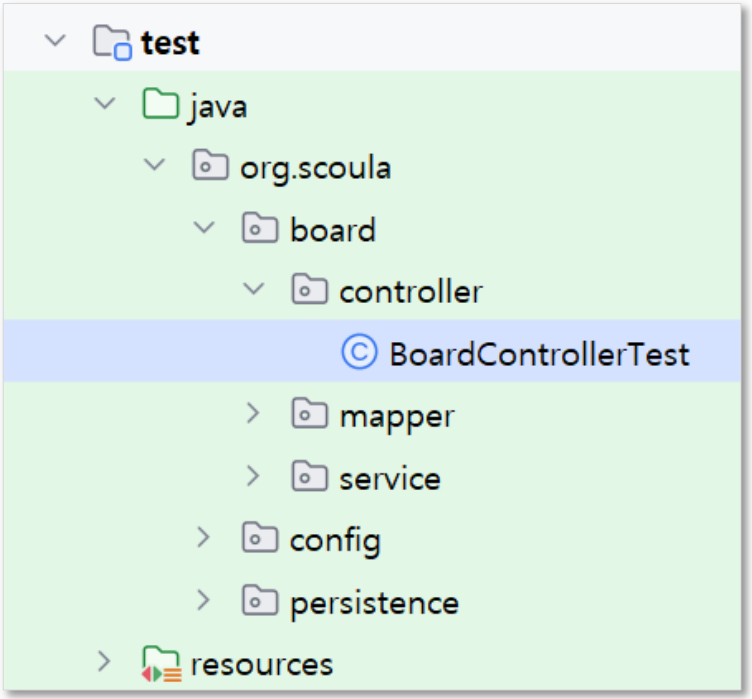
```
…

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

프레젠테이션(웹) 계층의 CRUD 구현

- **컨트롤러 테스트 클래스 BoardControllerTest 클래스의 기본 구조를 정의하세요.**
  - 매 테스트마다 MockMvc 객체가 초기화되도록 구성

test :: BoardControllerTest.java

```
…

@ExtendWith(SpringExtension.class)
@WebAppConfiguration
@ContextConfiguration(classes = {
    RootConfig.class,
    ServletConfig.class
})
@Log4j2
public class BoardControllerTest {
    @Autowired
    BoardService service;

    @Autowired
    private WebApplicationContext ctx;

    private MockMvc mockMvc;

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



프레젠테이션(웹) 계층의 CRUD 구현

- **목록 요청을 처리하는 getList() 메서드를 추가하고, 단위 테스트를 진행하세요.**
  - GET 요청
    - url: /board/list
    - Model 구성
      - 속성명 : list
      - 속성값: service의 getList()로 목록
    - View 이름: board/list

프레젠테이션(웹) 계층의 CRUD 구현

BoardController.java

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

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

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

        }
}
```

GET :: /board/list

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 제목3, 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 작성하는 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

프레젠테이션(웹) 계층의 CRUD 구현

- **새 글 등록 요청을 처리하는 create() 메서드를 추가하고, 단위 테스트를 진행하세요.**
  - 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")
public void create() {
        log.info("create");
}

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

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

        service.create(board);

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

GET :: /board/create

POST :: /board/create

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, wr
iter=user1, regDate=null, updateDate=null)
INFO : org.scoula.board.service.BoardServiceImpl - create.....BoardDTO(no=null, title=테스트 새글 제목, content=테스트 새글 내용, w
riter=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 요청을 처리하는 get() 메서드를 추가하고, 단위 테스트를 진행하세요.

- 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", "/updat
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

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 요청을 처리하는 update() 메서드를 추가하고, 단위 테스트를 진행하세요.**

  - 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

POST :: /board/update

```
@PostMapping("/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 요청을 처리하는 delete() 메서드를 추가하고, 단위 테스트를 진행하세요.

- 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")
public String delete(@RequestParam("no") Long no) {

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

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

POST :: /board/delete

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
```